Java

آموزش جاوا

مولف:مهندس افشین رفوآ

www.tahlildadeh.com

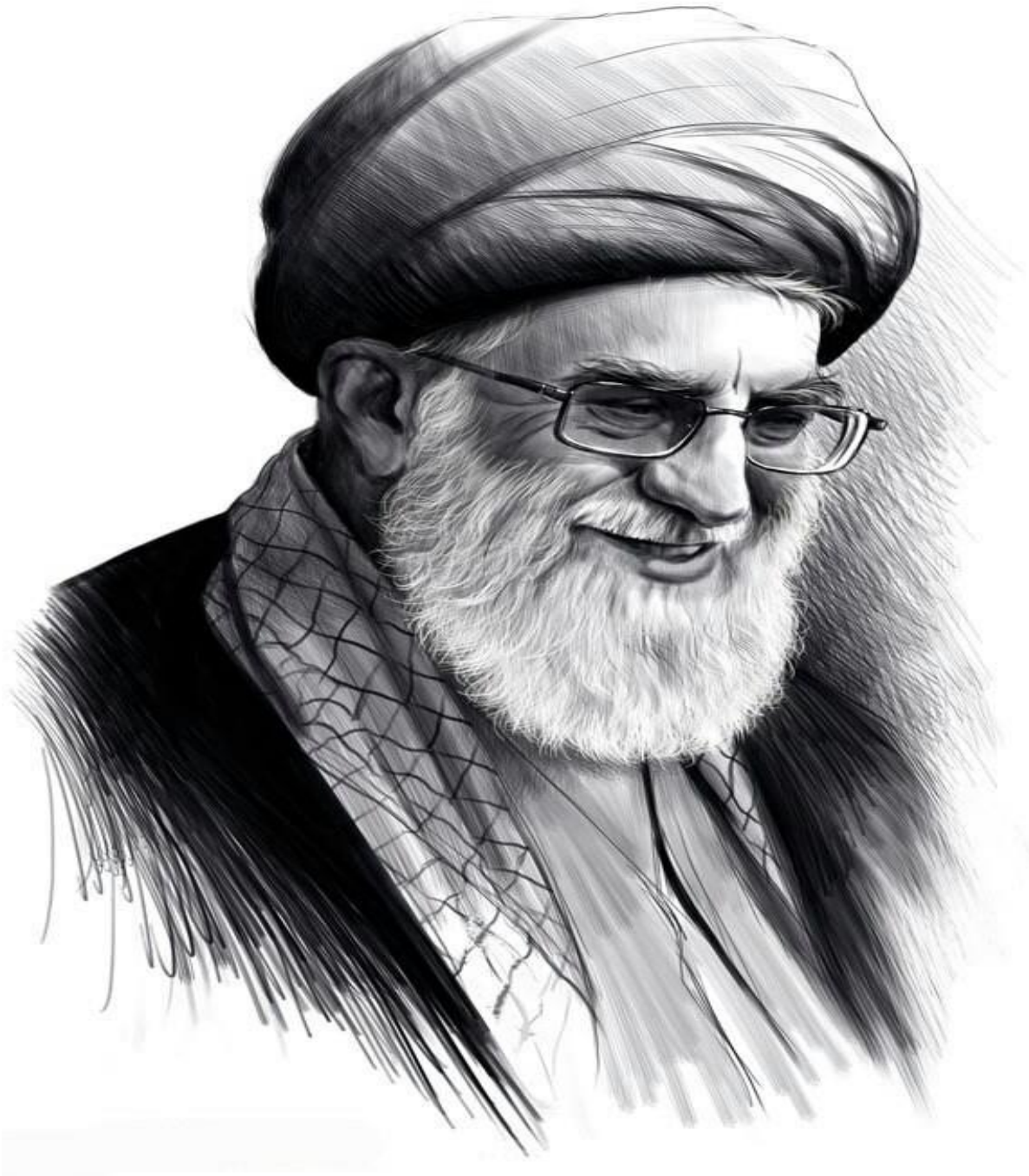
بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتا بیس در ایران

کتاب آموزش Java

نویسنده : مهندس افشین رفوآ



تقدیم به نائب امام عصر، حضرت آیت الله خامنه ای که عصا زدنش ضرب آهنگ حیدری دارد

آدرس آموزشگاه : تهران - خیابان شریعتی - بالا تر از خیابان ملک - جنب بانک صادرات - پلاک 651 طبقه دوم - واحد7
88146323 - 88446780 - 88146330

تقدیم به همه ی جویندگان علم که توان و امکان شرکت

در کلاس های ما را ندارند

فهرست

شروع کار با جاوا

یکی از مشکلات شروع کار با جاوا، نصب موارد موردنیاز شماست. حتی قبل از نوشتن یک خط از کد، دردسرها شروع می شوند. خوشبختانه بخش های بعدی این مورد را برای شما آسان تر کرده اند.

ما تصمیم داریم تمام کدهای خود را با استفاده از یک قطعه نرم افزار به نام NetBeans بنویسیم. این نرم افزار یکی از محبوب ترین IDEها Interface Development Environment (محیط گسترش اینترفیس) در جهان برای نوشتن برنامه های جاوا می باشد. شما به اختصار خواهید دید که این برنامه چگونه خواهد بود. اما قبل از که NetBeans کار کند، لازم است که مولفه ها و فایل های مورد نیاز جاوا را نصب کنید. اولین آن Java Virtual Machine(ماشین مجازی جاوا) نامیده می شود.

Java Virtual Machine

Javaیک سکوی مستقل می باشد و این به این معناست که روی هز سیستم عاملی اجرا می شود. بنابراین چه کامپیوتر شماWindows ، Linuxیا Mac OSرا اجرا می کند، همه ی اینها در جاو یکی است. دلیل اینکه می تواند روی هر سیستم عاملی اجرا شود به علت ماشین مجازی جاوا (Java Virtual Machine) می باشد . ماشین مجازی برنامه ای است که تمام کد شما را به درستی پردازش می کند .بنابراین لازم است که قبل از اجرای هر کد جاوا، این برنامه را (ماشین مجازی) نصب کنید.

جاوا متعلق به شرکتی به نام Sun Microsystems می باشد، بنابراین لازم است برای گرفتن ماشین مجازی جاوا، که به عنوان Java Runtime Environment (JRE) نیز شناخته می شود، وارد وب سایت Sun شوید. ابتدا این صفحه را امتحان کنید.

http://java.com/en/download/index.jsp

شما می توانید با کلیک کردن بر روی لینک "Do I have Java?"چک کنید که آیا JRE را سیستم خود دارید یا نه. این لینک را می توانید زیر دکمه ی Downloadدر بالای صفحه پیدا کنید (مگر اینکه Sun جای موارد را عوض کرده باشد.) وقتی که روی لینک کلیک می کنید، کامپیوتر شما برای JRE اسکن خواهد شد. سپس به

شما گفته می شود که این برنامه را دارید یا نه. اگر نداشته باشید، فرصت دانلود و نصب آن به شما داده می شود. یا اینکه می توانید لینک زیر را بررسی کنید.

http://java.com/en/download/manual.jsp

"manual" (دستی) در لینک های بالا به معنای دانلود دستی ("manual download") می باشد. صفحه به شما لینک های دانلود و دستور هایی برای سیستم عامل های مختلف ارائه می دهد.

پس از دانلود و نصب، لازم است کامپیوتر خود را ریست کنید. پس از انجام این کار شما برنامه ی Java Virtual Machine را خواهید داشت.

توسعه ی نرم افزار Kit در جاوا

در این مرحله هنوز نمی توانید برنامه ای بنویسید. تنها کاری که انجام داده اید، نصب نرم افزار است، بنابراین برنامه های جاوا روی کامپیوتر شما اجرا خواهند شد. برای نوشتن کد و امتحان کردن آن به نرم افزاری به نام Software Development kit احتیاج دارید.

این نرم افزار از سایت زیر قابل دانلود می باشد.

http://tahlildadeh.com

موردی که ما قصد استفاده از آن را داریم Java SE نامیده می شود (SE). مخفف Standard Edition می باشد، ویرایش استاندارد). روی لینک و سپس روی "Java SE (JDK) 6 Download" کلیک کنید. پس از آن وارد صفحه ای با یک لیست گیج کننده از گزینه ها برای دانلود می شوید. از آنجایی که قصد استفاده از NetBeans را داریم، این مورد را قرار دهید.

JDK 6 Update X with NetBeans 6.x

روی لینک Download کلیک کنید تا باز هم صفحه ی دیگری باز شود. روی top download کلیک کنید تا وارد صفحه ای شوید که در مورد سیستم عامل از شما سوال می پرسد. روی Continue کلیک کنید تا در نهایت دانلود مورد نظر را دریافت کنید. یک پیام هشدار، گرچه – این دانلود در هنگام نگارش بزرگ خواهد بود، بیشتر از 130 مگابایت! هنگامی که JDK و NetBeans را دانلود کردید، آن را روی سیستم خود نصب کنید.

برای نوشتن کد خود، قصد استفاده از NetBeans را داریم. به هر حال قبل از آغاز کار نرم افزار، در اینجا چگونگی کار کردن موارد در دنیای جاوا توضیح داده شده است.

چگونه برنامه ها در دنیای جاوا کار می کنند؟

شما در یک ویرایشگر متن کد حقیقی برای برنامه های خود می نویسید. (در NetBeansمحل خاصی برای نگارش کد شما وجود دارد.) این کد source code نامیده می شود و با file extension .java ذخیره می شود. سپس برنامه ای به نام Javac برای تبدیل source code به Java Byte Code استفاده می شود. این عمل compiling نامیده می شود. پس ازاینکه Javac فرایند کامپایل Java Byte Code را به پایان رسانده است، یک فایل جدید با extension .class ایجاد می کند. (حداقل اگر هیچ خطایی رخ ندهد این کار را انجام می دهد.) زمانی که فایل گروه ایجاد شده است، می تواند روی Java Virtual Machine اجرا شود . بنابراین

یک source code با extension .java ایجاد کنید.

از Java برای ایجاد یک فایل که در class. خاتمه می یابد، استفاده کنید.

گروه کامپایل شده را اجرا کنید.

NetBeansهمه ی فرایند ایجاد و کامپایل را برای شما کنترل می کند، گرچه در پشت صحنه source code های شما را گرفته و فایل جاوا ایجاد می کند. این برنامه Javac را آغاز کرده و فایل گروه را کامپایل می کند. سپس NetBeans می تواند برنامه ی شما را داخل نرم افزار خود اجرا کند. این امر از باز کردن یک پنجره ی ترمینال و تایپ کردن رشته های فرمان طولانی نجات می دهد.

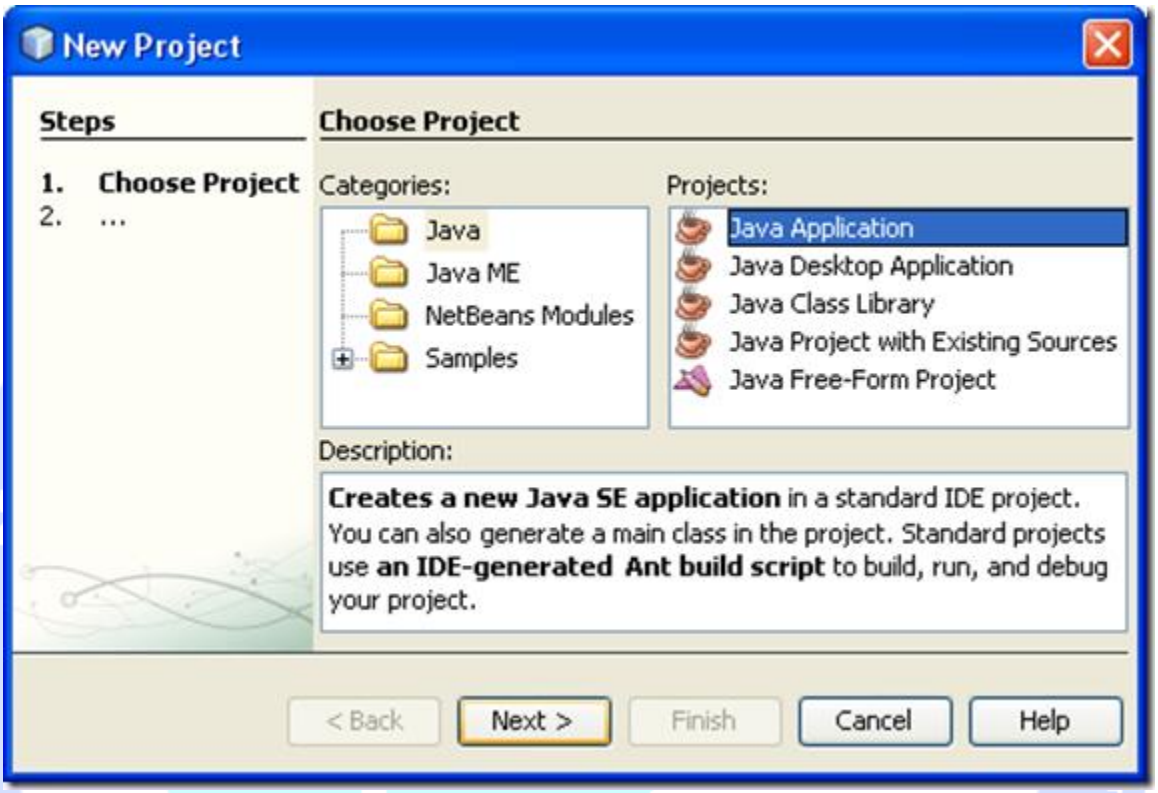
آموزش نرم افزار NetBeans

وقتی که برای اولین بار NetBeans را اجرا می کنید، صفحه ای مشابه لینک زیر خواهید دید.

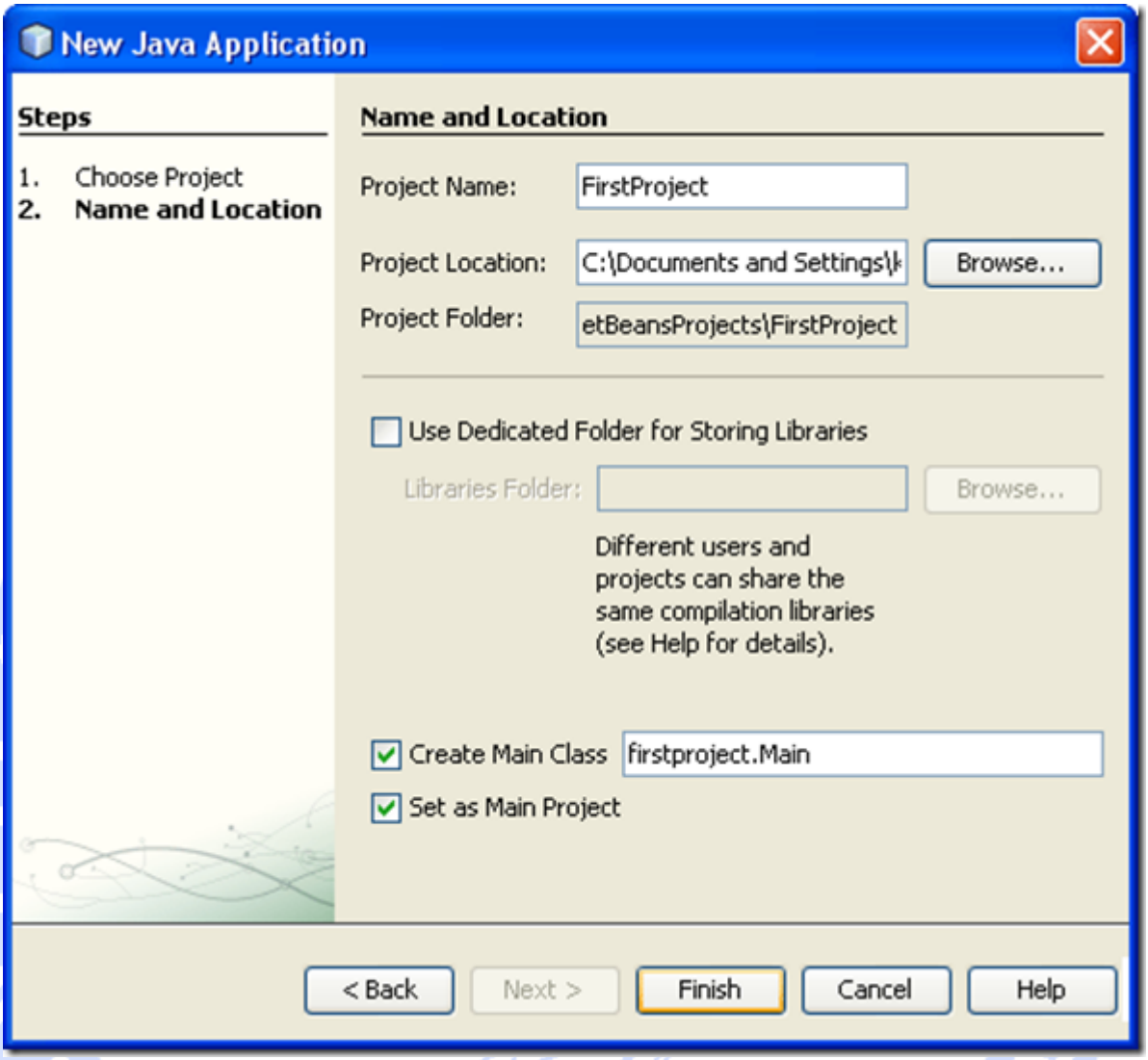
The NetBeans Software - First Start (38K)

از آنجایی که این برنامه خیلی سریع نیست، ممکن است مجبور شوید مدتی منتظر بمانید.

برای شروع یک پروژه ی جدید روی File > New Project از منوی NetBeans در بالا کلیک کنید. پس از آن صفحه ی زیر را مشاهده خواهید کرد.



ما قصد ایجاد یک Java Application را داریم، بنابراین گزینه ی Java را در زیر Categories و سپس Java Application در زیر Projects را انتخاب کنید.



در قسمت Project Name در قسمت بالا، یک نام برای پروژه ی خود تایپ کنید .به چگونگی تغییر متن در پایین دقت داشته باشید که برای هماهنگی با نام پروژه اتفاق می افتد) در تکست باکس در سمت راست Create Main Class)

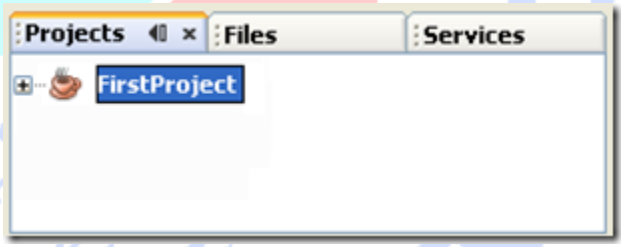
firstproject.Main

اگر آن را به این شکل رها کنیم، Class دارای نام Main خواهد بود. آن را به FirstProject تغییر دهید.

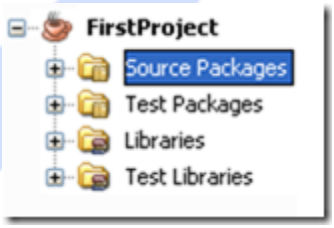
اکنون گروه ایجاد شده FirstProject نامیده می شود، که با F و P بزرگ نوشته می شود. پوشه نیز firstprojectنامیده می شود، اما با f و p کوچک.

موقعیت پیش فرض برای ذخیره ی پروژه ی شما درتکست باکس Project Location ظاهر می شود. اگر تمایل داشتید، می توانید آن را تغییر دهید NetBeans . نیز یک فولدر در همان موقعیت با نام پروژه ی شما ایجاد خواهد کرد. روی دکمه ی Finishکلیک کنید و NetBeans نیز به کار خود در ایجاد فایل های لازم برای شما ادامه خواهد داد.

وقتی که NetBeans به IDE بازمی گردد، نگاهی به بخش Projects در بالا و سمت چپ صفحه داشته باشید. (اگر نمی توانید آن را ببینید، روی Window > Projects از نوار منو در قسمت بالای نرم افزار کلیک کنید(



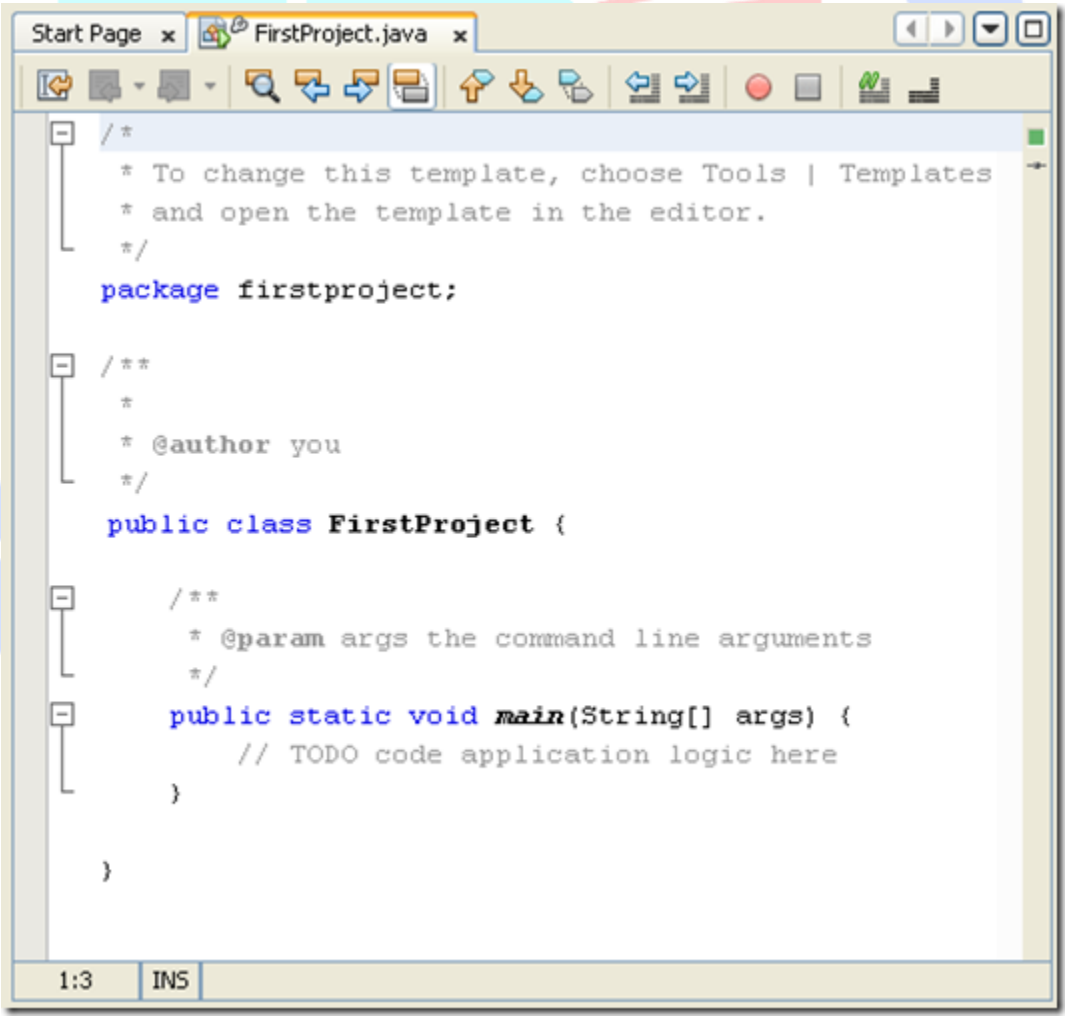
روی علامت به علاوه کلیک کنید تا پروژه ی خود را باز کنید، پس از آن موارد زیر را مشاهده خواهید کرد.



اکنون Source Packages را باز کنید تا مجددا نام پروژه ی خود را ببینید .آن را باز کنید، شما فایل Java را مشاهده خواهید کرد که source code شما می باشد.

همین source code باید در سمت راست، در تکست باکس بزرگ نمایش داده شود، که FirstProject.java نامیده می شود. اگر شما نمی توانید یک پنجره ی کد مشاهده کنید، روی FirstProject.java در بالای پنجره ی Projects خود دابل کلیک کنید.

پنجره ی برنامه نویسی (coding window) که ظاهر می شود، باید مشابه تصویر زیر باشد. (ما نام نویسنده را تغییر داده ایم)



موردی که باید به آن اشاره کرد این است که در اینجا گروه FirstProject نامیده می شود.

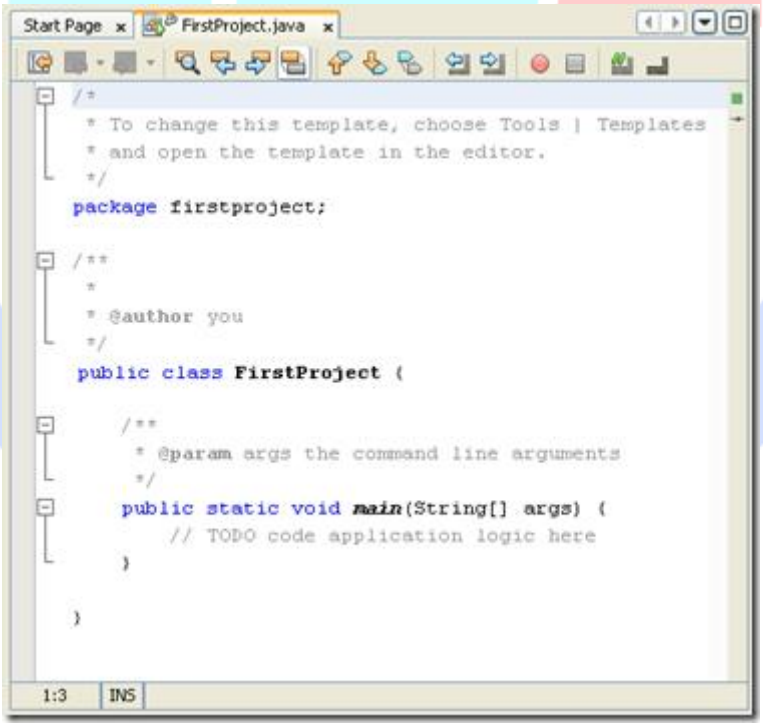
public class FirstProject {

این همان نام فایل java source در پنجره ی Project می باشد .FirstProject.java :وقتی که برنامه های خود را اجرا می کنید، کامپایلر انتظار دارد که فایل منبع و نام گروه با هم هماهنگ باشند. بنابراین اگر فایل java.با عنوان firstProject نامگذاری می شود، اما گروه FirstProject نامیده می شود، شما یک خطا در کامپایل دریافت خواهید کرد. این به خاطر این است که اولین f با حرف کوچک نوشته شده و دومین با حرف بزرگ است.

دقت داشته باشید که گرچه پوشه را نیز firsproject نامیده ایم، اما این لازم نیست. می توانستیم پوشه را someprogrammeنامگذاری کنیم. بنابراین نام پوشه نباید با نام فایل منبع جاوا یا گروه در فایل منبع یکی باشد. بلکه فقط نام فایل منبع جاوا و نام گروه است که باید با هم هماهنگ باشند.

آموزش دستورات جاوا

وقتی که در NetBeans یک پروژه ی جدید ایجاد می کنید، متوجه خواهید شد که برخی قسمت های متن خاکستری هستند با تعداد زیادی اسلش و ستاره.



بخش های خاکستری شده کامنت ها هستند. وقتی که برنامه اجرا می شود، کامنت ها نادیده گرفته می شوند. بنابراین شما می توانید هرچه تمایل دارید در داخل کامنت عای خود تایپ کنید. اما معمولا به این شکل است که کامنت هایی را ارائه می دهید که سعی دارد آنچه می خواهید انجام دهید را توضیح دهد. می توانید با تایپ کردن دو اسلش که با کامنت شما دنبال می شود، یک خط کامنت مجزا داشته باشید.

```
//This is a single line comment
```

اگر می خواهید بیشتر از یک خط داشته باشید نیز می توانید این کار را انجام دهید.

```
//This is a comment spreading
//over two lines or more
```

یا این کار را انجام دهید.

```
/* This is a comment spreading over two lines or more */
```

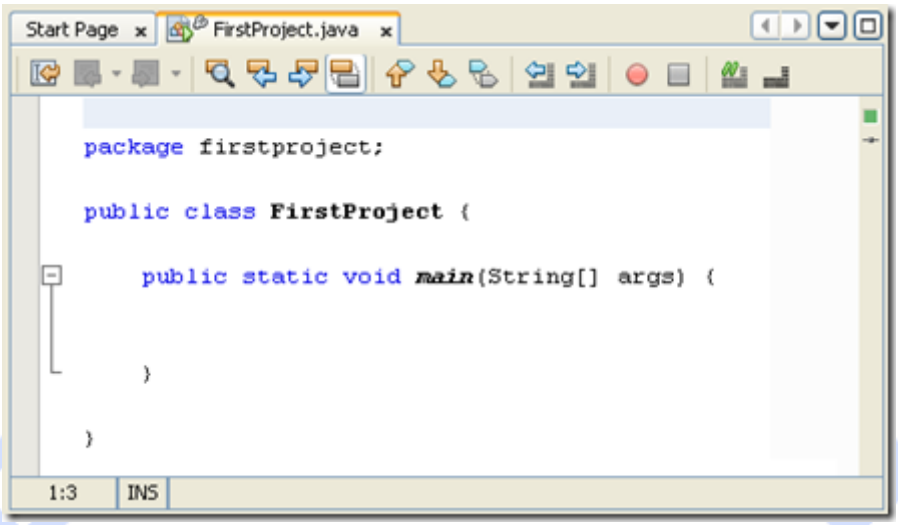
در کامنت بالا دقت داشته باشید که چگونه با /* شروع می شود. در عوض برای پایان کامنت نیز */ را داریم.

چیزی به نام کامنت Javadoc وجود دارد. می توانید دو نمونه از آن را در تصویر برنامه نویسی در صفحه ی قبل مشاهده کنید. یک کامنت Javadoc با یک اسلش دو به به جلو و دو ستاره (/**) شروع می شود و با یک ستاره و یک اسلش (*/) نیز به پایان می رسد. هر خط از کامنت با یک ستاره شروع می شود.

```
/** *This is a Javadoc comment */
```

کامنت های Javadoc برای کد داکیومنت (document code) استفاده می شوند .بنابراین کد داکیومنت شده می تواند به یک صفحه ی HTML تبدیل شود که برای دیگران مفید خواهد بود. با کلیک کردن بر روی Runاز منو در بالایNetBeans ، مشاهده می کنید که چگونه به نظر می رسند. از منوی Run عبارت Generate Javadocرا انتخاب کنید. به هرحال از آنجایی که هنوز کدی ننوشته اید، مورد زیادی برای مشاهده وجود ندارد.

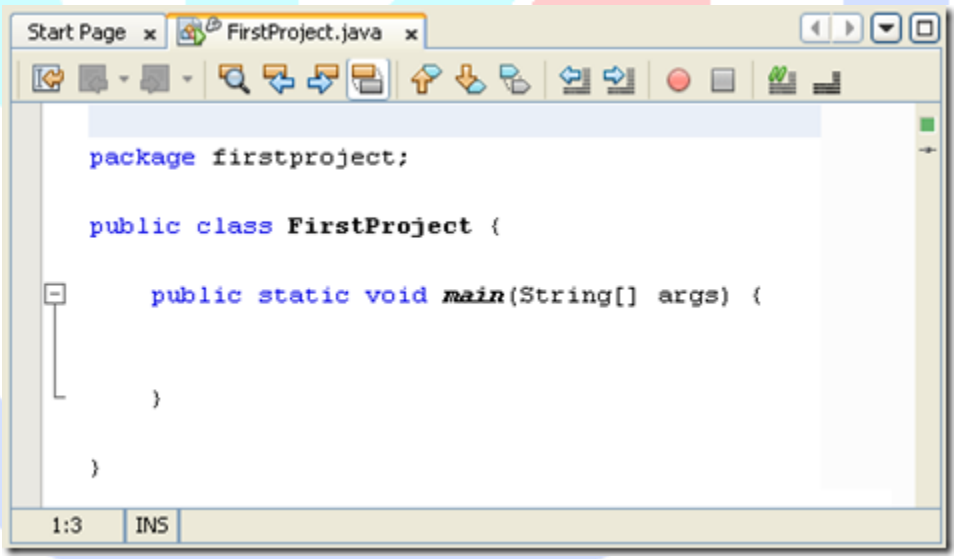
در این مرحله از کار برنامه نویسی خود می توانید کامنت هایی را که NetBeansبرای شما تولید می کند، حذف کنید. در اینجا دوباره برنامه ی ما را با کامنت های حذف شده مشاهده می کنید.



آموزش ساختار کدهای جاوا

ساختار کدهای جاوا

در بخش قبل کد خود را تا حدی مرتب کردید، در اینجا پنجره ی برنامه نویسی شما باید این گونه به نظر برسد.



مشاهده می کنید که ابتدا پوشه را نامگذاری کرده ایم. دقت داشته باشید که چگونه یک خط با نقطه ویرگول به پایان می رسد. اگر این نقطه ویرگول را نگذارید، برنامه کامپایل نخواهد شد.

```
package firstproject;
```

نام گروه نیز پس از آن قرار می گیرد.

```
public class FirstProject { }
```

می توانید یک گروه را به عنوان یک بخش کد (code segment) در نظر گیرید .اما باید به جاوا بگویید که بخش های کد کجا شروع و کجا به پایان برسند .این کار را می توانید با آکولاد انجام دهید. شروع یک بخش کد با آکولاد سمت چپ { و پایان آن با آکولاد سمت راست } انجام می شود. هرچیزی در داخل این آکولادها به آن بخش کد مربوط می شود.

آنچه در داخل این آکولادهاست، یک code segment دیگر است. این مورد

```
public static void main( String[ ] args ) { }
```

لغت main در اینجا قسمت مهمی است. هروقت یک برنامه ی جاوا آغاز می شود، به دنبال متودی به نام main می باشد. (یک متود در واقع بخشی از یک کد می باشد، در مورد اینها بعدها بیشتر فرا خواهید گرفت). سپس هر کدی را در داخل این آکولادها برای main اجرا می کند. اگر متود main را در برنامه های جاوای خود نداشته باشید، پیغام های خطا دریافت خواهید کرد. اما همانطور که از نام آن پیداست، نقطه ی ورودی اصلی برای برنامه های شما می باشد.

در حال حاضر قسمت های آبی قبل از لغت main می توانند نادیده گرفته شوند.

(به هرحال اگر علاقمند هستید، عمومی (public) به این معناست که متود در خارج از این گروه نیز قابل مشاهده است، و استاتیک نیز یعنی لازم نیست که یک آبجکت جدید ایجاد کنید و void یعنی اینکه مقداری گزارش نمی شود وفقط با همین مقدار ادامه می دهد. بخش های بین آکولادهای main با عنوان خط فرمان argumentمطرح می شوند)

نکته ی مهمی که باید به خاطر داشته باشید این است که ما گروهی با نام FirstProject داریم. این گروه حاوی یک متود به نام main می شود. هر دو دارای مجموعه آکولادهای مربوط به خود می باشند. اما بخش مهم کد به گروه FirstProject متعلق می باشد.

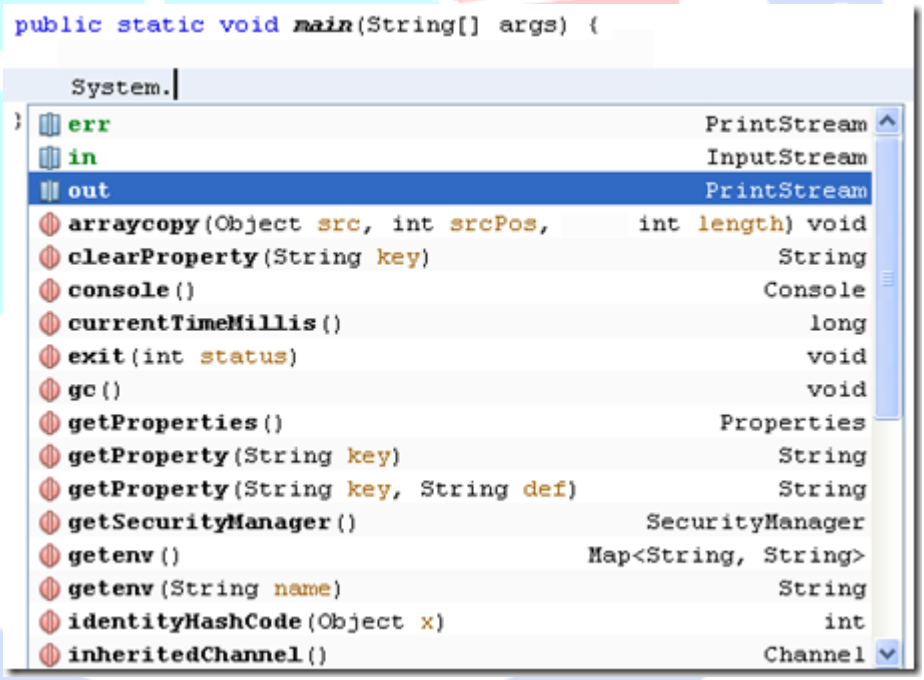
آموزش چاپ خروجی در جاوا

شما می توانید the code you have so far را اجرا کرده و آن را به یک برنامه تبدیل کنید. این برنامه کار خاصی انجام نمی دهد اما هنوز کامپایل خواهد شد. بنابراین اجازه بدهید یک خط از کد را اضافه کنیم، سپس می توانیم مشاهده کنیم که چگونه کار می کند.

متونی را به پنجره ی console خروجی می دهیم. خط زیر را به متد main خود اضافه کنید.

```
public static void main( String[ ] args ) {
        System.out.println( "My First Project" );
        }
```

وقتی که پس از System نقطه را تایپ می کنید، NetBeansنیز با نمایش لیستی از گزینه های موجود سعی می کند تا به شما کمک کند.



روی آن دابل کلیک کنید تا به کد خود بیفزایید. دوباره لیست گزینه ها ظاهر می شود.

println( )را انتخاب کنید. آنچه این گزینه انجام می دهد چاپ یک خط از متن زوی صفحه ی خروجی می باشد. اما نیاز خواهید داشت که متن خود را بین آکولادهای println قرار دهید. متن شما لازم است بین یک جفت علامت نقل قول قرار بگیرد.

```
public static void main(String[] args) {

    System.out.println("");
}
```

زمانی که نقل قول های خود را در محل قرار داده اید، متن خود را تایپ کنید.

```
public static void main(String[] args) {

    System.out.println("My First Project");
}
```

دقت داشته باشید که خط با یک نقطه ویرگول به پایان می رسد. هر خط کاملی از کد در Java نیاز به نقطه ویرگول در انتهای خود دارد. اگر این علامت را در انتهای آن قرار ندهید، برنامه کامپایل نخواهد شد.

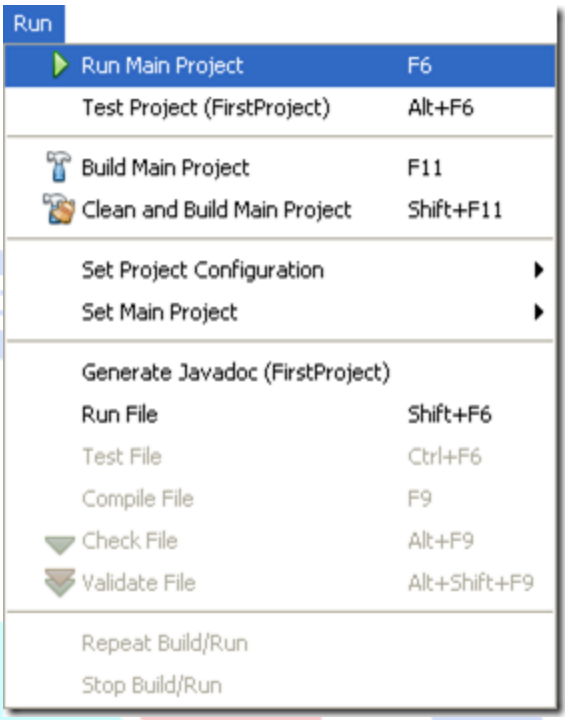
بسیارخوب، اکنون می توانیم جلوتر برویم و این برنامه را امتحان کنیم .ابتدا کار خود را ذخیره کنید. می توانید روی File > Save یا File > Save All کلیک کنید. یا روی آیکن Save در نوار ابزار NetBeans کلیک کنید.

آموزش اجرای برنامه های جاوا

اجرای برنامه های جاوا

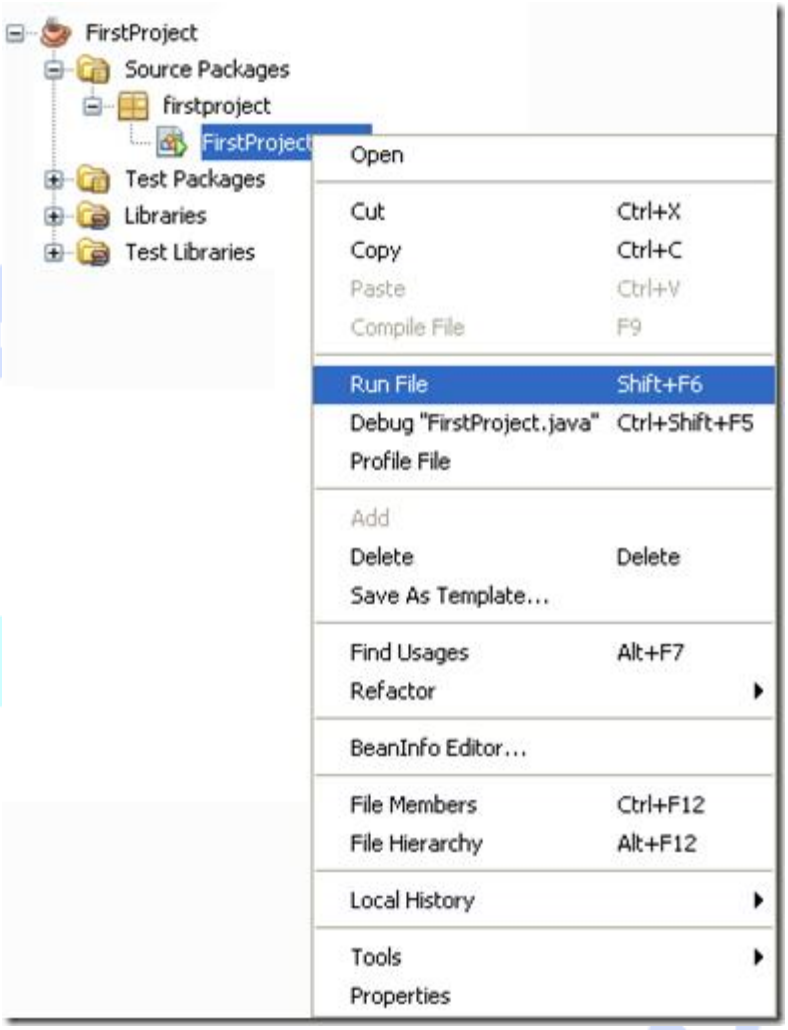
وقتی که برنامه ای را در NetBeans اجرا می کنید، این برنامه در پنجره ی Outputدر پایین صفحه ی شما، درست زیر کد، اجرا خواهد شد. به این دلیل است که لازم نیست یک پنجره ی ترمینال یا console را آغاز کنید – پنجره ی Output همان console می باشد.

روش های مختلفی برای اجرای برنامه در NetBeans وجود دارد. ساده ترین راه فشاردادن دکمه ی F6 روی صفحه کلید می باشد. همچنین می توانید با استفاده از منوها در بالای NetBeans برنامه ها را اجرا کنید. منوی Run را قرار داده و سپس Run Main Programme را انتخاب کنید.

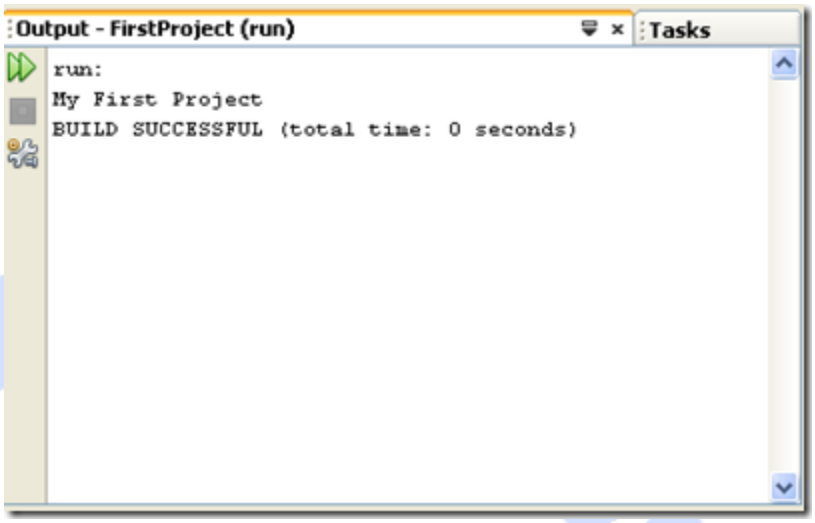


همچنین می توانید روی فلش سبز رنگ در نوار ابزار NetBeans کلیک کنید.

راه دیگر برای اجرای برنامه هایتان پنجره ی Projects می باشد. این روش اطمینان خواهد داد که source code درست قرار است که اجرا شود. به سادگی روی فایل منبع جاوای خود در پنجره ی projects کلیک راست کنید که منویی برای شما ظاهر خواهد شد. در این منو Run File را انتخاب کنید.



با استفاده از یکی از متودهای بالا برنامه ی خود را اجرا کنید. متوجه خواهید شد که در پنجره ی Output اتفاقاتی در حال افتادن است.

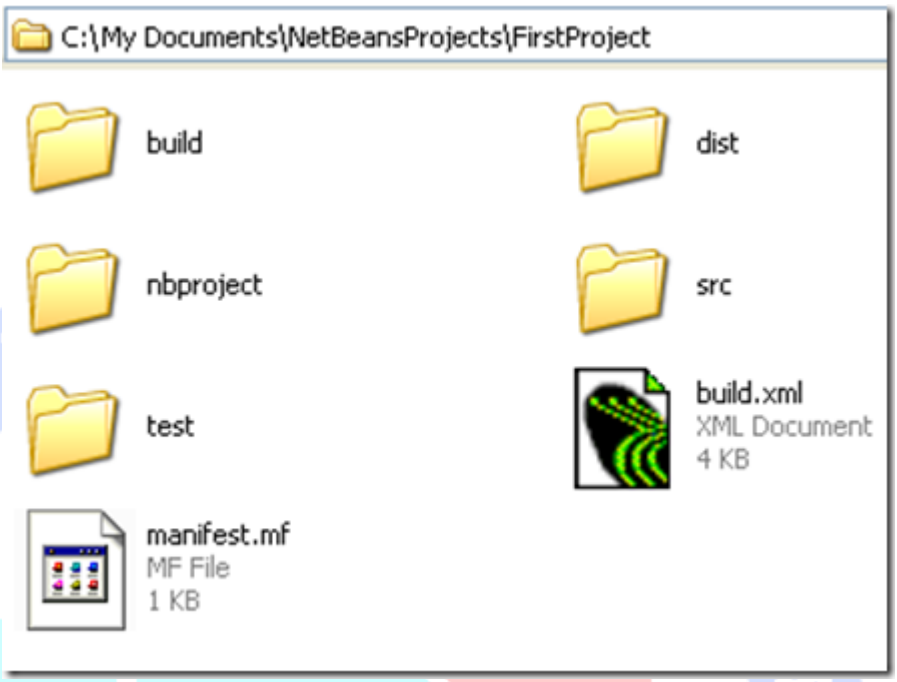


دومین خط در پنجره ی Output در بالا کد مورد نظر ما می باشد .My First Project :با کلیک کردن روی فلش های سبز رنگ در بالا و سمت چپ پنجره ی Output، می توانید آن را مجددا و با سرعت اجرا کنید.

آموزش به اشتراک گذاری کدهای جاوا

می توانید برنامه های خود را به دیگر افراد ارسال کنید، طوریکه بتوانند آنها را اجرا کنند. برای انجام آن نیاز به ایجاد یک فایل JAR دارید NetBeans .(Java Archive) می تواند همه ی این کار را برای شما انجام دهد. از منوی Run در بالا Clean and Build Main Project را انتخاب کنید.

وقتی کاری انجام می دهید، NetBeansآن را ذخیره می کند و سپس تمام فایل های لازم را ایجاد می کند. این برنامه فولدری به نام dist ایجاد کرده و تمام فایل ها را در آن قرار می دهد. اگر نگاهی به محل پروژه های NetBeansداشته باشید، فولدر dist را مشاهده خواهید کرد.



روی فولدر dist دابل کلیک کنید تا محتویات داخل آن را مشاهده کنید.

باید یک فایل JAR و یک تکست فایل README مشاهده کنید. تکست فایل حاوی دستورالعمل هایی در مورد چگونگی اجرای برنامه از یک پنجره ی terminal/console می شود.

اکنون که فراگرفته اید چگونه فایل های منبع جاوا را اجرا کنید، اجازه بدهید کمی برنامه نویسی کنیم.

آموزش متغیرهای جاوا

برنامه ها با دستکاری داده ی واقع در حافظه کار می کنند. داده می تواند عدد، متن، آبجکت، اشاره گرهایی به دیگر بخش های حافظه، و موارد دیگر باشد . داده نام گذاری می شود، طوریکه در صورت نیاز می تواند مجددا

فراخوانده شود . نام و مقدار آن با عنوان متغیر (Variable) شناخته می شوند. ما با مقادیر عددی شروع می کنیم.

برای ذخیره ی یک عدد در جاوا، باید گزینه های زیادی داشته باشید. تمام اعداد مانند 8، 10، 12 و غیره با استفاده از متغیر int ذخیره می شوند . (Intمخفف integer (عددصحیح) می باشد.). اعداد ممیز شناور مانند 8.4، 10.5، 12.8و غیره با استفاده از متغیر دوگانه ذخیره می شوند. شما ذخیره سازی را با یک علامت تساوی (=) انجام می دهید. اجازه بدهید چند مثال را بررسی کنیم (می توانید از کد FirstProject برای این مثال ها استفاده کنید.)

برای تنظیم یک عدد کامل (عدد صحیح)، موارد زیر را به متود main از بخش قبل اضافه کنید:

```
public static void main(String[ ] args) {
int first_number;
System.out.println("My First Project");
}
```

بنابراین برای اینکه به جاوا بگویید که می خواهید یک عدد کامل را ذخیره کنید، ابتدا لغت int را تایپ کنید که پس از آن یک فاصله قرار می گیرد. سپس لازم است برای متغیر عدد صحیح خود یک نام انتخاب کنید. شما می توانید هر نامی که می خواهید برای آنها انتخاب کنید، اما در این مورد قوانینی نیز وجود دارد.

نام متغیرها نباید با عدد شروع شود. بنابراین نام first_number درست می باشد اما نام 1 st_number درست نیست. می توانید اعداد را در هرجایی از نام متغیر به جز ابتدای آن قراردهید.

نام های متغیرها نمی تواند مشابه لغات کلیدی Java باشد. موارد بسیاری از این گونه وجود دارد که در NetBeansبه رنگ آبی در خواهند آمد، مانند int در بالا.

نمی توانید در نام های متغیرها فاصله داشته باشید. اطلاعیه ی متغیر int first number، خطایی را دریافت خواهد کرد. ما از متغیرهای تاکید شده استفاده کرده ایم، اما متداول این است که حرف اول با حروف کوچک انگلیسی و حرف بعدی یا بقیه ی حروف را با حرف بزرگ انگلیسی نوشته شوند:

firstNumber, myFirstNumber

نام های متغیرها موارد هوشمندی هستند، بنابراین firstNumber وFirstNumber نام های متفاوتی برای متغیرها می باشند.

برای ذخیره ی موردی در متغیری به نام first_number ، یک علامت تساوی و سپس مقداری مورد نظر برای ذخیره را تایپ کنید:

```
public static void main(String[ ] args) {
int first_number;
first_number = 10;
System.out.println("My First Project");
}
```

بنابراین این برنامه به جاوا می گوید که می خواهید مقداری از 10 را در متغیر عدد صحیح که first_number نامیده ایم، ذخیره کنید.

اگر تمایل دارید، می توانید تمام این کار را در یک خط انجام دهید:

```
public static void main(String[ ] args) {
int first_number = 10;
System.out.println("My First Project");
}
```

برای اینکه تمام موارد را در عمل ببینید، متود println را به این شکل تغییر دهید:

```
System.out.println( "First number = " + first_number );
```

آنچه ما اکنون در بین آکولادهای println داریم، متن های مستقیمی هستند که در علامت نقل قول محدود شده اند:

```
("First number = "
```

ما یک علامت به علاوه نیز داریم که با نام متغیرمان دنبال می شود:

```
+ first_number );
```

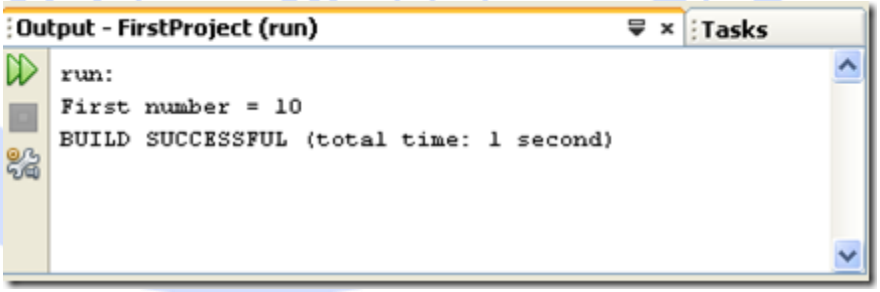
علامت به اضافه به معنای اتصال به یکدیگر "join together" می باشد . بنابراین ما متن و نام متغیر خود را به یکدیگر متصل می کنیم. این عملکرد به عنوان concatenation الحاق (مطرح است.

پنجره ی شما اکنون باید مانند زیر به نظر برسد (دقت داشته باشید که هر خط با نقطه ویرگول به پایان می رسد.):

```
package firstproject;

public class FirstProject {

  public static void main(String[] args) {

      int first_number;
      first_number = 10;

      System.out.println("First number = " + first_number );

    }

}
```

برنامه ی خودرا اجرا کنید و باید صفحه ی زیر را در پنجره ی Output در پایین مشاهده کنید:

بنابراین عددی را که در متغیر ذخیره کرده ایم، به نام first_number، خروجی می باشد پس از علامت تساوی.



اجازه بدهید چند جمع ساده را امتحان کنیم. چند متغیر intبه کد خود اضافه کنید، یکی برای ذخیره ی رومین عدد و دیگری برای ذخیره ی پاسخ.

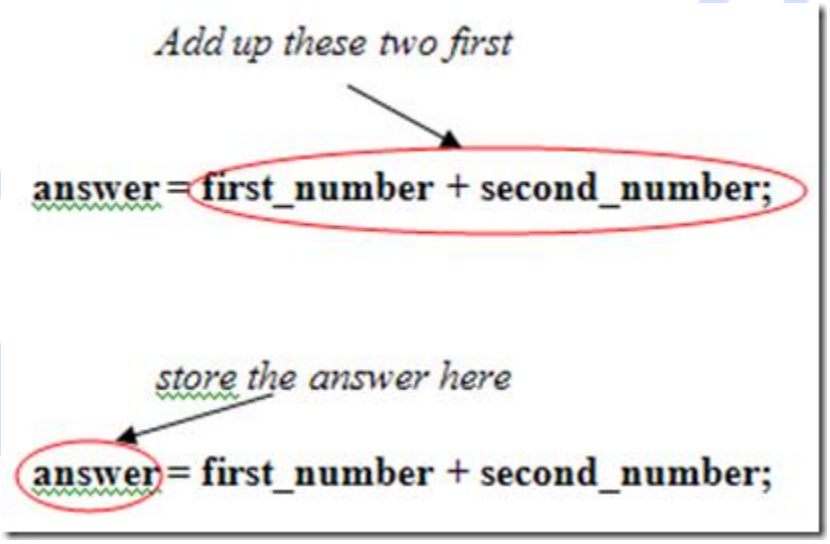
```
int first_number, second_number, answer;
```

دقت کنید که چگونه سه متغیر نام روی یک خط داریم. اگر متغیرها از یک نوع باشند، شما می توانید این کار را در جاوا انجام دهید (برای مورد ما نوع int). بنابراین نام هر متغیر با کاما (ویرگول) جدا می شود.

بنابراین می توانیم در متغیرهای جدید چیزی اضافه کنیم:

```
first_number = 10;
second_number = 20;
answer = first_number + second_number;
```

برای متغیر پاسخ، می خواهیم اولین عدد را به دومین عدد اضافه کنیم. جمع کردن با علامت به اضافه (+) انجام می شود. بنابراین Java مقادیر را در first_number و second_number به یکدیگر اضافه می کند . وقتی این کار تمام شد، مجموع را در متغیر واقع در سمت راست تساوی ذخیره می کند. بنابراین به جای اختصاص دادن 10 یا 20 به نام متغیر، عمل جمع را انجام خواهد داد و سپس اختصاص می دهد. در موردی که واضح نیست، اتفاقی مانند زیر رخ خواهد داد:



مورد بالا سازگار با این مورد می باشد:

```
answer = 10 + 20;
```

اما جاوا تقریبا می داند که در داخل دو متغیر first_number و second_numberچه چیزی وجود دارد، بنابراین شما می توانید فقط از نام ها استفاده کنید.

اکنون متود println خود را مانند زیر تغییر دهید:

```
System.out.println("Addition Total = " + answer );
```

مجددا یک متن مستقیم را که با علامت نقل قول احاطه شده، با نام متغیر ترکیب می کنیم. پنجره ی برنامه نویسی شما باید مانند زیر باشد:

```
public static void main(String[] args) {

    int first_number, second_number, answer;

    first_number = 10;
    second_number = 20;
    answer = first_number + second_number;

    System.out.println("Addition Total = " + answer );
}
```

```
Output - FirstProject (run)        Tasks
run:
Addition Total = 30
BUILD SUCCESSFUL (total time: 1 second)
```

بنابراین برنامه ی ما موارد زیر را انجام داده است:

یک عدد ذخیره کرده است .یک عدد دوم ذخیره کرده است .این دو عدد را با یکدیگر جمع کرده است .نتیجه ی جمع را در یک متغیر سوم ذخیره کرده است .نتیجه را چاپ کرده است .شما مستقیما نیز می توانید از اعداد

استفاده کنید .خط پاسخ را مانند زیر تغییر دهید ;answer = first_number + second_number + 12 :

برنامه ی خود را مجددا اجرا کنید. چه چیزی چاپ شده است؟ همان موردی است که انتظار داشتید؟ می توانید اعداد کاملا بزرگی در متغیر نوع intذخیره کنید. حداکثر مقدار 2147483647می باشد. اگر یک عدد منفی مورد نظر شماست، کمترین مقداری که می توانید داشته باشید 2147483648-می باشد .اگر اعداد بزرگتر و یا کوچکتر می خواهید، می توانید از متغیر عددی دیگری استفاده کنید double :، که آنها را در بخش بعدی مورد بررسی قرار می دهیم.

آموزش متغیر Double

متغیر double می تواند اعداد خیلی بزرگ (یا کوچک) را در خود داشته باشد .حداکثر و حداقل مقادیر 17 می باشد که با 307 صفر دنبال می شود.

متغیر double برای نگهداری مقادیر ممیزی شناور نیز استفاده می شود. یک مقدار ممیزی شناور عددی مانند 8.7، 12.5، 10.1 و غیره می باشد. به عبارت دیگر این عدد ممیزی در انتها دارد. اگر سعی کنید یک مقدار ممیزی را در یک متغیر int ذخیره کنید، NetBeansزیر کد پیش فرض را خط خواهد کشید. اگر سعی کنید برنامه را اجرا کنید، کامپایلر یک پیغام خطا ارائه می دهد. اجازه بدهید با متغیرهای double کمی تمرین کنیم.

از بخش int را به double تغییر دهید. بنابراین این مورد را نیز تغییر دهید:

```
int first_number, second_number, answer;
```

به خط زیر:

```
double first_number, second_number, answer;
```

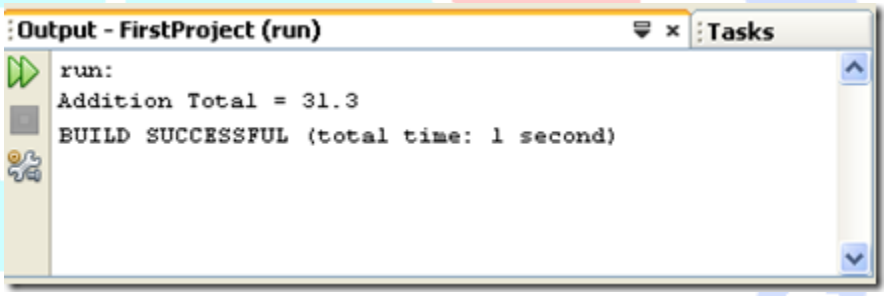
اکنون مقادیر ذخیره شده را تغییر دهید:

```
first_number = 10.5;
second_number = 20.8;
```

بقیه ی برنامه را همانطور که هست رها کنید. پنجره ی برنامه نویسی شما باید مشابه زیر باشد:

```
public static void main(String[] args) {

    double first_number, second_number, answer;

    first_number = 10.5;
    second_number = 20.8;
    answer = first_number + second_number;

    System.out.println("Addition Total = " + answer );
}
```

برنامه ی خود را مجددا اجرا کنید، پنجره ی خروجی باید مانند زیر باشد:



تغییر مقادیر ذخیره شده در first_number و second_number را امتحان کنید. از هر مقداری که تمایل دارید، استفاده کنید. برنامه ی خود را اجرا کرده و نتیجه را مشاهده کنید.

در بخش بعدی در مورد دو نوع متغیر دیگر جاوا فرا خواهید گرفت short :و.float

آموزش متغیرهای Short و Float در جاوا

دو نوع متغیر دیگری که می توانید استفاده کنید short و floatمی باشند. متغیر short برای ذخیره ی اعداد کوچکتر استفاده می شود و دامنه ی آن بین منفی 32768 و مثبت 32767 می باشد. به جای استفاده از int در کد خود مانند صفحات قبل، می توانستیم از متغیر short استفاده کنیم. اگر مطمئن هستید که متغیری را که مخواهید ذخیره کنید، بیشتر از 32767 و کمتر از 32768-نیست، باید فقط از متغیر short استفاده کنید.

متغیر double را که استفاده کردیم، می تواند اعداد ممیزی شناور واقعا بزرگ را ذخیره کند. به جای استفاهد ازdouble ، متغیر float نیز می تواند استفاده شود. هنگام ذخیره ی یک مقدار در یک متغیرfloat ، نیاز به حرف f در انتهای آن دارید. مانند مورد زیر:

```
float first_number, second_number, answer;
first_number = 10.5f;
second_number = 20.8f;
```

بنابراین حرف f بعد از عدد اما قبل از نقطه ویرگول در انتهای خط قرار می گیرد. برای مشاهده ی تفاوت بین floatو double مورد زیر را بررسی کنید.

حساب ساده (Simple Arithmetic) با متغیرهایی که استفاده کرده اید، می توانید از نمادهای زیر نیز برای انجام محاسبت استفاده کنید: + (علامت جمع) _ (علامت منفی) *(علامت ضرب که یک ستاره است.) / (علامت تقسیم که یک اسلش رو به جلو می باشد.) این تمرین را امتحان کنید: علامت جمع را که برای اضافه کردن first_number وsecond_number استفاده می شود، حذف کنید. آن را با نمادهای بالا جایگزین کنید، ابتدا علامت منفی، سپس علامت ضرب و در نهایت علامت تقسیم. پاسخ به مورد آخر، تقسیم، باید یک عدد واقعا بزرگ به شما ارائه دهد. عددی که باید برای تقسیم ارائه دهید 0.5048076923076923 می باشد. این به این دلیل است که شما از متغیر نوع double استفاده کرده اید .به هرحال double را به float تغییر دهید. سپس حرف f را به انتهای اعداد اضافه کنید. بنابراین کد شما باید مانند زیر باشد:

```
public static void main(String[] args) {

    float first_number, second_number, answer;

    first_number = 10.5f;
    second_number = 20.8f;
    answer = first_number / second_number;

    System.out.println("Total = " + answer );
}
```

وقتی که کد بالا را اجرا می کنید، اکنون پاسخ 0.5048077 می باشد. جاوا 6 عدد اول بعد از ممیز را گرفته و سپس بقیه را گرد کرده است double) .یک عدد 64 بیتی است و float فقط 32 بیتی می باشد(.

در بخش بعد در مورد اهمیت Operator Precedence فرا خواهید گرفت.

آموزش اولویت عملگرها در جاوا

البته که شما می توانید با بیشتر از دو عدد در جاوا محاسبه کنید. اما باید مراقب آنچه قرار است محاسبه شود، باشید. مورد زیر را به عنوان مثال در نظر بگیرید:

```
first_number = 100;

second_number = 75;

third_number = 25;

answer = first_number - second_number + third_number;
```

اگر محاسبه را از سمت چپ به راست انجام داده باشید، 75ـ100 می شود که پاسخ 25 است. سپس عدد سوم را که 25 است اضافه کنید. کجکوع 50 خواهد بود. به هرحال اگر مد نظر شما این نباشد چطور؟ اگر تمایل داشته باشید اعداد دوم و سوم را با هم اضافه کنید و سپس مجموع را از اولین عدد کسر کنید، چطور؟ بنابراین 25+75 است که پاسخ 100 می باشد. سپس آن را از اولین عدد کسر کنید که 100 می باشد. اکنون مجموع 0 خواهد بود.

برای اطمینان از اینکه جاوا کاری را انجام می دهد که شما می خواهید، می توانید از آکولا استفاده کنید. بنابراین اولین محاسبه مانند زیر خواهد بود:

```
answer = (first_number - second_number) + third_number;
```

این پنجره ی برنامه نویسی می باشد، بنابراین می توانید آن را امتحان کنید:

```
public static void main(String[] args) {

    int first_number, second_number, third_number, answer;

    first_number = 100;
    second_number = 75;
    third_number = 25;
    answer = (first_number - second_number) + third_number;

    System.out.println("Total = " + answer );
}
```

محاسبه دوم نیز به این شکل می باشد:

```
answer = first_number - (second_number + third_number);
```

پنجره ی کد آن را نیز در اینجا مشاهده می کنید:

```
public static void main(String[] args) {

    int first_number, second_number, third_number, answer;

    first_number = 100;
    second_number = 75;
    third_number = 25;
    answer = first_number - (second_number + third_number);

    System.out.println("Total = " + answer );
}
```

اکنون اجازه بدهید چند عمل ضرب و جمع را امتحان کنیم.

نمادهای ریاضی خود را به (که اپراتور نامیده می شوند) به جمع و ضرب تبدیل کنید:

```
answer = first_number + second_number * third_number;
```

تمام آکولادها را حذف کرده و سپس برنامه ی خود را اجرا کنید.

بدون آکولا تصور می کنید که Java از چپ به راست محاسبه را انجام می دهد .بنابراین تصور می کنید که عدد اول را به عدد دوم اضافه می کند تا 175 به دست آورد. سپس تصور می کنید که در عدد سوم ضرب می شود که 25 می باشد .بنابراین پاسخ 4375 خواهد بود. سپس برنامه را اجرا کنید. پاسخ حقیقی را که شما به دست می آورید 1975 می باشد. پس جریان چیست؟

دلیل اینکه جاوا پاسخ اشتباه ارائه می دهد Operator Precedence است .جاوا برخی از نمادهای ریاضی را مهم تر از بقیه در نظر می گیرد. این برنامه ضرب را مقدم به جمع می داند، بنابراین عملیات ضرب را قبل از جمع انجام می دهد، سپس جمع را انجام می دهد. بنابراین جاوا در حال انجام عملیات زیر می باشد:

```
answer = first_number + (second_number * third_number);
```

با قرار دادن آکولادها در جای درست مشاهده می کنید که عدد دوم در عدد سوم ضرب شده است. سپس مجموع به اولین عدد اضافه می شود. بنابراین حاصل 75 در 25 عدد 1875 می باشد. عدد 100 را اضافه کنید، که 1975 می باشد.

اگر آن را به روش دیگری می خواهید، فراموش نکنید که با استفاده از آکولادها به جاوا اعلام کنید:

```
answer = (first_number + second_number) * third_number;
```

تقسیم مشابه ضرب می باشد: جاوا ابتدا تقسیم را انجام می دهد وسپس جمع و یا تفریق را. خط پاسخ خود را به شکل زیر تغییر دهید:

```
answer = first_number + second_number / third_number;
```

پاسخی که به دست می آورید 103 می باشد. اکنون چند آکولاد اضافه کنید:

```
answer = (first_number + second_number) / third_number;
```

پاسخ این بار 7 خواهد بود. بنابراین بدون آکولادها، جاوا ابتدا تقسیم را انجام می دهد و سپس 100 را به مجموع اضافه می کند – این عملکرد از چپ به راست کار نمی کند.

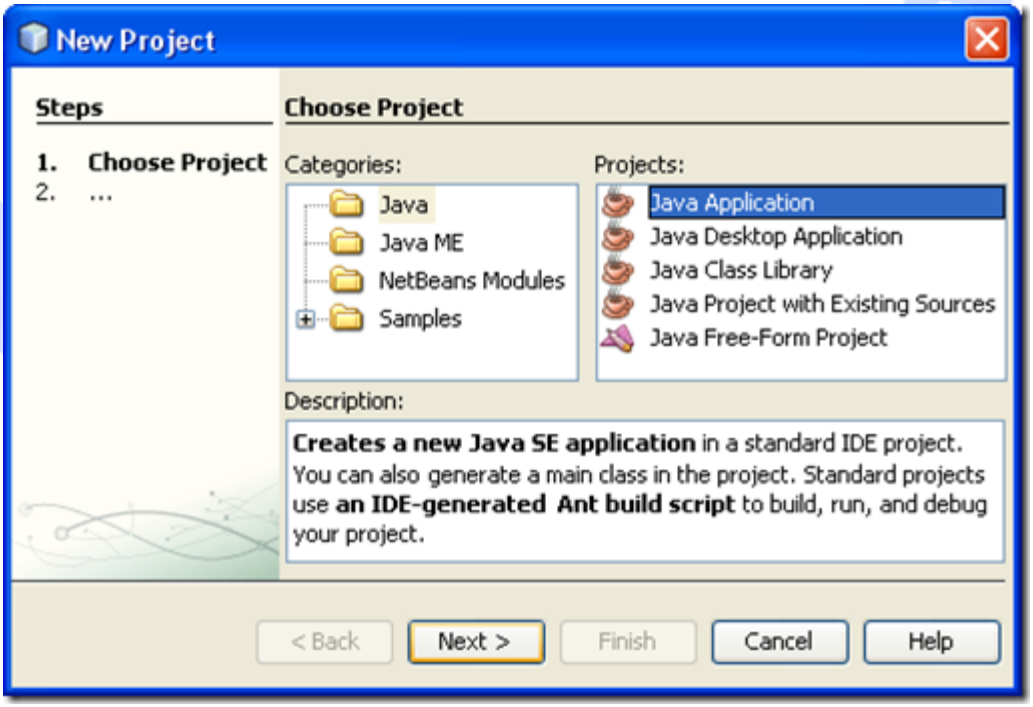
در اینجا لیستی از Operator Precedence را مشاهده می کنید:

ضرب و تقسیم – به طور مساوی رفتار می شوند، اما نسبت به جمع و تفریق دارای اولویت هستند .جمع و تفریق – به طور مساوی رفتار می شوند اما نسبت به ضرب و تقسیم اولویت پایین تری دارند .بنابراین اگر فکر می کنید که جاوا پاسخ اشتباه به شما می دهد، به یاد داشته باشید که Operator Precedence مهم می باشد و چند آکولاد اضافه می کند .در قسمت بعدی به چگونگی ذخیره ی مقادیر با استفاده از Java نگاهی خواهیم داشت.

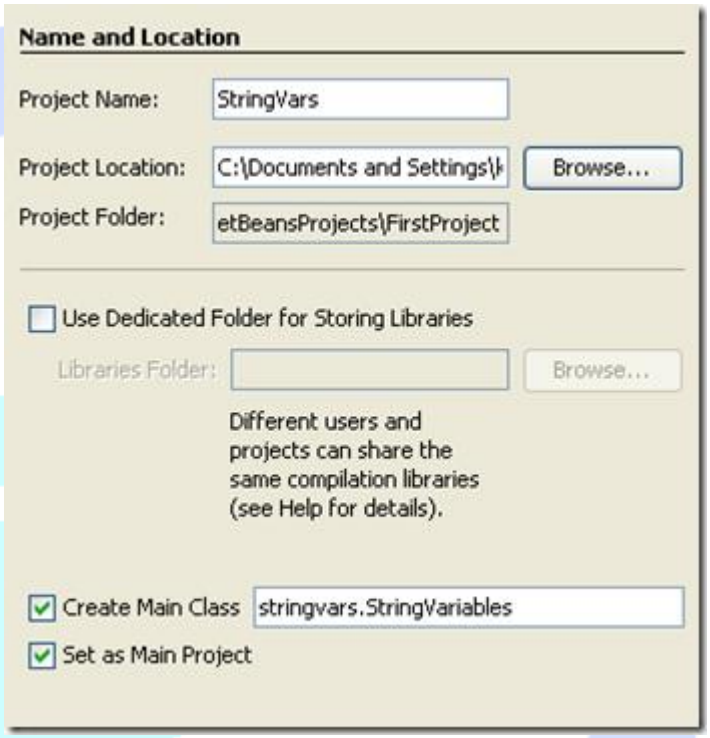
آموزش متغیر String در جاوا

همانند ذخیره ی مقادیر عددی، متغیرها می توانند متن را در خود حفظ کنند .شما میتوانید تنها یک کاراکتر یا چندید کاراکتر را ذخیره کنید. برای ذخیره ی تنها یک کاراکتر، متغیر char استفاده می شود. گرچه معمولا تمایل دارید بیشتر از یک کاراکتر را ذخیره کنید. برای انجام این کار نیاز به متغیر نوع رشته ای دارید.

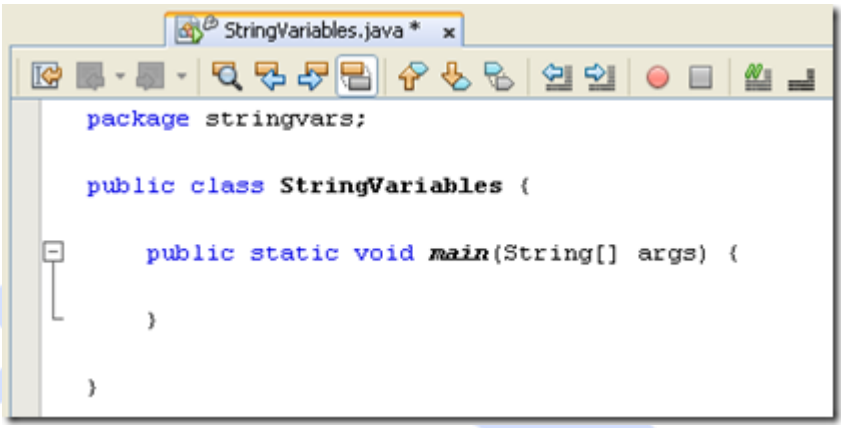
با کلیک کردن بر روی File > New Project از نوار منو در بالای NetBeans، یک پروژه ی جدید را آغاز کنید. وقتی دیالوگ باکس New Project ظاهر می شود، اطمینان حاصل کنید که Java و Java Application انتخاب شده اند:



روی Next کلیک کرده و StringVars را با عنوان نام پروژه تایپ کنید .اطمینان حاصل کنید که در قسمت Create Main Classیک تیک وجود دارد. سپس Mainرا پس از stringvars حذف کرده و در عوض StringVariablesرا تایپ کنید، مانند تصویر زیر:



بنابراین نام پروژه StringVars است و نام گروه نیز StringVariables می باشد .روی دکمه ی Finish کلیک کنید، پنجره ی برنامه نویسی شما مانند تصویر زیر خواهد بود (ما تمام کامنت های پیش فرض را حذف کرده ایم.) دقت داشته باشید که تمام حروف مربوط به نام پوشه با حروف کوچک نوشته شده اند(stringvars)، اما نام پروژه StringVars می باشد.



برای برقراری یک متغیر string ، لغت String را تایپ کنید که پس از آن نام متغیر قرار می گیرد. توجه داشته باشید که در لغت String حرف S بزرگ نوشته شده است. مجددا خط با یک نقطه ویزگول به پایان می رسد:

```
String first_name;
```

با تایپ کردن یک علامت تساوی، یک مقدار جدید را به متغیر string خود اختصاص دهید. پس از علامت تساوی متنی را که می خواهید ذخیره کنید، بین دو علامت نقل قول (") قرار می گیرد:

```
first_name = "William";
```

اگر ترجیه می دهید، می توانید تمام آن را در یک خط داشته باشید:

```
String first_name = "William";
```

متغیر دوم string را برقرار کنید تا یک نام یا نام خانوادگی را حفظ کنید:

```
String family_name = "Shakespeare";
```

برای اینکه هر دو نام را چاپ کنید، println( )زیر را اضافه کنید:

```
System.out.println( first_name + " " + family_name );
```

در بین آکولادهای println عبارت زیر را داریم:

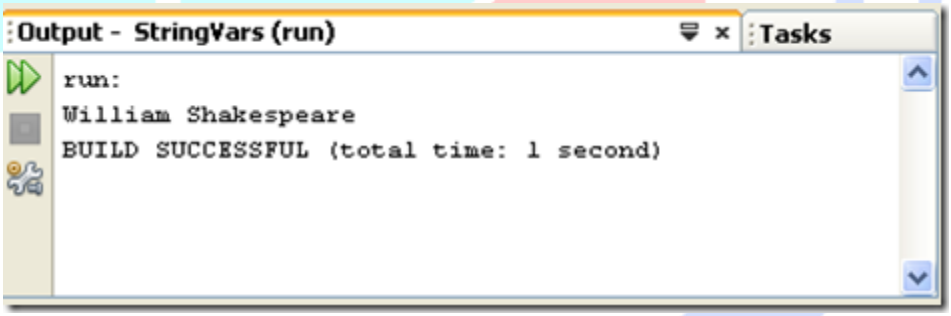
```
first_name + " " + family_name
```

هر آنچه در متغیر به نام first_nameوجود دارد، نسخه ی چاپی (print out) می نامیم. پس از آن یک علامت به علاوه وجود دارد که با یک فاصله دنبال می شود. این فاصله بین علامت های نقل قول احاطه شده است. به این روش جاوا در می یابد که ما قصد چاپ یک کاراکتر فاصله (space character) را داریم. پس از فاصله یک علامت به علاوه ی دیگر وجود دارد که با متغیر family_name دنبال می شود.

```
public static void main(String[] args) {

    String first_name = "William";
    String family_name = "Shakespeare";

    System.out.println(first_name + " " + family_name);
}
```

گرچه این مسئله کمی گیج کننده به نظر می رسد، اما ما فقط در حال چاپ یک نام، یک فاصله و سپس نام خانوادگی هستیم. پنجره ی برنامه نویسی شما باید مانند تصویر زیر به نظر برسد:

برنامه ی خود را اجرا کنید و پس از آن در پنجره ی Output تصویر زیر مشاهده خواهید کرد:



اگر شما در حال ذخیره ی تنها یک کاراکتر مجزا هستید، متغیر مورد نیاز شما char (با حرف کوچک) می باشد. برای ذخیره ی کاراکتر از علامت نقل قول (‘)به جای (“) استفاده می کنید. در اینجا مجددا برنامه را مشاهده می کنید، اما این بار با متغیر:char

اگر سعی کنید یک متغیر char را با علامت نقل قول جفت (“) ذخیره کنید، NetBeansزیر کد ناسازگار با قرمز خط خواهد کشید و خطای نوع ناسازگار (incompatible type)ارائه خواهد داد. به هرحال شما می توانید یک متغیر Stringتنها با یک کاراکتر مجزا داشته باشید. اما به علامت نقل قول دوگانه (“)نیاز دارید. بنابراین این مورد درست می باشد:

```
String first_name = "W";
```

اما مورد زیر درست نمی باشد:

```
String first_name = 'W';
```

ورژن دوم دارای یک علامت نقل قول مجزا می باشد، در حالیکه مورد اول دارای علامت نقل قول دوگانه (“) می باشد.

موارد بیشتری در مورد string ها وجود دارد و بعدها مجددا آنها را مشاهده خواهید کرد. اکنون اجازه دهید ادامه داده و چند ورودی از یوزر دریافت کنیم.

آموزش دریافت ورودی از کاربر در جاوا

یکی از نقاط قوتJava ، وجود کتابخانه های عظیمی از کدهای موجود برای شما می باشد. برای انجام کارهای خاص کدهایی نوشته شده است. تمام آنچه باید انجام دهید ارجاع به کتابخانه ی مورد نظر و سپس فراخوانی متد می باشد. گروه واقعا مفیدی که ورودی یک یوزر را کنترل می کند، گروه Scanner نامیده می شود. این گروه در کتابخانه ی java.util یافت می شود. برای استفاده از گروه Scanner لازم است به آن گروه در کد خود ارجاع کنید. این کار با لغت کلیدی import انجام می شود.

```
import java.util.Scanner;
```

عبارت import لازم است بالای عبارت عبارت Class قرار بگیرد.

```
import java.util.Scanner;

public class StringVariables {

}
```

این عبارت به جاوا می گوید که می خواهید از یک گروه خاص در یک کتابخانه ی خاص استفاده کنید – گروه Scannerکه در java.utilواقع شده است.

کار دیگری که باید انجام دهید ایجاد یک آبجکت از گروه Scanner می باشد) . یک گروه در واقع دسته ای از کدهاست. این کد تا زمانیکه یک آبجکت جدید از آن ایجاد نکنید، کاری انجام نمی دهد(.

برای ایجاد یک آبجکت Scanner جدید، کد مورد نیاز را در زیر مشاهده می کنید:

```
Scanner user_input = new Scanner( System.in );
```

بنابراین به جای متغیر int یا یک متغیر String ، در حال تنظیم یک متغیر Scannerمی باشیم، که مورد ما user_inputنامیده می شود. پس از یک علامت تساوی لغت کلیدی new را داریم که برای ایجاد آبجکت های جدید از یک گروه استفاده می شود. آبجکتی که در حال ایجاد آن هستیم از گروه Scanner می باشد . در بین آکولادها باید به جاوا اعلام کنید که از System Input (System.in) خواهد بود.

برای گرفتن ورودی یوزر، می توانید یکی از چندین متد موجود را در آبجکت جدید Scanner وارد عمل کنید. یکی از این متدها next نامیده می شود. این متد رشته ی بعدی متن را که یک یوزر روی صفحه کلید تایپ می کند، دریافت می کند:

```
String first_name;
first_name = user_input.next( );
```

بنابراین پس از آبجکت user_input یک نقطه تایپ می کنیم. سپس لیستی از متدهای موجود مشاهده خواهید کرد. روی next دابل کلیک کنید و سپس در انتهای خط یک نقطه ویرگول تایپ کنید. برای به جلو بردن یوزر می توانیم متن نیز تایپ کنیم:

```
String first_name;
System.out.print("Enter your first name: ");
first_name = user_input.next( );
```

دقت داشته باشید که مانند قبل از print به جای println استفاده می کنیم . تفاوت بین این دو این است که printlnپس از خورجی مکان نما را به یک خط جدید حرکت می دهد، اما print روی همان خط می ماند.

یک پیشروی برای نام خانوادگی نیز وارد می کنیم:

```
String family_name;
```

```
System.out.print("Enter your family name: ");
family_name = user_input.next( );
```

این همان کد است به جز اینکه اکنون جاوا هرآنچه را یوزر در قسمت متغیر family_nameبه جای متغیر first_nameتایپ می کند، ذخیره می کند.

برای چاپ ورودی می توانیم مورد زیر را اضافه کنیم:

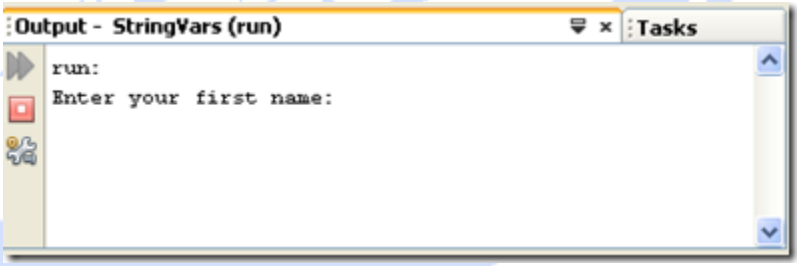
```
String full_name;
full_name = first_name + " " + family_name;
System.out.println("You are " + full_name);
```

یک متغیر String دیگر با عنوان full_name تنظیم می کنیم و هر آنچه در متغیرهای first_name و family_nameاست را ذخیره می کنیم. در بین این دو یک فاصله قرار می دهیم. خط آخر همه ی آن را در پنجره ی Output چاپ خواهد کرد.

بنابراین کد خود را طوری تطبیق دهید تا با تصویر بعدی هماهنگی داشته باشد:

```
package stringvars;

import java.util.Scanner;

public class StringVariables {

    public static void main(String[] args) {

        Scanner user_input = new Scanner(System.in);

        String first_name;
        System.out.print("Enter your first name: ");
        first_name = user_input.next();

        String family_name;
        System.out.print("Enter your family name: ");
        family_name = user_input.next();

        String full_name;
        full_name = first_name + " " + family_name;

        System.out.println("You are " + full_name);

    }
}
```

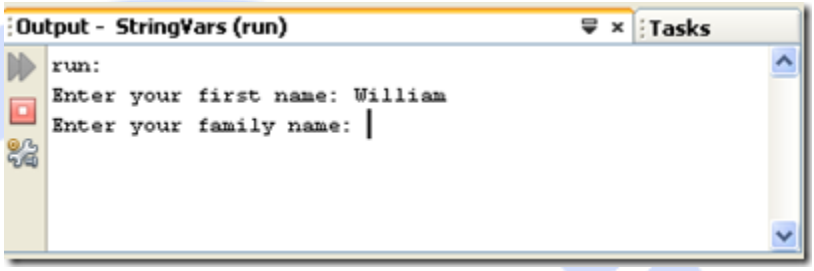
برنامه ی خود را اجرا کنید تا اینکه پنجره ی Output تصویر زیر را نمایش دهد:



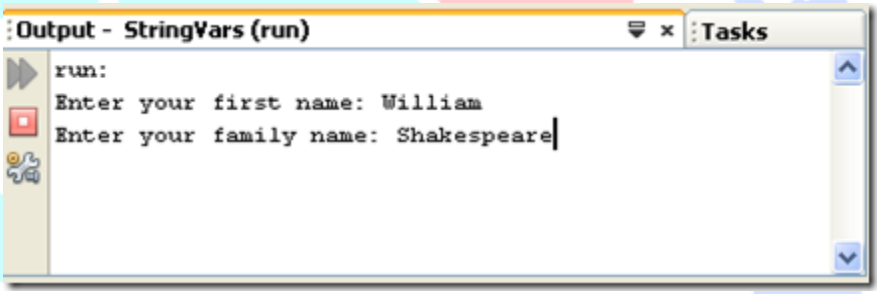
اکنون جاوا متوقف شده تا اینکه در صفحه کلید چیزی وارد کنید، و تا زمانیکه دکمه ی Enter صفحه کلید را فشار ندهید، پیشرفتی رخ نخواهد داد .بنابراین بعداز "Enter your first name:" را کلیک چپ کنید، مشاهده خواهید کرد که مکان نمای شما محو خواهد شد. یک نام تایپ کنید و سپس دکمه ی enter را فشار دهید.

پس از فشردن دکمه ی enter ، جاوا هر آنچه را که در سمت چپ علامت تساوی در متغیر نام تایپ و ذخیره شده، خواهد گرفت. در مورد ما این متغیر first_name نامیده می شد.

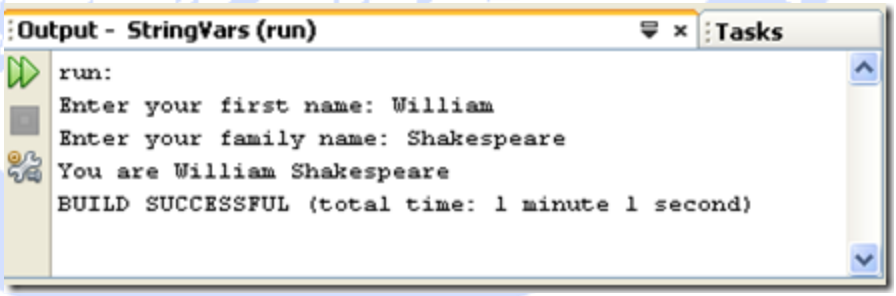
سپس برنامه وارد خط بعدی کد می شود:



یک نام خانوادگی تایپ کرده و سپس مجددا دکمه ی Enter را فشار دهید:



اکنون ورودی یوزر به پایان رسیده است و بقیه ی برنامه اجرا می شود. این خروجی هردو نام می باشد. نتیجه ی نهایی باید مشابه تصویر زیر باشد:



بنابراین ما از گروه Scanner برای گرفتن ورودی از یک یوزر استفاده می کنیم. هر آنچه تایپ شد در متغیرها ذخیره شد. نتیجه نیز در پنجره ی Output چاپ شد .در بخش بعدی نگاه کوتاهی به Option Panes خواهیم داشت.

آموزش گزینه های پنل جاوا

گروه موثر دیگری برای قبول ورودی یوزر و نمایش نتایج گروه JOptionPane (گزینه های پانل) می باشد. این گروه در کتابخانه ی javax.swing واقع شده است. گروه JOptionPane به شما اجازه می دهد تا جعبه های ورودی مانند تصویر داشته باشید:

و جعبه های پیغامی نیز مانند تصویر زیر:

اجازه بدهید کد خود را از the previous section تطبیق دهیم و چند گزینه ی پانل (option panes) داشته باشیم.

اولین کاری که باید انجام دهید مراجعه به کتابخانه ی مورد نظر است.

```
import javax.swing.JOptionPane;
```

این به شما می گوید که ما قصد استفاده از گروه JOptionPane را داریم که در کتابخانه ی javax.swing واقع شده است.

اگر تمایلی به تطبیق کد قبل ندارید، می توانید یک پروژه ی جدید برای این مورد آغاز کنید.(اکنون باید بدانید که چگونه یک پروژه ی جدید ایجاد کنید . فقط به یاد داشته باشید که نام گروه را از Main به نام دیگری تغییر دهید . ما قصد داریم که گروهی به نام InputBoxes داشته باشیم. نام پوشه ی ما userinputخواهد بود).

خط import را به پروژه ی جدید خود وارد کرده و پنجره ی کد شما باید مشابه تصویر زیر باشد:

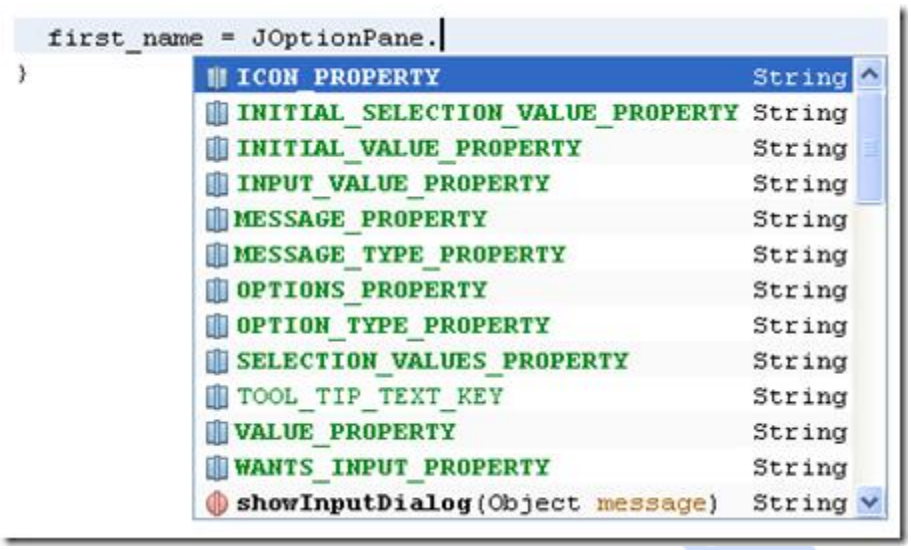
```
package userinput;
import javax.swing.JOptionPane;

public class InputBoxes {

    public static void main(String[] args) {

    }

}
```

(علت وجود خطوط موجی این است که ما هنوز از گروه JOptionPane استفاده نکرده ایم. زمانی که این کار را انجام دهیم این خطوط نیز محو خواهند شد) .

برای به دست آوردن یک جعبه ی وردی که یوزر در آن تایپ کند، می توانید از متد showInputDialog مربوط به JOptionPane استفاده کنید. درست مانند آخرین بار، ورودی را مجددا در متغیر نام (first name) ذخیره خواهیم کرد .بنابراین خط زیر را به متد اصلی خود اضافه کنید:

```
String first_name;
first_name = JOptionPane.showInputDialog("First Name");
```

به محض اینکه بعد از JOptionPane یک نقطه تایپ کنید، لیست زیر را مشاهده خواهید کرد:



روی showInputDialog دابل کلیک کنید. بین آکولادهایshowInputDialog ، پیغامی را تایپ کنید که می خواهید در بالای تکست باکس نمایش داده شود. ما "First name"را تایپ کرده ایم. مانند تمام رشته ها لازم است که این عبارت در داخل علامت های نقل قول قرار بگیرد.

کد زیر را اضافه کنید که پس از آن می توانیم نام خانوادگی یوزر را دریافت کنیم:

```
String family_name;
family_name = JOptionPane.showInputDialog("Family Name");
```

این دو را به یکدیگر متصل کرده و متن نیز به آن اضافه کنید:

```
String full name;
full_name = "You are " + first_name + " " + family_name;
```

برای نمایش نتیجه در یک پیغام، مورد زیر را اضافه کنید:

```
JOptionPane.showMessageDialog( null, full_name );
```

این بار ما از لیست showMessageDialog را می خواهیم. بین آکولادها ابتدا لغت null را مشاهده می کنیم. این لغت، لغت کلیدی جاوا می باشد که این پیغام ارتباطی با هیچ مورد دیگری در برنامه ندارد. پس از میرگول (کاما) متنی است که می خواهیم در جعبه ی پیغام نمایش دهیم. کل کد شما باید مانند تصویر زیر باشد:

```
package userinput;
import javax.swing.JOptionPane;

public class InputBoxes {

    public static void main(String[] args) {

      String first_name;
      first_name = JOptionPane.showInputDialog("First Name");

      String family_name;
      family_name = JOptionPane.showInputDialog("Family Name");

      String full_name;
      full_name = "You are " + first_name + " " + family_name;

      JOptionPane.showMessageDialog(null, full_name);
      System.exit(0);

    }
}
```

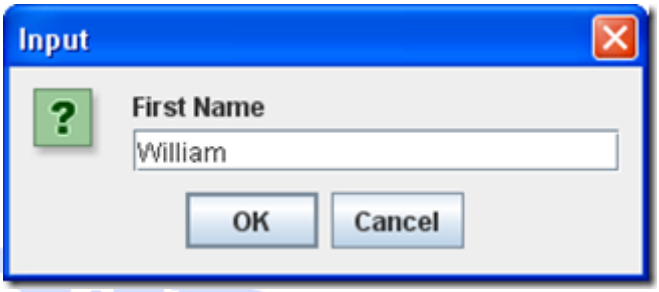
به خط واقع در پایین کد دقت کنید:

```
System.exit(0);
```

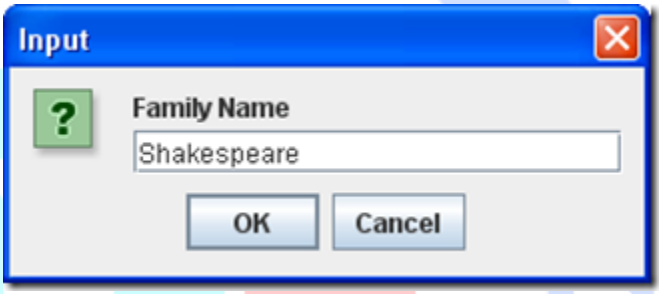
همانطور که از نامش پیداست، اطمینان می دهد که برنامه خارج می شود. اما همچنین با حذف آبجکت های ایجاد شده از حافظه، آن را برای ما مرتب می سازد.

اکنون کد خود را اجرا کنید. (راه دیگری برای اجرای برنامه ی خود در NetBeans، کلیک راست کردن بر روی هر قسمتی از پنجره ی برنامه نویسی می باشد. از منوی ظاهر شده Run File را انتخاب کنید).

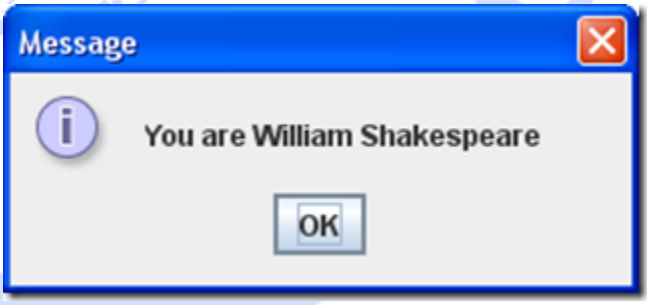
جعبه ی ورودی First Name را مشاهده خواهید کرد. چیزی در داخل آن تایپ کرده و سپس روی OK کلیک کنید:



وقتی که جعبه ی ورودی Family Name ظاهر می شود، یک نام خانوادگی تایپ کرده و روی ok کلیک کنید:



پس از کلیک کردن روی ok جعبه ی پیغام ظاهر خواهد شد:



برای اتمام برنامه روی ok کلیک کنید.

جعبه های Input و Message بیشتر نیز فرمت می شوند. مورد زیر را برای جعبه های Input خود امتحان کنید:

```
showInputDialog("First Name", "Enter Your First Name");
```

```
showInputDialog("Family", "Enter Your Family Name");
```

برای جعبه های Messages خود مورد زیر را امتحان کنید (مورد شما باید روی یک خط قرار بگیرد.):

```
showMessageDialog(null, full_name, "Name", JOptionPane.INFORMATION_MESSAGE);
```

به جای JOptionPane.INFORMATION_MESSAGE موارد زیر را امتحان کنید:

```
ERROR_MESSAGE
PLAIN_MESSAGE
QUESTION_MESSAGE
WARNING_MESSAGE
```

جعبه های Input فقط برای متن استفاده نمی شوند: آنها اعداد را نیز می پذیرند. برنامه ای بنویسید که یوزر را در دو برنامه،مساحت یک مستطیل و ارتفاع یک مستطیل، به جلو می برد. از یک جعبه ی پیغام برای محاسبه ی مساحت مستطیل استفاده کنید. (به یاد داشته باشد که مستطیل حاصلضرب طول در عرض می باشد.) به هرحال برای این تمرین به کمک نیز نیاز خواهید داشت.

کمک مربوط به تمرین:

شنا باید از متغیر String برای به دست آوردن اعداد خود از یوزر استفاده کنید:

```
String breadth;
breadth = JOptionPane.showInputDialog("Rectangle Breadth");
```

به هرحال نمی توانید دو رشته را با یکدیگر ضرب کنید. لازم است رشته ها را به اعداد صحیح تبدیل کنید. می توانید به شکل زیر یک رشته را به یک عدد صحیح تبدیل کنید:

```
Integer.parseInt( string_to_convert )
```

بنابراین ابتدا Integer را تایپ کرده و سپس یک نقطه تایپ کنید. بعد از آن ( )parseInt را تایپ کنید. بین آکولادهایparseInt ، نام متغیری را تایپ کنید که سعی در تغییر آن دارید.

یک متغیر int برای ناحیه تنظیم کنید. سپس می توانید روی همان خط ضرب کرده و اختصاص دهید؛

```
int area = Integer.parseInt( string_one ) * Integer.parseInt( string_two);
```

برای جعبه ی پیغام از الحاق (concatenation) استفاده کنید:

```
"answer = " + area
```

می توانید از هر نماد MESSAGE دیگری برای جعبه ی پیغام خود استفاده کنید.

اگر برای قاعده و ارتفاع مقادیر ممیزی شناور وارد کنید، برنامه اجرا نخواهد شد. چگونه این مشکل را حل می کنید؟

وقتی که تمرین بالا را حل کنید، آیا واقعا Integer.parseInt می خواهید؟ فکر می کنید از چه موارد دیگری می توانید استفاده کنید؟

بسیار خوب، ما ادامه خواهیم داد، اجازه بدهید عبارات IF را امتحان کنیم.

آموزش if در جاوا

برنامه نویسی که در حال حاضر در حال انجام آن هستید یک برنامه نویسی پی در پی می باشد، به این معنا که کد از بالا به پایین اجرا می‌شود. این برنامه نویسی خطی می باشد که در آن هر خط با شروع از اولین خط و پایان آخرین خط از کد خوانده خواهد شد.

اما همیشه تمایل ندارید برنامه هایتان اینگونه اجرا شوند. اغلب تمایل دارید کدها تنها تحت شرایط خاص اجرا شوند. برای مثال ممکن است بخواهید یک پیام در صورتی نمایش داده شود که یوزر کمتر از 18 سال سن داشته باشد و یا بالعکس پیام خاصی تنها در صورتی نمایش داده شود که یوزر بیشتر از 18 سال سن داشته باشد. شما می خواهید که جریان برنامه نویسی را برای خود کنترل کنید .این کار را می توانید با منطق شرطی انجام دهید.

منطق شرطی اساسا در مورد لغت IF می باشد: اگر یوزر کمتر از 18 سال سن دارد این پیغام را نمایش دهید، اگر یوزر بیشتر از بیشتر از 18 سال سن دارد، این پیغام را نمایش بده. خوشبختانه استفاده از منطق شرطی در جاوا ساده می باشد. اجازه بدهید با عبارات IF آغاز کنیم.

عبارات IF (IF Statements)

در برنامه نویسی که عبارت IF وجود دارد، اجرای یک مورد به جای مورد دیگر در هنگام اجرای کد بسیار متداول است. ساختار عبارت IF در جاوا به شکل زیر است:

```
if ( Statement ) {

}
```

عبارت را با لغت if (با حروف کوچک) و یک جفت آکولاد شروع می کنید. این کد، کدی می باشد که تنها زمانی می خواهید اجرا کنید که شرایط شما برقرار باشد. شرایط مربوط در داخل آکولادها قرار می گیرد:

```
if ( user < 18 ) {
}
```

این شرط به این معناست " اگر یوزر کمتر از 18 سال سن دارد." اما به جای عبارت کمتر از نشان (<) استفاده کرده ایم. اگر یوزر کمتر از 18 سال باشد، می خواهیم که برنامه ای اجرا شود، برای مثال پیغامی نمایش داده شود:

```
if ( user < 18 ) {

//DISPLAY MESSAGE

}
```

اگر یوزر کمتر از 18 سال نباشد، پیغام بین آکولادها اجرا نخواهد شد و برنامه به مسیر خود، به سمت خطوط پایین کد، ادامه خواهد داد. هر آنچه در داخل آکولادها تایپ می کنید، تنها در صورتی اجرا خواهند شد که شرایط درست باشند . این شرایط بین آکولادها قرار می گیرند.

قبل از امتحان این مورد، نماد تندنویسی دیگر <می باشد، این نماد به معنای بزرگتر از می باشد. عبارت IF در بالا برای یوزرهای بزرگتر از 18 سال قابل اصلاح می باشد:

```
if ( user > 18 ) {

//DISPLAY MESSAGE

}
```

تنها مورد جدید در این کد نماد >می باشد. اکنون شرایط مربوط به یوزرهای بالاتر از 18 سال می باشد.

اما شرایط برای یوزرهایی با سن دقیقا 18 سال درست نیست. اگر می خواهید شرایط در مورد افراد 18 سال یا بالای 18 سال درست باشد، می توانید از عبارت بزرگتر یا مساوی استفاده کنید. نمادها برای این شرایط بزرگتر از (>) و مساوی با (=) می باشند.

```
if ( user >= 18 ) {

//DISPLAY MESSAGE

}
```

همچنین می توانید این شرایط را برای کوچکتر یا مساوی با به روش زیر چک کنید:

```
if ( user <= 18 ) {

//DISPLAY MESSAGE

}
```

کد بالا حاوی یک نماد (>) می باشد که با علامت مساوی دنبال می شود.

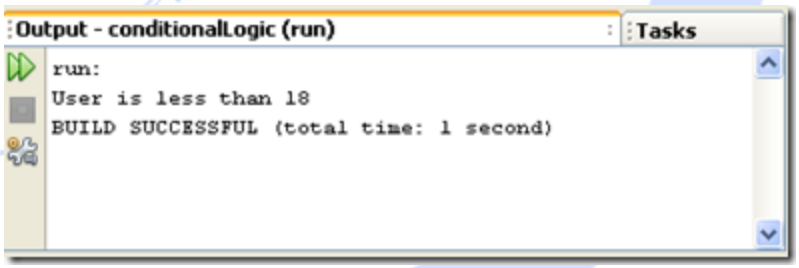
اجازه بدهید این مورد را در یک برنامه ی ساده امتحان کنیم.

یک پروژه ی جدید را با کلیک بر روی File > New Project از نوار منو در NetBeans آغاز کنید. شما می توانید پوشه و گروه خود را با هر نامی که می خواهید نام گذاری کنید. کد زیر را وارد کنید (نام پوشه ی ما conditionallogicو گروه ما نیز IFStatements نامیده می شود:).

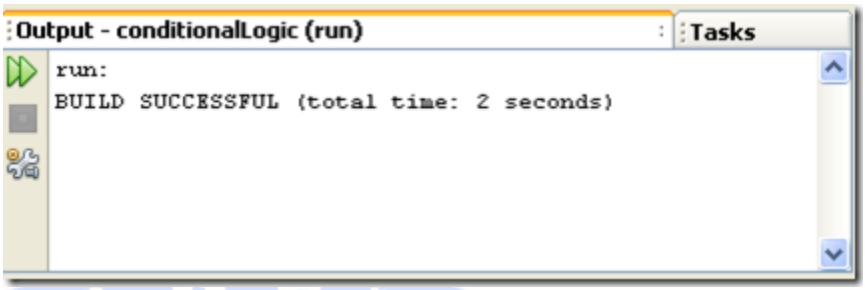
```
package conditionallogic;

public class IFStatements {

    public static void main(String[] args) {

        int user = 17;

        if (user < 18) {
            System.out.println("User is less than 18");
        }

    }
}
```

یک متغیر صحیح را تنظیم کرده ایم و مقداری برابر 17 به آن اختصاص داده ایم. عبارت IF مورد " کمتر از 18 سال" را چک می کند. بنابراین پیغام بین آکولادها باید چاپ شود.

برنامه ی خود را اجرا کرده و آن را بررسی کنید (NetBeans . معمولا برنامه را در متن بولد شده در پنجره ی Projects اجرا می کند و نه آن کدی که شما نمایش داده اید. برای اجرای کد در پنجره ی coding در قسمتی از کد کلیک راست کنید. از منوی ظاهر شده Run File را انتخاب کنید.) . تصویر زیر را در پنجره ی Output باید مشاهده کنید:



اکنون مقدار را برای متغیر یوزر از 17 به 18 تغییر دهید. برنامه ی خود را مجددا اجرا کنید. باید تصویر زیر را مشاهده کنید:

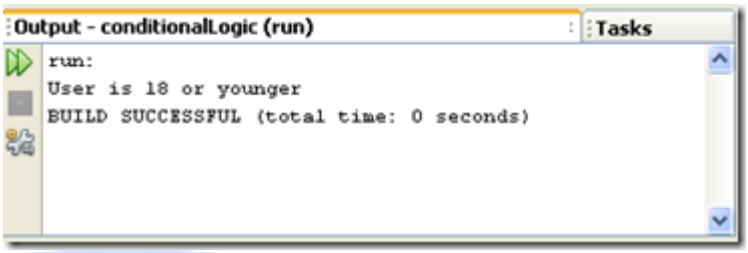


بنابراین برنامه به خوبی و بدون پیغام خطا اجرا خواهد شد. اینجاست که چیزی چاپ نمی شود. دلیل آن این است که کد پیغام بین آکولادهای عبارت IF وجود دارد و عبارت IF در حال بررسی مقادیر کمتر از 18 می باشد. اگر شرایط برقرار نباشد، جاوا آکولادها را کلا نادیده گرفته و به کار خود ادامه می دهد.

تمرین : نماد "کوچکتر از یا مساوی با" را با جایگزین نماد کوچکتر از کنید. پیغام خود را طوری تغییر دهید تا با مورد " یوزر 18 سال یا کمتر از 18سال است" هماهنگ باشد. برنامه ی خود را مجددا اجرا کنید. آیا پیغام را مشاهده می کنید؟ تمرین :مقدار یوزر را به 20 تغییر دهید. برنامه ی خود را مجددا اجرا کنید. آیا هنوز پیغام را مشاهده می کنید؟ شما می توانید بیشتر از یک عبارت IF در کد خود داشته باشید. کد زیر را امتحان کنید:

```
public static void main(String[] args) {

    int user = 18;

    if (user <= 18) {
        System.out.println("User is 18 or younger");
    }

    if (user > 18) {
        System.out.println("User is older than 18");
    }

}
```

این بار دو عبارت IF داریم. اولین عبارت مقادیر کمترا از یا برابر با 18را بررسی می کند. دومین عبارت مقادیر بزرگتر از 18 را بررسی می کند .وقتی که کد با مقداری از 18 یا کمتر از 18 برای نتغیر یوزر اجرا می شود، خروجی مانند زیر می باشد:



تغییر مقدار متغیر یوزر به 20 نیز نتیجه ی زیر را به دنبال خواهد داشت :بنابراین تنها یکی از عبارت های IF خروجی خط چاپی خواهد داشت و تمام آن بستگی به مقدار متغیر یوزر دارد . در قسمت بعدی با Conditional Logicادامه خواهیم داد.

آموزش دستورات if و else در جاوا

به جای استفاده از دو عبارت IF Statements می توانید از یک عبارت IF ... ELSE استفاده کنید. در اینجا ساختار یک IF ... ELSE را مشاهده می کنید:

```
if ( condition_to_test ) {
}
else {
}
```

اولین خط با یک if شروع می شود که پس از آن شرایطی قرار گرفته که قصد امتحان کردن آن را دارید. این شرایط بین دو پرانتز قرار می گیرد. مجددا کروشه هایی برای مجزا کردن انتخاب های مختلف استفاده می شوند. دومین انتخاب پس از لغت else و بین کروشه های خود قرار می گیرد. در اینجا مجددا کدی را مشاهده می کنید که سن یوزر را چک می کند:

بنابراین در اینجا فقط دو انتخاب وجود دارد: یوزر 18 سال یا کمتر از 18 سال سن دارد، یا اینکه بیشتر از 18 سال سن دارد. کد خود را طوری تنظیم کنید تا با تصویر بالا هماهنگ بوده و آن را امتحان کنید. اولین پیغام باید چاپ شود. اکنون مقدار متغیر مربوط به یوزر را به 20 تغییر داده و کد را اجرا کنید. پیغام بین کروشه های ELSEباید در پنجره ی Output نمایش داده شود.

```
IF … ELSE IF
```

شما می توانید این را برای بیشتر از دو انتخاب چک کنید. برای مثال اگر بخواهیم دامنه ی سنی بیشتری، مانند 19 تا 39 سال، را بررسی کنیک، چطور؟ برای بیشتر از دو انتخاب عبارت IF ... ELSE IF استفاده می شود.

ساختر یک عبارت IF ... ELSE IF به شکل زیر می باشد:

```
if ( condition_one ) {
}
else if ( condition_two ) {
}
else {
}
```

بخش جدید به شکل زیر می باشد:

```
else if ( condition_two ) {
}
```

بنابراین اولین IF برای امتحان کردن شماره یک به کار می رود (به عنوان مثال 18 و کمتر). پس از آن else if قرار می گیرد، که با یک جفت آکولاد دنبال می شود. شرط دوم بین این دو پرانتز قرار می گیرد. هر موردی که با دو شرط اول قرار نگیرد، برای آخرین else قرار خواهد گرفت. مجددا با استفاده از آکولادها، کد مجزا می شود، به این شکل که هرif ، else if یا else دارای آکولادهای مخصوص خود می باشد. نادیده گرفتن یکی از آنها منجر به پیغام خطا خواهد شد.

قبل از امتحام کردن کد جدید، لازم است با چند اپراتور شرطی دیگر آشنا شوید. مواردی که تاکنون استفاده کرده اید، عبارتند از:

> Greater Than < Less Than >= Greater Than or Equal To <= Less Than or Equal To

در اینجا چهار مورد دیگر را مشاهده می کنید که می توانید استفاده کنید:

&& AND || OR == HAS A VALUE OF ! NOT

اولین مورد دو نماد (ampersand) &می باشد که برای امتحان کردن همزمان چند شرط استفاده می شود. از آن می توانیم برای تست کردن دو محدوده ی سنی استفاده کنیم:

```
else if ( user > 18 && user < 40 )
```

در اینجا بررسی می کنیم اگر یوزر بیشتر از 18 سال و کمتر از 40 سال سن دارد. به یاد داشته باشید که سعی در بررسی موردی هستیم که در در داخل متغیر یوزر قرار دارد. شرط اول (user > 18) "Greater than 18" (بیشتر از 18 سال) می باشد و شرط دوم (user < 40) "Less than 40" (کمتر از 40 سال) است. بین این دو اپراتور (&&) AND را داریم. بنابراین تمام خط در حال بیان این عبارت است که " اگر یوزر کمتر از 18 سال و بیشتر از 40 سال است.("else if user is greater than 18 AND user is less than 40") ".

در یک لحظه وارد سه اپراتور شرطی دیگر خواهیم شد. اما در اینجا کد جدیدی را برای امتحان مشاهده می کنید:

برنامه ی خود را اجرا کرده و آن را امتحان کنید. شما باید قبل از پرینت گرفتن بتوانید حدس بزنید که چه چیزی چاپ می شود. از آنجایی مقدار مربوط به متغیر یوزر 21 می باشد، پیغامی بین آکولادهای مربوط به else if در پنجره ی Outputنمایش داده خواهد شد.

مقدار مربوط به متغیر یوزر را از 21 به 45 تغییر دهید. اکنون پیغام مربوط به بخش else از کد باید نمایش داده شود.

شما می توانید هر تعداد if part که تمایل دارید، اضافه کنید. فرض کنید قصد بررسی سن یوزر بین 45 یا 50 را داریم. می توانیم از دو اپراتور شرطی بالا استفاده کنیم. می توانیم چک کنیم اگر یوزر دارای مقدار 45 یا مقدار 50 می باشد:

```
else if (user == 45 || user == 50)
```

برای بررسی متغیر مربوط به یوزر بر اساس مقدار مورد نظر شما، از دو علامت مساوی بدون هیچ فاصله ای بین آنها استفاده کنید. جاوا بررسی را برای همان مقدار و نه مقدار دیگر انجام خواهد داد. از آنجایی که قصد بررسی سن یوزر 50 ساله را هم داریم، می توانیم شرایط دیگری در همان پرانتزها داشته باشیم == user :

.50این عبارت در واقع بیان می کند که " بررسی کنید اگر یوزر دارای متغیری با مقدار 50 می باشد." در بین این دو شرایط اپراتور OR وجود دارد. اینها دو کاراکتر pipe می باشند که روی صفحه کلید UK (انگلیسی) در

سمت چپ حرف z قرار دارند. مجددا فاصله ای بین آنها وجود ندارد. تمام خط بالا بیانگر این عبارت است که " اگر یوزر دارای مقدار 45 یا مقدار 50 باشد.".

در اینجا کد را بخش جدید else if مشاهده می کنید:

آن را برای خود امتحان کنید. مقدار متغیر یوزر را به 45 تغییر داده و کد خود را اجرا کنید. سپس آن را به 50 تغییر داده و مجددا کد را اجرا کنید . در هر دو مورد پیغام جدیدی باید نمایش داده شود.

اپراتورهای شرطی مختلفی با ترفندهای گوناگون مورد استفاده قرار می گیرند. اما شما فقط یک متغیر را برای یک شرایط خاص امتحان می کنید. این درواقع مربوط می شود به انتخاب اپراتور شرطی درست یا اپراتورهایی مانسب با کار آن.

IF Statementهای تو در تو

شما می توانید IF Statement ها را در هم قرار دهید. (این امر در مورد عبارت های IF ... ELSE و IF ... ELSE IF نیز استفاده می شود.) Nesting به معنای قرار دادن یک IF Statement در داخل یک عبارت دیگر می باشد. به عنوان مثال فرض کنید قصد بررسی سن فردی کمتر از 18 و بیشتر از 16 سال را دارید. می خواهید که برای بیشتر از 16 ساله ها یک پیغام متفاوت نمایش دهید .با اولین عبارت IF آغاز کنید.

```
if ( user < 19 ) {
System.out.println( "18 or younger");
}
```

برای بررسی بیشتر از 16 سال، می توانید یک عبارت IF دوم در داخل عبارتی که دارید، قرار دهید. فرمت مانند قبل می باشد:

```
if ( user < 19 ) {
if ( user > 16 && user < 19 ) {
System.out.println( "You are 17 or 18");
}
}
```

بنابراین اولین عبارت متغیر یوزر را اگر از 19 کمتر باشد، دریافت می کند.دومین عبارت حتی متغیر یوزر را محدودتر می کند، بین 16 تا 19. برای چاپ پیغام های مختلف، می توانید به جای عبارت IF بالا یک عبارت IF ELSE ...داشته باشید:

```
if ( user < 19 ) {
if ( user > 16 && user < 19 ) {
System.out.println( "You are 17 or 18");
}
else {
System.out.println( "16 or younger");
}
}
```

به محل همه ی کروشه ها دقت کنید: اگر یک مورد اشتباه باشد، برنامه ی شما اجرا نخواهد شد Nested IF . Statementمی توانند کمی گیج کننده باشند، اما تمام کاری که سعی در انجام آن دارید، محدود کردن انتخاب ها می باشد.

در بخش بعد، نوع متغیر Boolean را مشاهده خواهید کرد.

آموزش مقادیر Boolean

مقدار Boolean مقداری با دو انتخاب می باشد true :یاfalse ، yesیا no ، 1یا 0 .در جاوا نوعی متغیر برای مقادیر Boolean وجود دارد:

```
boolean user = true;
```

بنابراین به جای تایپ کردن int یا double یاstring ، فقط کافیست Boolean تایپ کنید(با b کوچک.) پس از نام متغیر خود می توانید یک مقدار true یا falseاختصاص دهید. دقت کنید که اپراتور مربوط به اختصاص دادن یک علامت تساوی (=) می باشد. اگر می خواهید بررسی کنید که یک متغیر مقداری از چیزی را دارد بخ دو علامت تساوی (= =) احتیاج دارید.

این کد ساده را امتحان کنید:

```
boolean user = true;
if ( user == true) {
```

```
System.out.println("it's true");
}
else {

System.out.println("it's false");

}
```

بنابراین اولیت عبارت IF متغیر یوزر را بررسی می کند، اگر یک مقدار true داشته باشد. بخش دیگر false بودن آن را بررسی می کند. نیازی به عبارت "else if (user = = false)"ندارید. در انتها اگر اگر چیزی true نباشد پس falseمی باشد. بنابراین می توانید از بخش دیگر استفاده کنید: تنها دو انتخاب با مقادیر Booleanوجود دارد.

تنها اپراتور شرطی ما در لیست اپراتور NOT می باشد. شما می توانید از این اپراتور با مقادیر Boolean استفاده کنید. نگاهی به کد زیر داشته باشید:

```
boolean user = true;

if ( !user ) {
System.out.println("it's flase");
}
else {
System.out.println("it's true");
}
```

این کد تقریبا مانند دیگر کدهای Boolean می باشد، به جز خط زیر:

```
if ( !user ) {
```

این بار قبل از متغیر یوزر اپراتور NOT را داریم. اپراتور NOT یک علامت تعجب (!) می باشد و قبل از متغیری قرار می گیرد که سعی در امتحان کردن آن دارید. این اپراتور negation را امتحان می کند که به معنای امتحان کردن متضاد مقدار حقیقی می باشد. از آنجایی که متغیر یوزر بر روی true تنظیم شده است، user!مقادیر falseرا امتحان خواهد کرد. اگر یوزر بر روی false تنظیم شده بود، user!مقادیر true را امتحان می کرد.

اینطور فکر کنید: اگر چیزی true نیست پس چه می تواند باشد؟ یا اگر false نیست پس چیست؟

در بخش بعدی به بررسی Java Switch Statements خواهیم پرداخت.

آموزش Switch در جاوا

روش دیگر برای کنترل جریان برنامه های شما switch statement می باشد. یک switch statementبه شما گزینه ی تست دامنه ای از مقادیر را برای متغیرهایتان ارائه می دهد. این عبارت می تواند به جای عبارت بلند و پیچیده ی else if ... ifاستفاده شود. ساختر عبارت switch مانند زیر می باشد:

```
switch ( variable_to_test ) {

case value:
code_here;
break;

case value:

code_here;

break;

default:

values_not_caught_above;



}
```

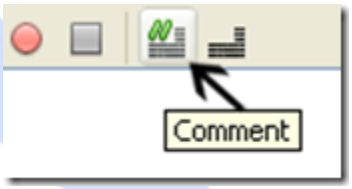
بنابراین شما با لغت switch شروع کرده اید که با یک جفت آکولاد دنبال می شود. تمام دیگر بخش های این عبارت در بین آکولادها قرار می گیرند. متغیری که می خواهید چک کنید بین آکولادهای switch قرار می گیرد. بنابراین شما یک جفت آکولاد دارید. بخش های دیگر عبارت switch نیز بین دو آکولاد قرار می گیرد. برای هر مقداری که می خواهید چک کنید نیاز به لغت case دارید. سپس مقداری را دارید که قصد بررسی آن را دارید:

case value

پس از case value: دو نقطه قرار می گیرد. سپس اگر مقدار هماهنگ باشد، آن چیزی را قرار می دهید که تمایل دارید اتفاق بیفتد. این کد شماست که می خواهید اجرا شود. برای گریز از هر مورد در عبارتswitch ، لغت break لازم است.

مقدار پیش فرض در انتها انتخابی می باشد. اگر مقادیر دیگری وجود دارند که می توانند در متغیر شما قرار بگیرند اما شما برای جای دیگری در عبارت switchامتحان نکرده اید، این مقادیر می توانند شامل شوند.

اگر همه ی آن گیج کننده می با شد در اینجا کد دیگری را مشاهده می کنید .می توانید برای این یا یک پروژه ی جدید ایجاد کنید و یا اینکه فقط روی کدی که دارید کامنت بدهید. یک راه سریع برای کامنت گذاشتن بر روی کد در NetBeansاز نوار ابزار در بالا می باشد. ابتدا کدی را که می خواهید کامنت بگذارید مشخص کنید. سپس روی آیکن کامنت کلیک کنید:



اما در اینجا کد آن را مشاهده می کنید:

```
public static void main(String[] args) {

    int user = 18;

    switch ( user ) {
        case 18:
            System.out.println("You're 18");
            break;
      case 19:
            System.out.println("You're 19");
            break;
      case 20:
            System.out.println("You're 20");
            break;
      default:
          System.out.println("You're not 18, 19 or 20");
    }

}
```

اولیین کاری که کد انجام می دهد تنظیم مقداری برای امتحان کردن می باشد .مجددا ما یک متغیر صحیح را تنظیم کرده و آن را user می نامیم. مقدار را بر روی 18 تنظیم کرده ایم. عبارت switch متغیر یوزر را چک کرده و آنچه در آن است را مشاهده خواهد کرد. سپس از طریق هر یک از عبارات مورد وارد خواهد شد .وقتی موردی

را پیدا می کند که هماهنگ می باشد، متوقف خواهد شد و کد را برای آن مورد اجرا خواهد کرد. سپس از عبارت switchگریز خواهد داشت.

برنامه را امتحان کنید. مقادیر مختلفی برای متغیر یوزر وارد کرده و نتیجه را مشاهده کنید.

متاسفانه نمی توانید برای دامنه ای از مقادیر بعد از مورد امتحان کنید و تنها برای یک مقدار میسر می باشد.

بنابراین نمی توانید این کار را انجام دهید:

```
case (user <= 18):
```

اما این کار را می توانید انجام دهید:

```
case 1: case 2: case 3: case 4:
```

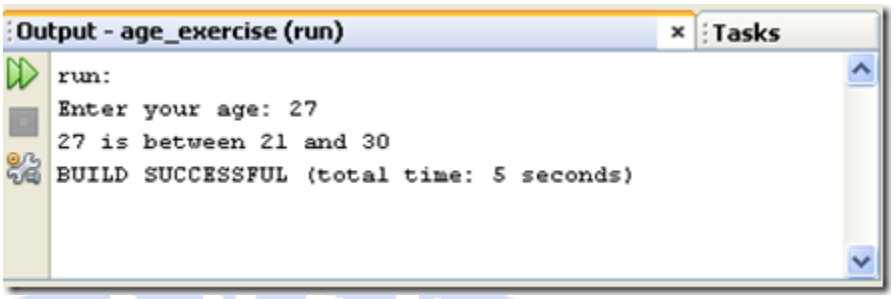
بنابراین خط بالا دامنه ی مقادیر را از 1 تا 4 تست می کند. اما شما باید هر مقدار را "spell out" کنید. (دقت داشته باشید که همه ی موردها و colon ها (دو نقطه) کجا قرار می گیرند(.

در اتمام این بخش در مورد منطق شرطی، تمرین های زیر را امتحان کنید:

برنامه ای بنویسید که ورودی یوزر را از console قبول می کند. برنامه ها باید یک عدد گرفته و سپس محدوده های سنی زیر را امتحان کنید: 0 تا 10، 11 تا 20، 21 تا 30، 30 و بیشتر. یک پیغام در پنجره ی Outputبه فرمت زیر نمایش دهید:

```
user_age + " is between 21 and 30"
```

بنابراین اگر یوزر 27 را به عنوان سن وارد کند، پنجره ی Output مانند زیر خواهد بود:



اگر یوزر 30 یا بیشتر از 30 باشد، فقط می توانید پیغام زیر را نمایش دهید:

```
"Your are 30 or over"
```

کمک برای این تمرین:

برای دریافت مقادیر رشته از یوزر، این کار را انجام دادید:

```
String age = user_input.next( );
```

اما متد ( )next برای رشته ها استفاده می شود. سنی که از یوزر دریافت می کنید باید یک عدد صحیح باشد، بنابراین نمی توانید از ( )next استفاده کنید.

اگر می خواهید چک کنید که یک رشته مشابه رشته ی دیگر می باشد، می توانید از متودی به نام equals استفاده کنید.

```
String user_name = "Bill";

if ( user_name.equals( "Bill" ) ) {
//DO SOMETHING HERE
}
```

در کد بالا یک متغیر String را تنظیم کرده و آن را user_name نامیده ایم. سپس مقدار "Bill" به آن اختصاص داده ایم. بین آکولادهای IF ، مجددا نام متغیر داریم که با یک نقطه دنبال می شود. پس از این نقطه

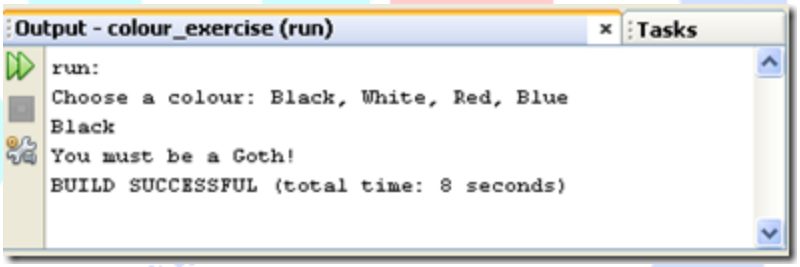
لغت "equals" قرار می گیرد. بین یک جفت آکولاد دیگر رشته ای را تایپ می کنید که سعی در امتحان آن دارید.

نکته: در هنگام بررسی اگر یک رشته همانند رشته ی دیگر می باشد، این دو رشته باید دقیقا هماهنگ باشند. بنابراین "Bill"با "bill"متفاوت می باشد . مورد اول دارای B بزرگ و مورد دوم دارای b کوچک می باشد.

برای این تمرین برنامه ای بنویسید که از یک یوزر می خواهد تا از بین چهار رنگ انتخاب کند: سیاه، سفید، قرمز یا آبی. بسته به رنگ انتخاب شده، از عبارت های IF ... ELSE IF برای نمایش یکی از پیغام های زیراستفاده کنید:

"BLACKشما باید یک Goth باشید" "WHITE".شما یک فرد پاک هستید" "RED".شما یک شخص سرگرم کننده و برون گرا هستید" "BLUE".شما طرفدار چلسی هستید، اینطور نیست؟"

وقتی برنامه ی شما به پایان می رسد، پنجره ی Output باید شبیه به تصویر زیر باشد:



بسیار خوب، اجازه بدهید ادامه داده و نگاهی به loop داشته باشیم. در بخش بعد سرعت را کمی بالا می بریم.

آموزش حلقه for در جاوا

همانطور که قبلا ذکر کردیم، برنامه نویسی که اکنون در حال انجام آن هستید یک برنامه نویسی پی در پی می باشد. این به این معناست که جریان صعودی می باشد، از بالا به پایین، با هر خط از کد که اجرا شده است، مگر اینکه شما درخواست دیگری از جاوا بکنید.

در بخش قبل مشاهده کردید که یک روش برای اینکه به جاوا بگویید هر خط از کد را اجرا نکند، استفاده از عبارت IF برای بخش های خاموش کد می باشد.

راه دیگر برای به هم ریختن جریان از بالا به پایین، استفاده از loop ها می باشد. یک برنامه نویسی loop آن برنامه نویسی می باشد که مجددا به عقب بازمی گردد. اگر بازگشت مجدد به عقب اجبار باشد، شما می توانید خطوط را به طور مکرر اجرا کنید.

به عنوان یک مثال فرض کنید که می خواهید اعداد از یک تا 10 را با هم جمع کنید. شما این کار را در جاوا به راحتی می توانید انجام دهید، مانند زیر:

```
int addition = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;
```

اما اگر بخواهید اعداد را از یک تا 1000 با هم جمع کنید، واقعا نمی خواهید از این متود استفاده کنید. در عوض می توانید از loop استفاده کنید تا مکررا وارد خطوط کد شوید تا اینکه به 1000 برسید. سپس می توانید از loop خارج شده و به راه خود ادامه دهید.

Java For Loops

با For Loops یکی از متداول ترین انواع loop ها آغاز خواهیم کرد. به نظر می رسد قسمت For از For Loops معنای خود را از دست داده باشد. اما می توانید آن را به این شکل در نظر بگیرید loop :برای یک مجموعه عدد از دفعات .ساختار For Loop مانند زیر می باشد:

```
for ( start_value; end_value; increment_number ) {
//YOUR_CODE_HERE
}
```

بنابراین بعد از لغت for با حروف کوچک، یک جفت آکولاد دارید. در داخل این آکولادها سه چیز لازم می باشد: مقدار آغازین برایloop ، مقدار پایانی برایloop ، و روشی برای به دست آوردن یک تعداد به تعداد دیگر. این فرایند افزایش عدد (increment number) نامیده می شود و معمولا 1 می باشد. اما لازم نیست اینطور باشد. اگر تمایل داشته باشید می توانید تا تعداد 10 نیز بالا بروید.

بعد از پرانتزها نیاز به یک جفت آکولاد دارید. آکولادها برای جداسازی آن بخشی از کد استفاده می شوند که می خواهید به طور مکرر اجرا شود. یک مثال می تواند این مورد را شفاف سازد.

برای این مورد یک پروژه ی جدید آغاز کنید. پروژه و گروه را می توانید به دلخواه خود نامگذاری کنید. (ما پروژه ی خود را "loops" و گروه را "ForLoops"نامیدیم.) اکنون کد زیر را اضافه کنید:

```
package loops;


public class ForLoops {

    public static void main(String[] args) {

        int loopVal;
        int end_value = 11;

        for (loopVal = 0; loopVal < end_value; loopVal++) {

            System.out.println("Loop Value = " + loopVal);
        }
    }

}
```

ما کار را با تنظیم یک مقدار صحیح آغاز کردیم که loopVal نامیده ایم. خط بعدی یک متغیر صحیح دیگر را تنظیم می کند. این متغیر برای مقدار نهایی loopاستفاده می شود و به 11 تنظیم شده است. آنچه ما تصمیم داریم انجام دهیم چاپ اعداد از 0 تا 10 می باشد.

در داخل پرانتزهای for loop خط زیر را مشاهده می کنید:

```
loopVal =0; loopVal < end_value; loopVal++
```

اولین قسمت به جاوا می گوید که در چه شماره ای looping را آغاز کنید. در اینجا ما یک مقدار صفر را به متغیر loopVal اختصاص می دهیم. این اولین عدد در loop می باشد. بخش بعد از برخی منطق های شرطی استفاده می کند:

```
loopVal < end_value
```

این عبارت می گوید که loopVal کمتر از end_value باشد for loop .به چرخش خود ادامه خواهد داد در حالیکه مقدار داخل متغیر loopVal کمتر از متغیری به نام end_value می باشد. تا زمانیکه کمتر بودن loopVal از end_value درست باشد، جاوا looping را روی کد در بین آکولادها حفظ خواهد کرد.

بخش نهایی بین پرانتزهای for loop مانند زیر می باشد:

loopVal++

آنچه در اینجا انجام می دهیم این است که به جاوا می گوییم چگونه از مقدار آغازین در loopVal به عدد بعدی در ترتیب حرکت کند. می خواهیم از 0 تا 10بشمریم. عدد بعد از 0 عدد 1 می باشد loopVal++ .راه کوتاه بیان عبارت "افزودن 1 به مقدار در متغیر" می باشد.

به جای بیان loopVal++ می توانیم عبارت زیر را نیز بنویسیم:

```
loopVal = loopVal + 1
```

در سمت راست علامت تساوی loopVal + 1 را داریم. بنابراین جاوا 1 را به هر چیزی که در متغیر loopVal ذخیره شده، اضافه می کند. زمانی که 1 را به مقدار اضافه کرده است، نتیجه را در داخل متغیر در سمت چپ تساوی ذخیره خواهد کرد. این مجددا متغیر loopVal می باشد. نتیجه این است که افزوده شدن 1 به loopValادامه می یابد. این فرایند افزایش متغیر نامیده می شود. نماد تندنویسی متغیر ++ بسیار متداول می باشد:

```
int some_number = 0;
some_number++;
```

وقتی که کد بالا اجرا شود، مقدار some_number مقدار 1 خواهد بود. این راه کوتاه بیان زیر می باشد:

```
int some_number = 0;
some_number = some_number + 1;
```

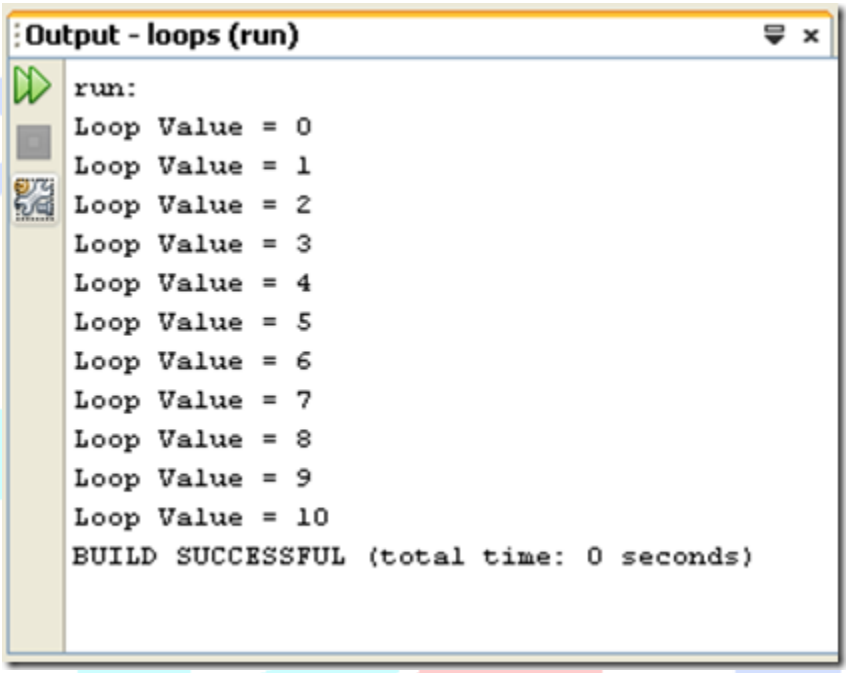
برای پوشاندن آن، for loopدر حال بیان مورد زیر می باشد:

Loop Start value: 0 Keep Looping While مقدار آغازین کمتر از 11 می باشد How to advance to the end value : افزودن 1 را به مقدار نهایی حفظ می کند.

در داخل پرانتزهای for loop عبارت زیر دیده می شود:

```
System.out.println("Loop Value = " + loopVal);
```

هر چیزی که در داخل متغیر loopValامی باشد، همراه با متن چاپ خواهد شد:



بنابراین برنامه را در یک loop گیر انداخته ایم و آن را مجبور کرده ایم به طور مکرر اجرا شود. با هر اجرای loop مقدار 1 به loopValاضافه می شود loop .به چرخش خود ادامه خواهد داد، در حالیکه مقدار در داخل loopValکمتر از مقدار در داخل end_value می باشد. هر چیزی که در داخل آکولادهای loop می باشد کدی است که پشت سر هم اجرا خواهد شد. و این تمام نکته ی loop می باشد: اجرای کد آکولاد به طور پیوسته.

ئر اینجا کدی را می بینید که اعداد را از 1 تا 10 به هم اضافه می کند. آن را امتحان کنید:

```
public static void main(String[] args) {

    int loopVal;
    int end_value = 11;
    int addition = 0;

    for (loopVal = 1; loopVal < end_value; loopVal++) {

        addition = addition + loopVal;
    }

    System.out.println("Total = " + addition);
}
```

پاسخی که در پنجره ی Output دریافت می کنید 55 می باشد. کد نیز کم و بیش همان کد مربوط به loop قبلی می باشد. ما همان دو متغیر loopVal و end_valueرا تنظیم شده در بالا داریم. همچنین یک متغیر صحیح سومی هم داریم که addition نامیده می شود. این مقدار مجموع 1 تا 10 را حفظ خواهد کرد.

بین پرانتزهای for loop تقریبا همان مقدار آخرین بار می باشد: ما در حال loopingهستیم در حالیکه loopValکمتر از end_value می باشد؛ و در حال افزودن 1 به متغیر loopVal می باشیم. تنها تفاوت این است که اکنون مقدار آغازین 1 می باشد.

بین آکولادها فقط یک خط کد وجود دارد:

```
addition = addition + loopVal;
```

این خط مجزا اعداد 1 تا 10 را با هم جمع می کند. اگر در مورد چگونگی کار کردن آن گیج شده اید، از سمت راست علامت تساوی آغاز کنید:

```
addition + loopVal;
```

اولین اجرایloop ، متغیر addition مقدار 0 را در خود حفظ می کند. در این حین متغیر loopVal مقدار 1 را در خود دارد. جاوا 0 را به 1 اضافه خواهد کرد. سپس نتیجه را در متغیر سمت چپ علامت تساوی ذخیره خواهد کرد. مجددا این متغیر addition (مجموع) می باشد. هرآنچه از قبل در متغیر addition حفظ شد (0)، پاک خواهد شد و با یک مقدار جدید (1) جایگزین می شود.

در دومین اجرایloop ، مقادیر در دو متغیر عبارتند از:

```
addition (1) + loopVal (2);
```

واضح است که پاسخ 1+2 عدد 3 می شود. بنابراین 3 مقدار جدیدی است که در سمت چپ تساوی ذخیره خواهد شد.

در سومین اجرای loop مقادیر جدید عبارتند از:

```
addition (3) + loopVal (3);
```

جاوا 3 را با 3 جمع خواهد کرد و 6 را در سمت چپ تساوی ها ذخیره می کند .این چرخش ادامه می یابد تا اینکه loopبه پایان برسد. نتیجه ی نهایی 55 می باشد.

(دقت داشته باشید که بعد از آخرین آکولاد ازloop ، خط چاپی ما خارج از for loopمی باشد .)

تمرین

کد خود را طوری تغییر دهید که جاوا اعداد را از 1 تا 100 اضافه کند. پاسخی که در یافت می کنید 5050 می باشد .تمرین :یک برنامه ی جدول زمانبندی بنویسید. برنامه باید از یوزر بخواهد تا یک عدد را وارد کند. سپس این عدد به عنوان دفعات مربوط به جدول استفاده خواهد شد. بنابراین اگر یوزر عدد 10 را وارد کند، جدول 10 مرتبه باید نمایش داده شود .

کمک برای این تمرین

for loop شما فقط به دو خط کد بین آکولادها نیاز دارد و یکی از آنها خط چاپ (print line) می باشد. شما تنها به یک خط مجزا نیاز دارید تا پاسخ های جدول خود را محاسبه کنید .تقریبا باید چگونگی گرفتن عدد از یوزر را بدانید. این کار در آکولادهای loop استفاده می شود تا پاسخ به ضرب شما را انجام دهد .جول شما تنها نیاز به رفتن از 1 تا 10 را دارد .تمرین از for loopبرای چاپ اعداد فرد از 1 تا 10 استفاده کنید. (برای انجام آسان این تمرین در مورد افزایش مقدار loop فکر کنید که سومین آیتم بین پرانتزها می باشد (.یکی از راه های سخت انجام تمرین بالا استفاده از اپراتوری است که هنوز با آن آشنایی ندارید – اپراتور .modulus این اپراتور

زمانی استفاده می شود که یک عدد را تقسم کرده و باقیمانده را نگاه می دارید. بنابراین 10 Mod 3 عدد 1 می باشد، زیرا 10 تقسیم بر 3 می شود 3. باقیمانده نیز 1 می باشد که آن را حفظ می کنید. اپراتور Modulus در جاوا نماد درصد می باشد که تا حدی گیج کننده است. مانند زیر:

```
int remainder;
int total = 10
remainder = total %3
```

بنابراین عددی را که می خواهید تقسیم کنید در ابتدا قرار می گیرد. سپس یک نماد درصد تایپ می کنید که پس از آن تقسیم کننده قرار می گیرد. پاسخ نیز باقیمانده می باشد.

در تمرین بالا می توانستید از 2 به عنوان عدد Mod استفاده کنید و سپس از عبارت IF در for loop برای امتحان باقیمانده استفاده کنید. (آیا متوجه می شوید که چرا 2 باید عدد Mod باشد؟)

در بخش بعد نگاهی به while loops خواهیم داشت.

آموزش حلقه while در جاوا

نوع دیگری از loop که می توانید در جاوا استفاده کنید While loop می باشد. درک While loop ها بسیار راحت تر از for loop می باشد. در اینجا مشاهده می کنید که چگونه به نظر می رسند:

```
while ( condition ) {

}
```

بنابراین با لغت while با حروف کوچک، آغاز می کنید. شرایطی را که می خواهید امتحان کنید بین پرانتز قرار می گیرد. یک جفت آکولاد نیز پس از آن قرار می گیرد و کدی را که می خواهید اجرا کنید بین آکولادها قرار می گیرد .به عنوان مثال در اینجا یک while loop را مشاهده می کنید که متنی را چاپ می کند:

```
int loopVal = 0;
while ( loopVal < 5) {
System.out.println("Printing Some Text");
loopVal++;
}
```

شرایط مورد امتحان بین پرانتزها قرار گرفته است. ما می خواهیم در حالیکه متغیری به نام loopVal کمتر از 5 می باشد، loopingرا ادامه دهیم. کد ما ابتدا در داخل آکولادها یک خط از متن را چاپ می کند. سپس نیاز به افزایش متغیر loopVal داریم. اگر این کار را انجام ندهیم یک loop نامحدود خواهیم داشت، همانطور که راهی برای loopVal وجود ندارد تا بیشتر از مقدار اصلی خود از 0 شود.

گرچه ما از یک شمارنده استفاده کرده ایم تا به شرایط نهایی برسیم، در حالیکه loop ها به بهترین شکل استفاده می شوند وقتی که شما واقعا نیازی به محاسبه ی مقدار ندارید، اما در عوض فقط در حال بررسی مقدار می باشد. برای مثال هنگامی که هیچ کدام از کلیدهای صفحه کلید استفاده نمی شوند، می توانید loopingرا ادامه دهید. این مورد در بازیها متداول می باشد. برای خروج از while loopو همچنین خود بازی، می توان از حرف "X" استفاده کرد. مثال دیگر loopingدور یک فایل متنی می باشد، در حالیکه هنوز انتهای فایل نرسیده است.

Do … While

do ... while loopدر ارتباط با while loop می باشد. این مورد مانند زیر می باشد:

```
int loopVal = 0;
do {
System.out.println("Printing Some Text");
loopVal++;
}
while ( loopVal < 5 );
```

مجددا جاوا آنقدر loop را می چرخاند تا به شرایط نهایی برسد. این بار بخش "while" در قسمت پایین قرار دارد. اما شرایط همان است looping – انجام شود، در حالیکه مقدار داخل متغیر loopVal کمتر از می باشد. تفاوت بین این دو کد بین آکولادهای do ... while است که نهایتا یک بار اجرا خواهد شد. با while loop شرایط حاصل خواهد شد. سپس جاوا از loop شما آزاد خواهد شد و حتی کد داخل آکولاد را اجرا نخواهد کرد. برای امتحان این مورد، ابتدا while loop را امتحان کنید. مقدار متغیر loopVal را به تغییر داده و سپس کد را اجرا کنید. باید متوجه شوید که متن چاپ نشده است. اکنون انجام loop را با مقدار5 برای loopVal امتحان کنید. متن یک بار چاپ خواهد شد و سپس جاوا از loopآزاد خواهد شد.

بسیار خوب، در اینجا looping را رها می کنیم. این موضوعی می باشد که نیاز است بیشتر به آن بپردازید. اما اگر هنوز در این موضوع کامل نیستید، نگران نباشید – در ادامه بیشتر فرا خواهید گرفت. در بخش بعدی نگاهی به موردی به نام array خواهیم داشت.

آموزش آرایه ها در جاوا

اگر می خواهید که به طور موثر کد گذاری کنید، مفهومی که در برنامه نویسی باید به آن عادت کنید array می باشد. در این بخش در مورد array ها و چگونگی استفاده از آنها فراخواهید گرفت.

یک Array چیست؟

تاکنون با متغیرهایی کار کردید که فقط یک مقدار را در خود حفظ می کنند . متغیرهای صحیح که تنظیم کرده اید، فقط یک عدد را حفظ می کنند و متغیرهای رشته نیز فقط یک رشته از متن را در خود دارند. یک array (ردیف) روشی برای حفظ بیشتر از یک مقدار در یک زمان می باشد که در واقع شبیه لیستی از آیتم ها می باشد. یک array را می توان به عنوان ستون هایی در یک صفحه ی گسترده در نظر گرفت. شما می توانید یک صفحه ی گسترئه تنها با یک ستون و یا تعداد زیادی ستون داشته باشید. داده ای که در یک ردیف مجزا حفظ می شود می تواند مشابه تصویر زیر باشد:

	Array_Values
0	10
1	14
2	36
3	27
4	43
5	18

مانند یک صفحه ی گسترده، arrayها برای هر ردیف دارای یک موقعیت عددی می باشند. این موقعیت ها در یک array از شروع شده و متداولا افزایش می یابند. هر موقعیت نیز در یک array می تواند یک مقدار در خود حفظ کند. در تصویر بالا موقعیت 0دارای مقدار 10و موقعیت 1 دارای 14، موقیت 2 دارای مقدار 36 و غیره می باشد.

برای برقراری یک array از اعداد، مشابه تصویر بالا، باید به جاوا اعلام کنید که چه نوع داده ای قرار است وارد arrayشود (اعداد صحیح، رشته ها، مقادیر Boolean و غیره). سپس لازم است که اعلام کنید که array دارای چند موقعین می باشد:

```
int[ ] aryNums;
```

تنها تفاوت بین برقراری یک متغیر عدد صحیح طبیعی و یک array یک جفت کروشه ای است که بعد از نوع داده قرار می گیرد. کروشه ها برای اعلام به جاوا در مورد این که شما قصد برقراری یک array را دارید، کافی هستند. نام arrayمربوط به تصویر بالا aryNums می باشد. درست مانند متغیرهای طبیعی، می توانید آنها را هر چیزی که می خواهید نام گذاری کنید (با همان استثنائاتی که قبلا ذکر کردیم(.

اما این مورد فقط به جاوا می گوید که شما قصد تنظیم یک array عدد صحیح را دارید. این برنامه تعداد موقعیت هایی را که array باید حفظ کند، اعلام نمی کند. برای انجام این کار باید یک array object جدید تنظیم کنید:

```
aryNums = new int[6];
```

شما با نام array شروع کرده اید که با علامت تساوی دنبال می شود. پس از علامت تساوی به لغت کلیدی newو سپس مجددا نوع داده ی خود، نیاز دارید.پس از نوع داده یک جفت کروشه قرار می گیرد. بین کروشه ها نیاز به اندازه ی arrayدارید. اندازه در واقع تعداد موقعیت هایی است که array باید حفظ کند.

اگر تمایل دارید، می توانید همه ی آنها را روی یک قرار دهید:

```
int[ ] aryNums = new int[6];
```

بنابراین ما به جاوا اعلام می کنیم که یک array را با 6 موقعیت در آن تنظیم کند. پس از اجرای این خط، جاوا مقادیر پیش فرض را برای array اختصاص خواهد داد. از آنجایی که ما یک array مقدار صحیح تنظیم کرده ایم، مقادیر پیش فرض برای همه ی 6 موقعیت 0 خواهد بود.

برای اینکه در یک array مقادیر را به موقعیت های مختلف اختصاص دهید، این کار را به یک روش طبیعی انجام می دهید:

```
aryNums[0] = 10;
```

در اینجا مقدار 10 به موقعیت 0 در یک array به نام aryNums اختصاص داده می شود. مجددا کروشه ها برای اشاره به هر موقعیت استفاده می شوند. اگر بخواهید مقدار 14 را به یک array با موقعیت 1 اختصاص دهید، کد مربوط مانند زیر خواهید بود:

```
aryNums[1] = 14;
```

و کد مربوط به مقدار 36 برای موقعیت 2 مانند زیر خواهد بود:

```
aryNums[2] = 36;
```

فراموش نکنید که از آنجایی که array ها از 0 شروع می شوند، سومین موقعیت در یک array دارای عدد شاخص 2 می باشد.

اگر بدانید که چه مقادیری قرار است در یک array قرار گیرند، می توانید آنها را مانند زیر تنظیم کنید:

```
int[ ] aryNums = { 1, 2, 3, 4 };
```

این متد در حال تنظیم یک array با استفاده از کروشه ها بعد از علامت تساوی، می باشد. در بین کروشه ها مقادیری را تایپ می کنید که array خواهد گرفت. اولین مقدار در موقعیت 0 و دومین مقدار در موقعیت 1 و غیره خواهد بود. توجه داشته باشید که هنوز پس از int نیاز به کروشه دارید، اما نیازی به لغت کلیدی new یا تکرار نوع داده و یا کروشه ها ندارید. اما این فقط مربوط می شود به نوع داده های مقادیرint ، رشته و مقادیر char. به عبارت دیگر نیاز به لغت کلیدی new دارید. بنابراین می توانید این کار را انجام دهید:

```
String[ ] aryStrings = {"Autumn", "Spring", "Summer", "Winter" };
```

اما این کار را نمی توانید انجام دهید:

```
boolean[ ] aryBools = {false, true, false, true};
```

برای تنظیم یک boolean array هنوز نیاز به لغت کلیدی new دارید:

```
boolean[ ] aryBools = new boolean[ ] {false, true, false, true};
```

برای به دست آوردن مقادیر مربوط بهarray ، نام array را تایپ کنید که با موقعیت array در یک کروشه دنبال می‌شود. مانند زیر:

```
System.out.println( aryNums[2] );
```

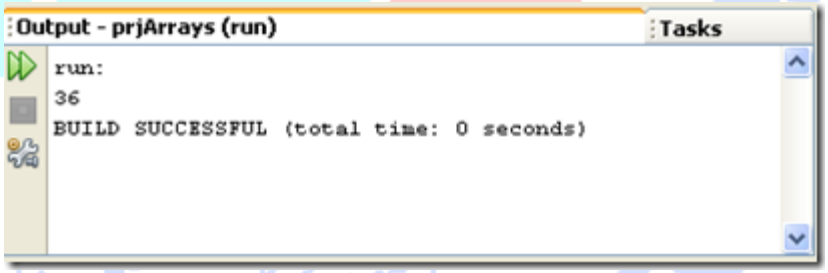
کد بالا هر مقداری که در ردیفی با موقعت 2 در یک array به نام aryNums باشد را چاپ می کند. اما اجازه بدهید کمی کدگذاذی تمرین کنیم.

یک پروژه ی جدید آغاز کرده و آن را به میل خود نامگذاری کنید. فراموش نکنید که نام گروه را به یک مورد واضح تغییر دهید.

کد زیر را در متود جدید Main تایپ کنید:

```
package prjarrays;

public class ArraysTest {

    public static void main(String[] args) {

        int[] aryNums;

        aryNums = new int[6];

        aryNums[0] = 10;
        aryNums[1] = 14;
        aryNums[2] = 36;
        aryNums[3] = 27;
        aryNums[4] = 43;
        aryNums[5] = 18;

        System.out.println( aryNums[2] );
    }

}
```

وقتی برنامه را اجرا می کنید باید تصویر زیر را در پنجره ی Output مشاهده کنید:



موقعیت عددی array را در print line از 2 به 5 تغییر دهید و در عوض 18 باید چاپ شود. در بخش بعدی نگاهی به چگونگی استفاده از array ها با loop ها خواهیم انداخت.

آموزش آرایه و حلقه در جاوا

Arrayها با loop ها در جای خود می آیند. در the previous section that مشاهده کردید که برای اختصاص دادن مقادیر به موقعیت های array از کد زیر استفاده می کردید:

```
aryNums[0] = 10;
```

اما اگر اعداد زیادی برای اختصاص دادن به یک array دارید، این کار اصلا عملی نمی باشد. به عنوان مثال یک برنامه ی قرعه کشی را در نظر بگیرید که باید اعداد 1 تا 49 را به موقعیت های یک array اختصاص دهد. به جای تایپ یک لیست طولانی از موقعیت ها و مقادیر array ، می توانید از یک loop استفاده کنید. در اینجا کدی را مشاهده می کنید که این کار را انجام می دهد:

```
package prjarrays;

public class ArraysTest {

    public static void main(String[] args) {

        int[] lottery_numbers = new int[49];
        int i;

        for (i=0; i < lottery_numbers.length; i++) {
            lottery_numbers[i] = i + 1;
            System.out.println( lottery_numbers[i] );
        }

    }
}
```

بنابراین یک array تنظیم می کنیم تا 49 مقدار صحیح را در خود حفظ کند. سپس کد loop قرار می گیرد. به شرط نهایی loop دقت داشته باشید.

```
i < lottery_numbers.length
```

Lengthیک پراپرتی از آبجکت های array می باشد که می توانید برای به دست آوردن اندازه ی array از آن استفاده کنید (تعداد موقعیت هایی که دارد.) .بنابراین، زمانی که مقدار در متغیر i کمتر از اندازه ی array می باشد، این loopبه چرخش متداول خود ادامه خواهد داد.

برای اختصاص دادن مقادیر به موقعیت ها درarray ، می توان از خط زیر استفاده کرد:

```
lottery_numbers[i] = i + 1;
```

به جای مقدار hard-code بین کروشه های مربوط به نامarray ، متغیری به نام i داریم. به یاد داشته باشید که این متغیر هر بار با چرخش loop یک واحد افزایش می یابد. بنابراین هر موقعیت فقط با استفاده از مقدار loopقابل دسترسی می باشد. مقداری که به هر موقعیت اختصاص داده شده i + 1 می باشد .بنابراین مجددا مقدار افزایش یافته ی loop را داریم، این بار با یک واحد افزوده به آن. از آنجایی که مقدار loop از 0 شروع می شود، این برنامه اعداد 1 تا 49 را به شما ارائه خواهد داد.

خط دیگر در loop تنها مقداری را که در هر موقعیت قرار دارد، چاپ می کند) .اگر تمایل داشتید می توانید کدی بنویسید که اعداد را در یک array قرار دهید. زمانی که مقادیر را در یک ردیف قرار دادید می توانید از 6 عدد اول به عنوان شماره های قرعه کشی استفاده کنید. از کد دیگری استفاده کنید برای مقایسه ی 6 شماره ی یک یوزر با شماره های برنده، اکنون یک برنامه ی قرعه کشی در اختیار دارید(.

در بخش بعدی چگونگی مرتب سازی array ها را مشاهده خواهید کرد.

آموزش مرتب سازی آرایه ها در جاوا

متودهای داخلی دیگر جاوا به شما اجازه می دهند تا array های خود را مرتب سازید. برای استفاده از متود مرتب سازی array ها، ابتدا نیاز به مراجعه به یک کتابخانه ی جاوا به نام Arrays دارید. شما این کار را با عبارت import انجام می دهید. می توانید آن را با your aryNums programme امتحان کنید .عبارت importزیر را اضافه کنید:

```
package prjarrays;

import java.util.Arrays;

public class ArraysTest {

    public static void main(String[] args) {

        int[] aryNums;
        aryNums = new int[6];

        aryNums[0] = 10;
        aryNums[1] = 14;
        aryNums[2] = 36;
        aryNums[3] = 27;
        aryNums[4] = 43;
        aryNums[5] = 18;

    }

}
```

```
import java.util.Arrays;
```

کد شما باید مشابه کد ارائه شده ی ما در زیر باشد:

اکنون که کتابخانه ی Arrays را وارد کرده اید، می توانید از متود مرتب سازی استفاده کنید. این کار بسیار ساده می باشد.

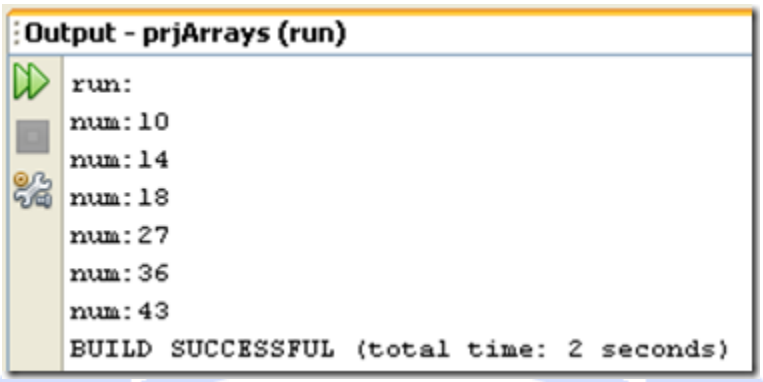
```
Arrays.sort( aryNums );
```

ابتدا لغت Array را تایپ کرده و سپس یک نقطه تایپ کنید. به محض اینکه نقطه را تایپ کردید، NetBeansلیستی از مواردی را نمایش خواهد داد که می توانید با array ها انجام دهید. لغت "sort" را تایپ کنید. سپس نام ردیف مورد نظر را که می خواهید مرتب کنید، بین یک جفت پرانتز قرار دهید. (توجه کنید که نیازی به کروشه بعد از نام array ندارید.)

و این برای مرتب سازی array کافی نمی باشد؛ در اینجا کدی را مشاهده می کنید که می توانید امتحان کنید:

```
package prjarrays;

import java.util.Arrays;

public class ArraysTest {

    public static void main(String[] args) {

        int[] aryNums;
        aryNums = new int[6];

        aryNums[0] = 10;
        aryNums[1] = 14;
        aryNums[2] = 36;
        aryNums[3] = 27;
        aryNums[4] = 43;
        aryNums[5] = 18;

        Arrays.sort(aryNums);

        int i;

        for (i=0; i < aryNums.length; i++) {
            System.out.println("num:" + aryNums[i]);
        }

    }
}
```

چرخش for loop در انتها ادامه پیدا می کند تا اینکه مقادیر را در هر موقعیت چاپ کند. وقتی کد اجرا می شود، خروجی مانند شکل زیر خواهد بود:



همانطور که مشاهده می کنید، arrayبا یک ترتیب صعودی مرتب شده است.

به هرحال مرتب سازی با ترتیب نزولی فقط با نوشتن کد مرتب سازی خودتان یا تغییر array به آبجکت های صحیح (Integer objects) و سپس وارد کردن از کتابخانه ی Collections امکان پذیر می باشد. اگر نیاز به یک مرتب سازی نزولی داشته باشید، در اینجا کدی را مشاهده می کنید که فقط این کار را انجام می دهد (اگر تمایل دارید می توانید از این کد رد شوید.):

آموزش آرایه ها و رشته ها در جاوا

شما می توانید رشته هایی از متن را در داخل array ها قرار دهید. این کار درست به روش اعداد صحیح انجام می شود:

```
String[ ] aryString = new String[5] ;

aryString[0] = "This";
aryString[1] = "is";
aryString[2] = "a";
aryString[3] = "string";
aryString[4] = "array";
```

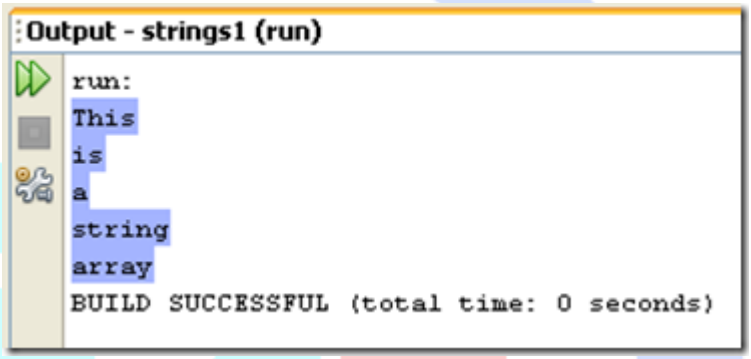
کد بالا یک رشته array را با 5 موقعیت تنظیم می کند. سپس متن به هر موقعیت در array اختصاص داده می شود.

در اینجا یک loop را مشاهده می کنید که در همه ی موقعیت های array قرار گرفته و هر چیزی را که در موقعیت ها وجود دارد، چاپ می کند:

```
int i;
for ( i=0; i < aryString.length; i++ ) {
System.out.println( aryString[i] );
}
```

هنگامی که مقدار متغیری بع نام I کمتر از طول array به نام aryString باشد، loopبه چرخش خود ادامه می دهد.

وقتی که برنامه ی بالا اجرا می شود، پنجره ی Output به شکل زیر خواهد بود:

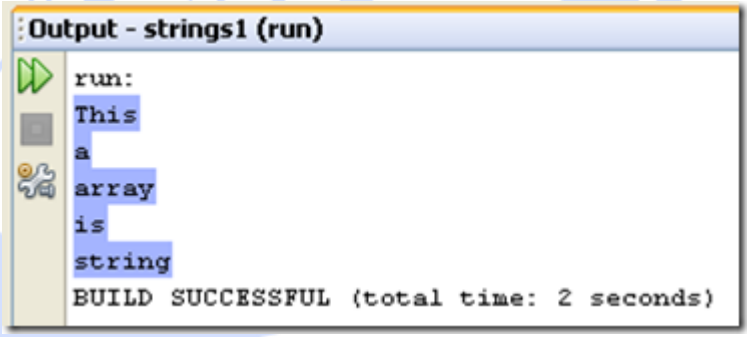


شما می توانید یک مرتب سازی روی رشته ی array ها اجرا کنید، درست مانند کاری که می توانید با اعداد صحیح انجام دهید. اما مرتب سازی یک ترتیب صعودی براساس حروف الفبا می باشد، به این معنا که aa قبل از ab قرار می گیرد. به هرحال جاوا برای مقایسه ی یک حرف در رشته ی شما با یک رشته ی دیگر از کاراکترهای یونیکد (Unicode) استفاده می کند. این به این معناست که حروف بزرگ قبل از حروف کوچک قرار می گیرند.

کد زیر را امتحان کنید:

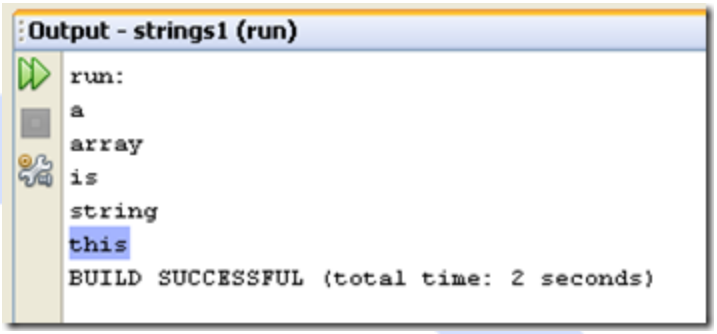
```
package strings1;
import java.util.Arrays;

public class StringArrays {

    public static void main(String[] args) {

        String[] aryString = new String[5] ;

        aryString[0] = "This";
        aryString[1] = "is";
        aryString[2] = "a";
        aryString[3] = "string";
        aryString[4] = "array";

        Arrays.sort(aryString);

        int i;
        for (i=0; i < aryString.length; i++){
            System.out.println( aryString[i] );
        }

    }
}
```

وقتی که برنامه اجرا می شود، پنجره ی Output تصویر زیر را نمایش خواهد داد:



گرچه ما array را مرتب کرده ایم، اما لغت This در ابتدا قرار می گیرد .اگر یک مرتب سازی الفبایی باشد، انتظار دارید که لغت a در ابتدا قرار بگیرد. این اتفاق می افتد اگر تمام حروف کوچک باشند. در کد برنامه نویسی

خود T بزرگ از لغت This را به t کوچک تغییر دهید. اکنون برنامه را مجددا اجرا کنید. پنجره ی Output به شکل زیر نمایش داده می شود:



همانطور که مشاهده می کنید، لغت this اکنون در پایین قرار دارد. در بخش بعد نگاه دقیق تری به رشته ها خواهیم داشت. بنابراین اکنون زیاد نگران این مورد نباشید. در عوض تمرین های زیر را بررسی کنید.

تمرین :یک array را تنظیم کنید تا مقادیر زیر را در خود داشته باشد .23, 6, 47, 35, 2, 14 :برنامه ای بنویسید تا میانگین این 6 عدد را به دست آورید. (می توانید ار اعداد صحیح برای این تمرین استفاده کنید که پاسخ شما را گرد خواهد کرد (.تمرین :با استفاده از مقادیر بالا، می توانید کاری کنید که برنامه ی شما بالاترین عدد را در array چاپ کند .تمرین :با استفاده از همان array در بالا می توانید کاری کنید که برنامه ی شما فقط اعداد فرد را چاپ کند .در بخش بعدی در مورد array های چند بعدی در جاوا فرا خواهید گرفت.

```
package prjarrays;

import java.util.Arrays;
import java.util.Collections;

public class ArraysTest {

    public static void main(String[] args) {

      int[] aryNums;
      aryNums = new int[6];

      aryNums[0] = 10; aryNums[1] = 14; aryNums[2] = 36;
      aryNums[3] = 27; aryNums[4] = 43; aryNums[5] = 18;

      //CREATE AN INTEGER OBJECT ARRAY
      Integer[] integerArray = new Integer[aryNums.length];

      //ASSIGN THE VALUES TO THE NEW ARRAY
      for (int i = 0; i < aryNums.length; i++) {
          integerArray[i] = new Integer(aryNums[i]);
      }

      //SORT DESCENDING
      Arrays.sort(integerArray,  Collections.reverseOrder() );

      //PRINT THE RESULTS
      for (int i =0; i < integerArray.length; i++) {
          System.out.println("num:" + integerArray[i]);
      }
    }
}
```

مطمئنا موافق هستید که این مورد کمی گیج کننده است . در بخش بعدی نگاهی به ردیف ها و رشته ها خواهیم داشت.

آموزش آرایه های چند بعدی در جاوا

Arrayهایی که تاکنون استفاده کرده اید تنها یک ستون داده دارند. اما می توانید یک array برای نگهداری بیشتر از یک ستون تنظیم کنید. این array ها چند بعدی نامیده می شوند. به عنوان مثال یک صفحه ی

گسترده را با ردیف ها و ستون ها در نظر بگیرید. اگر 6 ردیف و 5 ستون داشته باشید، صفحه ی گسترده ی شما می تواند 30 عدد را در خود داشته باشد، که ممکن است مانند تصویر زیر به نظر برسد:

	A	B	C	D	E
0	10	12	43	11	22
1	20	45	56	1	33
2	30	67	32	14	44
3	40	12	87	14	55
4	50	86	66	13	66
5	60	53	44	12	11

یک array چند بعدی موردی است که می تواند همه ی مقادیر بالا را در خود داشته باشد، که آنها را مانند زیر تنظیم کرده اید:

```
int[ ][ ] aryNumbers = new int[6][5];
```

این array به روش تنظیم یک array عادی تنظیم می شود، به جز اینکه در این مورد دو مجموعه کروشه دارید. اولین مجموعه از کروشه ها برای ردیف ها می باشد و دومین مجموعه نیز برای ستون ها می باشد. در خط بالا به جاوا اعلام کرده ایم که یک array با 6 ردیف و 5 ستون تنظیم کند. برای حفظ مقادیر در یک arrayچند بعدی، باید مراقب دنبال کردن ردیف ها و ستون ها باشید. در اینجا کدهایی را برای پر کردن اولین ردیف های اعداد از تصویر صفحه ی گسترده ی ما مشاهده می کنید:

```
aryNumbers[0][0] = 10;
aryNumbers[0][1] = 12;
aryNumbers[0][2] = 43;
aryNumbers[0][3] = 11;
aryNumbers[0][4] = 22;
```

بنابراین اولین ردیف، ردیف 0 می باشد. سپس ستون ها از 0 تا 4 می باشند که 5 آیتم هستند. برای پر کردن دومین ردیف کد زیر لازم می باشد:

```
aryNumbers[1][0] = 20;
aryNumbers[1][1] = 45;
aryNumbers[1][2] = 56;
aryNumbers[1][3] = 1;
aryNumbers[1][4] = 33;
```

اعداد ستون همان است، اما اعداد مربوط به ردیف همه 1 نی باشند.

تکنیک مورد نیاز برای دسترسی به همه ی آیتم ها در یک array چند بعدی، استفاده از یک loop داخل یکی دیگر می باشد. در اینجا کدی را مشاهده می کنید برای دسترسی به همه ی اعداد در بالا. این برنامه از یک loopبرای double استفاده می کند:

```
int rows = 6;
int columns = 5;

int i, j;

for (i=0; i < rows ; i++) {

    for (j=0; j < columns ; j++) {
        System.out.print( aryNumbers[i][j] + " " );
    }
    System.out.println( "" );
}
```

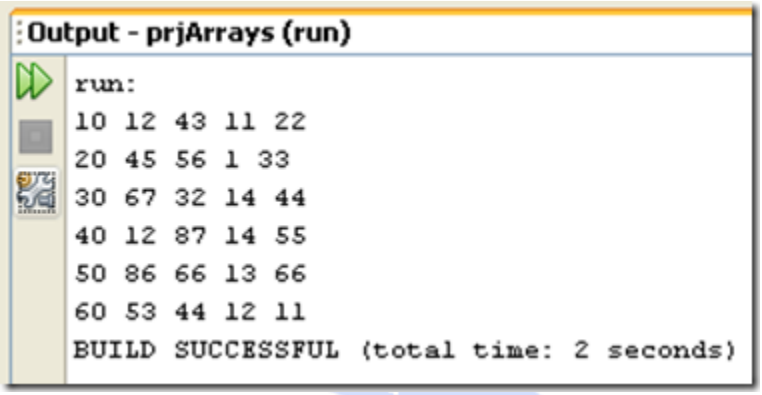
اولین for loop برای ردیف ها استفاده می شود و دومین for loop برای ستون ها. در اولین چرخش از اولین loop، مقدار متغیر i عدد 0 خوهد شد. کد داخل for loopیک loop دیگر می باشد. تمام این loop دوم هنگامی که مقدار متغیر i عدد 0 باشد، اجرا خواهد شد for loop . دوم از یک متغیر به نام j استفاده خواهد کرد. متغیرهای i و j برای دسترسی به array قابل استفاده می باشند.

```
aryNumbers[ i ][ j ]
```

بنابراین سیستم دو loop برای وارد کردن همه ی مقادیر به یک array چند بعدی استفاده می شود، ردیف به ردیف.

تمرین

برنامه ی فوق در جایی به پایان می رسد که در آن در حال نوشتن برنامه ای برای چاپ تمام مقادیر از صفحه ی گسترده هستیم. وقتی که این کار انجام می شود، پنجره ی Output چیزی شبیه به تصویر زیر خواهد بود:



Arrayهای چندبعدی می توانند گول زننده نیز باشند، اساسا به این دلیل که حفظ مسیر ستون ها و ردیف های شما سخت می باشد! در بخش بعدی در مورد لیست های array فرا خواهید گرفت.

آموزش استفاده از Array Lists در جاوا

اگر از تعداد آیتم هایی که قرار است در array قرار بگیرند، اطلاعی ندارید، ممکن ست بهتر باشد از چیزی به نام ArrayList استفاده کنید. یک ArrayListدر واقع یک ساختار دینامیک داده می باشد، به این معنا که آیتم ها قابل حذف و اضافه از لیست می باشند. یک array معمولی در جاوا یک ساختار استاتیک داده می باشد، زیرا شما اندازه ی اولیه ی array خود را دارید.

برای تنظیم یکArrayList ، ابتدا باید پوشه ای از java.util library وارد کنید:

```
import java.util.ArrayList;
```

سپس می توانید یک آبجکت ArrayList جدید ایجاد کنید:

```
ArrayList listTest = new ArrayList( );
```

توجه کنید که این بار نیازی به کروشه ندارید.

زمانی که آبجکت ArrayList جدید داشته باشید، می توانید با متودadd ، عناصری به آن اضافه کنید:

```
listTest.add( "first item" );
listTest.add( "second item" );
```

```
listTest.add( "third item" );
listTest.add( 7 );
```

بین پرانتزهای add آنچه را که می خواهید به ArrayList اضافه کنید، قرار دهید. به هرحال فقط می توانید آبجکت ها را اضافه کنید. سه آیتم اولی که به لیست بالا اضافه کرده ایم، آبجکت های String می باشند. چهارمین آیتم یک عدد می باشد. اما این عدد یک آبجکت از نوع صحیح خواهد بود.

آیتم ها در لیست می توانند توسط یک عدد Index و با استفاده از یک متود get مورد اشاره قرار بگیرند:

```
listTest.get( 3 )
```

این خط آیتم را روی لیست در Index position 3 خواهد گرفت. اعداد ایندکس شمارش را از 0 شروع می کنند، بنابراین این آیتم چهارمین آیتم خواهد بود.

شما می توانید آیتم ها را از یک ArrayList حذف هم بکنید:

```
listTest.remove(2);
```

یا می توانید از مقدار روی لیست استفاده کنید:

```
listTest.remove( "second item" );
```

حذف یک آیتم باعث تغییر اندازه ی ArrayList خواهد شد، بنابراین وقتی از عدد شاخص استفاده می کنید، وقتی سعی در به دست آوردن یک آیتم روی لیست دارید، باید مراقب باشید. اگر آیتم شماره ی 2 را حذف کرده ایم، بنابراین لیست فوق فقط حاوی 3 آیتم خواهد بود. سعی در به دست آوردن با عدد شاخص 3، منجر به بروز خطا خواهد شد.

برای وارد شدن به هر آیتم درArrayList ، می توانید چیزی به نام یک Iteratorرا تنظیم کنید. ایک گروه در کتابخانه ی java.util نیز یافت می شود:

```
import java.util.Iterator;
```

می توانید ArrayList را به یک آبجکت Iterator جدید ضمیمه کنید:

```
Iterator it = listTest.iterator( );
```

این خط یک آبجکت Iterator جدید تنظیم می کند که می تواند برای وارد شدن به آیتم ها در ArrayList به نام listTest استفاده شود. دلیل استفاده از یک آبجکت Iterator این است که این آبجکت دارای متودهایی به نام next و hasNext می باشد. می توانید از اینها در یک loop استفاده کنید:

```
while ( it.hasNext( ) ) {
System.out.println( it.next( ) );
}
```

متد hasNext یک مقدار Boolean را گزارش می دهد. اگر آیتمی در ArrayListوجود نداشته باشد، مقدار falseخواهد بود. متود بعدی می تواند برای وارد شدن همه ی آیتم ها در لیست استفاده شود.

برای تست کردن همه ی این تئوری کد زیر را امتحان کنید:

```
public static void main(String[] args) {

    ArrayList listTest = new ArrayList();

    listTest.add("first item");
    listTest.add("second item");
    listTest.add("third item");
    listTest.add(7);

    Iterator it = listTest.iterator();

    while (it.hasNext()) {
        System.out.println(it.next());
    }

    //  REMOVE AN ITEM FROM THE LIST
    listTest.remove("second item");

    //PRINT OUT THE NEW LIST
    System.out.println("Whole list=" + listTest);

    //GET THE ITEM AT INDEX POSITION 1
    System.out.println("Position 1=" + listTest.get(1));
}
```

به خطی که همه ی لیست را چاپ می کند، دقت داشته باشید:

```
System.out.println( "Whole list=" + listTest );
```

این خط به شما یک روش سریع برای مشاهده ی آیتم های روی لیستتان ارائه می دهد .وقتی کد اجرا می شود، پنجره ی Output موارد زیر را نمایش خواهد داد:

first item second item third item 7 Whole list=[first item, third item, 7] Position 1=third item

به طور خلاصه، وقتی از تعداد عناصری که قرار است در لیستی از آیتم ها قرار بگیرند، مطمئن نیستید، از ArrayListاستفاده کنید .اکنون بررسی array ها را رها می کنیم. در بخش بعدی به بررسی رشته ها خواهیم پرداخت.

آموزش رشته ها در جاوا

در مورد رشته های جاوا موارد بیشتری از آنچه با چشم دیده می شود وجود دارد. بر خلاف متغیرهای int یا متغیرهایdouble ، رشته ها آبجکت می باشند. و این در عمل به این معناست که با رشته های متن کارهایی می توانید انجام دهید که با متغیرهای double یا int نمی توانید انجام دهید. (همین مورد برای انواع داده ی boolean، byte، single، char، float، long و short نیز به کار می رود: آنها مانند رشته ها آبجکت نیستند)

قبل از اجرای رشته های متن، در اینجا اطلاعات اصلی مربوط به رشته ها را مشاهده خواهید کرد.

چگونگی ذخیره ی رشته ها در جاوا

یک رشته مجموعه ای از کاراکترهای Unicode می باشد که تحت عنوان نام یک متغیر حفظ می شوند. رشته ی زیر را در نظر بگیرید.

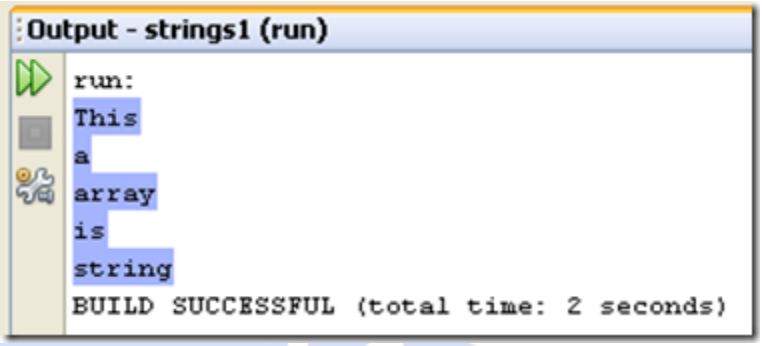
```
String someText = "Bill";
```

این خط به جاوا می گوید که یک رشته آبجکت با چهار کاراکتر"B" ، "i"، "l"و یک "l" دیگر برقرار کند. در مجموعه کاراکترUnicode ، این مقادیر عبارتند از\u0042 ، \u0069، \u006c، \u006c . \u006cمقادیر Unicode به عنوان اعداد هگزادسیمال ذخیره می شوند. حروف بزرگ A) تا (Z با استفاده از مقادیر \u0041تا \u005aذخیره می شوند، در حالیکه حروف کوچک a) تا (z با استفاده از مقادیر هگزادسیمال \u0061 تا \u007aذخیره می شوند.

در بخش قبل یک array وجود داشت که رشته های متن را حفظ می کرد. سپس array را مرتب سازی کردیم.

```
package strings1;
import java.util.Arrays;

public class StringArrays {

    public static void main(String[] args) {

        String[] aryString = new String[5] ;

        aryString[0] = "This";
        aryString[1] = "is";
        aryString[2] = "a";
        aryString[3] = "string";
        aryString[4] = "array";

        Arrays.sort(aryString);

        int i;
        for (i=0; i < aryString.length; i++){
            System.out.println( aryString[i] );
        }

    }
}
```

وقتی که برنامه اجرا می شود، خروجی مانند تصویر زیر می باشد.



قبلا ذکر کردیم که لغت "This" در ابتدا قرار می گیرد. اگر قرار است arrayبه ترتیب حروف الفبا مرتب شود، انتظار می رود که لغت "a" در ابتدا قرار بگیرد. این به این خاطر است که لغت "a" با حرف کوچک دارای مقدار هگزادسیمال u\0061 می باشد، که عدد دهدهی 97 می باشد. اما حرف بزرگ "T" دارای یک مقدار هگزادسیمال u\0054 می باشد که عدد دهدهی 84 است. 84 کمتر از 97می باشد، بنابراین "T" در ابتدا قرار می گیرد.

بسیار خود، اجازه بدهید در موزد اجرای رشته های متن کارهایی انجام دهیم. رشته متدهایی که مورد بررسی قرار خواهیم داد، عبارتند از

toUpperCasse toLowerCase compareTo IndexOf endWith, startsWith Substring Equals charAt trim valueOf

متدهای اول در بالا متدهای آسانی می باشند، که به حروف بزرگ و کوچک تبدیل می شوند. در بخش بعد در مورد آنها فرا خواهید گرفت.

آموزش تغییر به حروف بزرگ و کوچک در جاوا

تبدیل رشته های خود به حروف بزرگ و کوچک کار آسانی می باشد: تنها از متودهای داخلی toUpperCase و toLowerCaseاستفاده کنید.

برای این کار یک پروژه ی جدید را شروع کرده و کد زیر را به آن اضافه کنید:

```
package prjstrings;

public class StringManipulation {

    public static void main(String[] args) {

        String changeCase = "text to change";
        System.out.println( changeCase );

        String result;
        result = changeCase.toUpperCase();

        System.out.println( result );
    }
}
```

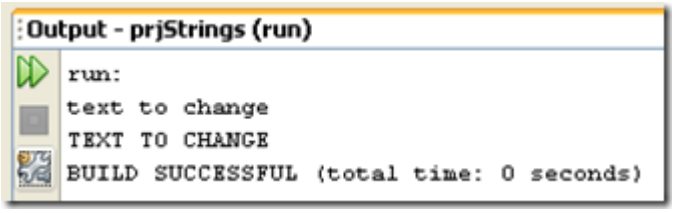
دو خط اول از کد یک متغیر String برای حفظ متن "text to change" تنظیم می کنند و سپس آن را چاپ می کنیم. خط سوم یک متغیر دوم String به نام نتیجه را تنظیم می کند. و خط چهارم می باشد که در آن تبدیل را انجام می دهیم:

```
result = changeCase.toUpperCase( );
```

برای استفاده از یک متود string ابتدا رشته ی مورد نظر را تایپ کنید .برای ما این رشته در متغیر changeCase نامیده می شود. پس از نام متغیر یک نقطه تایپ کنید، سپس لیستی از متودهای موجود را مشاهده می کنید که می توانید در رشته ی خود از آنها استفاده کنید toUpperCase .را انتخاب کنید(.متد پس از آن نیاز به پرانتز دارد).

پس از اینکه جاوا لغت را به حروف بزرگ تغییر داده است، ما رشته ی جدید را به متغیر نتیجه ی خود مرتب می کنیم.

وقتی برنامه اجرا می شود، پنجره ی Output نتیجه ی زیر را نمایش خواهد داد:



اما شما مجبور نیستید که لغت تغییر یافته را در یک متغیر جدید ذخیره کنید. این کار نیز انجام می شود:

```
System.out.println( changeCase.toUpperCase( ) );
```

در اینجا جاوا تنها با تغییر رشته و بدون نیاز به اختصاص دادن نتیجه به یک متغیر جدید، به کار خود ادامه خواهد داد.

در عوض اگر می خواهید فقط به حروف کوچک تغییر دهید، از متود toLowerCase استفاده کنید. این متود درست به روش متود toUpperCase استفاده می شود.

در بخش بعدی چگونگی مقایسه ی رشته ها را در جاوا مشاهده خواهید کرد.

آموزش مقایسه ی رشته ها

شما می توانید یک متود را با متود دیگر مقایسه کنید. (در هنگام مقایسه جاوا از مقادیر هگزادسیمال به جای خود حروف استفاده خواهد کرد.). به عنوان مثال اگر بخواهید لغت "Ape" را با لغت "App" مقایسه کنید تا ببینید کدامیک نخست قرار می گیرد، می توانید از یک رشته متود داخلی به نام compareTo استفاده کنید. اجازه بدهید چگونگی کار آن را بررسی کنیم.

برای این کار نیازی به شروع یک پروژه ی جدید ندارید: به راحتی روی کدی که دارید کامنت بگذارید یا آن را حذف کنید. اکنون کد زیر را به آن اضافه کنید:

```
package prjstrings;

public class StringManipulation {


    public static void main(String[] args) {

        int result;
        String Word1 = "Ape";
        String Word2 = "App";

        result = Word1.compareTo(Word2);

        if (result < 0) {
            System.out.println("Word1 is less than Word2");
        }
        else if (result > 0) {
            System.out.println("Word1 is more than Word2");
        }
        else if (result == 0) {
            System.out.println("The same word");
        }
    }

}
```

ما دو رشته متغیر تنظیم کرده ایم که حاوی لغا ت "Ape" و "App"می باشند. متود compareTo در کد بالا خط زیر می باشد:

```
result = Word1.compareTo( Word2 );
```

متد compareTo یک مقدار را گزارش می دهد. مقدار گزارش شده یا کمتر از 0 یا بیشتر از 0 و یا مساوی 0 می باشد. اگر Word1 قبل از Word2 قرار بگیرد، مقدار گزارش شده کمتر از 0 خواهد بود. اگر Word1 بعد از Word2قرار بگیرد، مقدار گزارش شده بزگتر از 0 خواهد بود. اگر هر دو لغت یکشان باشند، مقدار 0 گزارش خواهد شد.

بنابراین لازم است کقداری را اختصاص دهید که compareTo به یک متغیر بازمی گرداند. ما مقدار را در یک متغیر صحیح به نام نتیجه قرار می دهیم .عبارات IF در کد به سادگی تست می کنند تا مورد را در متغیر نتیجه مشاهده کنند.

به هرحال وقتی یک رشته از متن را با رشته ای دیگر مقایسه می کنید، جاوا به جای حروف واقعی، مقادیر هگزادسیمال تاکید شده را مقایسه می کند. از آنجایی که حروف بزرگ دارای مقدار هگزادسیمال کمتری نسبت به حروف کوچک هستند، حرف بزرگ A در App قبل از حرف کوچ a در ape قرار خواهد گرفت. می توانید این مورد را امتحان کنید. در کد خود "Ape" را به "ape"تغییر دهید. خروجی "Word1 is more than Word2" خواهد بود، به این معنا که از لحاظ الفبایی جاوا لغت ape را بعد از app قرار خواهد داد.

برای حل مشکل یک متود مرتبط به نام compareToIgnoreCase وجود دارد .همانطور که از نام آن پیداست حروف بزرگ و کوچک نادیده گرفته می شوند. با استفاده از این متود از لحاظ الفبایی لغت ape قبل از App قرار خواهد گرفت.

در بخش بعدی متد مفید دیگری به نام indexOfرا مورد بررسی قرار خواهیم داد.

آموزش استفاده از متد indexOf

متد indexOf برای قرار دادن یک کاراکتر یا یک رشته در داخل یک رشته ی دیگر استفاده می شود. به عنوان مثال می توانید از آن برای بررسی وجود کاراکتر @ در یک آدرس ایمیل استفاده کنید. اجازه بدهید از این مثال در کد استفاده کنیم.

مجددا می توانید روی کد خود کامنت بگذارید یا آن را حذف کنید. اما کد جدیدی برای امتحان کردن وجود دارد:

```
package prjstrings;

public class StringManipulation {

    public static void main(String[] args) {

        char ampersand = '@';
        String email_address = "meme@me.com";

        int result;
        result = email_address.indexOf( ampersand );

        System.out.println( result );
    }
}
```

میخواهیم وجود نماد @ را در آدرس ایمیل بررسی کنیم، بنابراین ابتدا یک متغیر char تنظیم کرده و به آن مقدار '@' اختصاص می دهیم. (به نماد علامت نقل قول متغیر char دقت داشته باشید.). پس از تنظیم یک آدرس ایمیل، یک متغیر نتیجه داریم، این یک متغیر int می باشد. دلیل این که نتیجه یک عدد صحیح می باشد این است که متود indexOf یک مقدار گزارش خواهد داد. این مقدار عدد مربوط به موقعیت کاراکتر علامت را در رشته ی آدرس ایمیل باز خواهد گرداند. در اینجا خط مربوطه را مشاهده می کنید:

```
result = email_address.indexOf( ampersand );
```

رشته ای که در حال جستجوی آن هستید در ابتدا قرار می گیرد. پس از یک نقطه، indexOfرا تایپ کنید. بین پرانتزهای indexOf گزینه های زیادی دارید :یکی از این گزینه ها تایپ کردن یک کاراکتر مجزا) یا نام متغیر (charمی باشد. ما متغیر علامت خود را (ampersand) بین پرانتزهای indexOf قرار داده ایم. جاوا موقعیت کاراکتر @ را در آدرس ایمیل به ما خواهد گفت. این مقدار را در متغیر نتیجه ذخیره خواهد کرد.

وقتی کد را اجرا می کنید خروجی 4 خواهد بود. ممکن است تصور کنید که علامت @ پنجمین کاراکتر در آدرس ایمیل می باشد، اما indexOf شمارش را از 0 شروع می کند.

به هرحال اگر کاراکتر در لغتی که آن را جستجو می کنید وجود نداشته باشد، indexOfمقدار 1ـ را گزارش می دهد. برای امتحان این مورد نماد @ را از آدرس ایمیل خود حذف کنید. سپس مجددا کد خود را اجرا کنید که 1ـ را به عنوان خروجی مشاهده خواهید کرد.

می توانید از مقدار گزارش شده ی 1ـ به نفع خود استفاده کنید. در اینجا مجددا کدی را مشاهده می کنید با یک عبارت IF که مقدار متغیر نتیجه را امتحان می کند:

```
package prjstrings;

public class StringManipulation {

    public static void main(String[] args) {

        char ampersand = '@';
        String email_address = "mememe.com";

        int result;
        result = email_address.indexOf( ampersand );

        if (result== -1 ) {
            System.out.println( "Invalid Email Address" );
        }
        else {
            System.out.println( "Email Address OK" );
        }

    }
}
```

بنابراین اگر نتیجه ی indexOf عدد 1ـ باشد، می توانیم به یوزر اجازه ی ادامه ی کار بدهیم.

می توانید از indexOf برای تست کردن بیشتر از یک کاراکتر استفاده کنید. کد زیر آدرس ایمیل را چک می کند اگر با ".com" تمام می شود:

```
public static void main(String[] args) {

    String dotCom = ".com";
    String email_address = "meme@me.com";

    int result;
    result = email_address.indexOf( dotCom );

    if (result== -1 ) {
        System.out.println( "Invalid Email Address" );
    }
    else {
        System.out.println( "Email Address OK ");
    }

}
```

کد تقریبا یکسان می باشد، به جز اینکه اکنون در حال استفاده از متغیر Stringبرای حفظ متنی هستیم که می خواهیم برای وجود (.com) و نه متغیر char آن را بررسی کنیم.

اگر متنی که در جستجوی آن هستیم در string که قبل از نقطه ی مربوط به indexOfیافت نشد، مجددا نتیجه ی 1ـ گزارش داده خواهد شد. در غیر این صورت indexOfموقعیت اولین کاراکتر هماهنگ سازی را گزارش خواهد داد. در کد فوق وقتی شمارش را از 0 شروع می کنید، نقطه هفتمین کاراکتر در آدرس ایمیل می باشد.

می توانید یک موقعیت شروع نیز برای جستجوهای خود تعیین کنید. در مثال آدرس ایمیل می توانیم جستجو برای ".com" را پس از نماد @ آغاز کنیم. در اینجا کدی را می بینید که موقعیت نماد @ را تعیین می کند و سپس از آن به عنوان موقعیت آغازین برای جستجوی ".com" استفاده می کند.

```
public static void main(String[] args) {

    char ampersand = '@';
    String dotCom = ".com";
    String email_address = "meme@me.com";

    int atPos = email_address.indexOf( ampersand );
    int result = email_address.indexOf(dotCom, atPos);

    if (result== -1 ) {
        System.out.println( "Invalid Email Address" );
    }
    else {
        System.out.println( "Email Address OK " + result );
    }
}
```

خط زیر در واقع خط جدید کد می باشد:

```
result = email_address.indexOf( dotCom, atPos );
```

تنها تفاوت افزودن یک متغیر اضافه بین پرانتزهای indexOf می باشد. ما هنوز رشته ای داریم که در جستجوی آن هستیم (که متنی است که در متغیر dotcomقرار می گیرد)، اما اکنون یک موقعیت ابتدایی برای جستجو داریم. این موقعیت مقدار متغیری به نام atPos می باشد. مقدار atPos با استفاده از indexOfبرای تعیین موقعیت نماد @ در آدرس ایمیل به دست می آید. بنابراین جاوا به جای شروع از 0، جستجو را از این موقعیت آغاز خواهد کرد که پیش فرض می باشد.

Ends With ... Starts With

برای برنامه ی فوق می توانید از متود داخلی endsWith استفاده کنید:

```
Boolean ending = email_address.endsWith( dotcom );
```

لازم است یک متغیر Boolean برای endsWith تنظیم کنید، زیرا متود پاسخ true یا false را گزارش خواهد داد. رشته ای که سعی در تست کردن آن دارید، بین پرانتزهای endsWith قرار می گیرد و متنی که در جستجوی آن هستید قبل از آن می آید. اگر متن در جستجوی رشته باشد، یک مقدار true گزارش داده می شود، در غیر اینصورت false خواهد بود. برای بررسی مقدار می توانید یک عبارت if ... else اضافه کنید:

```
System.out.println( "Invalid Email Address" );
}
```

```
else {
System.out.println( "Email Address OK " );
}
if (ending == false ) {
```

متود startsWith به روشی مشابه استفاده می شود.

```
Boolean startVal = email_address.startsWith( dotcom );
```

مجددا مقدار گزارش شده یک مقدار true یا false از Boolean می باشد.

در بخش بعدی چگونگی استفاده از متودی به نام substring را فرا خواهید گرفت .

آموزش متد Substring

یک متد واقعا مفید برای شما متود substring نامیده می شود. این متود به شما اجازه می دهد تا یک متن را از متن دیگر جدا سازید. برای مثال در برنامه ی فوق در آدرس ایمیل خود می توانیم پنج کاراکتر آخر را از کل آدرس جدا کرده و بررسی کنیم که آیا co.uk است یا نه.

برای تمرین با substring می خواهیم دو حرف اول از نام خانوادگی را تغییر دهیم و آنها را با دو حرف اول یک نام شخصی جابه جا سازیم و بالعکس .بنابراین اگر این نام را داشته باشیم:

"Bill Gates"

حروف "Ga" از "Gates" را با "Bi" از "Bill" جابه جا سازیم تا لغت "Bites"را بسازیم. بنابراین "Bi" از "Bill"با "Ga"از "Gates"جابه جا خواهد شد تا لغت "Gall" را بسازد. نام جدید چاپ شده "Gall Bites" خواهد بود.

ما از substring در قسمت بیشتر این برنامه استفاده خواهیم کرد Substring .به این صورت کار می کند:

```
String FullName = "Bill Gates";
String FirstNameChars = "";

FirstNameChars = FullName.substring( 0, 2 );
```

در مورد "Bill Gates" برای جستجو یک رشته تنظیم می کنید. رشته ای که سعی در جستجوی آن دارید، بعد از یک علامت تساوی قرار می گیرد. بعد از یک نقطه (dot)، نام متود، substring، را تایپ کنید. دو روش برای

استفاده از substringوجود دارد و تفاوت در اعداد بین پرانتزها می باشد. در کد فوق دو عدد 0 و 2 را داریم. این به این معناست که گرفتن کاراکترها در موقعیت 0 در رشته آغاز می شود و وقتی دو عدد دارید، این فرایند متوقف می شود. این دو کاراکتر بازگردانده می شوند و در متغیر FirstNameChars قرار می گیرند. اگر می خواهید درست در انتهای رشته قرار بگیرید، کافیست کار زیر را انجام دهید:

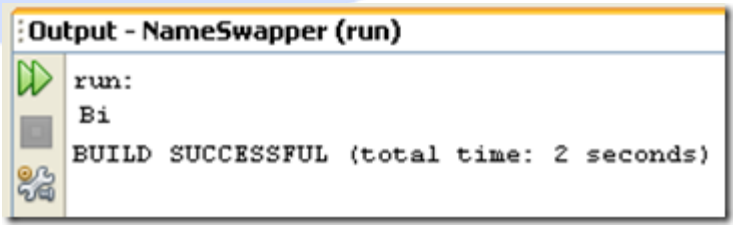
```
String test = FullName.substring( 2 );
```

این بار بین پرانتزهای substring تنها یک عدد داریم. اکنون جاوا از کاراکتر دو در رشته ی FirstName آغاز خواهد کرد و سپس کاراکترها را از موقعیت 2 درست در انتهای رشته می گیرد.

یک برنامه ی جدید آغاز کنید تا این برنامه را امتحان کنید. یک خط چاپ به انتهای کد خود اضافه کرده و کد شما نیز باید مانند زیر باشد:

```
package nameswapper;

public class NameSwap {

    public static void main(String[] args) {

        String FullName = "Bill Gates";
        String FirstNameChars = "";

        FirstNameChars = FullName.substring(0, 2);
        System.out.println(FirstNameChars);
    }

}
```

وقتی که برنامه اجرا می شود، پنجره ی Output باید شبیه به تصویر زیر باشد:



بنابراین متود substring به ما اجازه ی گرفتن دو کاراکتر اول نام "Bill" را داده است.

برای گرفتن دو کاراکتر اول، یک 0 و یک 2 بین پرانتزهای substring داشتیم. ممکن است تصور کنید برای گرفتن "Ga" از "Gates" می توانستیم این کار را انجام دهیم:

```
= FullName.substring(5, 2);
```

بعد از همه ی اینها هنوز دو کاراکتر می خواهیم. فقط این بار 5 به جاوا می گوید که از "G" در "Gates" آغاز کنید. (اولین موقعیت در یک رشته موقعیت 0 می باشد و نه 1). بنابراین از موقعیت 5 در این رشته آغاز کرده و دو کاراکتر را بگیرید.

به هرحال اجرای آن کد منجر به بروز خطا خواهد شد. این به این خاطر است که دومین عدد بین پرانتزهای substring به معنای تعداد کاراکترهایی که می خواهید بگیرید نیست، بلکه به معنای موقعیت نهایی در رشته می باشد. با مشخص کردن 2، به جاوا می گوییم که در کاراکتری در موقعیت 2 از رشته به اتمام برسد. از آنجایی که نمی توانید از موقعیت 6 به عقب بازگشته و به موقعیه 2 بروید، خطایی دریافت می کنید.

)نکته: اگر شمارش را در رشته ی "Bill" از 0 شروع کنید، ممکن است تصور کنید که موقعیت 2 حرف "l" می باشد و حدس شما درست است. اما substring قبل از کاراکتر در آن موقعیت آغاز می کند و پس از آن(.

بنابراین برای گرفتن "Ga" از "Gates" می توانید مانند زیر عمل کنید:

```
FullName.substring( 5, FullName.length( ) - 3 );
```

اکنون دومین عدد طول رشته منهای 3 کاراکتر می باشد. طول رشته تعداد کاراکترهایی است که آن رشته دارد "Bill Gates" . دارای 10 کاراکتر شامل فاصله، می باشد. اگر سه کاراکتر را کنار بگذارید، 7 کاراکتر خواهید داشت . بنابراین ما به substring می گوییم که در کارامتر 5 شروع شده و در کاراکتر 7 به پایان برسد.

این برنامه برای افرادی با نام "Bill Gates" به درستی کار می کند. اما اگر نام شما "Billy Gates" می بود، این برنامه به درستی کار نمی کرد . بنابراین کد فوق کاراکتر فاصله به علاوه ی حرف "G" را می گیرد که به هیچ

وجه کاراکترهای مورد نظر ما نیستند. ما می خواهیم که برنامه با هر دو نامی که وارد می شوند، کار کند.

بنابراین باید کمی زیرک تر باشیم.

کاری که می توانیم انجام دهیم این است که به موقعیت فاصله در هر دو نام دقت داشته باشیم. دو کاراکتری که می خواهیم از نام دوم بگیریم همیشه درست بعد از کاراکتر فاصله قرار می گیرند. ما به کدی احتیاج داریم که دو کاراکتر اول بعد از فاصله را بگیرد.

برای اشاره به موقعیت فاصله می توانیم از indexOf استفاده کنیم.

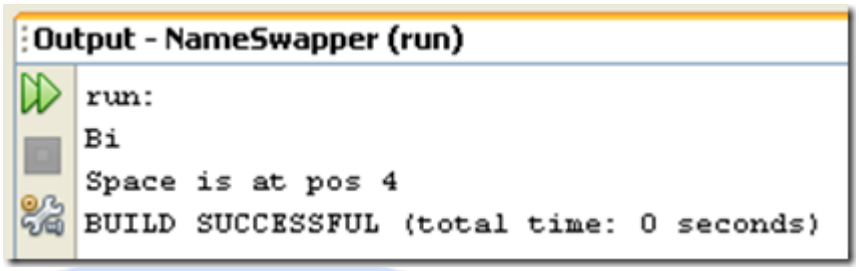
```
int spacePos = FullName.indexOf(" ");
```

برای تعیین کاراکتر فاصله، می توانید یک فاصله بین دو نماد نقل قول (") قرار دهید. این کاراکتر پس از پرانتزهای indexOf قرار می گیرد. مقدار بازگشت داده شده یک عدد صحیح خواهد بود که موقعیت اولین رویداد از کاراکتر فاصله در رشته ی FullName می باشد.

این تمرین را می توانید با اضافه کردن خط فوق به کد خود امتحان کنید: برای چک کردن Output یک خط چاپی اضافه کنید:

```
public static void main(String[] args) {

    String FullName = "Bill Gates";
    String FirstNameChars = "";

    FirstNameChars = FullName.substring(0, 2);
    System.out.println( FirstNameChars );

    int spacePos = FullName.indexOf(" ");
    System.out.println( "Space is at pos "  + spacePos);


}
```

برنامه را اجرا کنید تا خروجی زیر را مشاهده کنید:



بنابراین در این رشته فاصله در موقعیت 4 می باشد. ما می توانیم از این واقعیت برای گرفتن دو کاراکتر اول از "Gates"یا در واقع هر نام دوم دیگری استفاده کنیم. به جاوا می گوییم که از اولین کاراکتر بعد از فاصله آغاز و در دو کاراکتر بعدی به اتمام برساند.

```
FullName.substring( spacePos + 1, (spacePos + 1) + 2)
```

بنابراین دو عدد بین پرانتزهای substring عبارتند از:

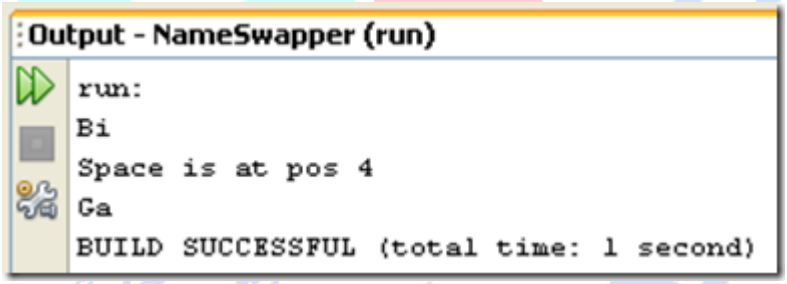
```
spacePos + 1, (spacePos + 1) + 2
```

ما می خواهیم که از اولین کاراکتر بعد از فاصله آغاز کنیم (space + 1) و دو کاراکتر بعد از این موقعیت به اتمام برسانیم که به این شکل می باشد2 + (spacePos + 1) :

خطوط زیر را به کد خود اضافه کنید (خطوطی که هایلایت شده اند. متد جدید substringدر دو خط قرار می گیرد، اما شما اگر تمایل داشته باشید می توانید روی یک خط قرار دهید.):

```
public static void main(String[] args) {

    String FullName = "Bill Gates";
    String FirstNameChars = "";


    FirstNameChars = FullName.substring(0, 2);
    System.out.println( FirstNameChars );

    int spacePos = FullName.indexOf(" ");
    System.out.println( "Space is at pos "  + spacePos);

    String SurNameChars = "";
    SurNameChars = FullName.substring(spacePos + 1,
                                      (spacePos + 1) + 2);

    System.out.println( SurNameChars );

}
```

وقتی که برنامه را اجرا می کنید، پنجره ی Output مانند زیر خواهید بود:



بنابراین اکنون ما "Bi" از Bill و "Ga" از Gates را داریم. آنچه اکنون باید انجام دهیم گرفتن بقیه ی کاراکترها از دو نام و سپس جابه جایی آنها می باشد.

مجددا می توانیم از substring برای گرفتن بقیه ی کاراکترها از نام اول استفاده کنیم:

```
String OtherFirstChars = "";
OtherFirstChars = FullName.substring( 2, spacePos );
System.out.println( OtherFirstChars );
```

و همچنین کاراکترهای باقیمانده از نام دوم:

```
String OtherSurNameChars = "";
```

```
OtherSurNameChars = FullName.substring((spacePos + 1) + 2,
FullName.length() );

System.out.println( OtherSurNameChars );
```

به اعداد بین پرانتزهای substring دقت کنید. برای گرفتن دیگر کاراکترهای نام اول، اعداد عبارتند از:

```
spacePos
```

این به جاوا می گوید که از موقعیت 2 شروع کرده و درست به موقعیت فاصله برود. به هرحال گرفتن بقیه ی نام دوم کمی دشوارتر خواهد بود:

```
(spacePos + 1) + 2, FullName.length( )
```

(spacePos1 + 1) + 2 موقعیت شروع سومین کاراکتر از دومین نام می باشد . ما می خواهیم که در طول رشته به پایان برسانیم که بقیه ی کاراکترها را دریافت خواهیم کرد.

خطوط های لایت شده ی زیر را به کد خود اضافه کنید:

```
public static void main(String[] args) {

  String FullName = "Bill Gates";
  String FirstNameChars = "";

  FirstNameChars = FullName.substring(0, 2);
  System.out.println( FirstNameChars );

  int spacePos = FullName.indexOf(" ");
  System.out.println( "Space is at pos "  + spacePos);

  String SurNameChars = "";
  SurNameChars = FullName.substring(spacePos + 1,
                                    (spacePos + 1) + 2);

  System.out.println( SurNameChars );

  String OtherFirstChars = "";
  OtherFirstChars = FullName.substring(2, spacePos);
  System.out.println( "other first: " + OtherFirstChars );

  String OtherSurNameChars = "";
  OtherSurNameChars = FullName.substring((spacePos + 1) + 2,
                                         FullName.length());

  System.out.println("other second: " + OtherSurNameChars );

}
```


خروجی مانند زیر خواهید بود:

```
Output - NameSwapper (run)
run:
Bi
Space is at pos 4
Ga
other first: ll
other second: tes
BUILD SUCCESSFUL (total time: 2 seconds)
```

اکنون تمام قسمت های نام را داریم. برای اتصال آنها به یکدیگر، می توانیم از الحاق (concatenation) استفاده کنیم:

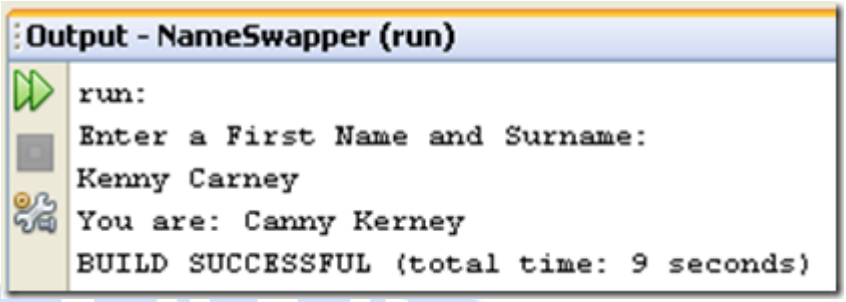
```
String NewName = "";
NewName = SurNameChars + OtherFirstChars + " " +
                         FirstNameChars + OtherSurNameChars;

System.out.println( "New Name = "  + NewName);
```

```
Output - NameSwapper (run)
run:
Bi
Space is at pos 4
Ga
other first: ll
other second: tes
New Name = Gall Bites
BUILD SUCCESSFUL (total time: 2 seconds)
```

می توانیم از خطوط چاپی رها شویم و از یک یوزر بخواهیم که نام اول و نام دوم را وارد کند. در اینجا برنامه ی جدید را مشاهده می کنید (تنها افزایش برای ورودی صفحه کلید می باشد که قبلا از آن استفاده کرده اید:(

```
package nameswapper;
import java.util.Scanner;

public class NameSwap {

 public static void main(String[] args) {

   Scanner user_input = new Scanner(System.in);

   System.out.println( "Enter a First Name and Surname:" );
   String FullName = user_input.nextLine();

   String FirstNameChars = "";
   FirstNameChars = FullName.substring(0, 2);

   int spacePos = FullName.indexOf(" ");
   String SurNameChars = "";
   SurNameChars = FullName.substring(spacePos + 1,
                                     (spacePos + 1) + 2);
   String OtherFirstChars = "";
   OtherFirstChars = FullName.substring(2, spacePos);

   String OtherSurNameChars = "";
   OtherSurNameChars = FullName.substring((spacePos + 1) + 2,
                                          FullName.length());

   String NewName = "";
   NewName = SurNameChars + OtherFirstChars + " " +
                            FirstNameChars + OtherSurNameChars;
   System.out.println( "You are: "  + NewName);
 }
}
```

وقتی برنامه را اجرا می کنید و نام و نام خانوادگی را وارد می کنید، پنجره ی Output باید مانند تصویر زیر باشد:



البته باید چند برنامه ی بررسی خطا نیز اضافه کنیم. اما خواهیم پذیرفت که یوزر می تواند یک نام و یک نام خانوادگی با یک فاصله بین آنها وارد کند . در غیر اینصورت برنامه با شکست مواجه می شود. به هرحال اجازه بدهید ادامه داده و نگاهی به متود equals داشته باشیم.

آموزش متد equals در جاوا

می توانید شباهت دو رشته را با یکدیگر چک کنید. به این منظور از متود equalsدر جاوا استفاده کنید. دز اینجا کد مربوط به آن را مشاهده می کنید:

```
public static void main(String[] args) {

    String email_address1 = "meme@me.cob";
    String email_address2 = "meme@me.com";
    Boolean isMatch = false;

    isMatch = email_address1.equals(email_address2);

    if (isMatch == true) {
        System.out.println( "Email Address Match ");
    }
    else {
        System.out.println("Email addresses don't match");
    }
}
```

در این کد سعی داریم شباهت دو آدرس ایمیل را بررسی کنیم. دو خط اول دو متغیر string را تنظیم می کنند، برای هر آدرس ایمیل یک متغیر. سومین خط یک متغیر Boolean برقرار می کند. این به این خاطر است که متود equals مقدار trueیا false را بازمی گرداند. چهارمین خط جایی است که در آن از متد استفاده می کنیم:

```
isMatch = email_address1.equals( email_address2 );
```

بین پرانتزهای متودequals ، رشته ای را قرار می دهید که سعی دارید آن را بررسی کنید. رشته ی دیگر قبل از متود تساوی قرار می گیرد. جاوا به شما می گوید که آیا این دو یکسان هستند یا نه .(true or false) عبارت IFاین موضوع را بررسی می کند.

به هرحال متود equals فقط آبجکت ها را مقایسه می کند. این برای رشته ها خوب می باشد، زیرا آنها نیز آبجکت می باشند. اما نمی توانید برای مقایسه ی متغیرهای int از متود equal استفاده کنید. برای مثال این کد منجر به بروز خطای زیر خواهد شد:

```
int num1 = 12;
int num2 = 13
Boolean isMatch = false;

isMatch = num1.equals(num2);
```

متغیر int نوع اولیه ی داده می باشد و نه یک آبجکت. شما می توانید نوع داده ی int را به یک آبجکت بازگردانید، گرچه:

```
int num1 = 12;
Integer num_1 = new Integer(num1);
```

در اینجا متغیر int با نام num1 به یک آبجکت عدد صحیح برگردانده شده است. به استفاده از کلمه ی کلیدی newدقت کنید. بین پرانتزهایInteger ، نوع اولیه ی داده ی int را قرار می دهید که قصد تغییر آن را به یک آبجکت دارید.

متد مفید دیگری در جاوا charAt می باشد. در بخش بعد به بررسی این متد خواهیم پرداخت .

آموزش متد charAt در جاوا

می توانید چک کنید که کدام کاراکتر مجزا در یک رشته ی خاص قرار می گیرد .متود charAt در جاوا به این منظور استفاده می شود. در اینجا کد مربوط به آن را مشاهده می کنید:

```
String email_address = "meme@me.com";
char aChar = email_address.charAt( 4 );
System.out.println( aChar );
```

این کد حرف مربوط به موقعیت 4 را در رشته ی آدرس ایمیل بررسی می کند. مقدار گزارش شده متغیری از نوع charمی باشد:

```
char aChar = email_address.charAt( 4 );
```

وقتی که کد فوق اجرا می شود، خروجی کاراکتر @ می باشد. عدد بین پرانتزهایcharAt ، آن موقعیتی در رشته می باشد که سعی در بررسی آن دارید . در اینجا قصد داریم به کاراکتری در موقعیت 4از رشته ی email_addressبرسیم. مجددا شمارش از 0 شروع می شود، درست مانند.substring

یک استفاده ی مفید از charAt گرفتن یک حرف از یک رشته متغیر میباشد که توسط یک یوزر تایپ شده و سپس به یک متغیر مجزای char تبدیل شده است. برای مثال می توانستید از یوزر بخواهید که Y تایپ کرده و ادامه دهد و یا یک N تایپ کرده و خارج شود. نگاهی به این کد داشته باشید:

```
public static void main(String[] args) {

    Scanner user_input = new Scanner(System.in);

    System.out.println( "Quit Y/N");

    String aString  = user_input.next();

    char aChar = aString.charAt(0);

    if (aChar == 'Y') {
        System.out.println( "OK, BYE BYE");
    }
    else {
        System.out.println( "Not Quitting");
    }
}
```

نمی توانیم برای دریافت یک حرف مجزا و ذخیره در یک متغیرchar ، مستقیما از گروه Scanner استفاده کنیم. بنابراین از متود ( )next برای گرفتن رشته ی بعدی استفاده می کنیم که یوزر وارد کرده است. یک next integer، next long، next double و حتی next Boolean وجود دارد. اما هیچگونه next char وجود ندارد. (به یاد داشته باشید که یک متغیر char یک عدد Unicode را به عنوان یک عدد صحیح ذخیره می کند(.

میتوانیم از charAt برای گرفتن یک کاراکتر از هر رشته ای که یوزر وارد می کند استفاده کنیم، حتی اگر یوزر یک حرف مجزا وارد کرده باشد:

```
char aChar = aString.charAt( 0 );
```

تمام آنچه می گوییم عبارت است از " گرفتن کاراکتر در موقعیت 0 در رشته ای به نام رشته ی aString و سپس ذخیره ی آن در متغیر"aChar

ما یک عبارت IF برای تست کردن آنچه در متغیر aChar می باشد، اضافه کرده ایم. (به استفاده از علامت نقل قول انفرادی (') در اطراف حرف Y دقت داشته باشید.)

آموزش متد جایگزینی (replace) در جاوا

متود جایگزینی (replace) برای جایگزین کردن همه ی وقایع مربوط به یک کاراکتر در یک رشته ی خاص استفاده می شود. این جمله را در نظر بگیرید:

"Where are you books?"

ما قصد داریم که لغت "your" را جایگزین "you"کنیم. کد مربوط را در اینجا مشاهده می کنید:

```
public static void main(String[] args) {

    String aString = "Where are you books?";

    String amend = aString.replace("you", "your");

    System.out.println( amend );
}
```

روش های زیادی در استفاده از متود replace وجود دارد و آنها در آنچه بین پرانتزهای متود قرار می گیرد با یکدیگر متفاوت هستند. ما یک ترتیبی از کاراکترها را با ترتیبی دیگر جایگزین می کنیم. کاما مانند کلمه ی with، این دو را از یکدیگر مجزا می کند.

همچنین می توانید یک کاراکتر مجزا را جایگزین کنید:

```
aString.replace( '£', '@' )
```

کد فوق "Replace £ with @" را می خواند.

(شما می توانید از چیزی به نام عبارت متداول (regular expression) در متودهای جایگزینی خود استفاده کنید، اما این مسئله خارج از این آموزش می باشد)

Trim

شما فضاهای سفید را می توانید در رشته های خود مرتب کنید. فضای سفید مواردی مانند کاراکترها، تب ها و کاراکترهای newline می باشند – به عبارت دیگر کاراکترهایی که شما نمی توانید ببینید. استفاده از متود trimآسان می باشد:

```
String amend = " white space ";
amend = amend.trim( );
```

بنابراین متود trim بعد از رشته ای که می خواهید اصلاح کنید، قرار می گیرد. کاراکترهای خالی قبل از لغت "white"و پس از "space" در کد بالا حذف خواهند شد.

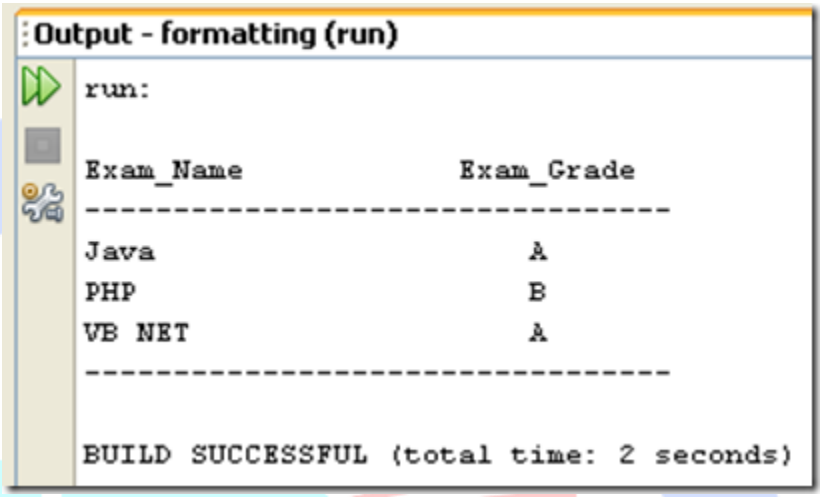
اگر از طرف یوزر ورودی دریافت می کنید، استفاده از trim روی رشته های وارد شده فکر بسیار خوبی می باشد .

اکنون اجرای رشته را رها کرده و ادامه می دهیم. بخش بعد رشته های فرمت شده از متن می باشند .

آموزش Formatted Strings

رشته های متن با استفاده از فرمان printf می توانند فرمت شده و خروجی شوند. فرمان printf مجموعه ای از کاراکترها با عنوان format specification (مشخصات فرمت) را درک می کند. سپس این فرمان یک رشته از

متن را گرفته و و آن را قالب بندی می کند، براساس مشخصات قالب تایید شده. به عنوان مثال فرض کنید که می خواهیم پنجره ی Output متنی را در ستون های مرتب نمایش دهد، مانند تصویر زیر:



اولین ستون از سمت چپ و دومین ستون از سمت راست مرتب شده اند. کد مربوط به Exam_Name و Exam_Grade مانند زیر می باشد:

```
String heading1 = "Exam_Name";
String heading2 = "Exam_Grade";
System.out.printf( "%-15s %15s %n", heading1, heading2);
```

برای به دست آوردن ستون مرتب شده از سمت چپ نیاز به یک نماد درصد، یک نماد منها، تعداد کاراکترها و سپس حرف s ("s" کوچک) دارید. بنابراین "%-15s"به معنای 15 کاراکتر مرتب شده از سمت چپ می باشد.

برای به دست آوردن یک ستون مرتب شده از سمت راست همان ترتیب کاراکترها استفاده می شوند، به جز علامت منها.

برای به دست آوردن یک newline نماد %n استفاده می شود. دقت داشته باشید که کاراکترها با علامت های نقل قول دوتایی احاطه شده اند.

پس از یک کاما، متنی را تایپ می کنید که قصد قالب بندی آن را دارید .اولین کاما در کد فوق مشخصات قالب (format specification)را از متنی که قرار است قالب بندی شود، جدا می سازد.

در اینجا جدول هایی از گزینه های مختلف را مشاهده می کنید:

"s%"	یک رشته را با تعداد کاراکترهای مورد نیاز قالب بندی می کند.
"%۱۵s"	رشته ای را با تعداد کاراکترهای مشخص و از سمت راست قالب بندی می کند.
"۱۵s-%"	رشته ای را با تعداد کاراکترهای مشخص و از سمت چپ قالب بندی می کند.

اگر بخواهید اعداد را قالب بندی کنید یا می توانید از کاراکتر "d" استفاده کنید یا، برای اعداد شناور ممیزی، از کاراکتر."f"

قالب بندی اعداد صحیح

"d%"	یک رشته را با تعداد اعداد مورد نیاز قالب بندی می کند.
"۴%d"	یک رشته را با تعداد مشخصی از اعداد صحیح قالب بندی می کند. اگر اعداد صحیح کافی نباشند، سمت چپ را با فاصله ها پر می کند.
	یک رشته را با تعداد مشخصی از اعداد صحیح قالب بندی می کند. اگر اعداد صحیح کافی نباشند، سمت چپ را با ۰ پر خواهد کرد.

قالب بندی اعداد ممیزی شناور:

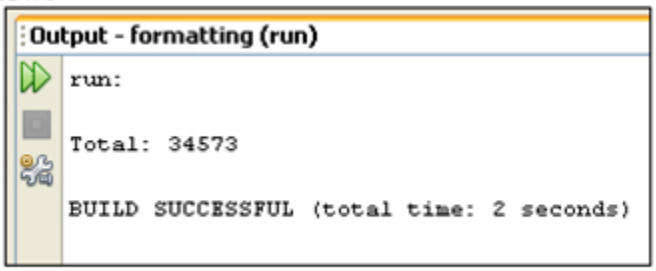
"f%"	یک رشته با تعداد اعداد مورد نیاز قالب بندی می کند. همیشه به شما ۶ مکان ممیزی می دهد.
"۲f.%"	یک رشته با تعداد اعداد مورد نیاز قالب بندی می کند. دو مکان ممیزی ارائه می دهد.
"%۱۰.۲f"	به دو مکان ممیزی قالب بندی می کند، اما تمام رشته ۱۰ کاراکتر را اشغال می کند. اگر اعداد کافی وجود نداشته باشند، سپس فضاها در سمت چپ اعداد استفاده می شوند.

در اینجا مثال هایی از کدهای مربوط بهString ، عدد صحیح و قالب بندی ممیز شناور (floating point) را مشاهده می کنید. آنها را برای خود امتحان کنید.

```
System.out.printf( "%s %d %n", "Total:", 34573);
```

The text "Total:" will be formatted as a string (%s), while the numbers 34573 will be formatted as numbers (%d).

Output Window:

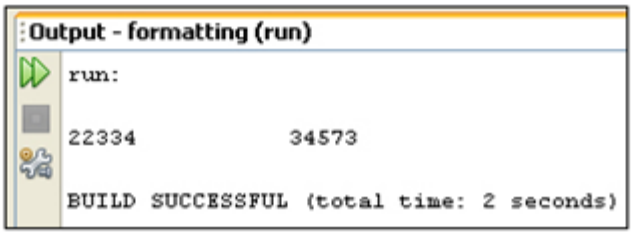


مانند فوق، اما اعداد 10 مکان را با فضاهایی در سمت چپ، اشغال می کنند.

```
System.out.printf( "%-10d %10d %n", 22334, 34573);
```

Two sets of numbers. The first one is left-justified; the second one is right-justified.

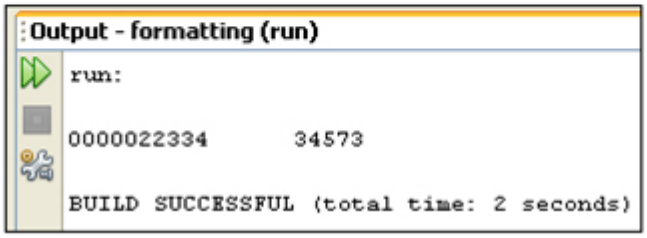
Output Window:




```
System.out.printf( "%010d %10d %n", 22334, 34573);
```

Two sets of numbers again. The first one is padded with leading zeros, this time. The second one is right-justified, but spaces are used as padding to the left instead of zeroes.

Output Window:

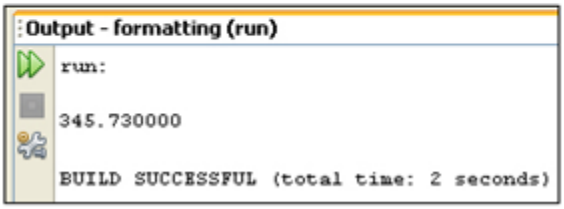


یک عدد ممیزی شناور را قالب بندی کرده و یک کاراکتر newline اضافه کنید. عدد ممیزی دارای شش مکان اعشاری خواهد بود.

```
System.out.printf( "%f  %n",  345.73);
```

Format a floating point number, and add a newline character. The floating point number will have six decimal places.

Output Window:

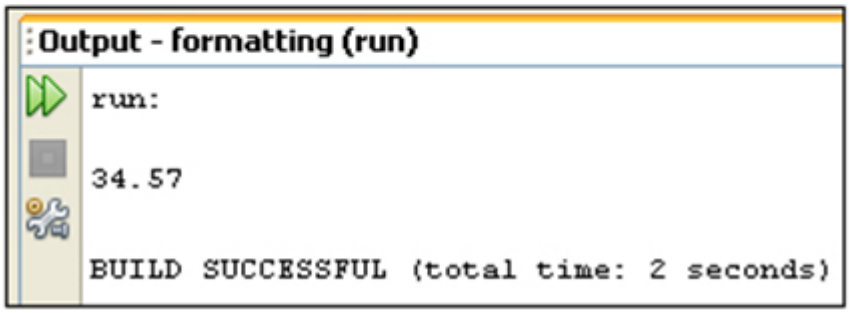


درست مانند فوق، اما تنها دو قسمت اعشاری را قالب بندی خواهد کرد.

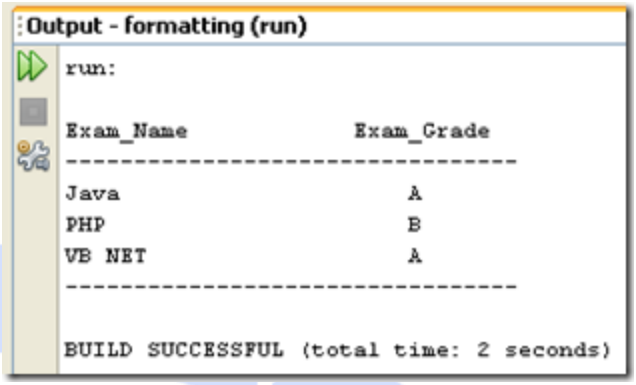
```
System.out.printf( "%.2f  %n",  34.573);
```

Same as above but will format to only two decimal places.

Output Window:



و سرانجام، در اینجا جدولی را از شروع این بخش قالب بندی مشاهده می کنید.



در اینجا نیز کد مربوط به قالب بندی فوق را مشاهده می کنید:

```
public static void main(String[] args) {

    String heading1 = "Exam_Name";
    String heading2 = "Exam_Grade";
    String divider = "---------------------------------";

    String course1 = "Java"; String grade1 = "A";
    String course2 = "PHP"; String grade2 = "B";
    String course3 = "VB NET"; String grade3 = "A";

    System.out.println("");
    System.out.printf("%-15s %15s %n", heading1, heading2);
    System.out.println(divider);

    System.out.printf("%-15s %10s %n", course1, grade1);
    System.out.printf("%-15s %10s %n", course2, grade2);
    System.out.printf("%-15s %10s %n", course3, grade3);

    System.out.println(divider);
    System.out.println("");

}
```

یک تمدین با قالب بندی داشته باشید و چگونگی به دست آوردن آن را مشاهده کنید. اگر پیغام خطا دریافت کردید، ممکن است قالب بندی نوع "s" را با قالب بندی نوع "d" اشتباه گرفته باشید .

متدهای جاوا

متد جاوا مجموعه ای از وضعیت هاست که برای اجرای یک عملکرد گروه بندی می شوند. به عنوان مثال وقتی متد System.out.println را فرا خوانی می کنید، درواقع سیستم چندین عبارت را برای نمایش یک پیغام در consoleاجرا می کند.

اکنون شما چگونگی ایجاد متودهای خود را با بازگشت مقادیر یا بدون بازگشت مقادیر، فراخوانی یک متود با پارامترها و بدون پارامترها، بارگذاری متودها با استفاده از همان نام ها و به کارگیری متود انتزاع در طراحی برنامه فرا خواهید گرفت.

ایجاد متد

با توجه به مثال زیر می توانید ترکیب یک متد را توضیح دهید

```
public static int funcName(int a, int b)
{
  // body
}
```

در اینجا:

- public static :اصلاح کننده
- int :نوع بازگشت
- funcName :نام عملکرد
- a, b :پارامترهای فرمال
- int a, int b :لیست پارامترها

متودها با عنوان Procedures یا Functions نیز شناخته می شوند:

- Procedures :هیچ مقداری را بازنمی گرداند.
- Functions :یک مقدار را بازمی گردانند.

تعریف متود حاوی یک تیتر و یک بدنه می شود. همین مورد در زیر نشان داده شده است:

مثال:

```
modifier returnType nameOfMethod (Parameter List) {
 // method body
}
```

ترکیب نمایش داده شده در بالا حاوی:

- Modifier :نوع access مربوط به متود را تعریف می کند و استفاده از آن انتخابی می باشد
- returnType :متود ممکن است یک مقدار را بازگرداند
- nameOfMethod :این نام متود می باشد. ویژگی خاص متود حاوی نام متود و لیست پارامترها می باشد.
- Parameter List :لیست پارامترها که حاوی نوع، ترتیب و تعداد پارامترهای یک متود می باشد. اینها انتخابی هستند، متود ممکن حاوی هیچ پارامتری نباشد.
- method body :بدنه ی متود تعریف می کند که متود با عبارات چه انجام می دهد.

مثال

در اینجا source code متود تعریف شده ی بالا به نام ()max می باشد. این متود دارای دو پارامتر num1 و num2می باشد و حداکثر را بین هر دو بازمی گرداند:

```
/** the snippet returns the minimum between two numbers */
public static int minFunction(int n1, int n2) {
   int min;
   if (n1 > n2)
      min = n2;
   else
      min = n1;

   return min;
}
```

فراخوانی متد

برای استفاده از یک متد ابتدا باید فراخوانده شود. دو روش در فراخوانی یک متد وجود دارد، به عنوان مثال متدی که یک مقدار را بازمی گرداند و متدی که هیچ چیز بازنمی گرداند.

فرایند فراخوانی ماود ساده می باشد. وقتی که برنامه ای یک متد را فرامی خواند، کنترل برنامه به فراخوانی متد تغییر می یابد. سپس این متد فراخوانده شده کنترل را به فراخواننده تحت دو شرط، بازمی گرداند، وقتی:

- بازگشت عبارت اجرا می شود
- اتمام متد در حال بستن گروه می باشد

متدهایی که پوچ (void) گزارش می دهند، برای فراخوانی یک عبارت در نظر گرفته می شوند. اجازه بدهید مثالی را در نظر بگیریم:

```
System.out.println("This is tutorialspoint.com!") ;
```

مقدار متد returning با مثال زیر قابل درک می باشد:

```
int result = sum(6, 9) ;
```

مثال

در زیر مثالی از چگونگی تعریف یک متد و چگونگی فراخوانی آن را مشاهده می کنید:

```
public class ExampleMinNumber{


   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      int c = minFunction(a, b);
      System.out.println("Minimum Value = " + c);
   }


   /** returns the minimum of two numbers */
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;


      return min;
   }
}
```

این مثال نتیجه ی زیر را تولید خواهد کرد:

Minimum value = 6

لغت کلیدی void

لغت کلیدی void به ما اجازه می دهد تا متدهایی را ایجاد کنیم که هیچ مقداری را بازنمی گرداند. در مثال زیر یک متد methodRankPoints را در نظر می گیریم. این متد یک متد void می باشد که هیچ مقداری را بازنمی گرداند .فراخوانی به یک متد void باید یک عبارت باشد، به عنوان مثال(methodRankPoints(255.7، که یک عبارت جاواست که با یک نقطه ویرگول به پایان می رسد، همانطور که در مثال زیر نشان داده شده است

مثال

```
public class ExampleVoid {

   public static void main(String[] args) {
      methodRankPoints(255.7);
   }

   public static void methodRankPoints(double points) {
      if (points >= 202.5) {
         System.out.println("Rank:A1");
      }
      else if (points >= 122.4) {
         System.out.println("Rank:A2");
      }
      else {
         System.out.println("Rank:A3");
      }
   }
}
```

این مثال نتیجه ی زیر را تولید خواهد کرد:

Rank:A1

انتقال پارامترها با مقدار

هنگام کار تحت فرایند فراخوانی، argumentباید منتقل شود. اینها باید به همان ترتیبی باشند که پارامترهای مربوطه در تعیین متود هستند. پارامترها می توانند با مقدار یا با مرجع منتقل شوند.

انتقال پارامترها با مقدار به معنای فراخوانی یک متود با یک پارامتر می باشد. از این طریق مقدار argument به پارامتر منتقل می شود.

برنامه ی زیر مثالی از انتقال پارامتر با مقدار را نشان می دهد. مقادیر argumentها، حتی پس از فراخوانی متود، بدون تغییر باقی می مانند.

```
public class swappingExample {


   public static void main(String[] args) {

      int a = 30;

      int b = 45;


      System.out.println("Before swapping, a = " +

                         a + " and b = " + b);


      // Invoke the swap method

      swapFunction(a, b);

      System.out.println("\n**Now, Before and After swapping values will be
same here**:");

      System.out.println("After swapping, a = " +

                         a + " and b is " + b);

   }
```

```
   public static void swapFunction(int a, int b) {

      System.out.println("Before swapping(Inside), a = " + a
                         + " b = " + b);
      // Swap n1 with n2
      int c = a;
      a = b;
      b = c;

      System.out.println("After swapping(Inside), a = " + a
                         + " b = " + b);
   }
}
```

این مثال نتیجه ی زیر را به دنبال دارد:

```
Before swapping, a = 30 and b = 45
Before swapping(Inside), a = 30 b = 45
After swapping(Inside), a = 45 b = 30

**Now, Before and After swapping values will be same here**:
After swapping, a = 30 and b is 45
```

Metho Overloading

وقتی که یک گروه دارای دو و یا چند متود هم نام اما با پارامترهای متفاوت می باشد، این امر method overloadingنامیده می‌شود، که متفاوت با overridingمی باشد. در overriding یک متد دارای همان نام، نوع، تعداد پارامترها و غیره می باشد.

اجازه بدهید مثال قبل را برای یافتن حداقل تعداد عدد صحیح در نظر بگیریم .اجاره بدهید بگوییم که می خواهیم حداقل تعداد نوع double را پیدا کنیم. سپس مفهوم overloading برای ایجاد دو یا چند متد هم نام اما با پارامترهای متفاوت، معرفی خواهد شد.

مثال زیر همین مورد را توضیح می دهد:

```
public class ExampleOverloading{


   public static void main(String[] args) {
      int a = 11;
      int b = 6;
      double c = 7.3;
      double d = 9.4;
      int result1 = minFunction(a, b);
      // same function name with different parameters
      double result2 = minFunction(c, d);
      System.out.println("Minimum Value = " + result1);
      System.out.println("Minimum Value = " + result2);
   }


  // for integer
   public static int minFunction(int n1, int n2) {
      int min;
      if (n1 > n2)
         min = n2;
      else
         min = n1;


      return min;
   }
```

```
  // for double
  public static double minFunction(double n1, double n2) {
    double min;
     if (n1 > n2)
        min = n2;
     else
        min = n1;

     return min;
  }
}
```

این مثال نتیجه زیر را به دنبال دارد :

Minimum Value = 6

Minimum Value = 7.3

متدهای overloading برنامه را خوانا می سازد. در اینجا دارای یک نام می باشند، اما پارامترهای متفاوت دارند. حداقل تعداد انواع صحیح(integer) و doubleنتیجه می باشد .

متد جاوا مجموعه ای از وضعیت هاست که برای اجرای یک عملکرد گروه بندی می شوند. به عنوان مثال وقتی متد System.out.println را فرا خوانی می کنید، درواقع سیستم چندین عبارت را برای نمایش یک پیغام در consoleاجرا می کند.

اکنون شما چگونگی ایجاد متدهای خود را با بازگشت مقادیر یا بدون بازگشت مقادیر، فراخوانی یک متد با پارامترها و بدون پارامترها، بارگذاری متدها با استفاده از همان نام ها و به کارگیری متد انتزاع در طراحی برنامه فرا خواهید گرفت.

استفاده از argument های Command-Line

گاهی اوقات وقتی برنامه ای را اجرا می کنید، تمایل خواهید داشت که اطلاعات را به برنامه انتقال دهید. این کار به وسیله ی انتقال خط فرمان argumentها به ( )main انجام می شود.

یک command-line argument اطلاعاتی است که مستقیما نام برنامه را در هنگام اجرا روی خط فرمان دنبال می کند. دسترسی به argument های خط فرمان در داخل یک برنامه ی جاوا بسیار آسان می باشد، آنها مانند رشته ها در ردیف String به ( )main منتقل می شوند.

مثال

برنامه ی زیر همه ی argument های خط فرمان را که از طریق آنها فراخوانده شد، نمایش می دهد:

```
public class CommandLine {


   public static void main(String args[]){

      for(int i=0; i
```

اجرای این برنامه را به شکلی که در زیر نشان داده شده، امتحان کنید:

```
java CommandLine this is a command line 200 -100
```

این برنامه نتیجه ی زیر را به دنبال دارد:

```
args[0]: this

args[1]: is

args[2]: a

args[3]: command

args[4]: line

args[5]: 200
```

```
args[6]: -100
```

سازنده ها

یک سازنده در هنگام ایجاد، یک آبجکت را مقدار دهی می کند. این سازنده هم نام گروه خود می باشد و از لحاظ نحوی مشابه یک متد است. به هرحال سازنده ها هیچ نوع بازگشتی مشخصی ندارند.

معمولا برای دادن مقادیر اولیه به متغیرهای نمونه که توسط گروه تعریف شده اند ، یا برای اجرای دیگر روش های راه اندازی که برای ایجاد یک آبجکت کامل ، از یک سازنده استفاده خواهید کرد.

همه ی گروه ها دارای سازنده هستند، چه شما برای آن تعریف کنید و چه نکنید، زیرا جاوا به طور خودکار یک سازنده ی پیش فرض ارائه می دهد که همه ی متغیرهای عضو را به صفر مقدار دهی می کند. به هرحال زمانی که شما سازنده ی خود را تعریف می کنید، سازنده ی پیش فرض دیگر مورد استفاده قرار نمی گیرد.

در اینجا مثال ساده ای را می بینید که از یک سازنده استفاده می کند.

مثال

```
// A simple constructor.

class MyClass {

   int x;
   // Following is the constructor

   MyClass() {

      x = 10;

   }

}
```

سازنده را برای آبجکت های اولیه فرا می خوانید، مانند زیر:

```
public class ConsDemo {
    public static void main(String args[]) {

      MyClass t1 = new MyClass();

      MyClass t2 = new MyClass();
```

```
      System.out.println(t1.x + " " + t2.x);

   }

}
```

اکثر اوقات، به سازنده ای نیاز خواهید داشت که یک یا چند پارامتر را می پذیرد. پارامترها به همان روشی به یک سازنده اضافه می شوند که به یک متد اضافه می شوند، تنها کافیست آنها را در داخل پرانتزها و پس از نام سازنده قرار دهید.

در اینجا مثال ساده ای را مشاهده می کنید که از یک سازنده استفاده می کند:

```
// A simple constructor.

class MyClass {

   int x;
   // Following is the constructor

   MyClass(int i ) {

      x = i;

   }

}
```

سازنده را برای مقدار دهی آبجکت ها فرا می خوانید، مانند زیر:

```
public class ConsDemo {
   public static void main(String args[]) {

      MyClass t1 = new MyClass( 10 );

      MyClass t2 = new MyClass( 20 );

      System.out.println(t1.x + " " + t2.x);

   }

}
```

این مثال نتیجه ی زیر را تولید می کند:

```
Variable Arguments(var-args)
```

JDK 1.5شما را قادر به انتقال یک متغیر عددی از argument های هم نوع به یک متود می سازد. پارامتر در متود مانند زیر اعلام می شود:

typeName… parameterName

در اعلام متود، نوع را که با یک (...) دنبال می شود، تعیین می کنید. فقط یک پارامتر variable-length ممکن است در یک متود تعیین شود و این پارامتر باید آخرین پارامتر باشد. هر پارامتر معمول دیگری باید قبل از آن قرار بگیرد.

```
public class VarargsDemo {
   public static void main(String args[]) {

      // Call method with variable args

            printMax(34, 3, 3, 2, 56.5);

      printMax(new double[]{1, 2, 3});

   }


   public static void printMax( double... numbers) {

   if (numbers.length == 0) {

      System.out.println("No argument passed");

      return;

   }


   double result = numbers[0];


   for (int i = 1; i <  numbers.length; i++)

      if (numbers[i] >  result)

      result = numbers[i];

      System.out.println("The max value is " + result);

   }
```

```
}
```

این مثال نتیجه ی زیر را به دنبال دارد The max value is 56.5 The max value is 3.0 :

متد finalize( ) :

ممکن است متودی را تعریف کنید که درست قبل از تخریب نهایی یک آبجکت به وسیله ی garbage collector، فراخوانده خواهد شد. این متود finalize( ) نامیده می شود و می تواند برای اطمینان دادن این مسئله استفاده شود که آیا به درستی به پایان رسیده.

برای مثال: ممکن است از finalize( ) برای اطمینان از بسته شدن یک فایل باز در آن آبجکت استفاده کنید.

هروقت لازم است آبجکتی از آن گروه را بازیابی کنند، زمان اجرای جاوا آن برنامه را فرا می خواند.

در داخل متد finalize( ) ، فعالیت هایی را تعیین می کنید که باید قبل از تخریب یک آبجکت اجرا شوند.

متد finalize( ) دارای این فرم کلی می باشد:

```
protected void finalize( )
{
   // finalization code here
}
```

در اینجا لغت کلیدی محافظت شده یک تعیین کننده است که با کد تعریف شده در خارج گروه آن، مانع دسترسی به finalize( ) می شود.

این به این معناست که شما از زمان اجرای finalize( ) و حتی از اجرای آن مطلع نمی شوید. برای مثال اگر برنامه ی شما قبل از وقوع garbage collection اتفاق بیفتد، finalize( )اجرا نخواهد شد.

این مثال نتیجه ی زیر را به دنبال دارد :

Minimum Value = 6 Minimum Value = 7.3

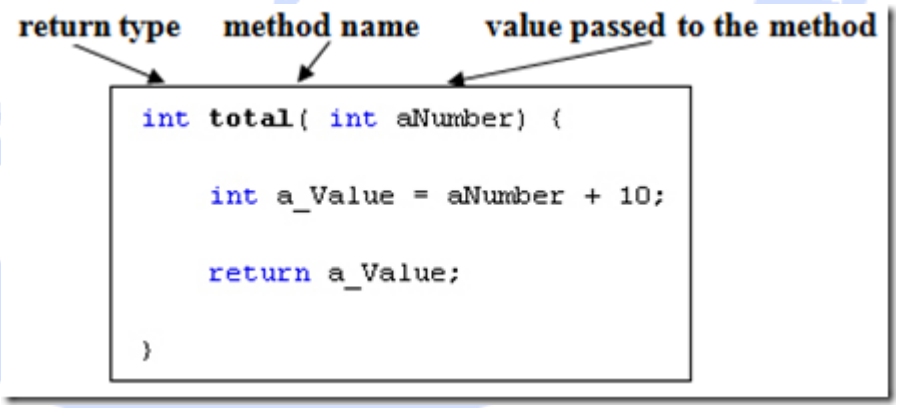
متدهای overloading برنامه را خوانا می سازد. در اینجا دارای یک نام می باشند، اما پارامترهای متفاوت دارند. حداقل تعداد انواع صحیح(integer) و doubleنتیجه می باشد.

آموزش نوشتن متدهای جاوا برای خودتان

در بخش های قبل از متدهایی استفاده کرده اید و مشاهده کرده اید که متدهای داخلی چقدر می توانند مفید باشند. در این بخش چگونگی نوشتن متدهایی را برای خودتان فراخواهید گرفت.

ساختار یک متد

یک متد گروهی کد می باشد که کار خاصی انجام می دهند. اما متدهایی هستند که به روش خاصی تنظیم شده اند. شما یک تیتر متد و یک بدنه ی متد دارید .تیتر جایی است که در آن به جاوا می گویید که چه مقداری را متد باز خواهد گرداند) یک مقدارint ، یک مقدارdouble ، یک مقدار string و غیره .(به همراه نوع مقدار بازگردانده شده به یک نام برای متد خود نیاز دارید که این نام نیز در تیتر قرار می گیرد. می توانید مقادیر را به متدهای خود انتقال دهید و این مقادیر بین دو پرانتز قرار می گیرند. بدنه ی متد جایی است که کد شما قرار می گیرد.



نوع گزارش شده ی متد (return type) در ابتدا قرار می گیرد که در کد بالا یک نوع int می باشد. پس از نوع متد نیاز به یک فاصله دارید که با نام متد شما دنبال می شود. متد بالا را totalنامیده ایم. بین پرانتزها به جاوا گفته ایم که ما در متد متغیری به نام aNumber توزیع می کنیم که یک عدد صحیح خواهد بود.

برای جدا کردن این متد از هر نوع کد دیگری، نیاز به یک جفت کروشه دارید .کد شما برای متد بین کروشه ها قرار می گیرد. به لغت return در متد بالا دقت کنید. مشخص است که نشان دهنده ی مقداری است که می خواهید پس از اجرای کد خود از متد خود گزارش دهید. اما این مقدار باید از همان نوع بازگشتی در تیتر متد باشد. بنابراین اگر متد را با int total شروع کرده باشید، مقدار بازگشتی نمی تواند یک رشته باشد.

گاهی اوقات اصلا نمی خواهید که جاوا موردی را بازگرداند. به Trim در بخش قبل فکر کنید. ممکن است که بخواهید فقط متد Trim به کار خود ادامه دهد و چیزی به شما بازگردانده نشود. یک متد که هیچ مقداری را به شما بازنمی گرداند، می تواند با لغت void تنظیم شود. در برخی موارد نیازی به لغت کلیدی returnنیست. در اینجا متدی را مشاهده می کنید که هیچ مقداری را باز نمی گرداند.

```
void print_text(String someText) {

    System.out.println( "Some Text Here" );

}
```

تمام کاری که متد بالا انجام میدهد چاپ کردن یک متن می باشد. این متد می تواند به کار خود ادامه دهد، بنابراین آن را با عنوان یک متد void تنظیم کرده ایم. هیچ مقدار گزارش داده ای وجود ندارد.

متدها نیاز به مقادیری که به آنها منتقل شود، ندارند. شما می توانید تنها چند کد را اجرا کنید. در اینجا یک متد) void خالی (بدون انتقال هیچ مقداری مشاهده می کنید.

```
void print_text() {

    System.out.println( "Some Text Here" );

}
```

و در تصویر زیر متد int را مشاهده می کنید که دارای هیچ مقداری برای انتقال نمی باشد.

```
int total() {

    int a_Value = 10 + 10;

    return a_Value;

}
```

همان طور که مشاهده می کنید پرانتزها در هر دو متد خالی هستند. اما هنوز لازم می باشند. اگر این پرانتزها را استفاده نکنید، با پیغام خطا روبرو خواهید شد.

در بخش بعد در مورد فراخوانی و فعال کردن متدها فرا خواهید گرفت.

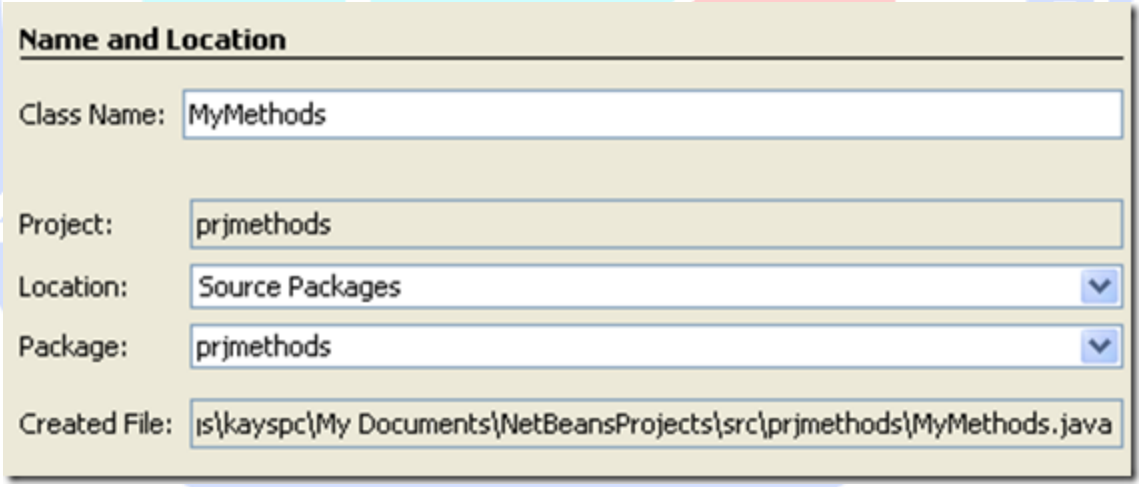
آموزش فراخوانی متد جاوا

متدها تا زمانی که آنها را فرا نخوانده و وارد عمل نکنید، کاری انجام نمی دهند. قبل از مشاهده ی چگونگی انجام این کار، اجازه بدهید گروه دیگری را به پروژه اضافه کنیم. به جای مسدود کردن گروه اصلی، می توانیم همه ی متدها را در آن قرا دهیم.(در مورد گروه ها در بخش بعد بیشتر فرا خواهید گرفت).

یک پروژه ی Java Application جدید آغاز کنید. پروژه ی خود را نام گذاری کرده و متد Main نیز تغییر نام دهید. سپس روی Finish کلیک کنید. در تصویر زیر پروژه ی خود را با عنوان prjmethods و گروه را TestMethodsنام گذاری کرده ایم.

```
package prjmethods;

public class TestMethods {

    public static void main(String[] args) {


    }

}
```

برای افزودن یک گروه جدید به پروژه ی خود، از منوی NetBeans روی File کلیک کنید. از منوی File عبارت New Fileرا انتخاب کنید. یک دیالوگ باکس برای شما ظاهر خواهد شد. در بخش Categories گزینه ی Javaو در بخش File Types گزینه ی Java Class را انتخاب کنید. سپس روی دکمه ی Next در پایین کلیک کنید. در مرحله ی دوم، یک نام برای گروه جدید خود تایپ کنید. ما گروه خود را MyMethods نام نهاده ایم. شما می توانید هر چیز دیگری را به عنوان پیش فرض قرار دهید.



بنابراین ما یک گروه دوم به نام MyMethods ایجاد می کنیم که در پروژه ی prjmethodsخواهد بود. روی دکمه Finish کلیک کنید و پس از آن فایل گروه جدید ایجاد خواهد شد. در نرم افزار NetBeans یک تب جدید با کامنت های پیش فرض در مورد چگونگی تغییر الگوها ظاهر خواهد شد. اگر تمایل داشته باشید، می توانید این کامنت ها را حذف کنید. سپس پنجره ای با کد زیر برای شما ظاهر خواهد شد.

```
package prjmethods;


public class MyMethods {


}
```

موردی که باید به آن توجه کنید این است که این بار متد Main وجود ندارد – تنها با یک گروه خالی با نامی که انتخاب کرده اید و یک جفت آکولاد برای کد شما. اجازه بدهید یکی از متدهای خود را اضافه کنیم. بنابراین کد زیر را به گروه خود اضافه کنید.

```
package prjmethods;

public class MyMethods {

      int total() {
         int a_Value = 10 + 10;

         return a_Value;
      }

}
```

این متد int می باشد که قبلا با نام total به آن پرداختیم. در این متد چیزی بین پرانتزها مبنی بر اینکه قصد توزیع هر مقداری بر روی آن نداریم، وجود ندارد. تمام آنچه متد انجام می دهد افزودن 10 + 10 و ذخیره ی پاسخ در متغیری به نام a_Value می باشد. این مقداری است که از این متد بازگردانده خواهد شد. مقداری که پس از لغت کلیدی گزارش شده است، باید با نوع بازگشتی از تیتر متد هماهنگ باشد. در مورد مثال ما این امر درست می باشد، زیرا هر دو از نوع int می باشند.

(این مسئله مهم می باشد که در ذهن داشته باشید که متغیر a_Value در خارج از متد total دیده نمی شود: هر متغیری که در داخل متد تنظیم شده باشد، در خارج آن متد قابل دسترس نمی باشد. این متغیر با عنوان متغیر local شناخته می شود – در واقع در داخل متد قرار می گیرد)

برای فراخوانی متد total تب TestMethods را در NetBeans انتخاب کنید، موردی با متد .Main قصد داریم متد total را از متد Main فرابخوانیم.

اولین کاری که باید انجام شود، ایجاد یک آبجکت جدید از گروه MyMethods می باشد. خط زیر را به متد Mainخود اضافه کنید.

```
package prjmethods;

public class TestMethods {

    public static void main(String[] args) {

        MyMethods test1 = new MyMethods();

    }

}
```

برای ایجاد یک آبجکت جدید از یک گروه، با نام گروه آغاز کنید، در مورد مثال ما گروه MyMethods نامیده می شود. این به جایint ، double، Stringو غیره می باشد. به عبارت دیگر نوع متغیری که در حال ایجاد آن هستید یک متغیر MyMethodsمی باشد. پس از یک فاصله، یک نام برای متغیر جدید MyMethodsتایپ کنید. ما مورد خود را test1 نامیده ایم.(زیر این مورد خط کشیده شده، زیرا هنوز با آن کاری نکرده ایم. این یک مورد NetBeans می باشد)

یک علامت تساوی بعد از آن قرار می گیرد که با لغت کلیدی new دنبال می شود که به معنای new object می باشد. پس از لغت کلیدی new یک فاصله قرار دهید که مجددا با نام گروه شما دنبال می شود. این بار پس از نام گروه نیاز به یک جفت پرانز دارید. خط را به روش معمول به پایان برسانید، با یک نقطه ویرگول

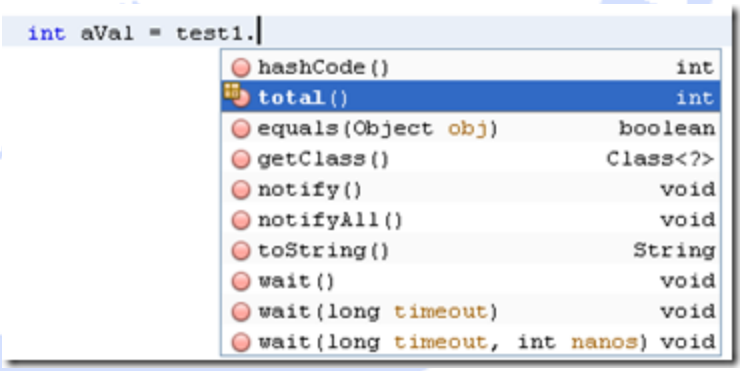
(semi-colon) آنچه در اینجا انجام داده ایم، ایجاد یک آبجکت MyMethods جدید با نام test1می باشد.

اکنون متد total در داخل گروه MyMethods از متد Main مربوط به گروه TestMethods در دسترس خواهد بود.

برای وارد عمل کردن متد، خطوط زیر را اضافه کنید.

```
public class TestMethods {

    public static void main(String[] args) {

        MyMethods test1 = new MyMethods();

        int aVal = test1.total();

    }
}
```

ما در حال تنظیم یک متغیر int با نام aVal می باشیم. پس از علامت تساوی نام گروه ما، test1، قرار می گیرد. برای دسترسی به متدها در گروه یک نقطه تایپ کنید NetBeans .جعبه ای را با متدهای موجود نمایش می دهد.

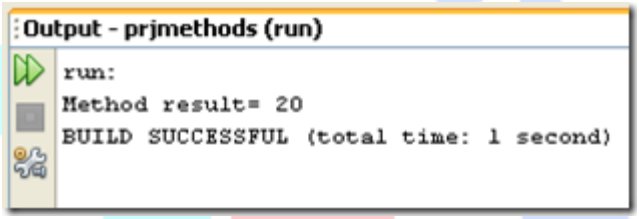


متغیر total در لیست موجود می باشد (دیگر متغیرها در متدها داخلی می باشند.). پرانتزها خالی هستند، زیرا متد ما مقادیر را نمی پذیرد، اما نوع بازگشتی، int، در سمت راست نمایش داده می شود. روی total دابل کلیک کنید تا آن را به کد خود اضافه کنید. سپس در انتهای خط یک نقطه ویرگول تایپ کنید.

در انتها یک خط چاپی اضافه کنید.

```
public class TestMethods {

    public static void main(String[] args) {

        MyMethods test1 = new MyMethods();

        int aVal = test1.total();

        System.out.println( "Method result= " + aVal );

    }
}
```

وقتی که کد اجرا می شود، پنجره ی Output صفحه ی زیر را نمایش خواهد داد.



بنابراین برای فراخوانی یک متد که یک مقدار را بازمی گرداند، دقت کنید که چه نوع مقداری توسط متد شما بازگردانده می شود. سپس این مقدار را به یک متغیر جدید اختصاص دهید، در مورد ما این متغیر aVal می باشد. اما وقتی که یک نقطه پس از نام آبجکت خود تایپ می کنید، متد باید در دسترس باشد.

به هرحال اگر متد شما از نوع void می باشد، نیازی به اختصاص دادن آن به متغیر جدیدی مانند aVal نمی باشد. به عنوان مثال به گروه MyMethods خود بازگشته و متد void را که قبلا بررسی کردید، اضافه کنید.

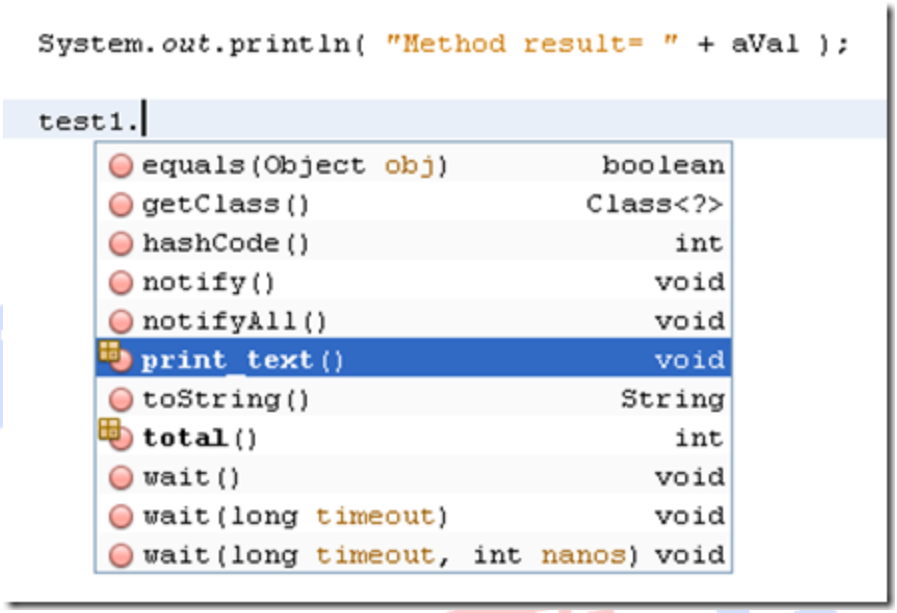
```
package prjmethods;

public class MyMethods {

     int total() {
        int a_Value = 10 + 10;

        return a_Value;
    }

     void print_text() {

        System.out.println( "Some Text Here" );
    }

}
```

این متد جدید print_text نامیده می شود. این متد نیز دارای پرانتزهای خالی می باشد که به آنها هیچ مقداری اختصاص نمی دهیم. تمام کاری که این متد انجام می دهد چاپ کردن متن می باشد.

زمانی که متد void را اضافه کرده اید، به گروه TestMethods بازگردید و خط زیر را اضافه کنید.

```
test1.print_text( )
```

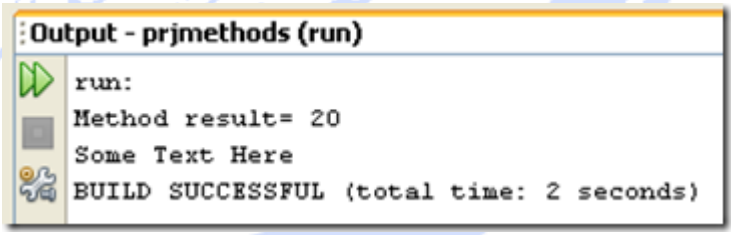
به محض اینکه نقطه را تلیپ کردید، باید متد جدید را مشاهده کنید که روی لیست ظاهر می شود.



اما متدهای ما که اکنون روی لیست هستند عبارتند از total و .print_text مقادیری که این متدها در سمت نمایش می دهند int و void می باشند.

از آنجایی که متد print_text یک متد خالی (void) می باشد، نیازی به تنظیم یک مقدار بازگشتی ندارید. تمام آنچه نیاز دارید نام آبجکت، یک نقطه (dot)و یک متد void می باشد که قصد فراخوانی آن را دارید. سپس جاوا تنها با اجرای کد در داخل متد شما ادامه خواهد داد.

کد خود را اجرا کنید و پنجره ی Output شما باید صفحه ای مانند زیر نمایش دهد.

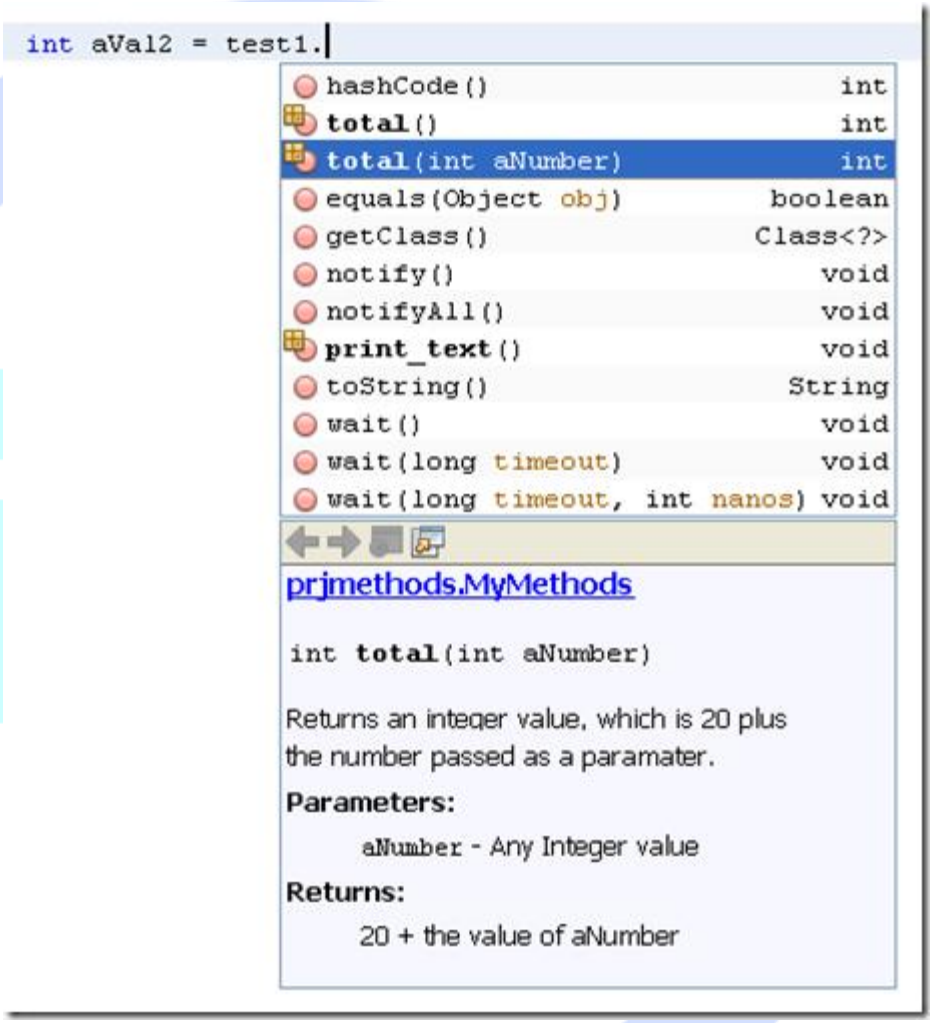


در بخش بعد نگاه دقیق تری به انتقال مقادیر به متدها خواهیم داشت.

آموزش انتقال مقادیر به متدهای جاوا

شما می توانید مقادیر را به متدهای جاوای خود طوری انتقال دهید که با این مقدار کاری انجام شود. این مقدار در بین پرانتزهای متد قرار می گیرد.

به گروه MyMethos خود بازگردید. اکنون دومین متد total را اضافه کنید.



اکنون دو متد هم نام داریم .total :تفاوت بین این دو این است که متد جدید دارای مقداری در بین پرانتزها می باشد. جاوا اجازه ی این کار را به شما می دهد که method overloading نامیده می شود. شما می توانید هر تعداد متد هم نام که می خواهید با هر مقدار بازگشتی داشته باشید. به هرحال نمی توانید یک نوع متغیر مشابه بین پرانتزها داشته باشید. بنابراین نمی توانید دو متد total داشته باشید که مقادیر int را بازمی گرداند که هر دوی آنها دارای مقادیر int در داخل پرانتزها هستند. به عنوان مثال نمی توانید این کار را انجام دهید.

```
int total( int aNumber ) {
        int a_Value = aNumber + 20;
         return a_Value;
        }
        int total( int aNumber ) {
        int a_Value = aNumber + 50;
        return a_Value;
        }
```

گرچه هر دو متد دو کار متفاوت انجام می دهند، هر دو دارای تیترهای یکسان متد می باشند. قبل از اینکه متد جدید خود را امتحان کنید، چند کامنت مستقیما در بالای متد اضافه کنید.

```
public static void main(String[] args) {

    MyMethods test1 = new MyMethods();

    int aVal = test1.total();

    System.out.println( "Method result= " + aVal );

    test1.print_text();

    int aVal2 = test1.total(30);

}
```

مشاهده خواهید کرد که کامنتها در یک لحظه چگونه عمل می کنند. اما param در متدهای فوق مخفف parameterمی باشد. یک پارامتر یک اصطلاح تخصصی برای مقدار بین پرانتزهای تیترهای متد می باشد. پارامتر ما aNumber نامیده می شود و این پارامتر دارای مقادیر صحیح می باشد. به استفاده ی کاراکتر @ قبل از param و returnدقت داشته باشید.

تمام کاری که ما با خود متد انجام می دهیم، انتقال یک مقدار صحیح و افزودن 20 به این مقدار انتقالی می باشد. مقدار بازگشتی مجموع هر دو می باشد.

اکنون به کد خود بازگردید و خط زیر را به آن اضافه کنید.

مثال

```
int aVal2 = test1.total(30);
```

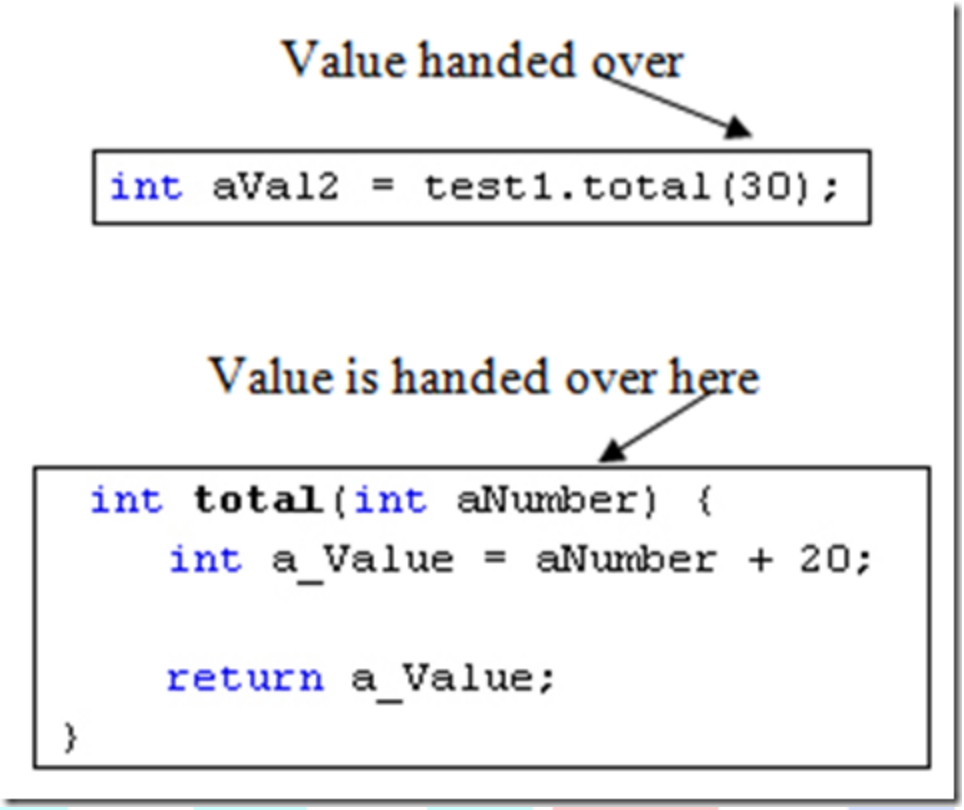
به محض اینکه نقطه را بعد از آبجکت test1 تایپ کردید، لیست پیش رو دوباره مشاهده خواهید کرد. متد جدید شما روی آن خواهد بود. روی متد جدید کلیک کنید تا آن را مشخص کنید و NetBeans مورد زیر را نمایش خواهد داد.

اکنون کامنت های اضافه شده در باکس آبی در زیر لیست متدها نمایش داده می شوند. هرکس دیگری که وارد متد شما شود، باید قادر به تشخیص کاری که انجام می دهد، باشد. خطوط @param و @return از کامنت ها پر شده و حالت bold می شوند.

اما زمانی که متد total2 را اضافه می کنید، عدد 30 را بین پرانتزها تایپ کنید. سپس یک نقطه ویرگول در انتهای خط تایپ کنید. اکنون متد اصلی شما باید مشابه زیر باشد.

```
public static void main(String[] args) {

    MyMethods test1 = new MyMethods();

    int aVal = test1.total();

    System.out.println( "Method result= " + aVal );

    test1.print_text();

    int aVal2 = test1.total(30);

}
```

30تایپ شده بین پرانتزهای متد جدید total تحویل داده شده و در متغیر aNumber قرار می گیرد.



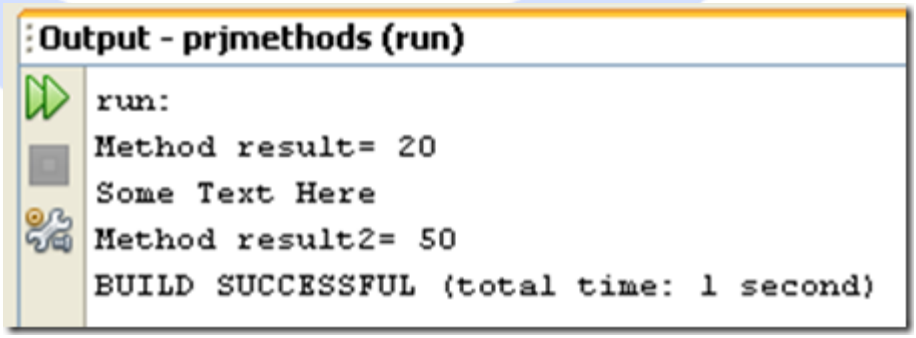
زمانی که مقدار را تحویل می دهید، متد می تواند وارد کار شود.

یک خط چاپی وارد کد خود کنید.

مثال

```
System.out.println( "Method result2= " + aVal2 );
```

سپس برنامه ی خود را اجرا کنید. پنجره ی Output صفحه ی زیر را نمایش می دهد.



بنابراین متدtotal ، 30را به 20 اضافه کرده و سپس پاسخ را به متغیری به نام aVal2 بازمی گرداند.

در بخش بعد، در مورد چگونگی انتقال بیشتر از یک مقدار به متدهای خود را فرا خواهید گرفت.

آموزش انتقال مقادیر چندگانه به متدها

شما می توانید بیشتر از یک مقدار را به متدهای خود انتقال دهید. متدهای زیر را به your MyMethods classاضافه کنید.

```
void print_text(String aString, int aVal) {

    System.out.println( aString  + aVal );

}
```

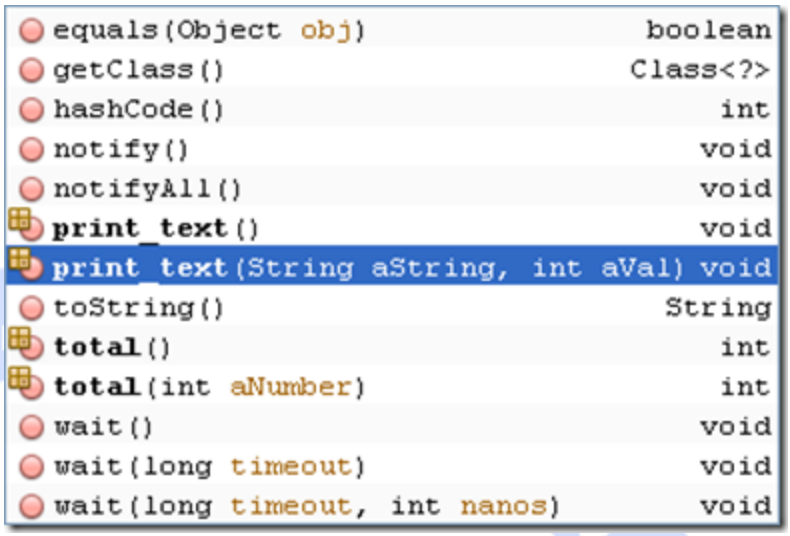
تمام آنچه این متد انجام می دهد، چاپ کردن یک مورد می باشد. بین پرانتزهای مربوط به نام متد، دو مقدار وجود دارد، یک متغیر String به نام aStringو یک متغیر int به نام .aVal وقتی این متد را فرا می خوانیم، ابتدا نیاز به رشته و سپس نیاز به یک عدد داریم. سعی کنید آن را به روش دیگری انجام دهید و پیغام خطا دریافت خواهید کرد.

به گروه TestMethods خود بازگردید و فراخوانی زیر را به متد انجام دهید.

مثال

```
test1.print_text( "The value was ", aVal2 );
```

دوباره متد print_text باید در لیست NetBeans نمایش داده شود.



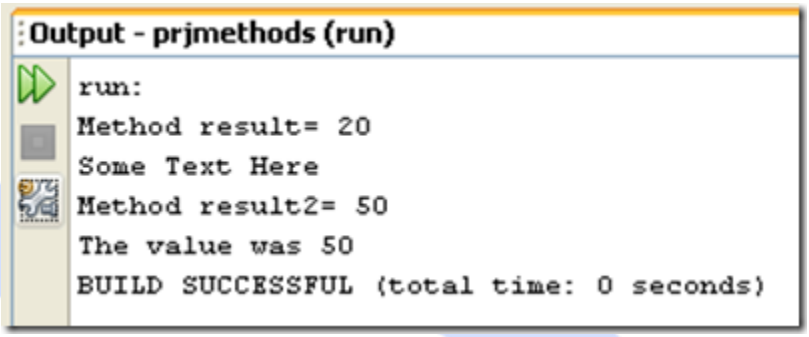
مقادیری (پارامترها) که تعیین می کنیم، همراه با متد بازگشتی، void، بین پرانتزها می باشند.

اما پنجره ی برنامه نویسی اصلی شما اکنون شبیه به تصویر می باشد.

```
public static void main(String[] args) {

    MyMethods test1 = new MyMethods();

    int aVal = test1.total();

    System.out.println( "Method result= " + aVal );

    test1.print_text();

    int aVal2 = test1.total(30);
    System.out.println( "Method result2= " + aVal2 );

    test1.print_text("The value was ", aVal2);

}
```

دو مقداری که انتقال می دهیم، توسط یک کاما مجزا می شوند. توجه داشته باشید که مقدار داخل aVal2 انتقال داده شده باشد. به هرحال نام متغیر داخل پرانتزهای print_text در واقع aVal می باشد. گرچه نام های دو متغیر متفاوت می باشند، اما این مسئله مشکلی ایجاد نمی کند. آنچه شما انجام می دهید انتقال مقادیر به متد می باشد. بنابراین متغیر aVal با همان مقدار aVal2 به پایان می رسد.

اکنون اگر برنامه ی خود را اجرا کنید، باید پنجره ی Output زیر را مشاهده کنید.



ما متدها را رها می کنیم. همانطور که پیش می رویم، متدهای بیشتری خواهیم نوشت، بنابراین شما باید زیاد تمرین کنید. بخش بعد در مورد گروه ها می باشد – برنامه نویسی آبجکت محور.

آموزش کلاس های جاوا

در این بخش چگونگی ایجاد آبجکت ها را فرا خواهید گرفت. یادگیری کلاس ها را تقریبا در بخش قبل آغاز کرده اید. اکنون به جزئیات بیشتری می پردازیم . در برخی موارد در مورد تفاوت بین یک آبجکت و کلاس گیج می شوید، گرچه وقتی درمورد یک کلاس صحبت می کنیم در واقع در مورد خود کد صحبت می کنیم، کدی که کاری انجام نمی دهد. وقتی کد را برای انجام کار فعال می سازید، این کد یک آبجکت می باشد.

وقتی کلاسی ایجاد می کنید، یک کد برای انجام یک کار خاص می نویسید. این کار ممکن است مربوط به یک کارمند باشد، اما فروش شرکت نیز در همان زمان شکل نمی گیرد. برای آمار فروش باید یک کلاس مجزا بنویسید. به این روش می توانید از کلاس کارمند (employee) در یک پروژه ی دیگر استفاده کنید. آمار فروش داده ی برکنار شده (redundant data) می باشند.

وقتی که سعی دارید نظریه های خود را در کلاس ها به کار برید، باید مسئله ی برکناری را در ذهن داشته باشید و از خود بپرسید: " آیا کدی در این کلاس وجود دارد که نیازی به آن در اینجا نیست؟"

در مورد مثال مربوط به این بخش، کلاسی خواهیم نوشت که شامل چند تمرین خواهد بود. روی وب سایت ما (www.tahlildadeh.com)چند امتحان وجود دارد که می توانید به رایگان به آنها دسترسی داشته باشید. در یک زمان سوال داده می شود و هر امتحان دارای 50 سوال می باشد. تمرین های مختلفی وجود دارند که می

توانید به آنها دسترسی داشته باشید. در مورد مثال ما، سوالات را ساده می کنیم. نام شخصی که امتحان می گیرد، ایمکه کدام امتحان گرفته شده، نمره از 50و یک درجه را ثبت می کنیم.

اجازه بدهید که یکی از موارد را شروع کنیم.

برای این مسئله یک پروژه ی جدید جاوا ایجاد کنید. پوشه ی exams را فرا بخوانید و سپس نام متود را از Mainبه ExamDetails تغییر دهید. سپس باید کد زیر را داشته باشید.

```
package exams;

public class ExamDetails {

    public static void main(String[] args) {

    }

}
```

کلاس دومی ایجاد خواهیم کرد تا داده ی امتحان را بررسی کنیم. بنابراین در NetBeans از نوار منو روی File کلیک کنید. از منوی File گزینه ی New File را انتخاب کنید Java. را در لیست Categories و Java Classرا در لیست File Typesمشخص کنید. سپس روی Finish کلیک کنید NetBeans. در پروژه ی شما یک کلاس دوم ایجاد خواهد کرد. شما می توانید کامنت های پیش فرض را حذف کنید.

در بخش بعد، به متغیرهای Field خواهیم پرداخت.

آموزش متغیرهای Field در جاوا

در بخش قبل در مورد متغیرهای داخل متدها صحبت کردیم. متغیرهایی که داخل متدها تنظیم می کنید، تنها در دسترس همان متدها می باشند و در متدهای دیگر قرار نمی گیرند، گفته می شود که دارای دامنه ی داخلی می باشند.

به هرحال می توانید متغیرهایی را در خارج از متدها تنظیم کنید که همه ی متدهای موجود در گروه شما می توانند ببینند. این متغیرها، متغیرهای Field یا متغیرهای (Instance نامیده می شوند. می توانید آنها را دقیقا به روش دیگر متغیرها تنظیم کنید. چهار فیلد زیر را گزوه جدید StudentResult اضافه کنید.

```
package exams;

public class StudentResults {

    String Full_Name;
    String Exam_Name;
    String Exam_Score;
    String Exam_Grade;

}
```

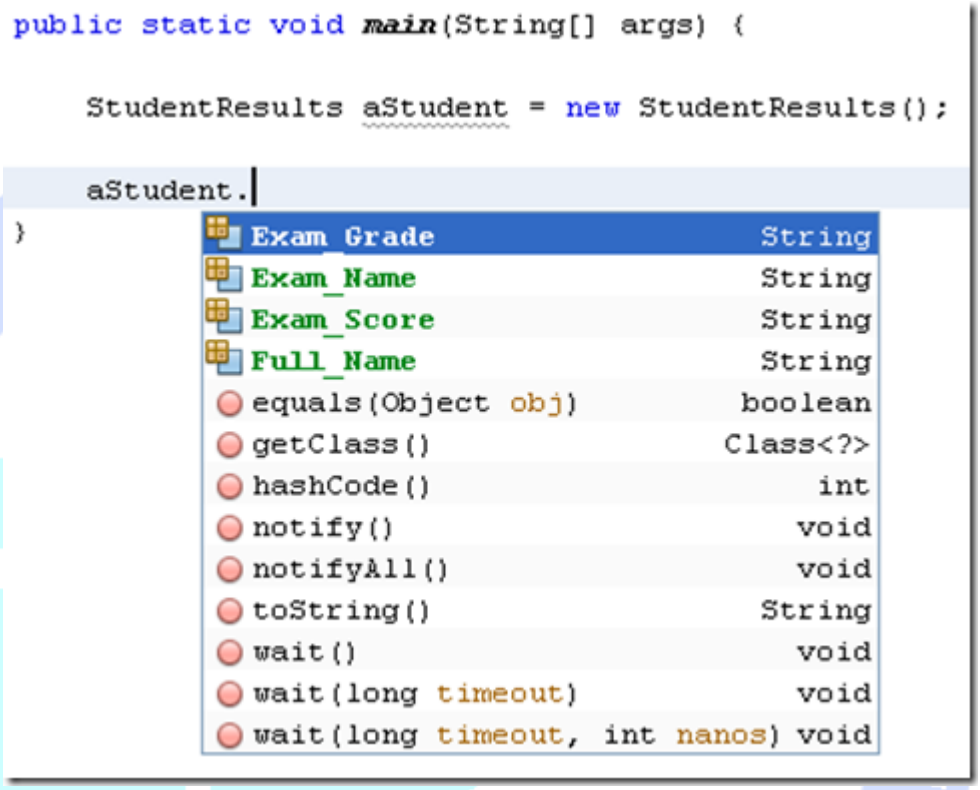
ما چهار رشته متغیر (چهار رشته فیلد) تنظیم می کنیم. همانطور که نام فیلدها نشان می دهد، رشته حاوی نام یک شخص، نام یک امتحان، یک نمره و یک درجه می شود. این چهار فیلد در دسترس همه ی متدهایی که در این گروه نوشتیم، قرار می گیرند و برای متدها داخلی نیستند. گفته می شود که دارای دامنه ی جهانی می باشند.

برای اینکه چگونگی جهانی بودن آنها را مشاهده کنیم، به گروه ExamDetails خود بازگردید، گروهی با متد اصلی .(main) برای ایجاد یک آبجکت جدید از گروه StudentResults ، کد زیر را اضافه کنید.

```
package exams;

public class ExamDetails {

    public static void main(String[] args) {

        StudentResults aStudent = new StudentResults();

    }

}
```

این همان کاری می باشد که در بخش قبل انجام دادهیم – استفاده از یک لغت کلیدی جدید برای ایجاد یک آبجکت جدید. نام آبجکت aStudent خواهد بود و از نوع StudentResults می باشد که گروه ماست.

در خط بعدی نام متغیر (aStudent) را که با یک نقطه (dot) دنبال می شود، تایپ کنید. به محض اینکه نقطه را تایپ کردید، NetBeansلیستی از متدها و پراپرتی های موجود در آبجکت شما را ارائه می دهد.



چهار فیلدی که تنظیم کردیم، روی لیست قرار دارند. اینها متد نیستند، بلکه چیزی به نام پراپرتی می باشند. این واقعیت که آنها روی لیست هستند، به این معناست که دارای دامنه ی جهانی می باشند. اگر دارای دامنه ی داخلی بودند، روی لیست قرار نمی گرفتند.

شما می توانید مقادیری را برای پراپرتی ها تنظیم کنید. این مورد را امتحان کنید: کد مشخص شده ی زیر را به متد اصلی خود اضافه کنید.

```
public class ExamDetails {

    public static void main(String[] args) {

        StudentResults aStudent = new StudentResults();

        aStudent.Exam_Name = "VB NET";

        String exam = aStudent.Exam_Name;
        System.out.println( exam );
    }

}
```

ما فیلد Exam_Name را از لیست انتخاب کرده ایم و به آن مقدار "VB NET" اختصاص داده ایم. سپس خط بعدی مقدار Exam_Name را از آبجکت aStudent می گیرد. نتیجه در متغیری به نام امتحان (exam) ذخیره شده و سپس چاپ می شود .وقتی برنامه را اجرا می کنید، خروجی رشته ی "VB Net" می باشد.

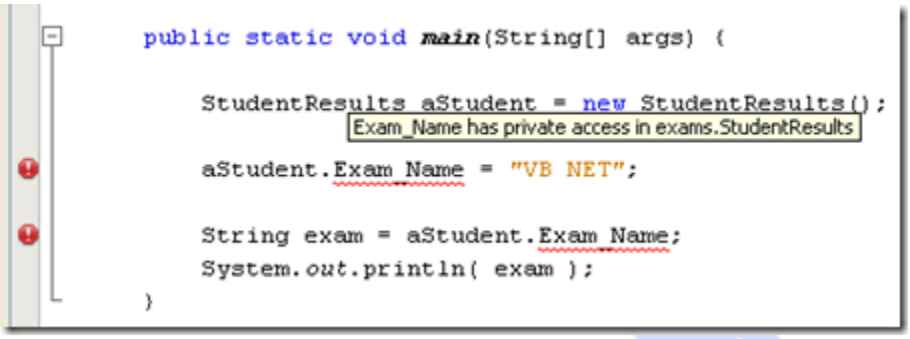
بنابراین چهار متغیری که تنظیم کرده ایم، در هر دو گروه در دسترس می باشند.

به هرحال جهانی ساختن متغیرهای فیلد، مانند این مورد، فکر خوبی نیست .شما تمایل به از دست دادن پیگیری مقادیری را دارید که در این متغیرها می باشند، بنابراین اشکال زدایی کد شما بسیار سخت تر می شود. محدود کردن دامنه ی متغیرهای فیلد، برنامه نویسی خوبی تلقی می شود.

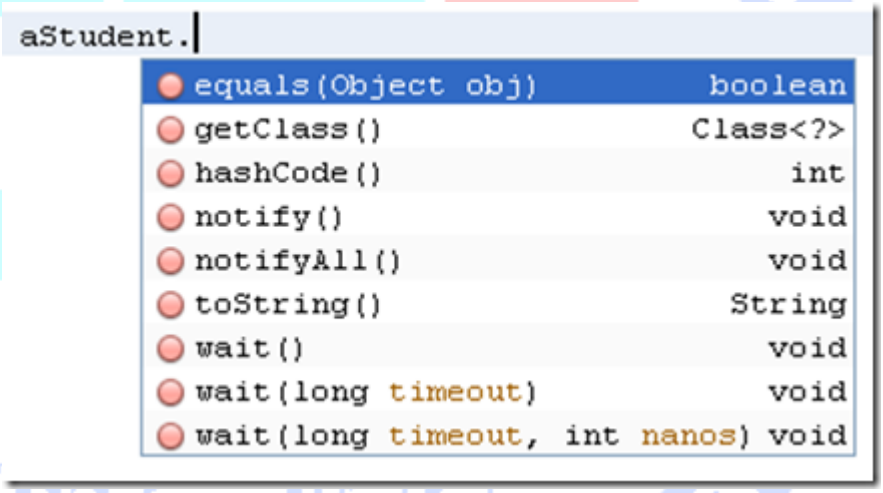
برای اینکه یک متغیر فیلد تنها در دسترس یک گروه خاص باشد، درست قبل از اعلام فیلد لغت کلیدی privateرا وارد کنید. کد را در گروه StudentResults مانند زیر تغییر دهید.

```
package exams;

public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

}
```

اکنون تنها کد موجود در گروه StudentResults می تواند این متغیرها را ببیند. برای بررسی به عقب، به متد اصلی خود بازگردید. باید چند اخطار و خطوط قرمز مشاهده کنید.



سه خط را در پایین حذف کنید aStudent. را و سپس یک نقطه تایپ کنید تا لیست NetBeans را مشاهده کنید.



همانطور که مشاهده می کنید، اکنون چهار متغیر فیلد محو شده اند. علت حذف آنها این است که دیگر دامنه ی جهانی ندارند و بنابراین از گروه ExamDetails قابل مشاهده نیستند.

در بخش بعد در مورد چیزی به نام class constructor خواهید آموخت.

آموزش سازنده در جاوا

سازنده در جاوا

از آنجایی که متغیرهای فیلد را خصوصی ساخته ایم، نیاز به راه دیگری برای اختصاص دادن مقادیر به آنها داریم. یک راه برای انجام آن استفاده از constructor(سازنده) می باشد. این دروافقع متودی است که می توانید برای تنظیم مقادیر اولیه ی متغیرهای فیلد استفاده کنید. وفتی آبجکت ایجاد می شود، ابتدا جاوا constructorرا فرا می خواند. سپس هر کدی که در این سازنده دارید، اجرا می شود. نیازی به هیچگونه فراخوانی خاصی به یک متود constructorندارید – در هنگام ایجاد یک آبجکت این اتفاق به طور خودکار انجام می شود.

متودهای Constructor هم نام گروه می باشند .Constructor زیر را به گروه StudentResults اضافه کنید.

```
package exams;

public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {

    }

}
```

بنابراین نام این Constructor (سازنده) نیز StudentResults می باشد. این دقیقا همان نام گروه می باشد. برخلاف متودهای عادی، گروه constructor ها به نوع بازگشتی مانند int یا double و کلا هیچ گونه گزارشی نیاز ندارد. به هرحال شما می توانید مقادیر را به constructor های خود انتقال دهید. اگر بخواهیم مقادیر را به متغیرهای فیلد انتقال دهیم، می توانیم مانند زیر عمل کنیم.

در اینجا دو متغیر String به پرانتزهای constructor اضافه کرده ایم. در داخل پرانتزها این مقادیر را به فیلدهای Full_Name و Exam_Grade اختصاص داده ایم. وقتی که یک آبجکت جدید ایجاد می کنید، نیاز به دو رشته بین پرانتزهای نام گروه دارید.

```
public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults( String name, String grade ) {

        Full_Name = name;
        Exam_Grade = grade;
    }
```

مثال

```
StudentResults aStudent = new StudentResults( "Bill Gates", "A" );
```

وقتی آبجکت ایجاد می شود، مقادیر "Bill Gates" و "A" در constructor بررسی می شوند.

به هرحال تنظیم چند مقدار پیش فرض برای متغیرهای فیلد فکر خوبی است. این مقادیر در هنگام ایجاد آبجکت، اختصاص داده می شوند. کد زیر را به constructor خود اضافه کنید.

```
package exams;

public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

}
```

اکنون هر وقت یک آبجکت StudentResults جدید ایجاد می شود، هر چهار متغیر فیلد ما دارای مقدار پیش فرض می باشند. دقت کنید که اکنون هیچ چیز بین پرانتزهای گروه constructor وجود ندارد.

در بخش بعد به دسترسی به گروه متغیرها خواهیم پرداخت.

آموزش دسترسی به متغیرهای گروه در جاوا

اکنون که چند مقدار پیش فرض داریم، می توانیم متدی را اضافه کنیم که مقادیر مختلفی را برای آنها تنظیم می کند. متد زیر را به گروه StudentResultsخود اضافه کنید.

```
public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

    String fullName(String aName) {

        Full_Name = aName;
        return Full_Name;
    }

}
```

این متد جدید fullName نامیده می شود و دارای یک متغیر String جدید به نام aName بین پرانتزهای آن می باشد. این متد کار بزرگی انجام نمی دهد و به خاطر سادگی در اینجاست. می توانستیم متدی داشته باشیم که کارهای بیشتری انجام دهد، از جمله چک کردن خطاها، اطمینان حاصل کردن در رابطه با یک مورد مناسب، بررسی رشته های خالی و غیره. اما نکته ی مهم این است که این متد برای فیلد Full_Name یک مقدار تنظیم می کند و این فیلد را به عنوان یک مقدار باز می گرداند. وقتی این متد را فرا می خوانیم، مقدار پیش فرض را برای Full_Name خواهد نوشت و یک مقدار جدید وارد خواهد کرد. هر آنچه در متغیر aName است، مقدار جدید Full_Name خواهد بود. اجازه بدهید این مورد را در عمل ببینیم.

روی back کلیک کنید تا به گروه ExamDetails بازگردید. دو خط زیر را به آن اضافه کنید.

مثال

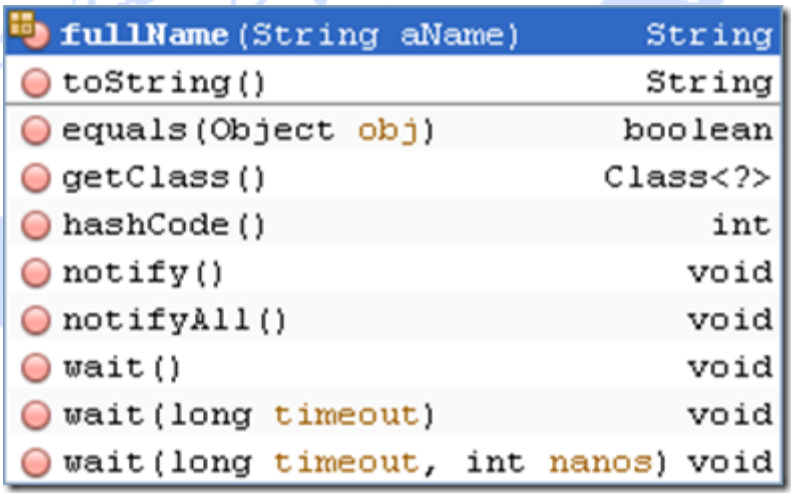
```
String sName = aStudent.fullName("Bill Gates");
System.out.println( sName );
```

کد مربوط به گروه ExamDetails باید مشابه زیر باشد.

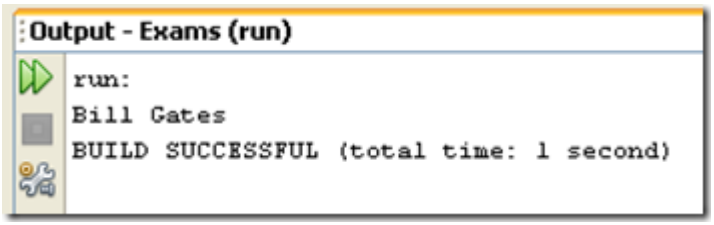
```
package exams;

public class ExamDetails {

    public static void main(String[] args) {

        StudentResults aStudent = new StudentResults();

        String sName = aStudent.fullName("Bill Gates");
        System.out.println( sName );

    }
}
```

آنچه در اینجا انجام می دهیم، فراخوانی متد fullName از آبجکت aStudent می باشد. ما در حال توزیع مقدار "Bill Gates" می باشیم. این مقدارمربوط به فیلد Full_Name نیز خواهد بود. (این مقدار می توانست برای خطاها بررسی شده، اصلاح شود و سپس در یک فیلد ذخیره شود.) سپس مقدار Full_Name بازگردانده شده و در متغیر sName ذخیره می شود.

درست مانند متدهایی که در بخش قبل ایجاد کردیم، گرچه متد fullName روی لیست NetBeans می باشد. به هرحال توجه داشته باشید که constructor در آنجا نیست.



برای امتحان آن کد خود را اجرا کنید. پنجره ی Output باید صفحه ی زیر را نمایش دهد.



بنابراین آنچه انجام داده ایم امتحان کردن یک مقدار برای یک متغیر فیلد در یک گروه به نام StudentResults می باشد. سپس به آن مقدار دسترسی پیدا کرده و آن را چاپ می کنیم.

حالا اجازه بدهید متدی وارد کنیم که واقعا کار مفیدی انجام می دهد. کاری که انجام خواهیم داد این است که به یک یوزر اجازه می دهیم تا یک کد امتحان دو حرفی وارد کند. سپس آن دو حرف را به متدی انتقال خواهیم داد که دو حرف را به نام یک امتحان بازمی گرداند. برای مثال اگر یک یوزر دو حرف "VB" را وارد کند، متد رشته ی "Visual Basic .NET" را گزارش خواهد داد. رشته ی بلندتر در متغیر فیلد Exam_Name ذخیره خواهد شد.

کد زیر را به گروه StudentResults ، درست زیر متد fullName اضافه کنید.

```
String fullName(String aName) {

    Full_Name = aName;
    return Full_Name;
}

    String examName(String examCode) {

        if (examCode.equals("VB")) {
            Exam_Name = "Visual Basic .NET";
        }
        else if (examCode.equals("JV")) {
            Exam_Name = "Java";
        }
        else if (examCode.equals("C#")) {
            Exam_Name = "C# .NET";
        }
        else if (examCode.equals("PH")) {
            Exam_Name = "PHP";
        }
        else {
            Exam_Name = "No Exam Selected";
        }

        return Exam_Name;
}
```

متد examName دارای یک متغیر رشته به نام examCode می باشد که بین پرانتزهای آن قرار می گیرد. این رشته آن دو حرف می باشد. خطوط IF ... ELSE IF بررسی می کنند تا ببینند کدام دو حرف در رشته می باشند. اگر یک هماهنگی برای دو دو حرف پیدا کردیم، تیتر بلندتر در فیلد Exam_Name قرار می گیرد .اگر هیچ گونه هماهنگی پیدا نشد، متن مربوط به فیلد "No Exam Selected" خواهد بود.

به گروه ExamDetails خود بازگشته و خط زیر را به آن اضافه کنید.

مثال

```
String exam = aStudent.examName("VB");
```

دوباره ما در حال فراخوانی متد می باشیم. به حروف "VB" دسترسی داریم . متد مقدار "Visual Basic .NET" را گزارش می دهد و سپس آن را در متغیر رشته ای که آن را exam نامیدیم، ذخیره می کند. یک print lineجدید اضافه کنید، کد شما باید مانند زیر باشد.

```
public class ExamDetails {

    public static void main(String[] args) {

        StudentResults aStudent = new StudentResults();

        String sName = aStudent.fullName("Bill Gates");
        String exam = aStudent.examName("VB");

        System.out.println( sName );
        System.out.println( exam );

    }
}
```

سپس کد خود را اجراکنید، Outoutمشابه زیر می باشد.

```
Output - Exams (run)
run:
Bill Gates
Visual Basic .NET
BUILD SUCCESSFUL (total time: 1 second)
```

این درس را در بخش بعد ادامه خواهیم داد، که صفحه کمی بلندتر می شود.

آموزش متودهای بیشتری در جاوا

بنابراین ما نام یک دانش آموز و نام یک امتحان داریم. هر دو در فیلد اسم ها در گروه StudentResults ذخیره می شوند. اکنون می توانیم یک نمونه ی امتحان نیز ذخیره کنیم.

متد جدید زیر را درست بعد از متد examName در گروه خود اضافه کنید.

```
String examScore(int aScore){

    Exam_Score = aScore + " out of 50";
    return Exam_Score;
}
```

این متد جدید examScore نامیده می شود با یک متغیر int به نام aScore بین پرانتزهای آن. این متد برای بازگرداندن یک مقدار String تنظیم شده است .خود متد نمره را با رشته ی " out of 50" ترکیب می کند. بنابراین اگر مقدار در aScore عدد 30 باشد، متن "30 out of 50" در فیلد Exam_Score ذخیره خواهد شد.

در گروه ExamDetails ، خط زیر را اضافه کنید.

مثال

```
String score = aStudent.examScore(30);
```

بنابراین متد جدید examScore را فرا می خوانیم و به آن مقدار 30 را می دهیم. مقدار در فیلد Exam_Score بازگردانده می شود، و سپس در یک متغیر رشته ذخیره می شود که ما آن را نمره (score) نامیده ایم.

یک متد print line جدید اضافه کنید، بنابراین کد شما شبیه به کد ما در زیر می شود.

```
public class ExamDetails {

    public static void main(String[] args) {

        StudentResults aStudent = new StudentResults();

        String sName = aStudent.fullName("Bill Gates");
        String exam = aStudent.examName("VB");
        String score = aStudent.examScore(30);

        System.out.println( sName );
        System.out.println( exam );
        System.out.println( score );
    }
}
```

وقتی برنامه اجرا می شود، پنجره ی Output شبیه زیر خواهد بود.

```
public class ExamDetails {

    public static void main(String[] args) {

        StudentResults aStudent = new StudentResults();

        String sName = aStudent.fullName("Bill Gates");
        String exam = aStudent.examName("VB");
        String score = aStudent.examScore(30);

        System.out.println( sName );
        System.out.println( exam );
        System.out.println( score );
    }
}
```

بنابراین ما نام دانش آموز، نام امتحان و نمره از 50 را در دست داریم. اکنون به پنجره ی خروجی می توانیم یک درجه نیز اضافه کنیم.

برای درجه از حروف مجزا استفاده می کنیمA : ، B, C ، D ، یا .E اگر دانش اموزی نمره ی 41 یا بیشتر بگیرد، حرف A را به آن اختصاص می دهیم. اگر نمره بین 31 و 40 باشد، درجه مربوطه B خواهد بود. برای نمره ی بین 21 تا 30 حرف C اختصاص داده می شود. درجه ی D نمره ی بین 11 تا 20 می باشد و حرف E برای نمرات بین 0 تا 10 می باشد.

برای محاسبه ی درجات بالا متد زیر را اضافه کنید).آن را به گروه StudentResults خود اضافه کنید(

```
private String getGrade(int aScore) {

    String examGrade = "";

    if (aScore >= 0 && aScore <= 10) {
        examGrade = "E";
    }
    else if (aScore >= 11 && aScore <= 20) {
        examGrade = "D";
    }
    else if (aScore >= 21 && aScore <= 30) {
        examGrade = "C";
    }
    else if (aScore >= 31 && aScore <= 40) {
        examGrade = "B";
    }
    else if (aScore >= 41 ) {
        examGrade = "A";
    }
    return "Grade is " + examGrade;
}
```

دقت داشته باشید که این متد privat(خصوصی) می باشد. درست مانند متغیرهای فیلد، خصوصی ساختن یک متد به این معناست که آن متد تنها در داخل این گروه قابل مشاهده می باشد و می تواند به وسیله ی گروه ExamDetails مشاهده شود.

برای به دست آوردن درجه، متد دیگری را در داخل گروه StudentResults تنظیم خواهیم کرد و از آن برای به دست آوردن درجه استفاده می کنیم. متد زیر را درست در بالای متد getGrade اضافه کنید. (گرچه اگر تمایل داشته باشید، می توانید آن را در زیر این متد نیز اضافه کنید: در جاوا هیچ فرقی نمی کند)

مثال

```
String examGrade(int aScore) {

        Exam_Grade = this.getGrade( aScore) ;
        return Exam_Grade;
        }
```

این متدی است که به جای متد getGrade از گروه ExamDetails فرا خوانی می کنیم. نام این متد جدید examGrade می باشد و مجددا آن را به نمره ی دانش آموز انتقال می دهیم. به این خط دقت کنید.

مثال

```
Exam_Grade = this.getGrade( aScore );
```

در اینجا متد getGrade فراخوانده می شود و آن را به نمره ای انتقال می دهیم که توزیع شده بود. فراخوانی یک متد از متد دیگر یک تمرین استاندارد می باشد، و به شما اجازه می دهد تا کد خود را ساده کنید. جایگزین آن داشتن متدهای خیلی بلند می باشد که خواندن آنها سخت است.

مورد دیگری که در خط بالا باید به آن توجه داشت لغت کلیدی this در جاوا می باشد. لغت کلیدی this به معنای "this class" می باشد و نه گروه دیگری که ممکن است دارای متد هم نام باشد. این امر از هر گونه سردرگمی جلوگیری می کند. این مسئله زیاد ضروری نیست و می توانید آن را نادیده بگیرید. فراخوانی متد هنوز بدون آن کار می کند.

مثال

```
Exam_Grade = getGrade( aScore );
```

گرچه نتیجه ی نهایی هنوز همان است: ما در حال ذخیره سازی چیزی در فیلد متغیر Exam_Grade می باشیم و آن عبارت خواهد بود از عبارت "Grade is" به علاوه ی یک حرف مربوط به درجه.

برای امتحان کردن متدهای جدید، خط زیر را به گروه ExamDetails اضافه کنید.

مثال

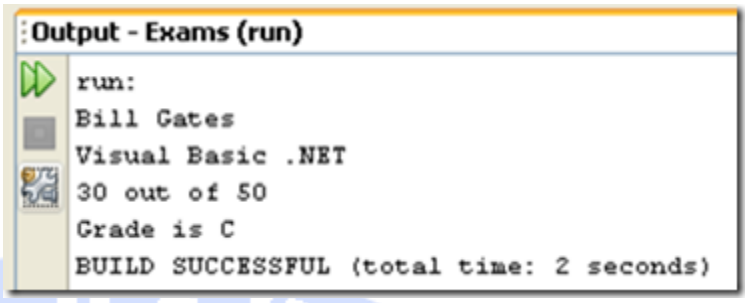
```
String grade = aStudent.examGrade(30);
```

این خط مقداری برابر 30 را به متد examGrade می دهد. سپس در فیلد متغیر Exam_Grade مقداری گزارش می‌شود و در متغیری به نام grade ذخیره می‌شود.

با یک print line ، گروه ExamDetails باید به شکل زیر باشد.

```
public class ExamDetails {

    public static void main(String[] args) {

        StudentResults aStudent = new StudentResults();

        String sName = aStudent.fullName("Bill Gates");
        String exam = aStudent.examName("VB");
        String score = aStudent.examScore(30);
        String grade = aStudent.examGrade(30);

        System.out.println( sName );
        System.out.println( exam );
        System.out.println( score );
        System.out.println( grade );
    }
}
```

برنامه ی خود را اجرا کنید تا پنجره ی Output را مشاهده کنید.



اگر برنامه ی شما به درستی کار نمی کند، در اینجا کد کامل مربوط به گروه StudentResults را مشاهده می کنید.

```
package exams;

public class StudentResults {

    private String Full_Name;
    private String Exam_Name;
    private String Exam_Score;
    private String Exam_Grade;

    StudentResults() {
        Full_Name = "No Name Given";
        Exam_Name = "Unknown";
        Exam_Score = "No Score";
        Exam_Grade = "Unknown";
    }

    String fullName(String aName) {

        Full_Name = aName;
        return Full_Name;
    }

    String examName(String examCode) {

        if (examCode.equals("VB")) {
            Exam_Name = "Visual Basic .NET";
        }
        else if (examCode.equals("JV")) {
            Exam_Name = "Java";
        }
        else if (examCode.equals("C#")) {
            Exam_Name = "C# .NET";
        }
        else if (examCode.equals("PH")) {
            Exam_Name = "PHP";
        }
        else {
            Exam_Name = "No Exam Selected";
        }

        return Exam_Name;
    }
```

```
    String examScore(int aScore){

        Exam_Score = aScore + " out of 50";
        return Exam_Score;
    }

    String examGrade(int aScore) {
        Exam_Grade = this.getGrade(aScore);
        return Exam_Grade;
    }

    private String getGrade(int aScore) {

        String examGrade = "";

        if (aScore >= 0 && aScore <= 10) {
            examGrade = "E";
        }
        else if (aScore >= 11 && aScore <= 20) {
            examGrade = "D";
        }
        else if (aScore >= 21 && aScore <= 30) {
            examGrade = "C";
        }
        else if (aScore >= 31 && aScore <= 40) {
            examGrade = "B";
        }
        else if (aScore >= 41 ) {
            examGrade = "A";
        }
        return "Grade is " + examGrade;
    }

}
```

در بخش بعد در مورد Inheritance فرا خواهید گرفت.

آموزش وراثت در جاوا

مفهوم مهم دیگر در برنامه نویسی Object Oriented (آبجکت محور) مفهوم Inheritanceمی باشد. با چند مثال برنامه نویسی مفهوم Inheritance واضح تر خواهد شد. اما لزوما دارای یک گروه به عنوان گروه اصلی می باشد (که گروه superنامیده می شود.) و یک گروه دیگر به عنوان زیرمجموعه ی گروه اصلی (که گروه subنامیده می شود). گفته می شود که گروه زیرمجموعه از گروه اصلی استخراج شده است. دلیل داشتن یک گروه زیرمجموعه حفظ اطلاعات به صورت مجزا می باشد. گروه زیرمجموعه می تواند تمام متدها و فیلدها را از گروه اصلی خود دریافت کند، اما سپس می تواند کار خود را انجام دهد.

به عنوان یک مثال ازinheritance ، یک گروه زیرمجموعه ایجاد خواهیم کرد که اطلاعات مربوط به تاییدیه ها را بررسی می کند. اگر دانش آموزی رتبه ی "A" دریافت کند، گواهی Excellence Certificate of Excellence به او اعطا خواهیم کرد، اگر دانش آموزی رتبه ی "B" دریافت کند، به او گواهی دستاورد Certificate of) (Achievementاعطا خواهیم کرد. برای هر رتبه ی دیگری هیچ گواهی اعطا نمی گردد. اما نکته ی مربوط به گروه زیرمجموعه، نگهداری جدا از هم داده ی مربوط به گواهی و داده ی مربوط به امتحان می باشد. به هرحال ممکن است تمتیل داشته باشیم به اطلاعاتی در مورد امتحان دسترسی داشته باشیم، اطلاعاتی از قبیل امتحانی که برگزار شده بود. حتی می توانیم به متدهایی دسترسی داشته باشیم که یک نمره را به رتبه تبدیل می کنند و همه مربوط به زیر مجموعه می باشند.

بنابراین با کلیک کردن بر روی File > New File از منوی NetBeans یک گروه جدید ایجاد کنید. وقتی که دیالوگ باکس ظاهر می شود، Javaرا در زیر تیتر Categories و Java Class را زیر File Types کلیک کنید. روی Next کلیک کرده و Certificatesرا به عنوان نام گروه جدید خود وارد کنید. روی Finish کلیک کنید.

وقتی که گروه جدید شما ایجاد شده است، یک فیلد خصوصی String وارد کرده و آن را certificate بنامید. گروه جدید شما باید به شکل زیر باشد.

```
package exams;

public class Certificates {

    private String certificate;

}
```

برای ایجاد یک sub class (زیرمجموعه) از گروه اصلی جاواsuper class ، لغت کلیدی extends استفاده می شود .بنابراین لغت کلیدی "extends" گروه اصلی مورد نظر را که می خواهید گسترش دهید، دنبال می کند. ما می خواهیم که یک گروه زیرمجموعه از StudentResults ایجاد کنیم. گروه StudentResults گروه اصلی خواهد بود و گروه Certificates گروه زیرمجموعه می باشد.

پس از "public class Certificates" در کد خود "extends StudentResults" را اضافه کنید. بنابراین کد شما باید مانند زیر باشد.

```
package exams;

public class Certificates extends StudentResults {

    private String certificate;

}
```

اکنون شما یک زیرمجموعه دارید که کد را از گروه StudentResults دریافت کرده است.

درست مانند گروهStudentResults ، می توانیم برای این گروه Certificates جدید یک سازنده (constructor)ایجاد کنیم. وقتی که از گروه یک آبجکت ایجاد می کنیم، اول از همه جاوا سازنده ی ما را فرا می خواند.

به هرحال تنها یک سازنده می تواند فرا خوانده شود. اگر یک سازنده ی جدید را از گروه Certificates فرا بخوانیم، همه ی آن مقادیر پیش فرض که برای فیلدهای گروه StudentResults تنظیم کردیم، تنظیم

نخواهند شد. برای رسیدن به این، یک لغت کلیدی به نام super وجود دارد. این لغت از گروه اصلی به سازنده یک فراخوانی می فرستد. سازنده ی زیر را به گروه Certificates خود اضافه کنید.

```
package exams;

public class Certificates extends StudentResults {

    private String certificate;

    Certificates() {
        super();
        certificate = "No Certificate Awarded";
    }
}
```

نام سازنده هم نام با گروه می باشد .Certificates :اولین خط کد بین پرانتزها گروه اصلی می باشد) به پرانتزهای بعد از super توجه کنید .(.وقتی که این خط اجرا می شود، همه ی فیلدهای پیش فرض برقرار شده در StudentResults، تنظیم خواهند شد.

خط دوم کد در سازنده یک مقدار پیش فرض به نام certificate برای فیلد String تنظیم می کند.(شما در واقع می توانید بیشتر از یک سازنده تنظیم کنید. چگونگی انجام این کار را در بخش های بعد مشاهده خواهید کرد(

برای امتحان کردن گروه جدید خود، به گروه ExamDetails بازگردید، گروهی با متد .main روی هر کدی که تاکنون داشته اید، کامنت بگذارید. یک راه سریع برای انجام این کار های لایت کردن تمام کد و سپس کلیک کردن روی آیکن comments در نوار ابزار NetBeans می باشد.

برای رهایی از کامنت ها، دوباره کد را های لایت کرده و روی آیکن uncomment کلیک کنید.

اکنون برای ایجاد یک آبجکت جدید از گروه خود، خط زیر را اضافه کنید.

مثال

```
Certificates c1 = new Certificates();
```

پنجره ی کد شما باید به شکل زیر باشد.

```
package exams;

public class ExamDetails {

    public static void main(String[] args) {

//          StudentResults aStudent = new StudentResults();
//          String sName = aStudent.fullName("Bill Gates");
//          String exam = aStudent.examName("VB");
//          String score = aStudent.examScore(30);
//          String grade = aStudent.examGrade(30);
//          System.out.println( sName );
//          System.out.println( exam );
//          System.out.println( score );
//          System.out.println( grade );

            Certificates c1 = new Certificates();

    }
}
```

نام آبجکت c1 می باشد و از نوع Certificates می باشد.

برای بررسی این موضوع که می توانید به متدهایی از گروه StudentResults دسترسی داشته باشید، خط زیر را به آبجکت جدید c1 اضافه کنید.

مثال

```
String exam = c1.examName("VB");
```

این درست همان کاریست که قبلا انجام دادید: فراخوانی متد examName از گروه . StudentResults به

هرحال این بار از آبجکت Certificates به جای آبجکت StudentResults استفاده می کنید.

یک print line اضافه کنید، کد شما باید به شکل زیر باشد.

```
public class ExamDetails {

    public static void main(String[] args) {

//          StudentResults aStudent = new StudentResults();
//          String sName = aStudent.fullName("Bill Gates");
//          String exam = aStudent.examName("VB");
//          String score = aStudent.examScore(30);
//          String grade = aStudent.examGrade(30);
//          System.out.println( sName );
//          System.out.println( exam );
//          System.out.println( score );
//          System.out.println( grade );

            Certificates c1 = new Certificates();

            String exam = c1.examName("VB");
            System.out.println( exam );
    }
}
```

برنامه را اجرا کنید تا پنجره ی Output را به شکل زیر مشاهده کنید.

```
Output - Exams (run)
run:
Visual Basic .NET
BUILD SUCCESSFUL (total time: 1 second)
```

بنابراین متد مربوط به گروه اصلی (the super class) به عمل فرا خوانده شده است. اکنون می توانیم یک متد به گروه زیر مجموعه اضافه کنیم . (the sub class) متد زیر را به گروه Certificates خود اضافه کنید، درست در زیر .constructor

```
package exams;

public class Certificates extends StudentResults {

    private String certificate;

    Certificates() {
        super();
        certificate = "No Certificate Awarded";
    }

    String certificateAwarded(int aScore) {

        String aGrade = examGrade(aScore);

        if (aGrade.equals("Grade is A") ) {
            this.certificate = "Certificate of Excellence";
        }
        else if (aGrade.equals("Grade is B") ) {
            this.certificate = "Certificate of Achievement";
        }
        else if (aGrade.equals("Grade is C") ) {
            this.certificate = "Certificate of Achievement";
        }
        else  {
            this.certificate = "No Certificate Awarded";

        }

        return this.certificate;
    }
}
```

این متد certificateAwarded نامیده می شود و برای بازگرداندن یک مقدار Stringتنظیم شده است. در داخل پرانتزهای مربوط به متد، یک نمونه ی نمره ی آزمون توزیع می کنیم.

اولین خط متد عبارت است از

مثال

```
String aGrade = examGrade(aScore);
```

متد examGrade یک متد در گروه اصلی می باشد و متدی می باشد که در StudentResultsتنظیم کردیم. به یاد داشته باشید که این متد برای بازگردانی یک رتبه و عباراتی مانند"Grade is A" ، "Grade is B"و غیره تنظیم شده بود. اکنون آن را از زیرمجموعه فرا می خوانیم IF Statement . مقدار مربوط به رشته ی aGradeرا برای مشاهده ی آنچه در آن است، بررسی می کند. بسته به مقدار، یک رشته ی جدید بازگردانده می شود ویک تاییدیه ی مشخص به آن اختصاص داده می شودExcellence :، Achievementیا بدون تاییدیه.

روی back برای بازگشت به گروه ExamDetails کلیک کنید و خط زیر را اضافه کنید.

مثال

```
String award = c1.certificateAwarded(50);
```

این خط متد جدید را فرا خوانده و به آن مقدار 50 اختصاص می دهد. نتیجه به رشته ای که award نامیده ایم، بازگردانده می شود.

روش خط چاپ را در کد خود به این روش تطبیق دهید.

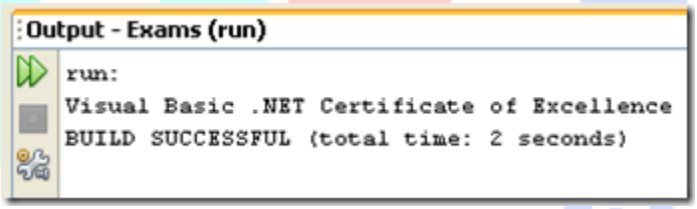
مثال

```
System.out.println( exam + " " + award );
```

گروه ExamDetails باید به شکل زیر باشد. (تمام کامنت ها را حذف کرده ایم)

```
package exams;

public class ExamDetails {

    public static void main(String[] args) {

            Certificates c1 = new Certificates();

            String exam = c1.examName("VB");
            String award = c1.certificateAwarded(50);

            System.out.println( exam  + " " + award );
    }
}
```

و در اینجا پنجره ی Output را در هنگام اجرای برنامه مشاهده می کنید.



بنابراین برای نگهداری جزئیات آزمون جدا از جزئیات مربوط به گواهی، از Inheritanceاستفاده کرده ایم. گروه زیرمجموعه به متدهای گروه اصلی خود دسترسی دارد. بنابراین ما قادر به گرفتن خروجی از هر دو هستیم.

بنابراین Inheritance نمایش فواید گروه اصلی می باشد. شما می توانید با قرار دادن داده در یک گروه زیرمجموعه، آن را مجزا نگه دارید. اما زیرمجموعه به نحوی در ارتباط با گروه اصلی می باشد و می تواند به تمام یا بخشی از کد آن دسترسی داشته باشد.

در بخش بعد به چگونگی بررسی خطاها در جاوا خواهیم پرداخت.

آموزش بررسی خطای جاوا

در حالت کلی خطاها در دو دسته قرار می گیرند: خطاهای طراحی زمان (Design-time)و خطاهای منطقی (Logical). متوقف کردن خطاهای Design-time آسان می باشد، زیرا NetBeans زیر آنها خط می کشد. اگر خطا مانع اجرای برنامه شود، NetBeansزیر آن را با قرمز خط می کشد. خطاهای Logical خطاهایی هستند که شما به عنوان یک برنامه نویس ایجاد می کنید. برنامه اجرا خواهد شد، اما از آنجایی که شنا در برنامه نویسی اشتباه کرده اید، باعث می شود که کل برنامه دچار مشکل شود. به طور مختصر مثال هایی از خطاهای زمان اجرا را مشاهده خواهید کرد. همچنین چگونگی بررسی آنها را نیز فراخواهید گرفت. اما ابتدا به چگونگی رسیدگی جاوا به این خطاها خواهیم پرداخت.

استثنائات(Exceptions)

در جاوا خطاها توسط یک آبجکت Exceptionمورد بررسی قرار می گیرند. گفته می شود که Exceptionوارد می شوند و کار شما گرفتن آنها می باشد. شما می توانید این کار را با بلوک try ... catch انجام دهید. بلوک try ... catchمانند زیر می باشد.

مثال

```
try {
        }
        catch ( ExceptionType error_variable ) {
        }
```

قسمت try از try ... catch به معنای امتحان کردن این کد می باشد. اگر اشتباهی رخ دهد، جاوا وارد بخش catchمی شود. در این بخش جاوا موارد بین پرانتزها را بررسی می کند تا رسیدگی به خطاها را کنترل کند. اگر نوع اصلاح سازی Exception را داشته باشید، هر موردی که بین کروشه های catch می باشد، اجرا خواهد شد. اگر نوع اصلاح سازی Exception را نداشته باشید، جاوا از مورد پیش فرض برای نمایش یک پیغام خطا استفاده می کند.

به عنوان مثال، یک برنامه ی جدید کنسول (console application) ایجاد کنید. آن را هر چه میلید نام گذاری کنید. در کد مربوط به متود Main کد زیر را وارد کنید.

مثال

```
try {
```

```
int x = 10;
int y = 0;
int z = x / y;
System.out.println( z );
}
catch ( Exception err ) {
System.out.println( err.getMessage( ) );
}
```

در بخش try ازcatch try ... ، سه عدد صحیحx ، yو z را تنظیم کرده ایم. سعی داریم y را به x تقسیم کنیم و سپس پاسخ را چاپ کنیم.

اگر اشتباهی رخ بدهد، بخش catch موجود می باشد. بین پرانتزهای این بخش عبارت زیر موجود می باشد.

Exception err

نوع Exception که استفاده می کنید، در ابتدا قرار می گیرد. در این مورد از آبجکت Exception error استفاده می کنیم. این آبجکت یک نوع "catch all" از Exception می باشد و تمرین خوبی برای برنامه نویسی نیست. ما آن را در یک لحظه به یک نوع خاص تغییر خواهیم داد.

پس از نوع Exception خود، یک فاصله و سپس نام یک متغیر را دارید. ما متغیر خود را err نامیده ایم، اما شما می توانید به دلخواه خود آن را نامگذاری کنید.

در کروشه هایcatch ، یک عبارت چاپ داریم. اما به آنچه بین پرانتزهای println وجود دارد دقت کنید.

err.getMessage( )

getMessageمتودی است در دسترس آبجکت های . Exception همانطور که از نام آن پیداست، پیغام خطای مربوط به Exception را دریافت می کند.

برنامه ی خود را اجرا کرده و آن را امتحان کنید. کد شما باید مشابه کد زیر باشد.

```
package errorhandling;

public class errorChecking {

    public static void main(String[] args) {

        try {
            int x = 10;
            int y = 0;
            int z = x / y;

            System.out.println( z );
        }
        catch (Exception err) {
            System.out.println(err.getMessage());
        }
    }
}
```

و پنجره ی Output باید مورد زیر را نمایش دهد.

مثال

```
run:
        / by zero
        BUILD SUCCESSFUL (total time: 1 second)
```

خود خطا، خطایی که به وسیله ی getMessageتولید شد، خط وسط می باشد.

/ by zero

به عبارت دیگر یک تقسیم به وسیله ی خطای صفر. جاوا به شما اجازه ی تقسیم عدد به صفر را نمی دهد، از این رو پیغام خطا می دهد.

کد خود را به شکل زیر تغییر دهید.

مثال

```
double x = 10.0; double y = 0.0; double z = x / y;
```

بقیه ی کد بدون تغییر می باشند. برنامه ی خود را اجرا کرده و آن را امتحان کنید.

مجددا یک پیغام خطا در پنجره ی Output نمایش داده خواهد شد که به شکل زیر می باشد.

مثال

```
run:
        Infinity
        BUILD SUCCESSFUL (total time: 1 second)
```

این بار به خاطر اینکه نتیجه یک عدد خیلی بزرگ می باشد، جاوا برنامه را متوقف خواهد کرد.

خطاهایی که شامل اعداد می شوند، نباید توسط یک نوع Exception "catch all" بررسی شوند. یک نوع خاص به نام ArithmeticException وجود دارد .لغت Exception را از بین پرانتزهای بلوک catch حذف کنید . ArithmeticExceptionرا جایگزین آن کنید. اکنون برنامه را دوباره اجرا کنید.

نباید تفاوتی با خطای نمایش داده شده در پنجره ی Output مشاهده کنید .اما با محدود کردن نوع خطای مورد انتظار، در حال انجام یک تمرین برنامه نویسی خوب می باشید.

در بخش بعد به Stack Trace خواهیم پرداخت.

آموزش عملکرد پشته در جاوا

در جریان عادی یک برنامه، وقتی ماشین مجازی جاوا (Java Virtual Machine) در حال اجرای کد شما می باشد، یک متد بعد از دیگری اجرا خواهد شد که با متد main آغاز می شود. وقتی در ابتدای صف برنامه نویسی نوبت به یک متد رسیده باشد، گفته می شود که متد در راس stack می باشد. پس از اینکه تمام متد اجرا می شود، از stack گرفته شده تا متد دیگر در صف متدها جایگزین آن شود. برای توضیح اصول، کد برنامه ی خود را به شکل زیر تغییر دهید.

```
package errorhandling;

public class errorChecking {

    public static void main(String[] args) {

        System.out.println("Starting Main method");
        m1();
        System.out.println("End Main method");
    }

    static void m1() {
        System.out.println("Method One - m1");
        m2();
    }

    static void m2() {
            System.out.println("Method Two - m2");
    }
}
```

اکنون یک متد Main و دو متد دیگر در دست داریم که عبارتند m1 :و .m2 وقتی که در ابتدا برنامه آغاز می شود، متد Main در بالای stack می باشد. به هرحال در داخل متد Main یک فراخوانی برای متد m1 وجود دارد. وقتی که این متد فراخوانده می شود، در بالای stack قرار می گیرد. پس از آن متد m1 متد m2را فرا می خواند. وقتی که متد m2 فراخوانده می شود، در بالای stack قرا گرفته و m1 را به طور موقت کنر می گذارد. پس از اتمامm2 ، کنترل دوباره به m1بازمی گردد. وقتی m1 تمام می شود، در بالای stack خاموش می شود و کنترل دوباره به متد Mainبازمی گردد.

برنامه ی خود را اجرا کرده و پنجره ی Output را بررسی کنید تا موارد چاپ شده را مشاهده کنید

```
Starting Main method
Method One - m1
Method Two - m2
End Main method
```

اگر در متد m2 اشتباهی رخ دهد، JVMبه جستجوی هرگونه کنترل خطا می پردازد، از جمله یک گروه ... try .catchاگر هیچگونه کنترل خطایی وجود نداشته باشد، Exceptionبه m1 تحویل داده خواهد شد تا رویارویی آن با خطا را مشاهده کنید. در m1 هیچگونه بررسی خطایی وجود ندارد، بنابراین مجددا Exception به stack منتقل می شود، این بار به متد Main منتقل می شود. اگر متد Main با Exception هماهنگ نباشد، یک پیغام خطای عجیب در پنجره ی Output دریافت خواهید کرد. به عنوان یک مثال متد m2 را با مورد زیر تطبیق دهید.

مثال

```
static void m2( ) {
        int x = 10;
        int y = 0;
        double z = x / y;
        System.out.println( z );
        System.out.println("Method Two - m2");
        }
```

متد دوباره حاوی خطای تقسیم بر صفر می باشد. اکنون کد شما باید مشابه کد ما در زیر باشد.

```
Starting Main method
Method One - m1
Method Two - m2
End Main method
```

برنامه را اجرا کرده و مشاهده کنید که در پنجره ی Output چه اتفاقی می افتد.

```
Starting Main method
Method One - m1
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at errorhandling.errorChecking.m2(errorChecking.java:20)
        at errorhandling.errorChecking.m1(errorChecking.java:14)
        at errorhandling.errorChecking.main(errorChecking.java:8)
Java Result: 1
```

آنچه مشاهده می کنید stack trace نامیده می شود. سه خط آبی که زیر آنها خط کشیده شده به متد شما و جایی که یافت می شوند اشاره دارند.

```
package_name.class_name.method_name
```

مورد بالا جایی است که ابتدا خطا در آن اتفاق می افتد، در .m2 جاوا در جستجوی این مورد بود تا توسط یک ArithmeticExceptionکنترل شود که جایی است که در آن تقسیمات بر صفر گرفته می شوند. در m1 ، m2 و یا main هیچ بررسی خطایی وجود نداشت. بنابراین برنامه به کنترل کننده ی خطای پیش فرض خروجی می دهد.

متد m1 را به شکل زیر تغییر دهید.

مثال

```
try {
        System.out.println("Method One - m1");
        m2( );
        }
        catch (ArithmeticException err) {
        System.out.println( err.getMessage( ) );
        }
```

اکنون متد m2 را در یک بلوک try قرار داده ایم. در بخش catch از نوع Exceptionاستفاده کرده ایم که در stack trace – ArithmeticExceptionگزارش شد.

دوباره کد را اجرا کنید، پنجره ی Output صفحه ی زیر را نمایش خواهد داد.

```
Starting Main method
Method One - m1
/ by zero
End Main method
```

دقت داشته باشید که پیغام خطا با عنوان "/ by zero" چاپ شده است. تمام متد m2 اجرا نشد، اما در جایی که خطا اتفاق افتاد، متوقف شد. سپس کنترل به m1بازگردانده شد. از آنجایی که یک بلوک catch برای رسیدگی به خطا وجود داشت، JVMنیازی به کنترل کننده ی پیش فرض ندید، اما پیغام را بین کروشه های catchچاپ کرد.

به هرحال خود برنامه متوقف نشد. کنترل به متد Main بازمی گردد که در آن متد m1 فراخوانده می شود. خط آخر در متد Main که "End Main method" را چاپ کرده، اجرا شد. این برنامه تاثیرات مهمی دارد. فرض کنید که شما به مقداری از m1 نیاز داشتید، زیرا تصمیم داشتید با آن در Main کاری انجام دهید .مقدار در آنجا نیست و ممکن است برنامه ی شما به شکلی که انتظار می رود رفتار نکند.

اما اگر در پنجره ی Output یک stack trace مشاهده می کنید، به یاد داشته باشید که اولین خط جایی است که مشکل در آن اتفاق افتاد. بقیه ی خطوط جایی است که Exception به stack تحویل داده می شود، که معمولا در متد main تمام می شود.

در بخش بعد به Logic Errors (خطاهای منطقی) خواهیم پرداخت.

آموزش خطاهای منطقی در جاوا

خطاهای منطقی خطاهایی هستند که شما به عنوان یک برنامه نویس دچار آنها می شوید، وقتی کد به شکلی که انتظار دارید اجرا نمی شود. بررسی این خطاها ممکن است دشوار باشد. خوشبختانه NetBeans دارای ابزار داخلی برای کمک به شما در کنترل این مشکل می باشد.

ابتدا کد زیر را امتحان کنید.

```
package errorhandling2;

public class errorChecking2 {

    public static void main(String[] args) {
        int LetterCount = 0;
        String check_word = "Debugging";
        String single_letter = "";
        int i;

        for (i = 0; i < check_word.length(); i++) {

            single_letter = check_word.substring(1, 1);

            if (single_letter.equals("g")) {
                LetterCount++;
            }
        }

        System.out.println("G was found " + LetterCount + " times.");
    }
}
```

با استفاده از برنامه ای که دارید یا با شروع یک پروژه ی جدید، آن را برای خود تایپ کنید. کاری که در اینجا سعی داریم انجام دهیم شمردن حرف g در لغت "Debugging" می باشد. مشخصا پاسخ 3 می باشد. به هرحال وقتی برنامه را اجرا می کنید، پنجره ی Output عبارت زیر را چاپ می کند.

```
"G was found 0 times."
```

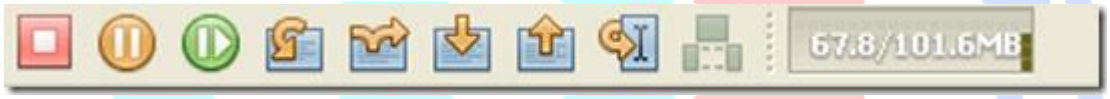
بنابراین ما جایی در کد خود خطایی انجام داده ایم. اما کجا؟ برنامه خوب اجرا می‌شود و هیچ Exceptionsبرای ما در پنجره ی Output ارائه نمی دهد .بنابراین ما چه کار می توانیم بکنیم؟

برای بررسی مشکلات مربوط به کد خود، NetBeansبه شما اجازه ی اضافه کردن چیزی به نام Breakpoint را می دهد.

برای افزودن یک Breakpoint جدید، روی حاشیه ی پنجره ی مربوط به کد کلیک کنید.

```
package errorhandling2;

public class errorChecking2 {

    public static void main(String[] args) {
        int LetterCount = 0;
        String check_word = "Debugging";
        String single_letter = "";
        int i;

        for (i = 0; i < check_word.length(); i++) {
```

از منوی NetBeans ، روی Debug errorhandling2 > Debug کلیک کنید (یا هر آنچه پروژه ی خود را نامیده اید NetBeans . (. وارد breakpoint خواهد شد . این برنامه حالا اجرای کد را متوقف کرده است. همچنین باید یک نوار ابزار ظاهر شده ی جدید مشاهده کنید.



سه دکمه ی اول به شما اجازه ی متوقف کردن کامل یا موقت و ادامه دادن بخش عیب زدایی (debugging) را می دهد. پنج دکمه ی دیگر اجازه ی وارد شدن به کد، رد شدن از کد، خروج از کد یا رسیدن به مکان نما می دهد.

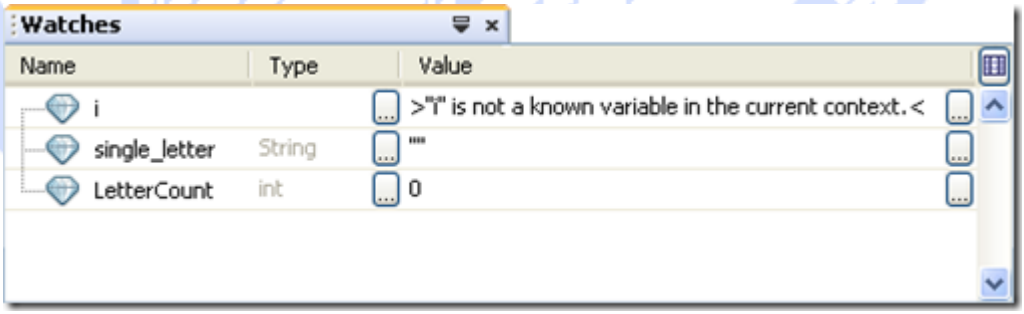
می توانید با فشار دادن دکمه F5 ادامه دهید. کد با Breakpoint نیز باید مانند قبل طبیعی اجرا شود. بنابراین بخش عیب زدایی به پایان خواهد رسید.

وقتی بخش عیب زدایی به پایان می رسد، در Breakpoint روی back کلیک کنید تا از آن رهایی یابید. اکنون به for loop یک Breakpoint اضافه کنید.

```
public class errorChecking2 {

    public static void main(String[] args) {
        int LetterCount = 0;
        String check_word = "Debugging";
        String single_letter = "";
        int i;

        for (i = 0; i < check_word.length(); i++) {

            single_letter = check_word.substring(1, 1);
```

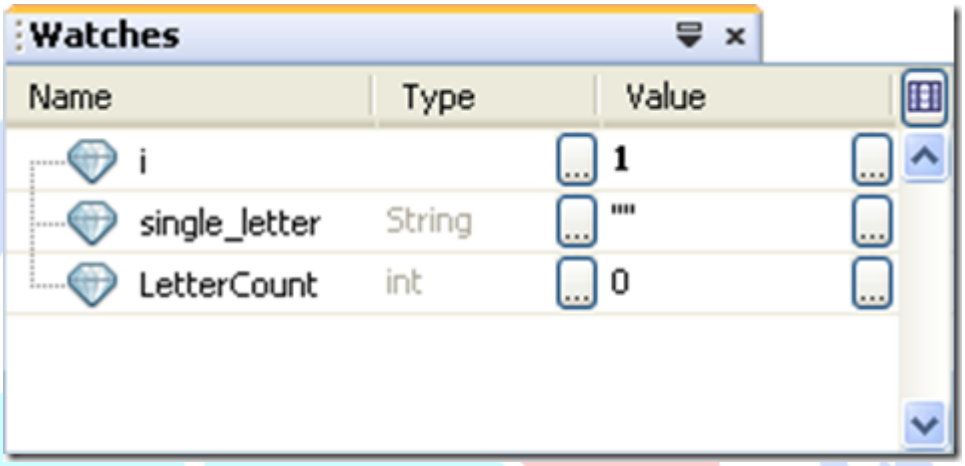
اکنون روی Debug > New Watch کلیک کنید. یک Watch به شما اجازه ی پیگیری آنچه در یک متغیر است را می دهد. بنابراین حرف i در دیالوگ باکس Watch تایپ کرده و روی OK کلیک کنید.

یک Watch دیگر اضافه کرده و سپس single_letter را تایپ کنید. روی OK کلیک کنید Watch .سوم را اضافه کرده و LetterCount را تایپ کنید. شما باید سه متغیر زیر را در پایین صفحه ی خود داشته باشید.



اکنون در نوار ابزار روی آیکن Step Into فشار دهید.

یا فقط دکمه ی F7 را فشار دهید. به فشار دادن دکمه ی F7 ادامه دهید تا مشاهده کنید در پنجره ی Watch چه اتفاقی می افتد. باید متوجه شده باشید که متغیر i هر بار 1 واحد افزایش می یابد. اما دو متغیر دیگر بدون تغییر باقی می مانند.



از آنجایی که در متغیر تک حرفی چیزی وجود ندارد، بنابراین LetterCount نمی تواند بیشتر از صفر باشد. بنابراین مشکل خود را یافته ایم – استفاده از یک رشته ی زیرمجموعه (substring) ممکن است اشتباه باشد، چرا که هیچ کاراکتری نمی گیرد.

بخش عیب زدایی را متوقف کنید و خط رشته ی زیر مجموعه (substring) را مانند زیر تغییر دهید.

```
single_letter = check_word.substring( i, i + 1 );
```

اکنون عیب زدایی را دوباره آغاز کنید. به فشار دادن دکمه ی F7 ادامه دهید تا روی خطوط مربوط به for loop بروید. این بار باید تغییر متغیرهای sinle_letterو LetterCountرا مشاهده کنید.

وقتی کد به پایان می رسد، باید پنجره ی Output را مشاهده کنید که عبارت زیر را نمایش می دهد.

```
"G was found 3 times."
```

اکنون ما پاسخ صحیح را داریم.

بنابراین اگر موارد طوریکه با کد شما طراحی شدند، پیش نمی روند، تنظیم یک Breakpoint و چند Watch را برای متغیرهای خود امتحان کنید. سپس یک بخش اشکال زدایی را آغاز کنید.

در بخش بعد به بررسی یک موضوع متفاوت خواهیم پرداخت: چگونگی باز کردن فایل های متن در جاوا.

آموزش چگونگی خواندن فایل متن در جاوا

دستکاری فایل های متن مهارتی است که در کار برنامه نویسی به خوبی با آن آشنا می شوید. در این بخش در مورد چگونگی باز کردن و نوشتن یک فایل متن فرا خواهید گرفت. اما منظور ما از فایل متن فایلی با متن داخل آن می باشد – متن ساده. می توانید در ویندوز کامپیوتر در برنامه هایی مانندNotepad ، TextEditدرMac ، Geditدر محیط Linux/Gnome یک فایل متن ایجاد کنید.

اولین کاری که انجام خواهیم داد باز کردن یک فایل متن و خواندن محتوای آن می باشد.

خواندن فایل متن: (Text File)

برای این کار یک پروژه ی جدید آغاز کنید. پوشه ی textfilesو گروه FileDataرا فرا بخوانید. درست در زیر خط پوشه و قبل از نام گروه یک عبارت مهم اضافه کنید:

```
import java.io.IOException;
```

پنجره ی کد گذاری شده ی شما مانند زیر خواهد بود:

```
package textfiles;
import java.io.IOException;

public class FileData {

    public static void main(String[] args)  {

    }

}
```

برای مقابله با اشتباهات، عبارت زیر را به متود اصلی اضافه کنید(متن در حالت bold نوشته شده است:)

```
public static void main(String[ ] args) throws IOException {
}
```

ما به جاوا می گوییم که متود اصلی (main) خطای IOException را ایجاد می کند و اینکه این برنامه باید با آن مقابله کند. بعدها برای نمایش یک پیغام خطای مناسب برای یوزر بلوک try ... catch را اضافه می کنیم، البته باید اشتباهی رخ دهد.

برای باز کردن فایل متن اجازه بدهید یک گروه جدید ایجاد کنیم. بنابراین از منوی NetBeans در بالا روی File > New File کلیک کنید. یک فایل جدید Java Class ایجاد کرده و آن را ReadFile بنامید. وقتی این گروه جدید ایجاد شد، سه عبارت مهم زیر را اضافه کنید:

```
import java.io.IOException;
import java.io.FileReader;
import java.io.BufferedReader;
```

بنابراین گروه جدید شما باید مشابه زیر باشد:

```
package textfiles;

import java.io.IOException;
import java.io.FileReader;
import java.io.BufferedReader;

public class ReadFile {

}
```

(زیر خطوط وارد شده خط کشیده شده است، زیرا هنوز با آنها هیچ کاری انجام نداده ایم. این یک ویژگی NetBeans می باشد.)

از این گروه یک آبجکت جدید برای خواندن یک فایل ایجاد می کنیم. سازنده ی (constructor) زیر را به همراه یک فیلد خصوصی String به نام path به کد خود اضافه کنید:

```
public class ReadFile {

    private String path;

    public ReadFile(String file_path) {
        path = file_path;
    }
}
```

تمام کاری که در اینجا انجام می دهیم، انتقال به نام یک فایل و سپس توزیع نام فایل به مسیر فیلد می باشد.

اکنون کاری که لازم است انجام دهیم ایجاد متودی است که تمام خطوط کد را از فایل متن باز می گرداند. این خطوط در یک ردیف حفظ می‌شوند. بیانیه ی زیر را که فایل را باز می کند، اضافه کنید:

```
public class ReadFile {

    private String path;

    public ReadFile(String file_path) {
        path = file_path;
    }

    public String[] OpenFile() throws IOException {


    }
}
```

نگران خطوط قرمز نباشید: این خطوط با اضافه کردن کد جدید از بین خواهند رفت NetBeans .آن را به این دلیل اضافه کرده که عبارت بازگشتی نداشته ایم.

دقت کنید که متود بر اساس بازگرداندن یک String array از طریق زیر تنظیم شده است:

```
public String[ ]
```

این) array ردیف (حاوی تمام خطوط فایل متن خواهد بود.

همچنین دقت کنید که به انتهای تیتر متود عبارت "throws IOException" را اضافه کرده ایم. هر متودی که مربوط به خواندن فایل های متن می شود به یکی از این موارد نیاز دارد. جاوا هر خطایی را در بالای خط وارد می کند، و این خطاها در متود اصلی (main) گرفته خواهند شد.

برای خواندن کاراکترهای فایل متن FileReader استفاده می شود. این برنامه بایت ها را از یک فایل می خواند که هر بایت یک کاراکتر مجزا می باشد. شما می توانید به جای خواندن کاراکترهای مجزا، تمام متن را بخوانید. برای انجام این کار می توانید FileReader را در جایی به نام BufferedReader توزیع کنید . BufferedReaderدارای متودی به نام ReadLine می باشد. همانطور که از نام این متود پیداست، برای خواندن تمام خطوط و نه برای کاراکترهای مجزا استفاده می شود. آنچه BufferedReader انجام می دهد، ذخیره کردن کاراکترها در حافظه (buffer) می باشد طوری که به سادگی قابل اجرا باشند.

خطوط زیر را اضافه کنید که FileReader و BufferedReader را تنظیم می کند:

```
public String[] OpenFile() throws IOException {

    FileReader fr = new FileReader(path);
    BufferedReader textReader = new BufferedReader(fr);

}
```

در اینجا در حال ایجاد دو آبجکت هستیم: یکی آبجکت FileReader می باشد که ما fr نامیده ایم؛ دیگری نیز آبجکت BufferedReader می باشد که textReader نامیده می شود.

FileReaderبرای باز کردن نیاز به نام فایل دارد. برای ما، مسیر و نام فایل در متغیر فیلد به نام path نگهداری می شود.

BufferedReaderآبجکت FileReader خود را پرانتزها قرار می دهد. همه ی کاراکترهای فایل نیز در حافظه نگهداری می شوند که منتظر اجرا می باشند .آنها تحت عنوان نام متغیر textReader حفظ می شوند.

قبل از اینکه بتوانیم خطوط متن را بخوانیم، نیاز به تنظیم یک array داریم. هر موقعیت در array نیز می تواند یک خط کامل از متن را در خود حفظ کند. بنابراین دو خط زیر را به کد خود اضافه کنید:

مثال

```
int numberOfLines = 3;

String[ ] textData = new String[numberOfLines];
```

اکنون تعداد خطوط فایل متن را فقط تا 3 تنظیم خواهیم کرد. واضح است که فایل های متن می توانند هر تعداد خط را در خود داشته باشند، و معمولا ما تعداد را نمی دانیم. بنابراین این مسئله را خیلی زود تغییر خواهیم داد و یک متود مجزا خواهیم نوشت که تعداد خطوط را در یک فایل متن به دست می آورد.

دومین خط از کد جدید یک String array تنظیم خواهد کرد. تعداد موقعیت های یک array (اندازه ی آن)، به تعداد خطوط تنظیم می شود. این عدد را بین کروشه قرار می دهیم.

برای قرار دادن تمام خطوط فایل در موقعیت های مختلف یکarray ، نیاز به یک loop داریم Loop . هر خط از متن را گرفته و هر خط را در یک array قرار می دهد. خطوط زیر را به کد خود اضافه کنید:

مثال

```
int i;

for (i=0; i < numberOfLines; i++) {

textData[ i ] = textReader.readLine();

}
```

پنجره ی کد گذاری شما مانند زیر خواهد بود:

```
public String[] OpenFile() throws IOException {

    FileReader fr = new FileReader(path);
    BufferedReader textReader = new BufferedReader(fr);

    int numberOfLines = 3;
    String[] textData = new String[numberOfLines];

    int i;

    for (i=0; i < numberOfLines; i++) {
        textData[i] = textReader.readLine();
    }

}
```

for loop از 0 شروع می شود و تا کمتر از خطوط خطوط می باشد. (به یاد داشته باشید که موقعیت های array ها در 0 شروع می شوند. سه خط در موقعیت های 0، 1 و 2 ذخیره خواهند شد.)

خطی که به خطوط متن دسترسی یافته و آنها را در یک array ذخیره می کند، خط زیر می باشد:

```
textData[i] = textReader.readLine( );
```

بعد از علامت تساوی خط زیر را خواهیم داشت:

```
textReader.readLine( );
```

آبجکت textReader را که تنظیم کزده ایم، تمام کاراکترهای فایل متن را در حافظه (the buffer) حفظ می کند. می توانیم از متود readLine برای خواندن یک خط کامل از حافظه استفاده کنیم. پس از خوانده شدن خط، خط را در موقعیت یک array ذخیره می کنیم:

```
textData[i]
```

متغیری به نام i هر بار loop را افزایش خواهد داد، بنابراین وارد کل خطوط ذخیره شده ی متن در array می شود.

اکنون فقط دو خط دیگر از کد باید به متود اضافه شوند. بنابراین این خطوط را به کد خود اضافه کنید:

```
textReader.close( );

return textData;
```

خط گزارش شده، تمام array را گزارش می دهد. دقت کنید که برای نام array نیازی به کروشه نیست .وقتی کد را اضافه کرده اید، تمام خطوط زیر کد باید محو شوند و متود شما باید شبیه به تصویر زیر باشد :

```
public String[] OpenFile() throws IOException {

    FileReader fr = new FileReader(path);
    BufferedReader textReader = new BufferedReader(fr);

    int numberOfLines = 3;
    String[] textData = new String[numberOfLines];

    int i;

    for (i=0; i < numberOfLines; i++) {
        textData[i] = textReader.readLine();
    }

    textReader.close();
    return textData;
}
```

به هرحال هنوز مشکل تعداد خطوط وجود دارد. در اینجا این را به 3 تنظیم کرده ایم. آنچه نیاز داریم این است که وارد فایل متن شده و خطوط آن را بشمریم. بنابراین متود زیر را به گروه ReadFile اضافه کنید:

```
int readLines() throws IOException {

    FileReader file_to_read = new FileReader(path);
    BufferedReader bf = new BufferedReader(file_to_read);

    String aLine;
    int numberOfLines = 0;

    while (( aLine = bf.readLine()) != null) {
            numberOfLines++;
    }
    bf.close();

    return numberOfLines;

}
```

متد جدید readLines نامیده می شود و طوری تنظیم شده تا یک مقدار صحیح گزارش دهد. این عدد صحیح تعداد خطوط یک فایل متن می باشد. دقت کنید که این متود دارای یک بخش IOException در تیتر متود می باشد.

کد برای متد یک FileReader دیگر و BufferedReader دیگر تنظیم می کند .Loop خطوط متن را گرد کرده و سپس عبارات زیر را داریم:

مثال

```
while ( ( aLine = bf.readLine( ) ) != null ) {

numberOfLines++;

}
```

while loopکمی گیج کننده به نظر می رسد. اما در واقع فقط می گوید " هر خط از متن را خوانده و وقتی به یک مقدار تهی (null value) می رسید، توقف کنید." (اگر خط زیادی در یک فایل متن وجود نداشته باشد، جاوا یک مقدار تهی را گزارش می دهد.) در داخل آکولادها یک شمارشگر به نام numberOfLines را وارد کرده ایم.

دو خط آخر کد حافظه ی بافر به نام bf را تحت فشار می گذارد، و تعداد خطوط را گزارش می دهد.

برای وارد کردن این متود به مرحله ی عمل، این خط را در متود OpenFile خود تغییر دهید:

```
int numberOfLines = 3;
```

آن را به شکل زیر تغییر دهید:

```
int numberOfLines = readLines( );
```

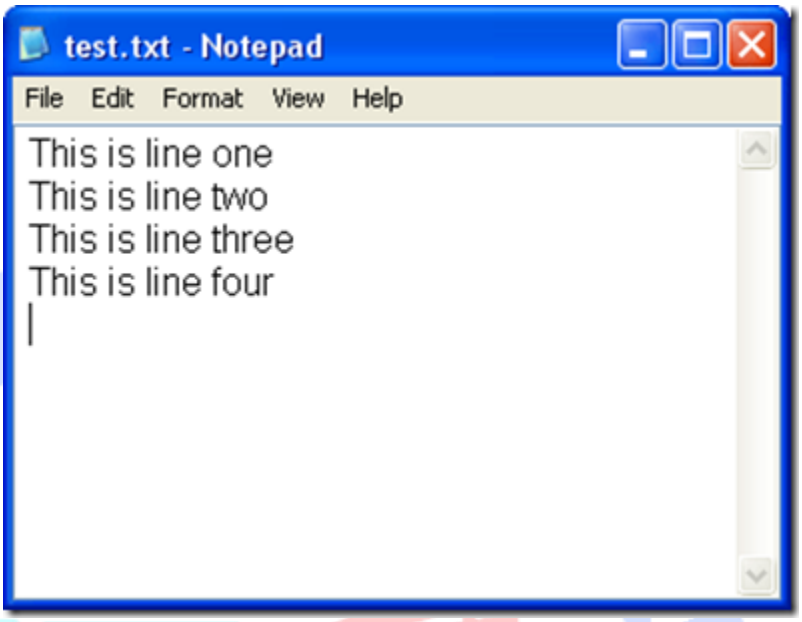
بنابراین به جای کدگذاری تعداد خطوط، می توانیم متود جدید خود را فراخوانده و تعداد خطوط را در هر فایل متنی به دست آوریم.

بسیار خوب، زمان آن رسیده که گروه جدید را به کار گرفته و مشاهده کنیم که آیا فایل متن را باز می کند یا نه.

به گروه FileData بازگردید، گروهی با متود اصلی در آن. یک متغیر رشته تنظیم کنید تا نام فایل متن را در خود داشته باشد:

```
package textfiles;
import java.io.IOException;

public class FileData {

    public static void main(String[] args) throws IOException {

        String file_name = "C:/test.txt";

    }
}
```

در این مرحله نیاز به ایجاد یک فایل متن در جایی از کامپیوتر خود دارید، ما این فایل ساده را در Notepad در ویندوز سیستم ایجاد می کنیم:



نام فایل متن "test.txt"می باشد. یک فایل متن مشابه روی کامپیوتر خود ایجاد کنید. دقت کنید که آن را کجا ذخیره می کنید، زیرا به مسیر فایل نیز احتیاج دارید:

```
String file_name = "C:/test.txt";
```

بنابراین فایل test.txtدر درایو C ذخیره می شود. اگر برای حفظ فایل فولدری به نام MyFiles ایجاد کردیم، مسیر "C:/MyFiles/test.txt"خواهد بود. اگر لازم باشد، می توانید مسیر فایل خود را تغییر دهید.

کار دیگری که باید انجام دهید، ایجاد آبجکت جدید از گروه ReadFile می باشد. سپس می توانیم متودی را فرا بخوانیم که فایل را باز می کند. اما می توانیم این کار را در یک بلوک try ... catch انجام دهیم. کد زیر را درست زیر خط متغیر String ضافه کنید:

```
public static void main(String[] args) throws IOException {

    String file_name = "C:/test.txt";

    try {
        ReadFile file = new ReadFile(file_name);
        String[] aryLines = file.OpenFile();

    }
    catch (IOException e) {
          System.out.println( e.getMessage() );
    }
}
```

پرانتزهای مربوط به بلوک try ... catch را فراموش نکنید. برای بخش try یک جفت آکولاد و برای بخش catch یک جفت دیگر نیاز دارید. برای بخش try خطوط زیر را داریم:

مثال

```
ReadFile file = new ReadFile( file_name );

String[ ] aryLines = file.OpenFile( );
```

خط اول یک آبجکت جدید ReadFile به نام file تنظیم می کند. بین پرانتزهایReadFile ، متغیر file_name را اضافه می کنیم. این برای ارائه ی constructorبه مسیر فایل مورد نیاز کافی می باشد.

خط دوم از کد یک ردیف رشته (String array) به نام aryLines تنظیم می کنیم. بعد از علامت تساوی متود OpenFileاز گروه ReadFile را فرا خواندیم .اگر با موفقیت فایل متن را باز کند، سپس ردیف خطوط متن در یک array aryLines جدید توزیع خواهد شد.

به هرحال اگر اشتباهی رخ دهد، یک خطا وارد خط شده و در بخش catch از بلوک try ... catch به پایان می رسد:

مثال

```
catch ( IOException e ) {
```

```
System.out.println( e.getMessage() );
}
```

بعد از لغتcatch ، یک جفت پرانتز وجود دارد. در داخل پرانتزها عبارت زیر را داریم:

IOException e

کاری که این عبارت انجام می دهد، تنظیم متغیری به نام e می باشد که از نوع IOException می باشد.

آبجکت IOException دارای متودهایی از نوع خود می باشد که می توانید از آنها استفاده کنید. یکی از این متودها getMessage می باشد. این متود به یوزر اطلاعاتی در مورد اشتباه رخ داده ارائه می دهد.

قبل از اینکه مثالی در مورد یک پیغام خطا مشاهده کنیم، اجازه بدهید از طریق همه ی خطوط فایل متن loop انجام دهیم و هر کدام را چاپ کنیم. کد زیر را به بخش try از بلوک try ... catch اضافه کنید:

مثال

```
int i;
for ( i=0; i < aryLines.length; i++ ) {
System.out.println( aryLines[ i ] ) ;
}
```

پنجره ی کد گذاری شما باید مشابه زیر باشد:

```
public static void main(String[] args) throws IOException {

    String file_name = "C:/test.txt";

    try {
        ReadFile file = new ReadFile(file_name);
        String[] aryLines = file.OpenFile();

        int i;
        for (i=0; i < aryLines.length; i++) {
            System.out.println(aryLines[i]);
        }
    }
    catch (IOException e) {
          System.out.println( e.getMessage() );
    }
}
```

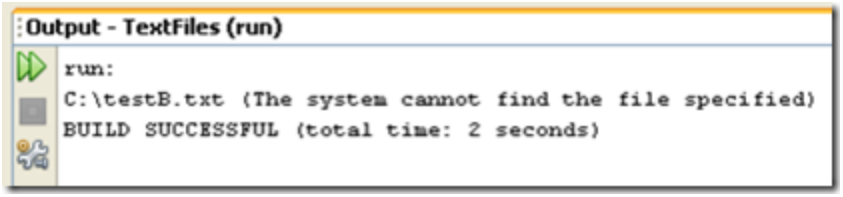
وقتی که برنامه اجرا می شود، پنجره ی Output مورد زیر را چاپ خواهد کرد:

همانطور که مشاهده می کنید، هر خط از فایل متن چاپ شده است.

برای امتحان کردن کد بخش چک کننده ی خطا(error checking) ، نام فایل متن را موردی که می دانید ایجاد نشده، تغییر دهید. سپس کد خود را مجددا اجرا کنید. در پنجره ی Output در تصویر زیر، مشاهده می کنید که فایل متن ما به testBتغییر یافته است و نمی تواند یافت شود:

```
Output - TextFiles (run)
run:
C:\testB.txt (The system cannot find the file specified)
BUILD SUCCESSFUL (total time: 2 seconds)
```

اگر تمایل دارید، می توانید پیغام خطای خود را به بخش catch اضافه کنید:




```
catch (IOException e) {
        System.out.println( "Sorry, dude - no can do!" );
}
```

گرچه بهتر است آن را به جاوا واگذار کنید . در بخش بعد به چگونگی نوشتن فایل متن با استفاده از کد جاوا، خواهیم پرداخت .

آموزش نوشتن در یک فایل

نوشتن در یک فایل کمی آسانتر از خواندن یک فایل می باشد. برای نوشتن یک فایل از چند گروه داخلی استفاده خواهیم کرد: گروه FileWriter و .PrintWriter

با کلیک کردن بر روی File > New File از منوی NetBeans یک گروه جدید در پروژه ی خود ایجاد کنید. در بخش Categories از دیالوگ باکس Java و Class را از لیست File Types انتخاب کنید. روی دکمه ی Nextدر پایین کلیک کنید .برای نام گروه WriteFile را تایپ کرده و سپس روی Finish کلیک کنید. سه عبارت زیر را به کد خود وارد کنید:

```
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;
```

گروه جدید شما باید مشابه زیر باشد:

```
package textfiles;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class WriteFile {


}
```

مجددا عبارت هایی که زیر آنها خط کشیده شده وجود دارند، زیرا هنوز از گروه وارد شده استفاده نکرده ایم.

وقتی در یک فایل می نویسید، یا می توانید از ابتدا آغاز کرده و هر موردی را بنویسید. یا می توانید از انتها آغاز کرده و به فایل ضمیمه کنید. گروه FileWriterبه شما اجازه می دهد تا روش آن را تعیین کنید. یک فیلد اضافه خواهیم کرد که مقدار ضمیمه را برای گروه FileWriter تنظیم می کند. ما یک فیلد هم برای تنظیم نام فایل اضافه خواهیم کرد.

بنابراین دو فیلد زیر را به اضافه ی یکconstructor ، به کد خود اضافه کنید:

```
public class WriteFile {

    private String path;
    private boolean append_to_file = false;

    public WriteFile(String file_path) {
        path = file_path;
    }
}
```

فیلد Boolean در واقع append_to_file نامیده می شود و روی مقدار falseتنظیم کرده است. این یک مقدار پیش فرض برای گروه FileWriter می باشد و به این معناست که ضمیمه نمی خواهید، اما هر چیزی را در فایل حذف می کند.

سازنده (constructor) مقداری را برای فیلد path تنظیم می کند که نام و موقعیت فایل می باشد. این برنامه یک آبجکت جدید از گروه WriteFile را توزیع خواهد کرد.

به هرحال همانطور که در بخش قبل ذکر شد، می توانید بیشتر از یک سازنده در کد خود تنظیم کنید. می توانیم سازنده ی دوم را در تنظیم کرده و آن را در یک مقدار ضمیمه انتقال دهیم. به این روش یک یوزر یا می تواند از اولین سازنده استفاده کند و یا اینکه نام یک فایل و یا نام یک فایل و یک مقدار ضمیمه را توزیع کند. بنابراین مورد زیر را در زیر اولین سازنده اضافه کنید:

مثال

```
public WriteFile( String file_path , boolean append_value ) {
path = file_path;
append_to_file = append_value;
}
```

اکنون سازنده ی دوم دارای دو مقدار در بین پرانتزها می باشد: یک مسیر فایل و یک مقدار ضمیمه. اگر می خواهید به این فایل ضمیمه کنید، می توانید در هنگام ایجاد یک آبجکت جدید از این سازنده استفاده کنید. اگر فقط می خواهید فایل متن را بنویسید، می توانید از اولین سازنده (constructor) استفاده کنید.

پنجره ی کد شما باید شبیه به تصویر زیر باشد:

```
package textfiles;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class WriteFile {

 private String path;
 private boolean append_to_file = false;

 public WriteFile(String file_path) {
     path = file_path;
 }

 public WriteFile( String file_path, boolean append_value ){
     path = file_path;
     append_to_file = append_value;
 }

}
```

برای نوشتن روی فایل، متود زیر را در زیر سازنده های خود اضافه کنید:

```
public void writeToFile( String textLine ) throws IOException {

}
```

این متود نیازی به بازگشت یک مقدار ندارد، بنابراین آن را void می سازیم. در بین پرانتزهای مربوط به نام متود، یک متغیر String به نام textLineرا داریم. مشخص است که این متنی است که می خواهیم در یک فایل بنویسیم. مجددا واضح است که باید "throws IOException" را اضافه کنیم چرا که برای کنترل خطاهای مربوط به file-writing لازم است.

اولین چیزی که در متود لازم داریم یک آبجکت FileWriter می باشد FileWriter . مراقب باز کردن فایل درست و مرتب کردن متن به عنوان بایت می باشد. خط زیر را به متود writeToFile خود اضافه کنید:

```
FileWriter write = new FileWriter( path , append_to_file);
```

بنابراین ما یک آبجکت FileWriter جدید با نام write ایجاد می کنیم. بین پرانتزهای FileWriter نام و موقعیت فایل به علاوه ی مقدار ضمیمه را انتقال می دهیم. این مقدارtrue (ضمیمه شده به یک فایل) یا false) ضمیمه نشده (خواهد بود. اگر فایلی با نامی که انتقال داده اید، وجود نداشته باشد، FileWriterبرای شما ایجاد می کند.

به هرحال FileWriter بایت ها را می نویسد. اما می توانیم به کمک گروه PrintWriterیک متن ساده به FileWriterارائه دهیم PrintWriter .چند متود ساده برای این کار در دست دارد. اما در هنگام ایجاد آبجکت از گروه نیاز به نام یک FileWriter دارد. بنابراین خط زیر را به متود خود اضافه کنید:

```
PrintWriter print_line = new PrintWriter( write );
```

آبجکت PrintWriter ما در واقع print_line نامیده می شود. بین پرانتزهایPrintWriter ، نام آبجکت FileWriterخود را اضافه می کنیم.

برای افزودن متن به یک فایل، نام آبجکت PrintWriter را تایپ کرده که با یک نقطه (dot) دنبال می شود:

```
print_line.
```

به محض اینکه این نقطه را تایپ می کنید، NetBeansلیستس از گزینه های متغیر را نمایش خواهد داد:

```
FileWriter write = new FileWriter(path, append_to_file);

PrintWriter print_line = new PrintWriter(write);

print_line.
```

```
print(Object obj)                    void
print(String s)                      void
print(boolean b)                     void
print(char c)                        void
print(char[] s)                      void
print(double d)                      void
print(float f)                       void
print(int i)                         void
print(long l)                        void
printf(String format,    PrintWriter
printf(Locale l,         PrintWriter
println()                            void
println(Object x)                    void
println(String x)                    void
println(boolean x)                   void
println(char x)                      void
println(char[] x)                    void
```

گزینه های خیلی زیادی در لیست وجود دارند.

یکی از گزینه هایی که استفاده خواهیم کرد، یکی از متودهای printf می باشد. این متود به شما اجازه می دهد تا یک رشته ی قالب بندی شده از متن را به PrintWriter انتقال دهید. یک دلیل خوب برای استفاده ازprintf ، بررسی کاراکترهای خط جدید می باشد. کاراکترهای خط جدید بسته به سیستم عاملی که استفاده می کنید، متفاوت می باشند. ویندوز کاراکترهای \r\n را برای یک خط جدید اضافه خواهد کرد. اما سیستم Unix فقط از \nاستفاده می کند. استفاده از عملکرد printf کد گذاری صحیح را، بدون توجه به سکو، تضمین خواهد داد.

خط زیر را به کد خود اضافه کنید:

مثال

```
print_line.printf( "%s" + "%n" , textLine);
```

به متود printf دو مورد را تحویل داده ایم: فرمت مربوط به متن، رشته ای که روی فایل بنویسیم. هر دوی این موارد با استفاده از یک ویرگول (comma) از یکدیگر جدا می شوند. به این دو دقت کنید:

```
"%s" + "%n"
```

این خط فایل متن را بسته و منابعی را که استفاده می کرد، آزاد می کند.

اکنون گروه WriteFile مانند زیر به نظر خواهد رسید:

```
package textfiles;

import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;

public class WriteFile {

  private String path;
  private boolean append_to_file = false;

  public WriteFile(String file_path) {
      path = file_path;
  }

  public WriteFile( String file_path, boolean append_value ) {
      path = file_path;
      append_to_file = append_value;
  }

public void writeToFile(String textLine) throws IOException {

    FileWriter write = new FileWriter(path, append_to_file);
    PrintWriter print_line = new PrintWriter(write);

    print_line.printf("%s" + "%n", textLine);

    print_line.close();

  }
}
```

برای اینکه گروه جدید خود را امتحان کنید) گروهی با متود اصلی (main به گروه FileData بازگردید. برای ایجاد یک آبجکت جدید از گروهWriteFile ، خط زیر را اضافه کنید:

```
WriteFile data = new WriteFile( file_name , true );
```

بنابراین یک آبجکت جدید WriteFile به نام data تنظیم کرده ایم. بین پرانتزهای WriteFile دو مورد اضافه کرده ایم: نام فایل و یک مقدار .true این امر فراخوانی سازنده های دوم (constructor) را تضمین خواهد کرد. اگر بخواهیم فقط فایل را بنویسیم، کد آن مانند زیر خواهد بود:

```
WriteFile data = new WriteFile( file_name );
```

از آنجایی که مقدار ضمیمه ی پیش فرض را با عنوان false تنظیم کرده ایم، اگر بخواهیم کل محتوا را بنویسیم، تنها به نام فایل احتیاج داریم.

برای فراخوانی متود writeToFile از آبجکتWriteFile ، خط زیر را اضافه کنید:

```
data.writeToFile( "This is another line of text" );
```

برای تغییر متن بین پرانتزهای مربوط به متود آزاد هستید.

برای اینکه به یوزر اجازه بدهید متوجه اتفاقی که افتاده شود، باید چیزی از پنجره ی Output چاپ کنید:

```
System.out.println( "Text File Written To" );
```

اکنون کد FileData باید مانند زیر باشد (چند کامنت اضافه کرده ایم.):

```
package textfiles;
import java.io.IOException;

public class FileData {

    public static void main(String[] args) throws IOException {

        String file_name = "C:/test.txt";
        try {
            ReadFile file = new ReadFile(file_name);
            String[] aryLines = file.OpenFile();
            int i;
            for (i=0; i < aryLines.length; i++) {
                System.out.println(aryLines[i]);
            }
        }
        catch (IOException e) {
              System.out.println( "Sorry, dude - no can do!" );
        }

        //============================
        //  WRITE TO A FILE
        //============================
        WriteFile data = new WriteFile(file_name, true);
        data.writeToFile("This is another line of text");

        System.out.println("Text File Written To");
    }
}
```

اگر تمایل داشته باشید، می توانید برای نوشتن متن یک بخش catch ... try دیگر نیز اضافه کنید. به جای مورد بالا، آن را مانند زیر تغییر دهید:

```
//============================
//   WRITE TO A FILE
//============================
try {
   WriteFile data = new WriteFile(file_name, true);
   data.writeToFile("This is another line of text");
}
catch (IOException e) {
      System.out.println( e.getMessage() );
}
```

اکنون کد خود را اجرا کرده تا آن را امتحان کنید. شما باید به وسیله ی پیغامی که فایل متن نوشته، محتوای فایل متن خود را پنجره ی Output مشاهده کنید:

```
Output - TextFiles (run)
run:
This is line one
This is line two
This is line three
This is line four
Text File Written To
BUILD SUCCESSFUL (total time: 0 seconds)
```

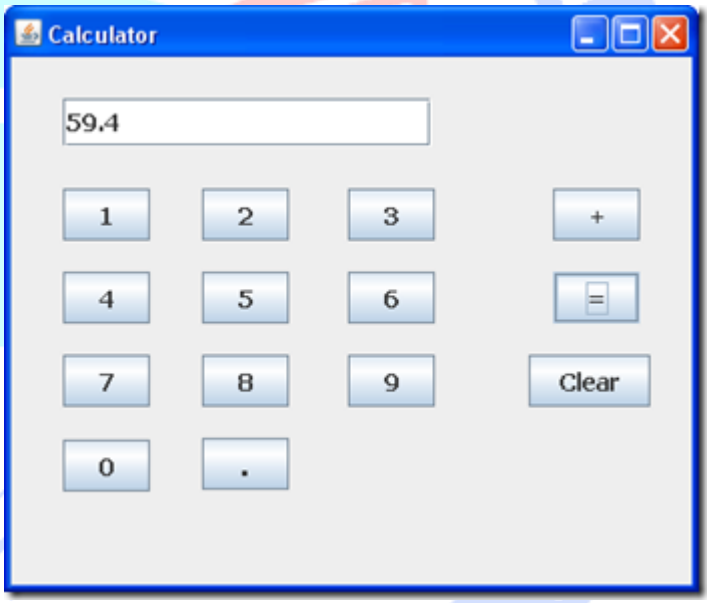
مجددا برنامه را اجرا کرده و یک خط جدید مشاهده خواهید کرد. (می توانید به کدی که در فایل متن نوشته می شود، کامنت بگذارید.)

```
Output - TextFiles (run)
run:
This is line one
This is line two
This is line three
This is line four
This is another line of text
BUILD SUCCESSFUL (total time: 2 seconds)
```

و این تمام آن مورد می باشد_ اکنون می توانید یک فایل متن نوشته و محتوای آن را بخوانید. در بخش بعد ادامه خواهیم داد و به برنامه نویسی با فرما های جاوا خواهیم پرداخت.

ایجاد فرم جاوا با NetBeans IDE

در جاوا لازم نیست هر چیزی را به یک پنجره ی نهایی انتقال دهید. در این بخش یک برنامه ی ماشین حساب خواهید نوشت که از این فرما ها استفاده خواهد کرد. این فرم دارای دکمه ها و یک جعبه ی متن می باشد. با یک ماشین حساب ساده آغاز خواهیم کرد که تنها عملکرد جمع را انجام می دهد و سپس توانایی های خود را گسترش داده به عملکردهای منها، تقسیم و ضرب. ماشین حساب چیزی مشابه تصویر زیر خواهد بود:



(تصویر بالا از یک Windows XP گرفته شده است. اگر یک سیستم عامل Linux یا Apple Mac داشته باشید، ماشین حساب شما کمی متفاوت به نظر خواهد رسید.)

اجازه بدهید که شروع کنیم.

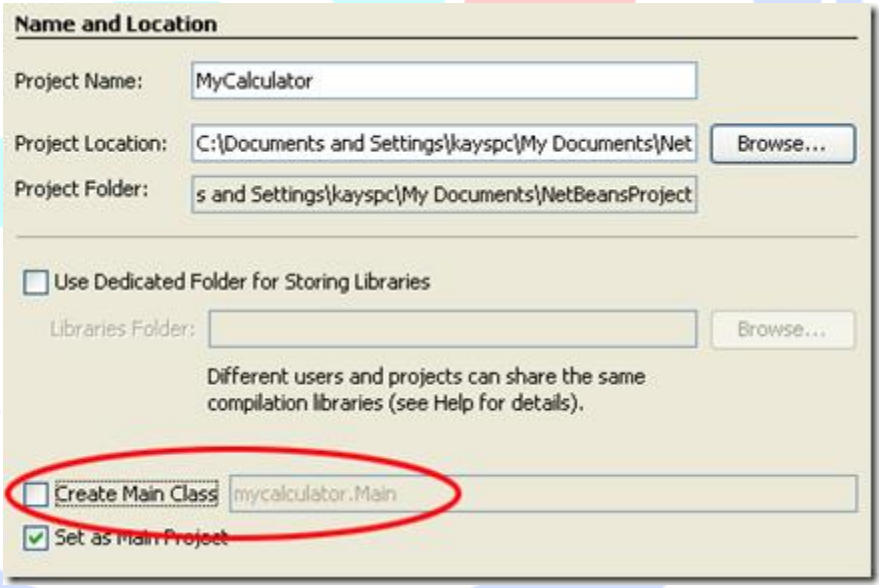
فرم های Java وNetBeans

توسعه ی یک Graphic User Interface (GUI) با استفاده از جاوا می تواند به نوبه ی خود یک کار هنری باشد، همانطور که موارد زیادی برای استفاده وجود دارند: مولفه ها(Components) ، ظروف

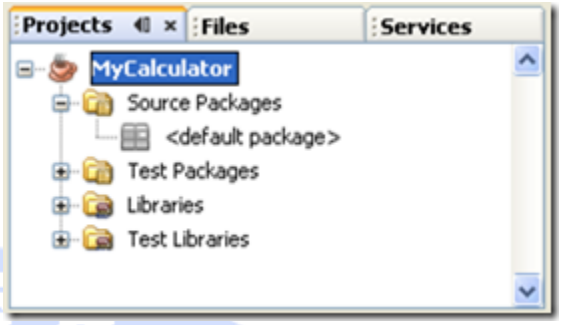
(Containers)، مدیران چیدمان (Layout Managers) و کلیات بیشتر در کنار اینها. به هرحال محیط توسعه ی NetBeans ایجاد فرم ها را به شدت آسان ساخته است و از آن برای درگ و دراپ کردن کنترل ها روی یک چارچوب استفاده خواهیم کرد.

به جای وارد شدن از طریق بندهای تئوریGUI ، مستقیما وارد آن خواهیم شد . با کلیک کردن بر روی File > New Projectاز منوی NetBeans در بالا، یک پروژه ی جدید برای این کار ایجاد کنید. از لیست ظاهر شده را انتخاب کرده Java Application > Javaو سپس روی دکمه ی Next کلیک کنید.

در مرحله ی دوم از wizard عبارت MyCalculator را به عنوان نام پروژه تایپ کنید. در پایین نیز عبارت "Create main class"را از حالت انتخاب خارج کنید. این امر به این دلیل است که وقتی ما یک فرم را اضافه کردیم، یک متود mainبرای ما توسط NetBeans ایجاد خواهد شد. مرحله ی 2 از wizard باید مانند زیر به نظر باشد:



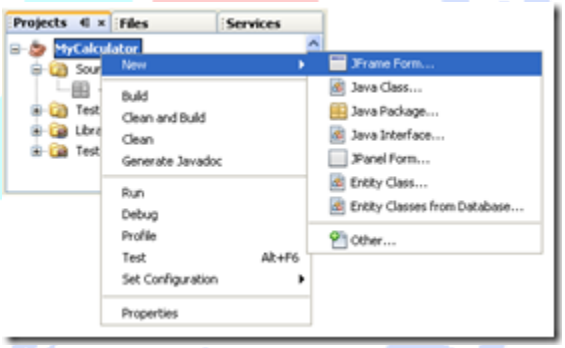
روی دکمه ی Finish کلیک کنید و NetBeans پروژه و نه چیز بیشتری، ایجاد خواهد کرد. در سمت چپ NetBeansنگاهی به محدوده ی Projects نگاهی داشته باشید که باید چیزی مانند تصویر زیر را مشاهده کنید (اگر نمی توانید محدوده ی Projects را ببینید، روی Window > Projectsاز منوی NetBeans در بالا کلیک کنید.):



معمولا یک فایل java. تحت عنوان Source Packages وجود دارد. اما از آنجایی که بخش "Create main class"را از حالت لنتخاب در آوردیم، هیچ فایل گروه جاوایی در آنجا وجود ندارد.

کاری که در اینجا انجام خواهیم داد افزودن یک فرم به پروژه می باشد. فرم ایجاد شده در گروه جاوای خود ایجاد می شود.

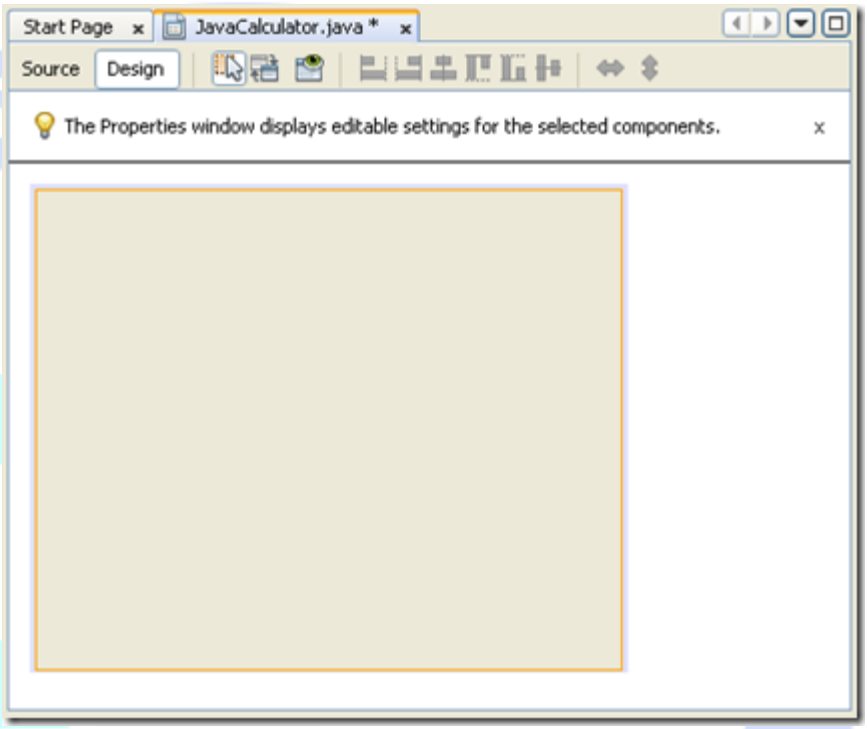
برای افزودن یک فرم، روی نام پروژه در پنجره ی Projects راست کلیک کنید. یک منو ظاهر خواهد شد:



از منوی ظاهر شده New > JFrame Form را انتخاب کنید. وقتی این کار را انجام می دهید باید صفحه ی زیر برای شما ظاهر شود:

در اینجا نامی برای گروه و یک نام برای پوشه از شما خواسته می شود. ما تقریبا پروژه را ایجاد کرده و آن را MyCalculatorمی نامیم. نام پوشه و گروه وارد پروژه خواهند شد. بنابراین برای بخش)Class Nameنام گروه(، عبارت JavaCalculator را تایپ کنید. در پوشه ی خالی تکست باکس jCalculatorرا تایپ کنید. بنابراین ما در حالت ایجاد گروهی به نام JavaCalculator هستیم که در پوشه ی jCalculator و در پروژه ی MyCalculatorمی باشد.

وقتی wizard مربوط به بخش قبل تمام شد، در پنجره ی اصلی یک فرم خالی ایجاد خواهد کرد.



در حال حاضر این فرم خالیست و یک مستطیل نارنجی رنگ آن را احطه کرده است. این مستطیل به این معناست که فرم انتخاب نشده است. در فرم روی back کلیک کنید، پس از آن دوباره مستطیل نارنجی را مشاهده می کنید.

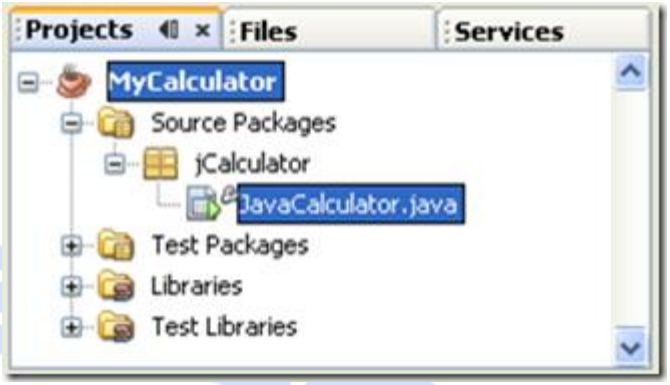
به دو دکمه ی Source و Design در بالا دقت کنید. در حال حاضر در Design هستید. برای مشاهده ی کد زیر روی Source کلیک کنید.

```
package jCalculator;

/**...*/
public class JavaCalculator extends javax.swing.JFrame {

    /** Creates new form JavaCalculator */
    public JavaCalculator() {
        initComponents();
    }

    /**...*/
    @SuppressWarnings("unchecked")
    Generated Code

    /**...*/
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new JavaCalculator().setVisible(true);
            }
        });
    }

    // Variables declaration - do not modify
    // End of variables declaration

}
```

شما می توانید برای مشاهد یا پنهان کردن کد می توانید علامت های به اضافه را باز کنید یا ببندید. به هرحال دقت داشته باشید که متود main ایجاد شده است. وقتی که برنامه آغاز می شود، این متود main است که فرا خوانده خواهد شد. این برنامه یک آبجکت جدید از فرم ما ایجاد می کند و پراپرتی قابلیت رویت بودن آن را روی true تنظیم می کند.

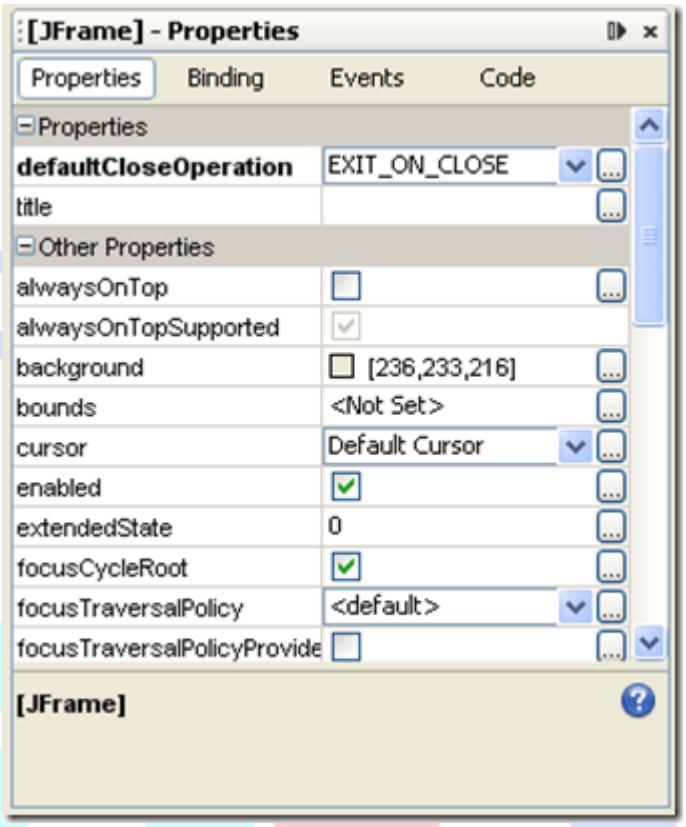
اما NetBeans همه ی این کد را برای شما تولید می کند. در غیر این صورت مجبور خواهید بود تمام آن را خودتان انجام دهید.

مجددا نگاهی به بخش Projects در سمت چپ داشته باشید. مشاهده خواهید کرد که یک پوشه و یک فایل classاضافه شده اند.



در بالا روی دکمه ی Design روی back کلیک نید. شما فرم خالی خود را دوباره مشاهده خواهید کرد. در سمت راست دو بخش مشاهده خواهید کرد: یک پالت (Palette) با کنترل های زیاد در آن و یک بخش properties. Paletteباید مانند شکل زیر باشد.

بخش properties نیز باید مانند تصویر زیر باشد.

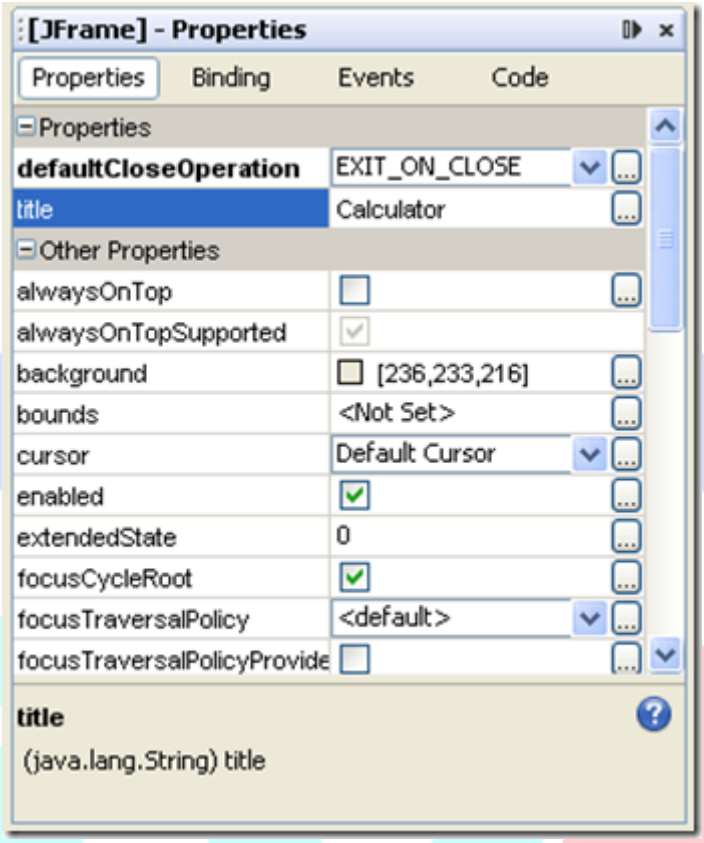


(اگر نتوانستید آنها را ببینید، از منوی NetBeans روی Window> Palette و Window > Properties کلیک کنید)

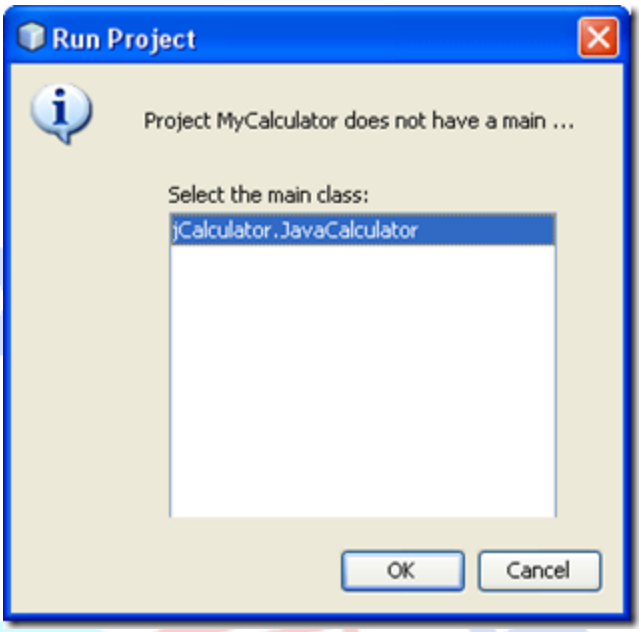
یک پراپرتی از یک آبجکت لیستی از مواردی است که می توانید با آن کارهایی انجام دهید، از جمله تنظیم رنگ زمینه، تنظیم متن، تنظیم فونت و موارد بسیار دیگر. اجازه بدهید تیتر را تغییر دهیم.

مطمئن شوید که فرم شما انتخاب شده است. اگر چنین باشد، فرم انتخاب شده باشد، با یک مستطیل نارنجی رنگ احاطه خواهد بود. بخش properties در بالا JFrameرا اعلام خواهد کرد.

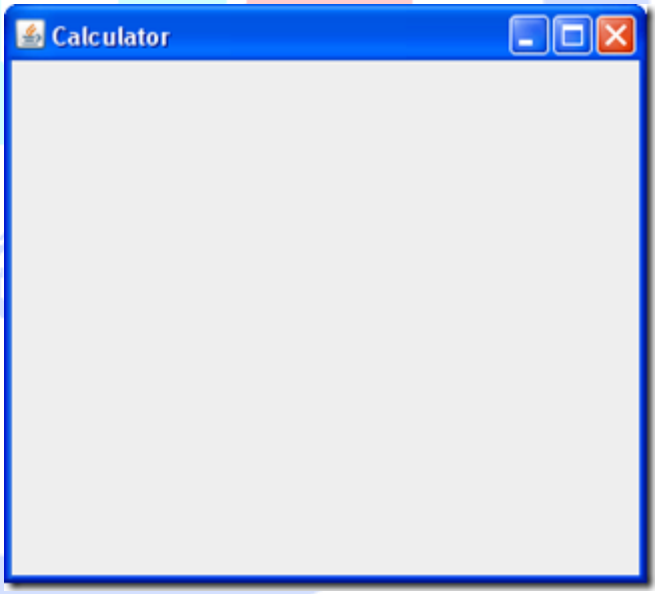
در داخل بخش titleکلیک کرده و Calculator را تایپ کنید.



سپس دکمه ی enter را فشار دهید. برای مشاهده ی اثر تغییر پراپرتی تیتر (title)، برنامه را به حالت معمول اجرا کنید. وقتی برنامه را برای اولین بار اجرا می کنید، NetBeansیک دیالوگ باکس نمایش خواهد داد که از شما در مورد متود اصلی مورد استفاده خواهد پرسید.



شما تنها یک متود دارید که در گروه JavaCalculator قرار دارد، بنابراین فقط روی OK کلیک کنید. سپس فرم مورد نظر شما باید ظاهر شود.



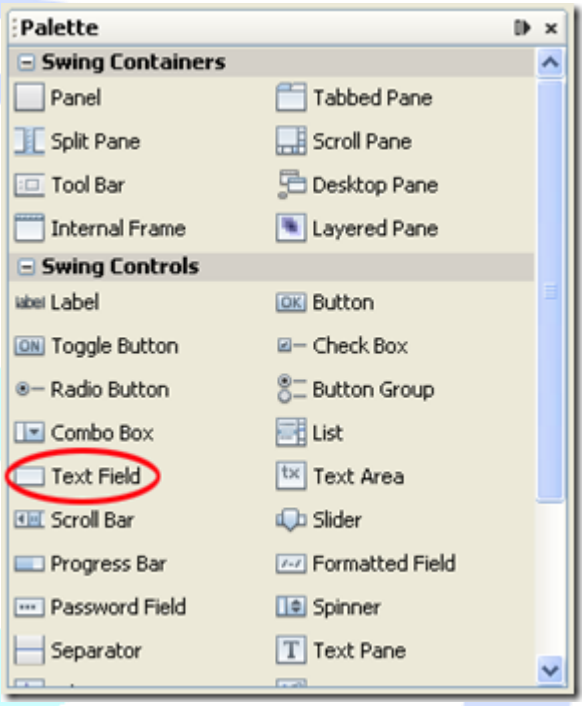
فرم دارای عنوانی است که تایپ کرده اید، به علاوه دارای آیکن های پیش فرض دیگری برای بزرگ کردن، کوچک کردن و یا بستن. برای بستن برنامه روی X کلیک کرده و به محیط Design بازگردید.

در بخش بعد یک تکست باکس به فرم اضافه خواهیم کرد.

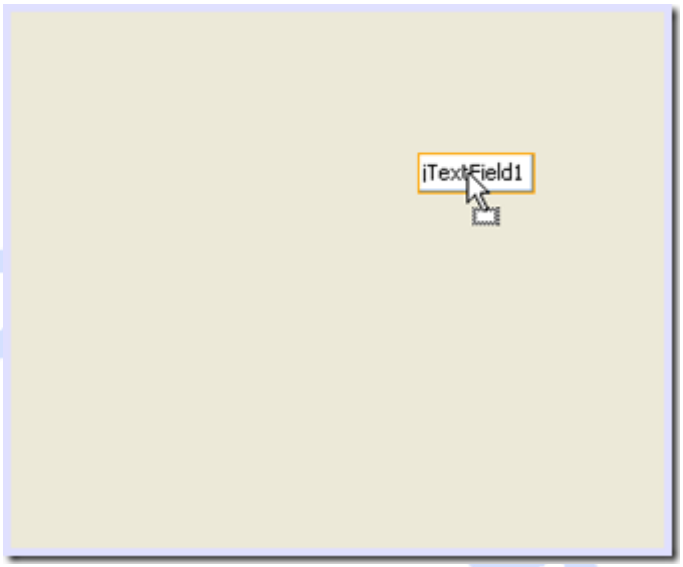
آموزش افزودن TextBox به فرم جاوا

آنچه اکنون فرم ما نیاز دارد، یک تکست باکس می باشد. اجازه بدهید ابتدا تکست باکس را اضافه کنیم.

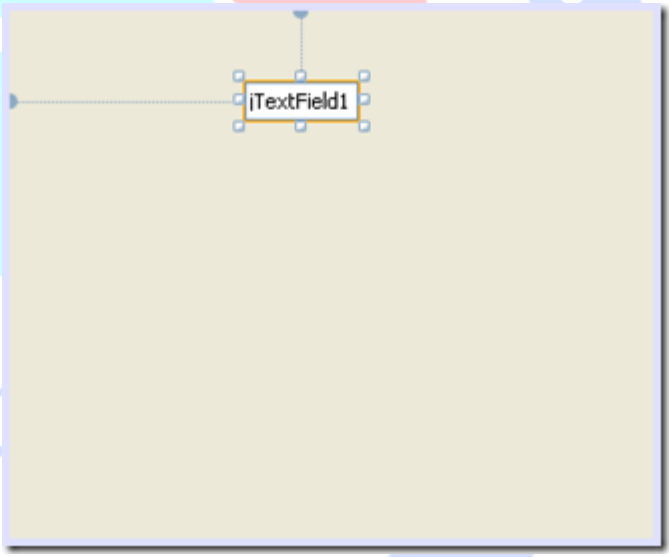
کنترل Text Field را در Palette قرار دهید.



کنترل های واقع در NetBeans Palette می توانند روی فرم درگ شوند .بنابراین برای انتخاب آن روی Text Fieldکلیک کنید. اکنون دکمه ی چپ ماوس را روی Text Field رو به پایین نگاه دارید. ماوس را در همین حالت نگاه داشته و کنترل را روی فرم درگ کنید.

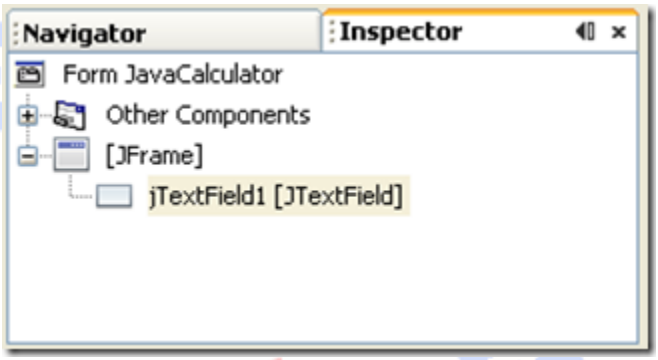


اجازه بدهید هر جایی روی فرم قرار بگیرد.

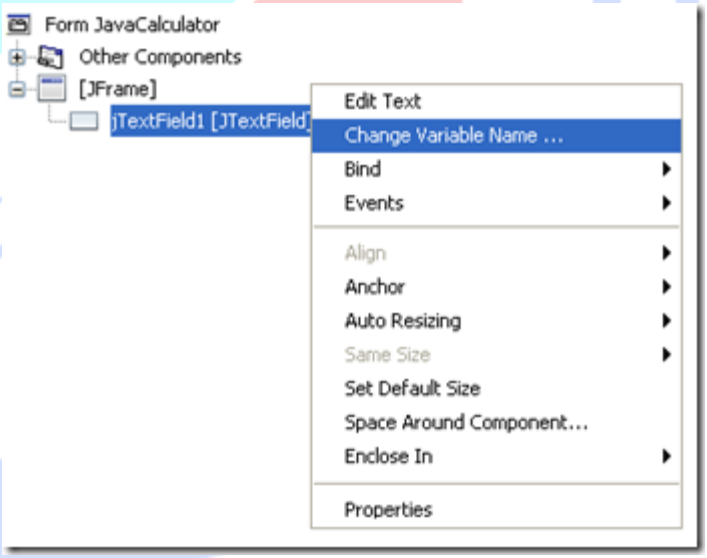


مربع های اطراف متن برای تنظیم اندازه می باشند. شما می توانید دکمه ی سمت چپ ماوس را روی یکی از مربع ها رو به پایین نگاه داشته و آن را برای طول یا عرض جدید درگ کنید. خطوط نقطه چین نشان دهنده ی موقعیت می باشند، یکی برای موقعیت چپ و دیگری برای موقعیت بالا. دقت کنید که نام پیش فرض برای فیلد متن jTextField1 می باشد که اجازه بدهید آن را تغییر دهیم.

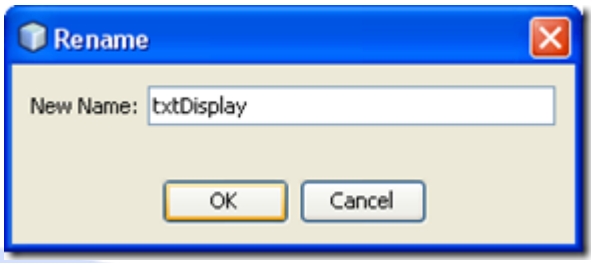
با فیلد متن انتخاب شده، نگاهی به محدوده ی Inspector در پایین سمت چپ، داشته باشید. (اگر نمی توانید محدوده ی Inspector را مشاهده کنید، از نوار منوی NetBeans روی Window > Navigating > Inspector کلیک کنید)



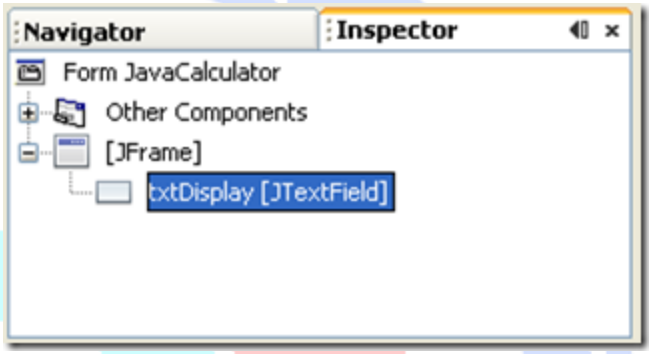
همانطور که مشاهده می کنید، jTextField در Inspector انتخاب شده است .این بخش به شما نشان می دهد که در فرم های خود چه آبجکت هایی دارید. همچنین از این قسمت می توانید یک آبجکت را تغییر نام دهید. برای انجام چنین کاری روی jTextField1 راست کلیک کنید. از منوی ظاهر شده، Change Variable Name را انتخاب کنید.



وقتی که روی Change Variable Name کلیک کنید، یک دیالوگ باکس ظاهر می شود. یک نام جدید برای Text Field تایپ کنید. آن را txtDisplay بنامید.



روی OK کلیک کنید. وقتی این کار را انجام می دهید، NetBeansفیلد متن شما را تغییر نام خواهد داد.



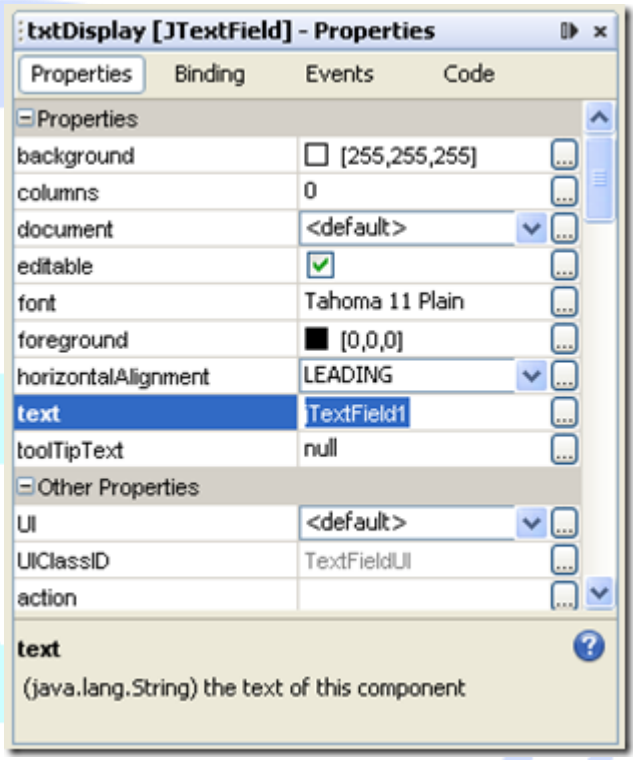
اکنون با کلیک کردن روی دکمه ی Source در پنجره ی اصلی، دوباره به کد خود نگاهی داشته باشید. وقتی که کد شما ظاهر می شود، به سمت پایین اسکرول کنید. مشاهده خواهید کرد که یک متغیر فیلد خصوصی جدید اضافه شده است.

```
/**...*/
public static void main(String args[]) {
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new JavaCalculator().setVisible(true);
        }
    });
}

    // Variables declaration - do not modify
    private javax.swing.JTextField txtDisplay;
    // End of variables declaration
```

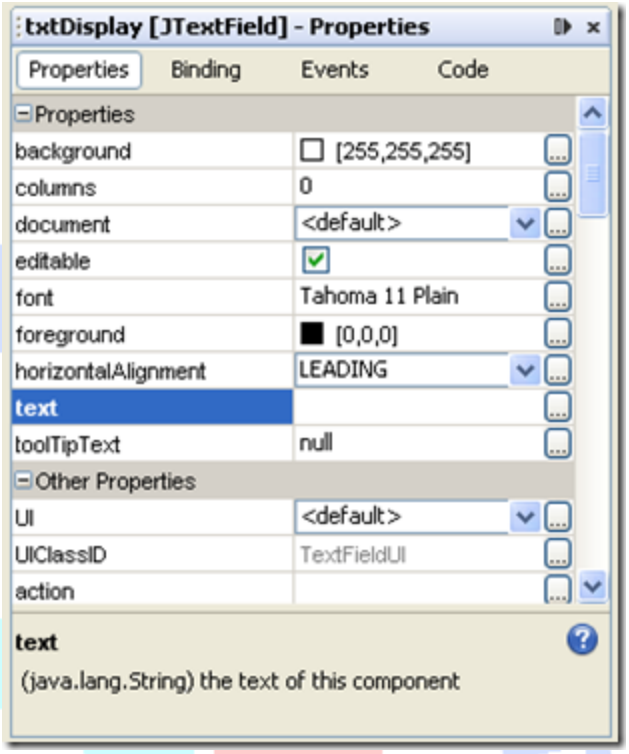
بنابراین یک متغیر JTextField به نام txtDisplay تنظیم شده است. بخش "javax.swing"مرجعی به مجموعه ای از پوشه هاست که در توسعه ی GUI استفاده می شوند Swing .ایجاد فرم ها و کنترل های روی فرم ها را برای شما آسان تر می سازد.

روی دکمه ی Design در بالا کلیک کنید تا به فرم خود برگردید. در حال حاضر فیلد متن دارای چند متن پیش فرض می باشد. شما می توانید با تغییر پراپرتی متن از فیلد متن، متن خود را اضافه کنید.

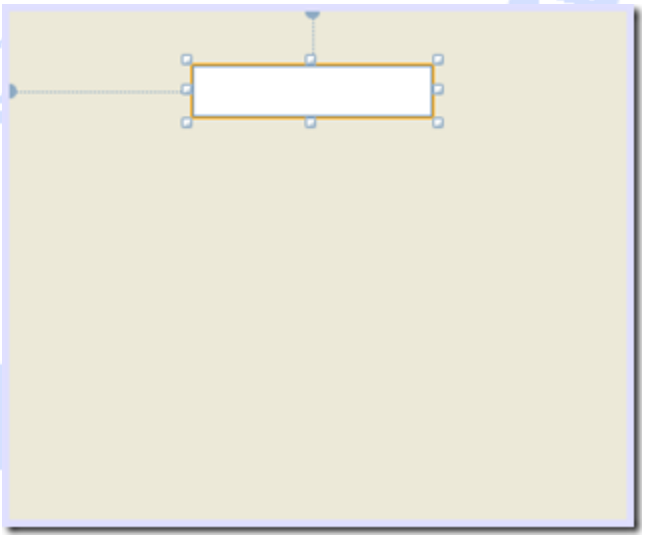
مطمئن شوید که فیلد متن شما در فرم انتخاب شده است. اکنون متن را در پنجره ی properties قرار دهید.



متن پیش فرض jTextField1 را حذف کرده و آن را خالی رها کنید.



سپس در صفحه کلید دکمه ی enter را فشار دهید. نگاهی به آبجکت فیلد متن خود روی فرم داشته باشید. ممکن است اندازه ی آن تغییر پیدا کرده باشد. از دسته های تنظیم اندازه برای تغییر اندازه ی آن استفاده کنید. مشاهده خواهید کرد که اصلا متنی در آن وجود ندارد.



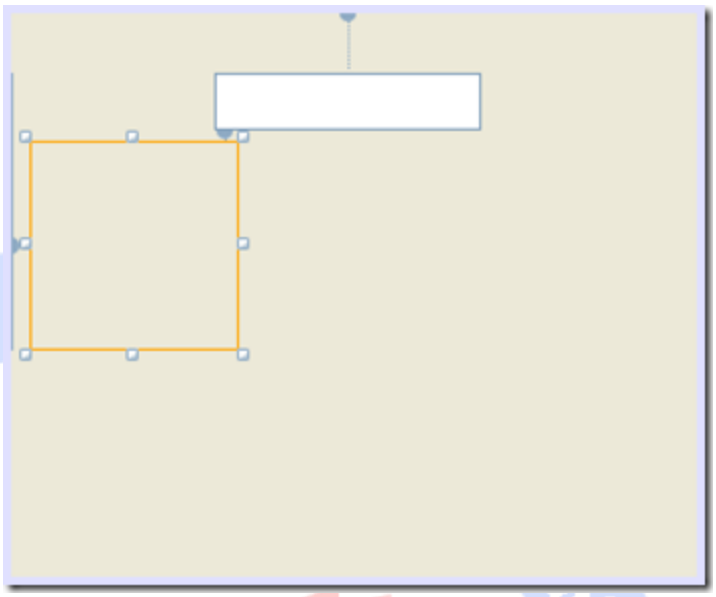
فیلد text در ماشین حساب ما، مشخصا برای خروجی ماشین حساب استفاده می شود. اما بدون دکمه ها کار نخواهد کرد. چگونگی افزودن آنها را در بخش بعد مشاهده خواهید کرد.

آموزش افزودن دکمه به فرم در جاوا

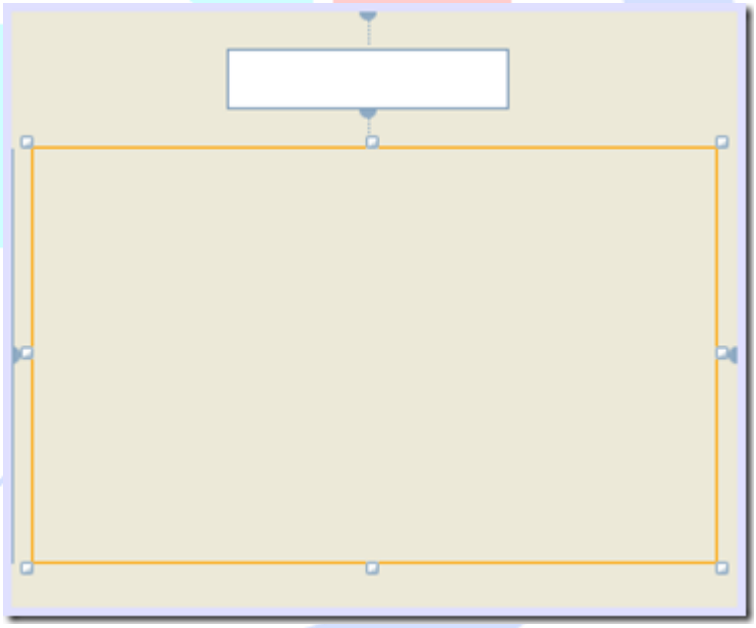
به همان روش فیلدهای text می توانید یک دکمه نیز به یک فرم اضافه کنید – درگ و دراپ کردن. به هرحال، همانطور که قصد اضافه کردن دکمه های زیادی داریم، افزودن دکمه به کنترلی به نام Panel نیز خوب می باشد. اگر لازم است دکمه ها را حرکت دهید، کافیست Panel را جابه جا کنید. تمام دکمه با Panel جابه جا خواهند شد.

بنابراین کنترل Panel را در داخل Palette قرار دهید.

یکی از آنها را روی فرم خود درگ کنید. نمی توانید Panel را ببینید، زیرا که آنها همرنگ فرم خواهند بود. اما می توانید آن را انتخاب کنید.

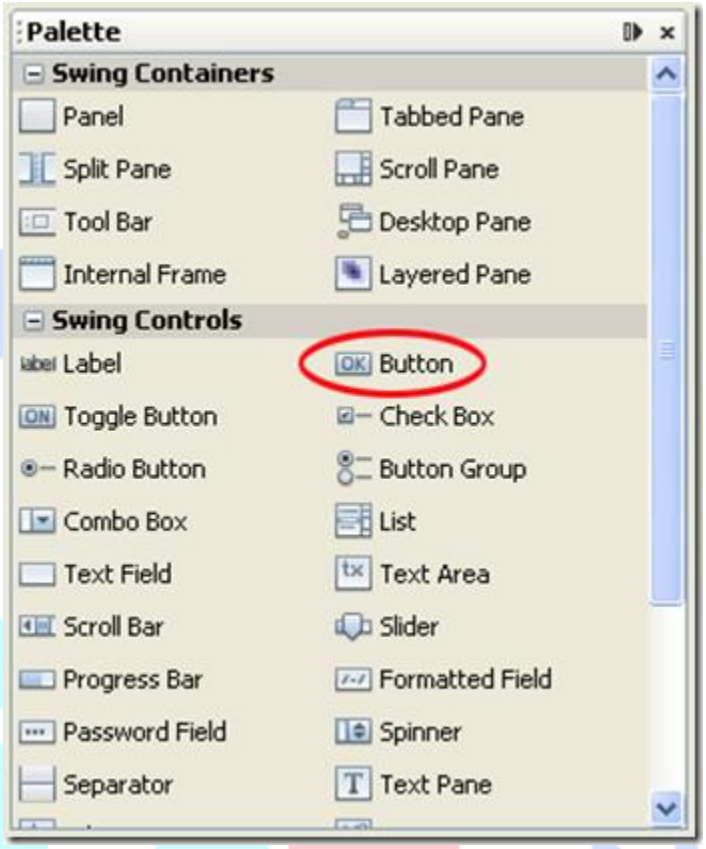


دسته های تغییر اندازه را درگ کرده تا Panel قسمت بیشتر فرم را پر کند.

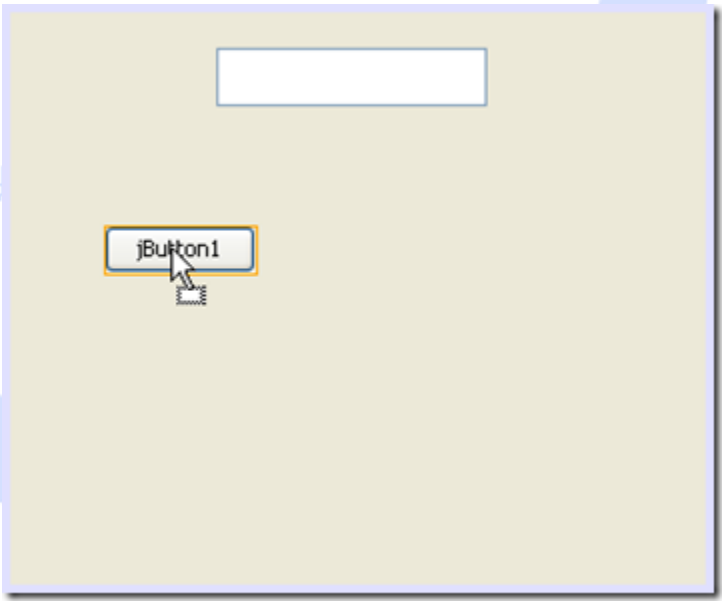


اکنون می توانیم یک دکمه اضافه کنیم.

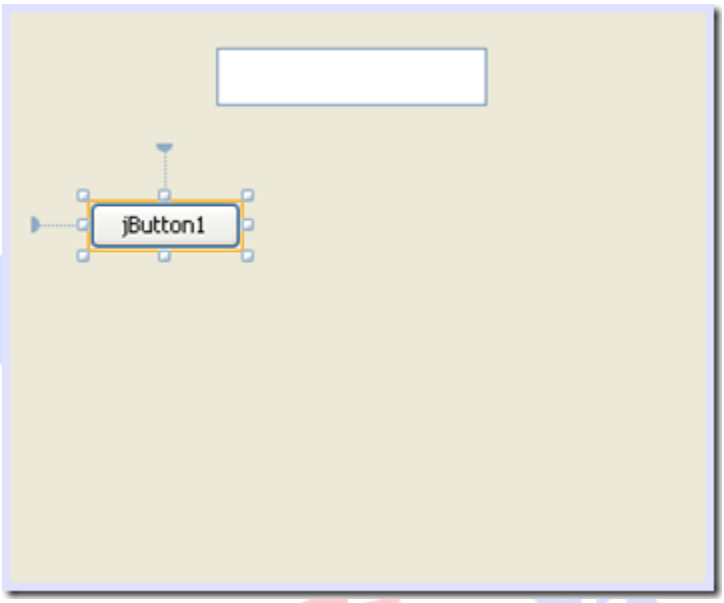
کنترل Button را در داخل Palette قرار دهید.



یکی از آنها را به کنترل Panel درگ کنید.

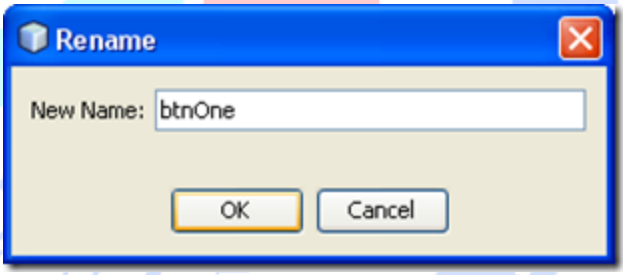


سپس دکمه ای با دسته های تغییر اندازه و موقعیت خطوط مشاهده خواهید کرد.

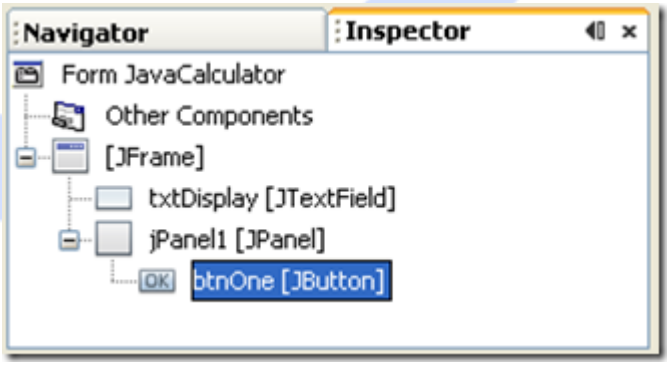


نام پیش فرض دکمه jButton1می باشد. این نام را همان طور که برای فیلد textانجام دادید، تغییر دهید.

در قسمت Inspector راست کلیک کرده و گزینه ی Change variable nameرا انتخاب کنید. نام دکمه را به btnOne تغییر دهید.



نام دکمه در Inspector تغییر خواهد کرد.



یک خط جدید از کد نیز افزوده شده است.

```
// Variables declaration - do not modify
private javax.swing.JButton btnOne;
private javax.swing.JPanel jPanel1;
private javax.swing.JTextField txtDisplay;
// End of variables declaration
```

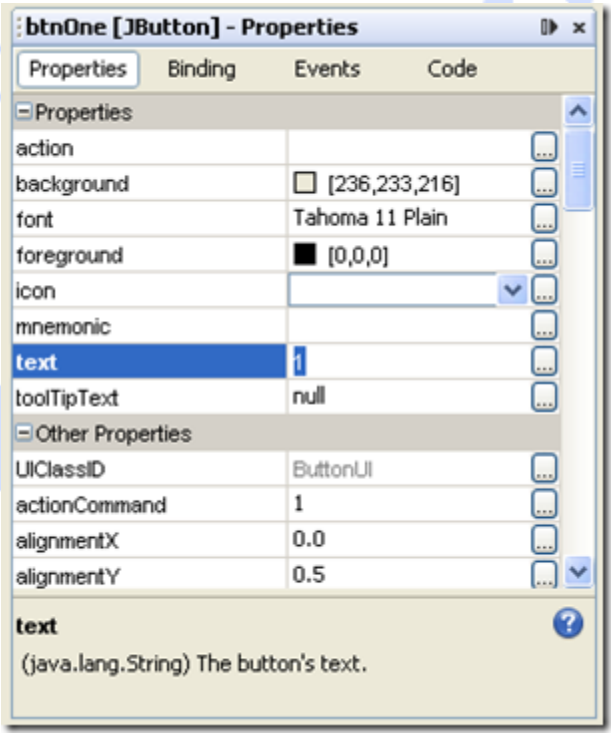
نام متغیر جدید btnOne است که یک آبجکت JButton می باشد که یک کنترل Swingاست. همچنین به متغیر تنظیم شده برای panel دقت داشته باشید.(ما این متغیر را در نام پیش فرض رها کرده ایم(

در بخش بعدی چگونگی تغییر پراپرتی های دکمه های Java را فرا خواهید گرفت.

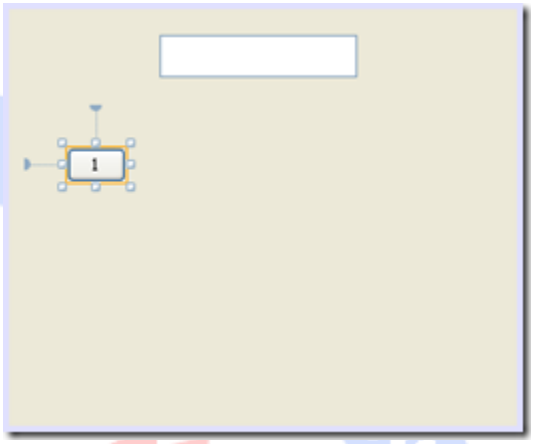
آموزش چگونگی تغییر پراپرتی های دکمه در جاوا

در بخش قبل چگونگی افزودن یک دکمه به فرم جاوا را مشاهده کردید. در این بخش چگونگی تغییر پراپرتی های آن را مشاهده خواهید کرد.

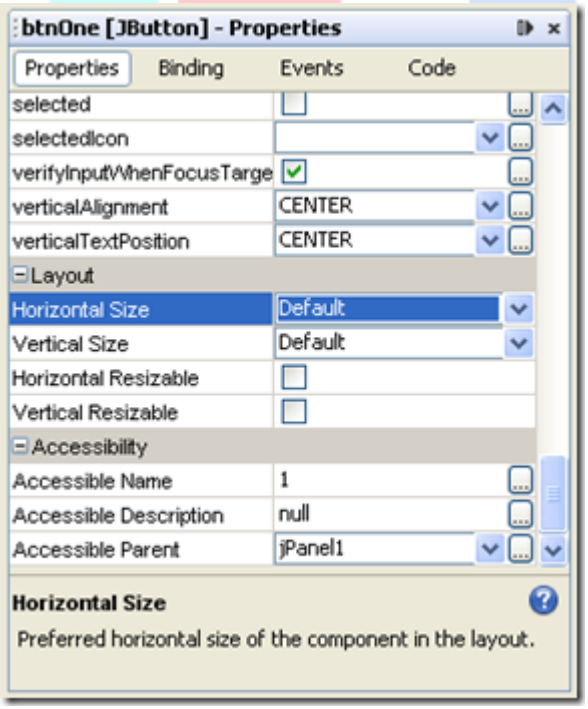
به ویو Design برگردید و اطمینان حاصل کنید که دکمه ی شما انتخاب شده است. می توانیم متن روی دکمه را تغییر دهیم. آنچه نیاز داریم یک عدد می باشد، یک عدد برای هر دکمه روی ماشین حساب. پراپرتی text برای دکمه ی text استفاده می شود. بنابراین این پراپرتی را در پنجره ی properties قرار دهید .پیش فرض را حذف کرده و به جای آن عدد 1 را تایپ کنید.



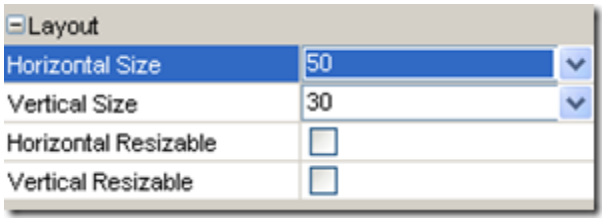
دکمه ی enter را روی صفحه کلید فشار دهید و پس از آن مشاهده خواهید کرد متن روی دکمه تغییر کرده است. این متن تغییر اندازه نیز خواهد کرد.



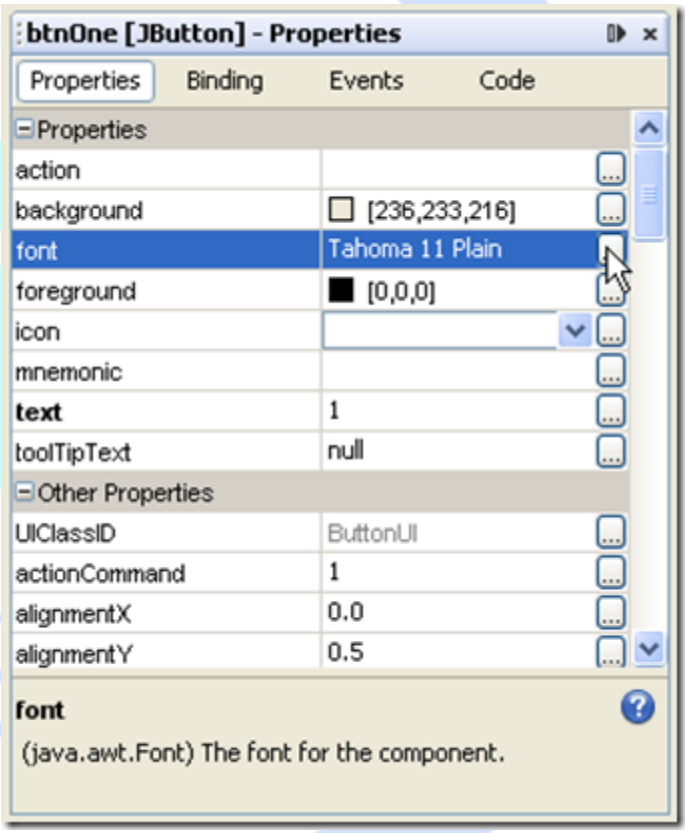
در پنجره ی properties می توانید سایز دکمه را تغییر دهید Horizontal Size .و Vertical Size را زیر تیتر Layoutقرار دهید.



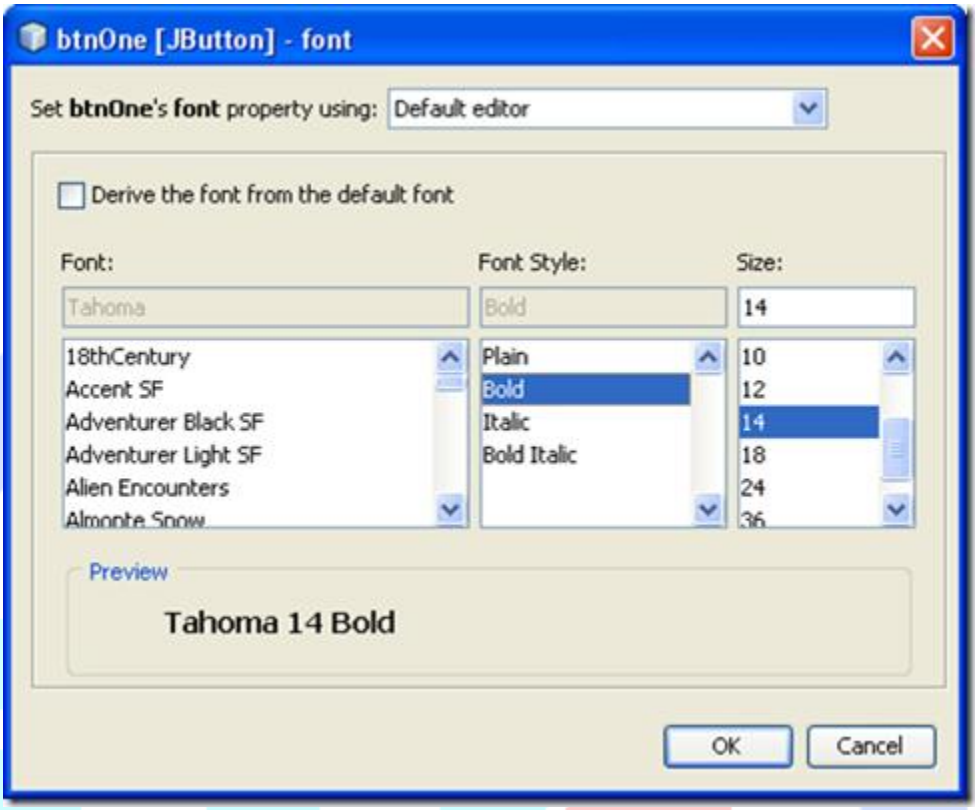
این مقادیر را از حالت پیش فرض به 30 و 50 تغییر دهید.



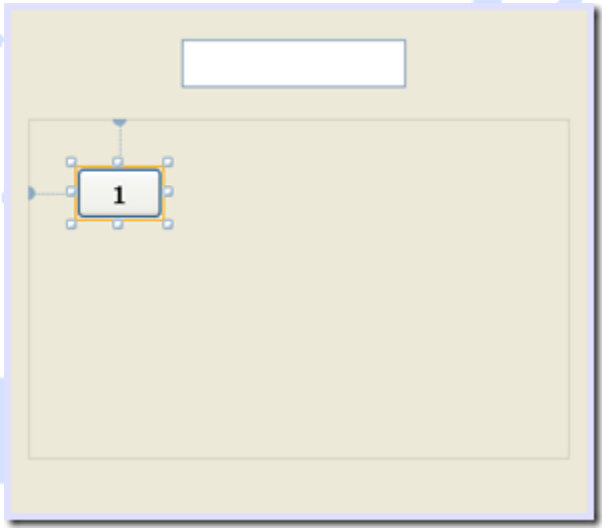
همچنین می توانید شکل فونت و اندازه ی فونت را در پنجره ی properties تغییر دهید. مطمئن شوید که دکمه ی مورد نظر انتخاب شده است. اکنون پراپرتی fontرا قرار دهید. روی دکمه ی کوچک در سمت راست ردیف کلیک کنید.



وقتی روی دکمه ی font کلیک می کنید، باید دیالوگ باکس ظاهر شده را مشاهده کنید. اندازه ی فونت را به 14 تغییر داده و آن را bold کنید.



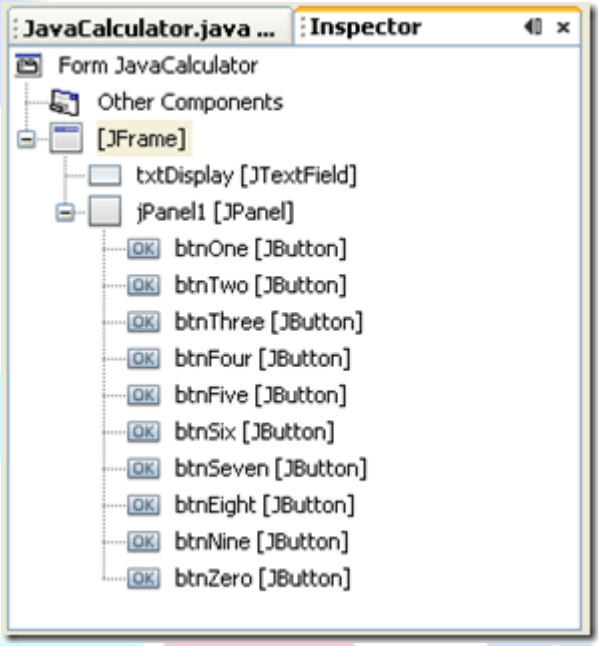
اکنون فرم شما باید مانند تصویر زیر باشد). برای مشاهده ی اوت لاین مربوط بهpanel ، کافیست ماوس خود را روی آن حرکت دهید(



اکنون لازم است به همین روش 9 دکمه ی دیگر نیز اضافه کنید، اعداد از 2 تا 9 و همچنین 0. نام هر متغیر را بهbtnTwo ، btnThree، btnFourو غیره تغییر دهید. متن پیش فرض را حذف کرده و به جای آن یک عدد

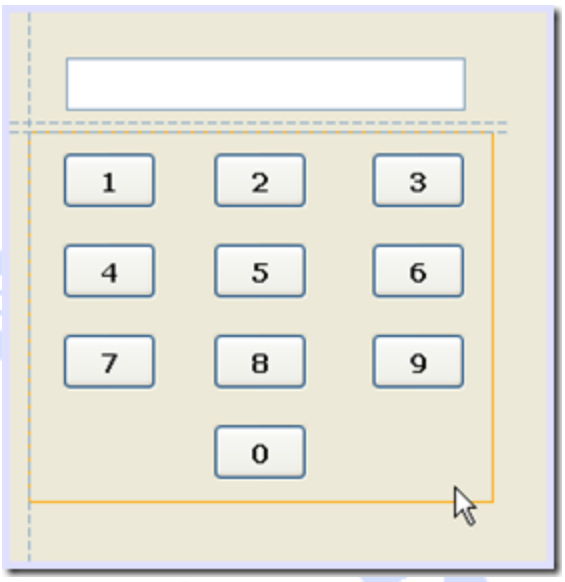
وارد نمایید. سپس اندازه ی هر دکمه را به 50 و 30 تغییر دهید. سرانجام پراپرتی font مربوط به هر دکمه را تغییر دهید.

وقتی کار شما تمام شده است، بخش Inspectoer باید مشابه تصویر زیر باشد.

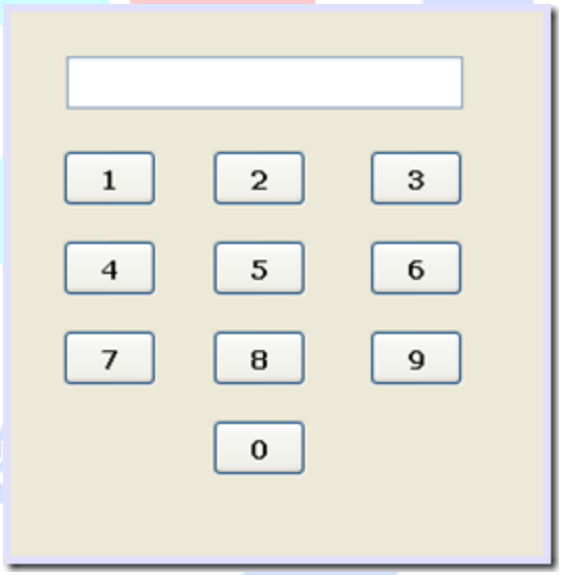


)توجه کنید که دکمه ی 0 دارای نام btnZero می باشد(

اگر دکمه های شما در محل درستی نباشند، روی یک دکمه کلیک کنید تا آن را انتخاب کنید. دکمه ی چپ ماوس خود را به پایین نگاه دارید و سپس در یک موقعیت جدید در panel درگ کنید. برای جابه جا کردن تمام دکمه ها به طور همزمان، panelرا انتخاب کرده و آن را به یک موقعیت جدید درگ کنید.



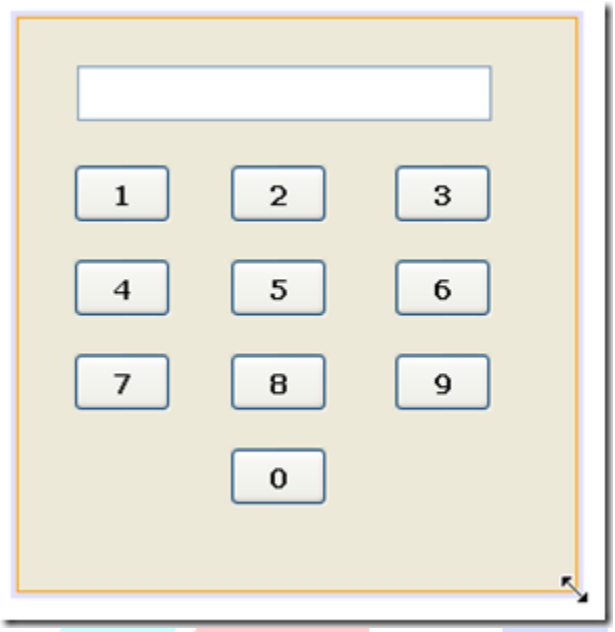
وقتی از فرم خود راضی هستید، باید مشابه تصویر زیر باشد). فیلد text را تغییر اندازه می دهیم(



فقط سه دکمه ی دیگر اضافه کنید: یک دکمه ی به اضافه، یک دکمه ی مساوی و یک دکمه ی پاک کردن. این دکمه ها را در panel خود اضافه خواهیم کرد و آنها را به سمت راست ماشین حساب حرکت خواهیم داد.

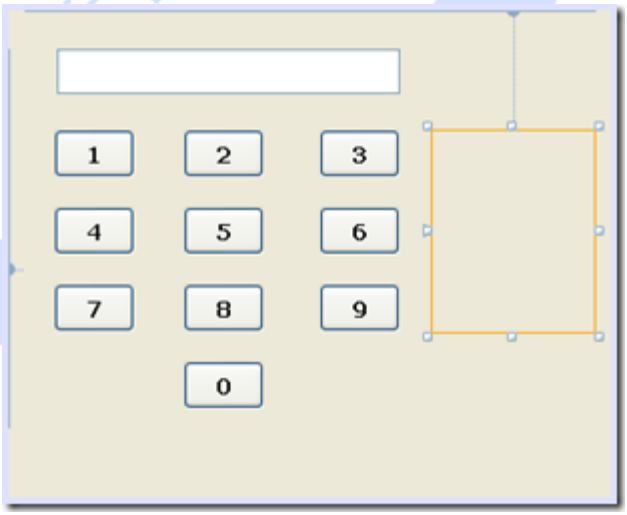
همانطور که مشاهده می کنید، ماشین حساب ما در سمت راست جایی ندارد. اما به راحتی می توانید یک فرم را تغییر اندازه بدهید.

فقط روی فرم کلیک کرده و نه روی فیلد panel یا .text اگر این کار را به درستی انجام دهید، یک مستطیل نارنجی رنگ مشاهده خواهید کرد که اطراف فرم را احاطه کرده است. اکنون ماوس خود را به لبه های فرم حرکت دهید. نشانگر ماوس تغییر شکل می دهد، همانطور که در تصویر زیر مشاهده می کنید.

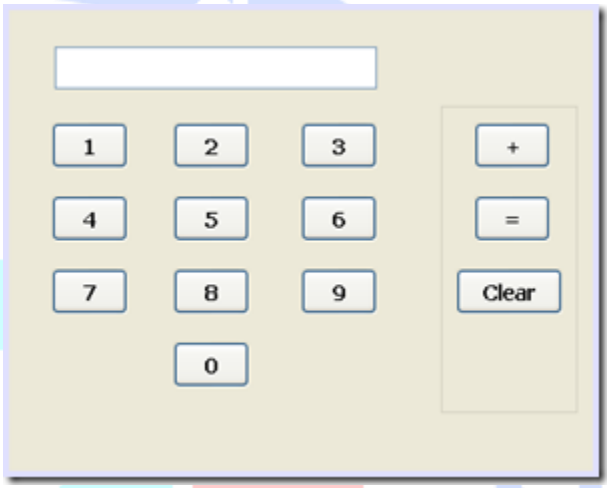


وقتی که نشانگر ماوس تغییر شکل می دهد، دکمه ی چپ ماوس خود را رو به پایین نگاه دارید. اکنون به بک سایز جدید خطوط را درگ کنید.

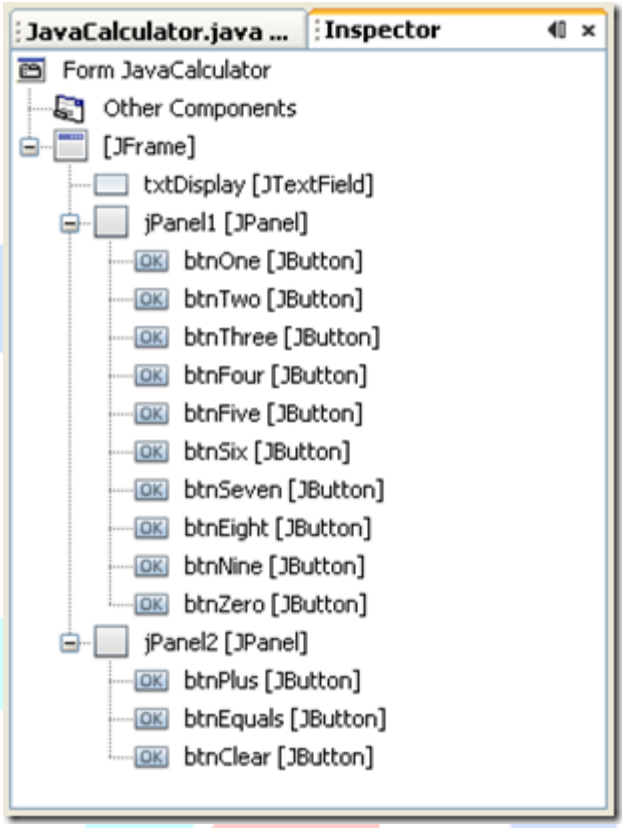
به فضایی که ایجاد کرده اید، یک panel اضافه کنید.



به panel جدید سه دکمه اضافه کنید. نام متغیرها را بهbtnPlus ، btnEqualsو btnClear تغییر دهید. برای دکمه ها و متن، یک علامت + برای دکمه ی به اضافه، یک علامت = برای دکمه ی تساوی و لغت "clear" برای دکمه ی پاک کردن تایپ کنید. فونت را به عدد دکمه ها تغییر دهید، 14 .boldاندازه ی دکمه های به اضافه و مساوی نیز باید با دکمه ی اعداد یکی باشد: 50 و 30 .برای دکمه ی Clear آن را به 70 و 30 تغییر دهید. پس از تمام اینها فرم شما باید مانند تصویر زیر باشد.



بخش Inspector از NetBeans نیز باید مانند تصویر زیر باشد.



در این مرحله می توانید فرم خود را اجرا کنید تا ببینید چطور به نظر می رسد.

ظاهر بسیار خوبی دارد. گرچه وقتی روی دکمه ها کلیک کنید هیچ اتفاقی نمی افتد. ما به طور کوتاه چند کد اضافه خواهیم کرد. ابتدا یک لغت ئر مورد رویدادها. (Events)

آموزش رخدادهای فرم در جاوا

در بخش قبل ماشین حساب خود را در جاوا طراحی کردید. در این بخش رویدادهایی برای آبجکت های فرم جاوا را بررسی خواهیم کرد.

از لحاظ برنامه نویسی، یک رویداد به زمانی گفته می شود که یک اتفاق خاص بیفتد. برای فرم ها رویداد به معنای کلیک کردن دکمه ها، جابه جایی ماوس، وارد کردن متن به فیلدtext ، بستن برنامه و موارد بسیار دیگر.

رویدادها آبجکت هایی در جاوا هستند. آنها از مجموعه ای از گروه ها گرفته می شوند که در java.util.EvenObjectذخیره شده اند. وقتی که یک دکمه کلیک می شود، گفته می شود که آن دکمه منبع رویداد می باشد. اما عمل کلیک کردن یک آبجکت رویداد را تولید می کند. سپس آبجکت رویداد در جستجوی آبجکتی است که منتظر کلیک کردن ماوس یا ضربه زدن به کلید یا هر اتفاق دیگری است که می تواند در یک فرم اتفاق بیفتد. به عنوان مثال یک فیلد text می تواند منتظر ضربه ی یک کلید در صفحه کلید باشد. یا باکس drop down می تواند منتظر آیتم انتخاب شده در باکس باشد.

آبجکت های مختلفی رویدادهای متفاوتی به وجود می آورند. برای یک دکمه، رویدادی که شروع می شود ActionListenerمی باشد. برای یک فیلد text این رویداد KeyEventمی باشد. به موارد مشابه فکر کنید، آبجکت رویداد مسئول انتقال پیام ها بین آبجکت های فرم می باشد که رویدادی بر روی آنها اتفاق افتاده و یا آبجکت هایی که منتظر رویداد هستند.

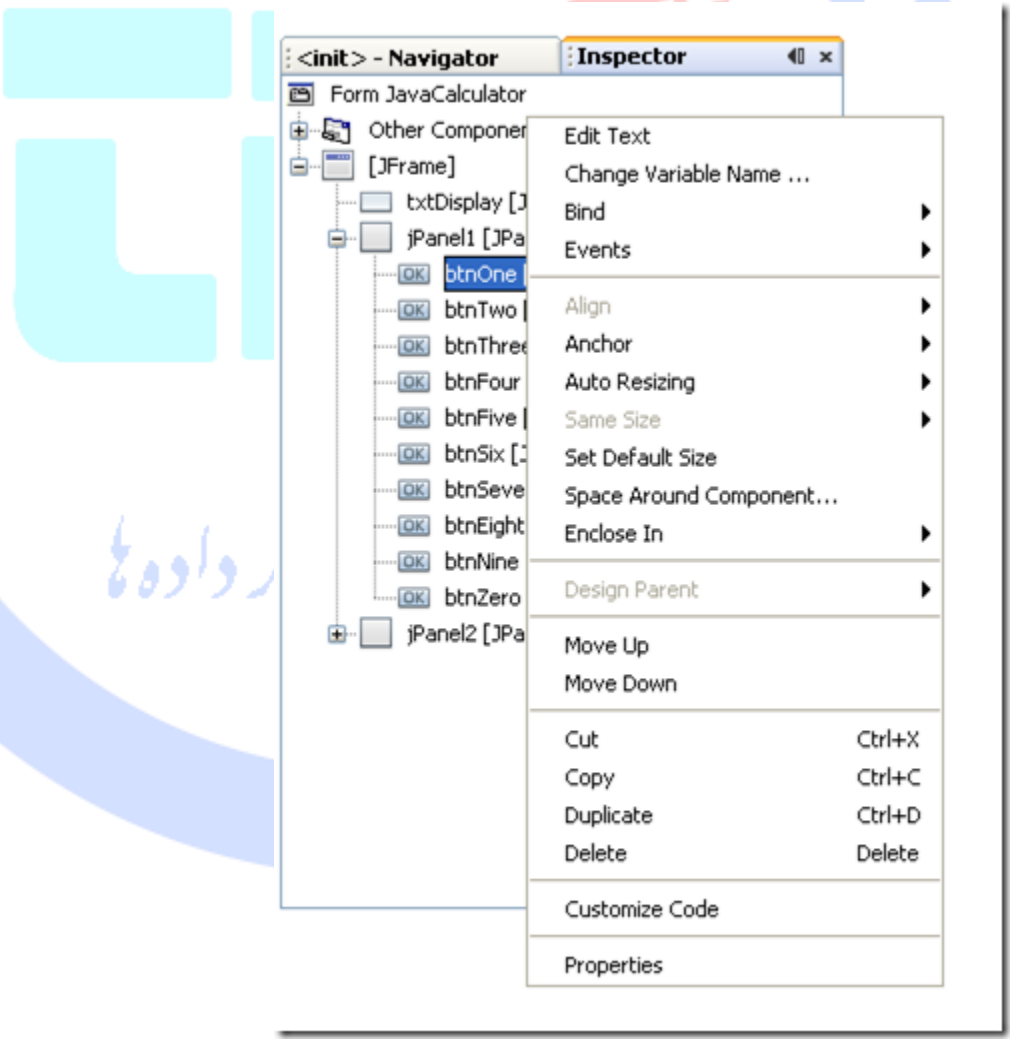
اگر همه ی اینها کمی پیچیده به نظر رسیدند، نگران نباشید. اما چند مثال برنامه نویسی می تواند تمام موارد را واضح سازد. ابتدا صحبتی در مورد کار کردن ماشین حساب.

اگر بخواهیم 3 و 2 را بایکدیگر جمع کنیم (2+3)، ابتدا لازم است روی دکمه ی 3 کلیک کنیم.سپس عدد 3 روی فیلد text ظاهر خواهد شد. پس از آن دکمه ی به اضافه کلیک خواهد شد و این به برنامه در مورد این واقعیت که قصد جمع کردن چند مورد را داریم، پیغام خواهد داد وهمچنین فیلد متن را برای عدد بعدی آماده خواهد کرد. عدد بعدی 2 می باشد و ما این مقدار را همراه با 3 ذخیره می کنیم. برای به دست آوردن مجموع دکمه ی

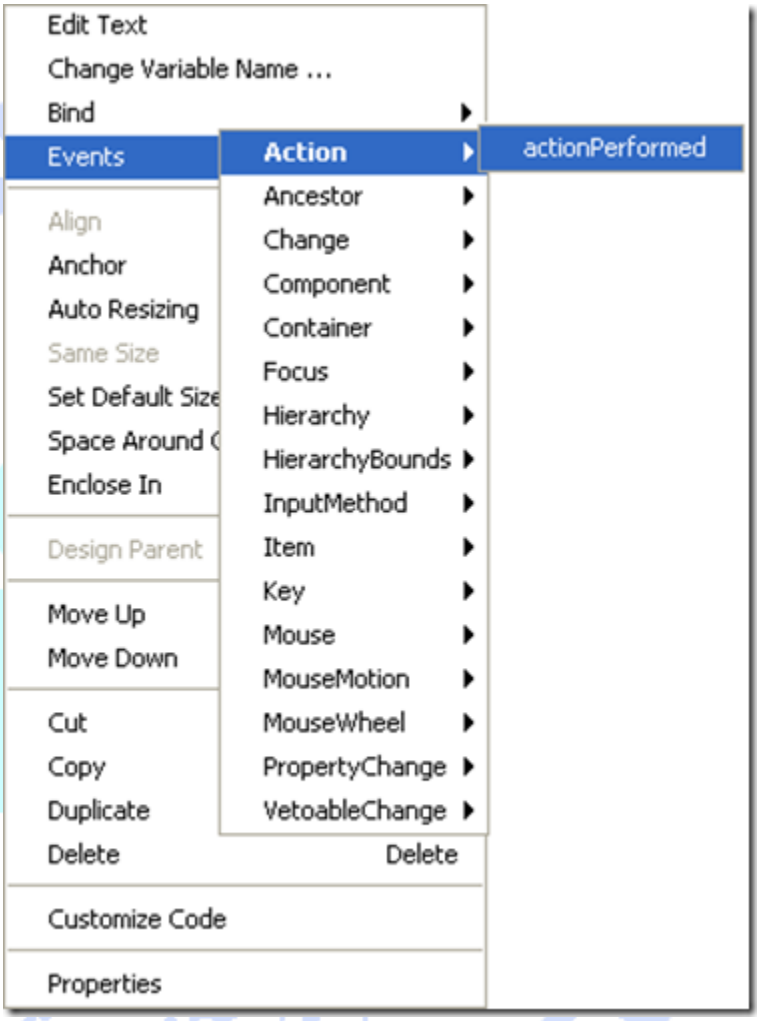
تساوی کلیک می شود و در واقع در اینجا اعداد ذخیره شده را به دست می آوریم (2 , 3) و آنها را با یکدیگر جمع می کنیم. سرانجام پاسخ در فیلد متن ذخیره می شود.

اولین مشکل چگونگی دستیابی به اعداد روی دکمه ها می باشد. این کار را می توانیم با بازگرداندن پراپرتی متن مربوط به دکمه انجام دهیم. زمانی که متن را داشته باشیم می توانیم آن را در تکست باکس قرار دهیم. اما برای زمان کلیک کردن دکمه به یک ActionEvent احتیاج داریم.

در ویو Design در NetBeans دکمه ی شماره ی 1 خود را انتخاب کنید. اکنون نگاهی به پنجره ی Inspector در پایین سمت چپ، داشته باشید. دکمه ی btnOne خود را جایگذاری کنید. حالا کلیک راست کنید تا منوی زیر را مشاهده کنید.



از منوی پیش رو Event را انتخاب کرده و باز از منوی جدیدتر روی Action و سپس روی actionPerformed کلیک کنید.



وقتی که روی actionPerformed کلیک می کنید، محل کد برای btnOne ایجاد خواهید کرد.

مثال

```
private void btnOneActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        }
```

اولین خط کمی بلند می باشد. اما این خط فقط یک متود با یک آبجکت ActionEventبین پرانتزها می باشد.

وقتی دکمه کلیک می شود، هر آنچه بین پرانتزها نوشته ایم، اجرا خواهند شد.

Writing code for the numbers buttons on our Java Calculator

برای به دست آوردن متن از آبجکت فرم، می توانید ازمتود getText استفاده کنید. بنابراین برای دریافت متن از btnOne ، می توانیم از کد زیر استفاده کنیم.

```
String btnOneText = btnOne.getText( );
```

برای دریافت متن از فیلد متن نیز می توان از کد زیر استفاده کرد.

```
String textfieldText = txtDisplay.getText( );
```

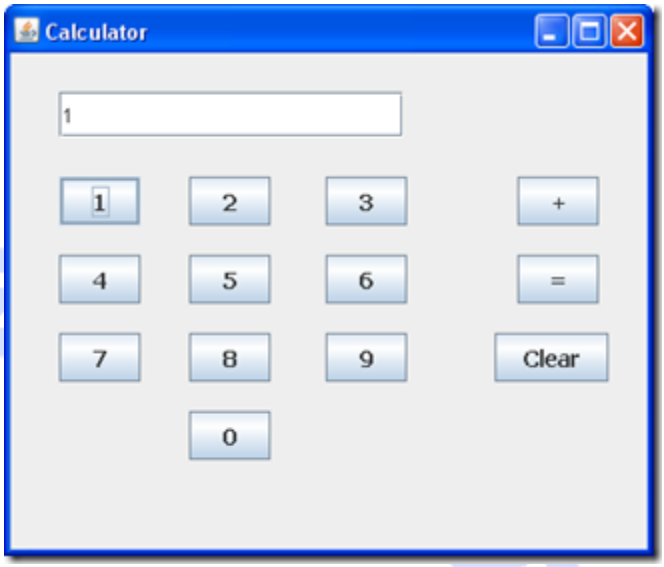
به هرحال اگر بخواهیم چیزی را در داخل فیلد متن قرار دهیم، متود مورد استفاده setText می باشد.

```
txtDisplay.setText( btnOneText );
```

آن را امتحان کنید. این کد را به محل کد اضافه کنید.

```
private void btnOneActionPerformed(java.awt.event.ActionEvent evt) {

        String btnOneText = btnOne.getText( );

        txtDisplay.setText(btnOneText);

        }
```

برنامه ی خود را اجرا کرده و آن را تست کنید. روی دکمه ی 1 کلیک کنید و پس از آن باید عدد 1 را در قسمت فیلد متن مشاهده کنید.



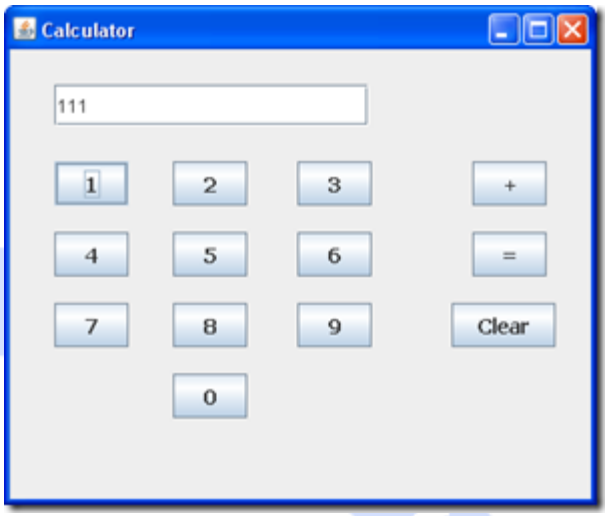
به هرحال مشکلی در اینجا وجود دارد. اگر بخواهید عدد 11 یا عدد 111 را وارد کنید چطور؟ وقتی که به طور مکرر روی یک دکمه کلیک می کنید، اتفاق دیگری نمی افتد. همیشه یک بار عدد 1 تایپ می شود و مهم نیست که چند بار این عدد را کلیک کردید.

دلیل آن این است که کد ما آنچه را از قبل در فیلد متن بوده، حفظ نمی کند و فقط عدد 1 را در آنجا قرار می دهد. اگر می خواستیم عدد 11 را تایپ کنیم، می بایست 1 اول را حفظ کنیم. برای انجام چنین کار باید متن را از فیلد متن گرفته و آن را با دکمه ی text ترکیب کنید. اولین خط کد را به این شکل اصلاح کنید.

```
String btnOneText = txtDisplay.getText() + btnOne.getText();
```

حالا می گوییم که متن را از فیلد متن گرفته و آن را با دکمه ی text ترکیب کنید. نتیجه را در متغیری به نام btnOneTextذخیره کنید.

برنامه ی خود را دوباره اجرا کرده و چند بار روی دکمه ی 1 کلیک کنید. پس از آن باید مشاهده کنید که فیلد متن مجموعه ای از 1 ها را نمایش خواهد داد.



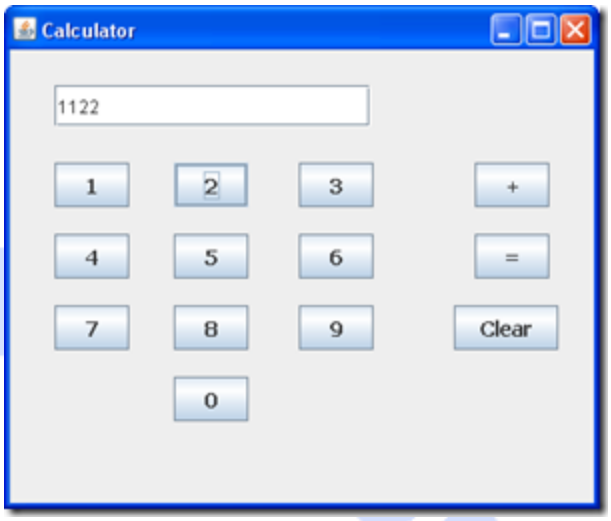
می توانیم همان کد را برای همه ی اعداد روی ماشین حساب اضافه کنیم. به ویو design برگردید. روش دیگر اضافه کردن یک Action به یک دکمه این است که به جای راست کلیک کردن در بخشInspector ، بر روی دکمه دابل کلیک کنید .بنابراین روی دکمه ی عدد 2 دابل کلیک کنید. یک کد ایجاد خواهد شد، درست مانند دفعه ی قبل.

کد زیر را به آن اضافه کنید.

```
private void btnTwoActionPerformed( java.awt.event.ActionEvent evt ) {

        String btnTwoText = txtDisplay.getText() + btnTwo.getText();

        txtDisplay.setText( btnTwoText );

        }
```

تنها تفاوت با کد قبل، نام متغیر String می باشد(اکنون btnTwoText نامیده می شود) و این واقعیت که ما متن را از btnTwo دریافت می کنیم. بین پرانتزهایsetText ، نام متغیر String را انتقال می دهیم.

دوباره برنامه ی خود را اجرا کنید. اکنون باید قادر به کلیک کردن روی دکمه های 1 و 2 باشید و همچنین متنی در فیلد متن ظاهر شود.



همان کد را برای تمام دکمه های ماشین حساب خود اضافه کنید. سپس برای دریافت stub code روی روی هر دکمه دابل کلیک کنید و کد دوخطی را به هر دکمه اضافه کنید. لازم ایت که نام متغیر String را تغییر دهید. در اینجا کد مربوط به دکمه ی 3 را مشاهده می کنید.

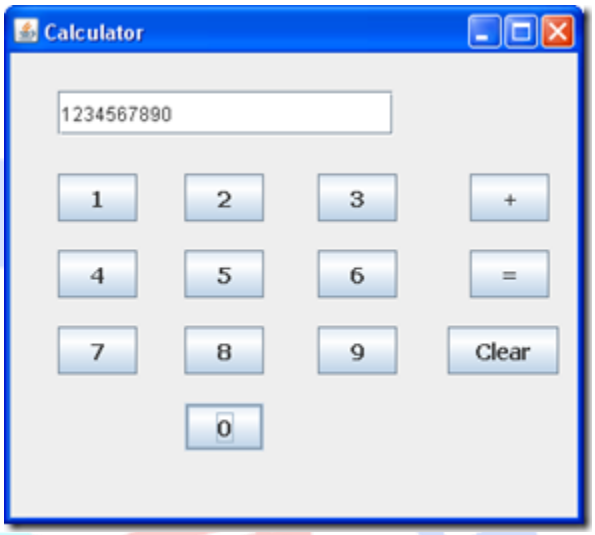
```
String btnThreeText = txtDisplay.getText() + btnThree.getText();

        txtDisplay.setText( btnThreeText );
```

می توانید کدی را که دارید کپی و پیست کنید. سپس فقط نام متغیر String را تغییر دهید، نام دکمه بعد از بخش btn و بخش بین پرانتزهای .setText اگر موردی را مشاهده کردید که زیر آن خط کشیده شده، باید بدانید که در جایی اشتباهی رخ داده است.

وقتی که کار شما تمام شد، کد مربوط به همه ی دکمه ها باید مانند زیر باشد.

```
private void btnOneActionPerformed(java.awt.event.ActionEvent evt) {

    String btnOneText = txtDisplay.getText() + btnOne.getText();
    txtDisplay.setText(btnOneText);
}

private void btnTwoActionPerformed(java.awt.event.ActionEvent evt) {

    String btnTwoText = txtDisplay.getText() + btnTwo.getText();
    txtDisplay.setText(btnTwoText);
}

private void btnThreeActionPerformed(java.awt.event.ActionEvent evt) {

    String btnThreeText = txtDisplay.getText() + btnThree.getText();
    txtDisplay.setText(btnThreeText);
}

private void btnFourActionPerformed(java.awt.event.ActionEvent evt) {
    String btnFourText = txtDisplay.getText() + btnFour.getText();
    txtDisplay.setText(btnFourText);
}

private void btnFiveActionPerformed(java.awt.event.ActionEvent evt) {
    String btnFiveText = txtDisplay.getText() + btnFive.getText();
    txtDisplay.setText(btnFiveText);
}

private void btnSixActionPerformed(java.awt.event.ActionEvent evt) {
    String btnSixText = txtDisplay.getText() + btnSix.getText();
    txtDisplay.setText(btnSixText);
}

private void btnSevenActionPerformed(java.awt.event.ActionEvent evt) {
    String btnSevenText = txtDisplay.getText() + btnSeven.getText();
    txtDisplay.setText(btnSevenText);
}

private void btnEightActionPerformed(java.awt.event.ActionEvent evt) {
    String btnEightText = txtDisplay.getText() + btnEight.getText();
    txtDisplay.setText(btnEightText);
}

private void btnNineActionPerformed(java.awt.event.ActionEvent evt) {
    String btnNineText = txtDisplay.getText() + btnNine.getText();
    txtDisplay.setText(btnNineText);
}

private void btnZeroActionPerformed(java.awt.event.ActionEvent evt) {
    String btnZeroText = txtDisplay.getText() + btnZero.getText();
    txtDisplay.setText(btnZeroText);
```

ماشین حساب خود را اجرا کرده و آن را تست کنید. باید بتوانید تمام اعداد را از 0 تا 9 وارد کنید.



در بخش بعد کد جاوا را برای دکمه ی به اضافه در ماشین حساب خواهید کرد.

آموزش برنامه نویسی دکمه جمع ماشین حساب در جاوا

اکنون که دکمه های عدد در ماشین حساب جاوا کار می کنند، کار بعدی رسیدگی به دکمه ی به علاوه در ماشین حساب می باشد. تنها چیزی که دکمه ی به علاوه برای کار کردن نیاز دارد، ذخیره کردن عدد موجود در فیلد متن می باشد. این ذخیره سازی ثبت اولین عددی است که قرار است به آن اضافه شود. پس از آن نیز کلیک شدن دکمه ی به علاوه را ثبت می کنیم و نه دکمه های دیگری مثل تفریق و یا تقسیم و یا ضرب.

برای ذخیره سازی مقدار نیاز به نیاز به تنظیم یک فیلد متغیر داریم، که در واقع متغیری است خارج از هر کد دکمه ای. این برنامه طوری است که همه ی دکمه ها می توانند آنچه را در دکمه ذخیره شده، ببینند.

متغیر زیر را در بالای پنجره ی کد گذاری (coding) اضافه کنید(می توانید آن را در پایین، همراه با متغیرهای فیلد NetBeans که برای آبجکت های فرم تنظیم شده، قرار دهید؛ اما ما متغیرهای خود را مجزا نگهداری می کنیم)

```
private double total1 = 0.0;
```

در اینجا code window را مشاهده می کنید.

```
public class JavaCalculator extends javax.swing.JFrame {

    private double total1 = 0.0;

    /** Creates new form JavaCalculator */
    public JavaCalculator() {...}

    /**...*/
```

بنابراین ما متغیری به نام total1 را تنظیم می کنیم. نوع متغیر double می باشد و مقدار پیش فرض آن نیز 0.0 می باشد.

برای ذخیره سازی مقدار مربوط به فیلدtext ، نیاز به دریافت متن داریم .اما لازم است آن را از String به Doubleتبدیل کنیم. این کار را می توانید با متود parseDouble method از آبجکت Double انجام دهید.

```
Double.parseDouble( txtDisplay.getText( ) )
```

بین پرانتزهایparseDouble ، متن را از فیلد متن txtDisplay دریافت می کنیم.

به هرحال وقتی مقدار را از فیلد متن به یک متغیر total1 ذخیره می کنیم، لازم است آنچه را در total1 است را نیز حفظ کنیم. همچنین می توانیم فیلد متن را پاک کنیم و برای عدد دوم آماده کنیم.

بنابراین به ویو Design در NetBeans بازگردید. روی دکمه ی به علاوه دابل کلیک کنید تا code stub را تولید کنید. اکنون دو خط زیر را به دکمه ی به علاوه اضافه کنید.

```
total1 = total1 + Double.parseDouble( txtDisplay.getText( ) ) ;

        txtDisplay.setText("");
```

بین پرانتزهایsetText ، یک جفت علامت نقل قول ("") بدون هیچ فاصله ای بین آنها، وجود دارد. این برای پاک کردن فیلد متن کافیست.

در حال حاضر این برنامه مربوط به دکمه ی به علاوه می باشد، بعدا به بررسی آن خواهیم پرداخت. گرچه تمام کاری که انجام می دهیم ذخیره ی یک عدد در متغیر total1 و حفظ آنچه در این متغیر است، می باشد. زمانی که عدد ذخیره می شود، فیلد متن را پاک کرده ایم. اکنون یوزر می تواند عدد دوم را به عدد اول اضافه کند.

در بخش بعد به دکمه ی تساوی کد اضافه می کنیم.

آموزش برنامه نویسی برای دکمه تساوی در جاوا

اکنون که برای دکمه ی به علاوه کد دارید، می توانیم به دکمه ی تساوی بپردازیم.

پس از اینکه یوزر عدد دوم را انتخاب کرد، دکمه ی تساوی باید کلیک شود .کلیک کردن بر روی دکمه ی تساوی پاسخ جمع را تولید خواهد کرد .برای ذخیره سازی خروجی این محاسبه، می توانیم یک فیلد متغیر دیگر تنظیم کنیم. خط زیر را به بالای کد خود اضافه کنید.

```
private double total2 = 0.0;
```

پنجره ی شما باید مانند زیر باشد.

```
public class JavaCalculator extends javax.swing.JFrame {

    private double total1 = 0.0;
    private double total2 = 0.0;

    public JavaCalculator() {
        initComponents();
    }
```

برای دریافت پاسخ این جمع، هر آنچه در حال حاضر در total1 ذخیره شده است، گرفته و آن را به هر آنچه در فیلد متن وجود دارد، اضافه می کنیم .مجددا لازم است رشته را فیلد متن تجزیه کرده و آن را به یک double تبدیل می کنیم.

به ویو Design برگردید و روی دکمه ی تساوی دابل کلیک کنید. در code stub که ایجاد شده، خط زیر را اضافه کنید.

```
total2 = total1 + Double.parseDouble( txtDisplay.getText( ) );
```

این خط متن را از فیلد متن گرفته و string را به double تبدیل می کند .سپس نتیجه به total1 اضافه می شود و پس از آن پاسخ در متغیر total2 ذخیره می شود.

کار دیگری که باید انجام دهیم، نمایش پاسخ در ماشین حساب در فیلد متن می باشد. به هرحال اکنون که فیلدها متن را و نه اعداد را حفظ می کنند، باید دوباره double را به string تبدیل کنیم. اگر سعی کنید مستقیما یک مقدار doubleرا در فیلد متن ذخیره کنید، پیغام خطا دریافت خواهید کرد.

برای تبدیل یک double به متن می توانید از متود toString از آبجکت Double استفاده کنید. خط زیر را درست در زیر اولین خط اضافه کنید.

```
txtDisplay.setText( Double.toString( total2) );
```

این تبدیل بین پرانتزهای setText انجام می شود. اما اگر تمایل داشته باشید می توانید یک متغیر جدید تنظیم کنید.

```
String s1 = Double.toString( total2 );

        txtDisplay.setText( s1 );
```

اما نتیجه همان است : تبدیل double به یک string.

خط آخر برای دکمه های تساوی می تواند متغیر total1 را حذف کند. زمانیکه متغیر total1 حذف شد، یک محاسبه ی جدید آغاز می شود. اینجا خطی را که می توان اضافه کرد، مشاهده می کنید.

```
total1 = 0;
```

اکنون سه خط مربوط به دکمه های تساوی را می توانید مشاهده کنید.

```
total2 = total1 + Double.parseDouble( txtDisplay.getText( ) ) ;

        txtDisplay.setText( Double.toString(total2) );

        total1 = 0;
```

در بخش بعد کدی برای دکمه ی Clear خواهیم نوشت. همچنین چند مثال را بررسی خواهید کرد.

آموزش برنامه نویسی دکمه clear ماشین حساب در جاوا

تنها موردی که کد گذاری آن باقیمانده است دکمه ی Clear می باشد. برای این کار لازم است متغیر total2 را حذف کنیم و متن را در فیلد متن در یک رشته ی خالی تنظیم کنیم.

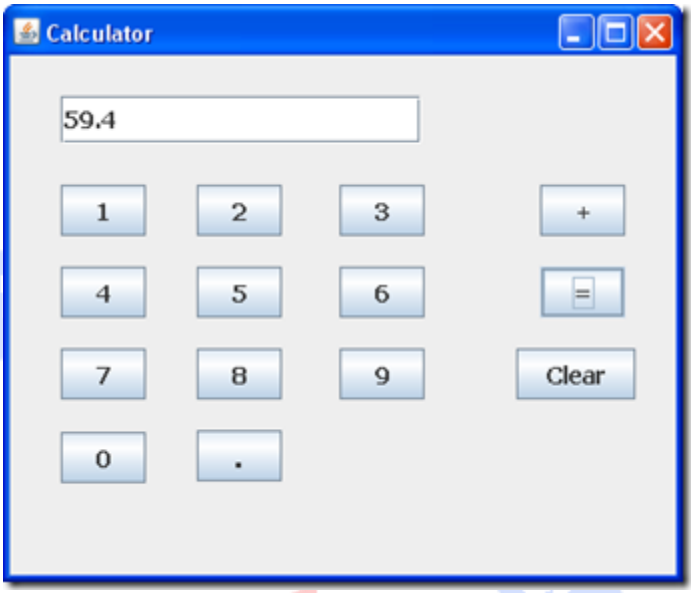
در ویو Design روی دکمه ی Clear دابل کلیک کنید تا code stub را ایجاد کنید. اکنون این دو خط را دکمه ی Clear اضافه کنید.

```
total2 = 0;
        txtDisplay.setText("");
```

زمانیکه این خطوط را اضافه کردید، می توانید ماشین حساب خود را امتحان کنید. برنامه را اجرا کرده و جمع اعداد را تست کنید. ماشین حساب شما باید اعداد را به درستی با هم جمع کند.

به هرحال یک مورد نادیده گرفته شده است – علامت ممیز . (point) در حال حاضر ماشین حساب شما نمی تواند اعدادی مانند 23.6 + 35.8 را با هم جمع کند . بنابراین یک دکمه ی ممیز به فرم خود اضافه کنید. کد را برای آن بنویسید. (در واقع این همان کدی است که برای دکمه های اعداد استفاده کردید)

پراپرتی فونت را برای فیلد متن تنظیم کنید، 14 .bold وقتی تمرین های بالا را کامل کردید، ماشین حساب شما باید چیزی شبیه به تصویر زیر باشد.



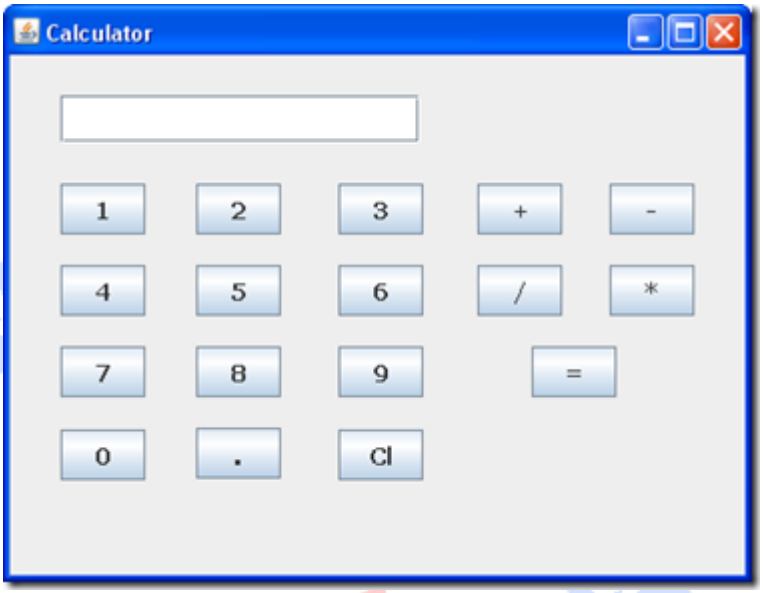
تبریک – اکنون شا یک برنامه ی ماشین حساب جاوا نوشته اید. بسیار خوب، این ماشین حساب فقط می تواند جمع کند، اما یک شروع می باشد.

آنچه انجام خواهیم داد ساخت یک تفریق، تقسیم و ضرب می باشد.

آموزش دکمه های ضرب ، تفریق و تقسیم در ماشین حساب

اکنون که دکمه ی جمع در ماشین حساب جاوا کار می کند، می توانیم دکمه های تفریق، تقسیم و ضرب را اضافه کنیم. درست مانند دکمه ی جمع، این دکمه ها نیز کاری انجام نمی دهند: هنوز دکمه ی تساوی تمام کارها را انجام خواهد داد. تنها کاری که اپراتورهای دکمه ها انجام می دهند، ثبت دکمه ای است که کلیک می شود: جمع، تقسیم، تفریق و یا ضرب.

اولین کاری که باید انجام دهید، قرار دادن چند دکمه در فرم می باشد. در تصویر زیر، دکمه ی Clear را جابه جا کرده ایم و تمام اپراتورهای مربوط به دکمه ها را در سمت راست panel قرار می دهیم. به راحتی می توانید طرح خود را داشته باشید.



زمانی که دکمه های جدید را اضافه کرده اید، متغیرهای پیش فرض را به btnSubtract، btnDivideو btnMultiplyتغییر نام دهید. (روش دیگری برای تغییر نام متغیر کلیک راست کردن روی دکمه می باشد. سپس منویی ظاهر خواهد شد "Change Variable Name" .را انتخاب کنید(.

تکنیکی که برای دریافت دکمه ی کلیک شده استفاده خواهیم کرد، ذخیره کردن متن دکمه در field variable می باشد. سپس می توانیم از یک عبارت switch برای امتحان کردن کاراکتر در field variable استفاده کنیم. اگر این کاراکتر علامت + باشد، می توانیم عمل جمع را انجام دهیم. اگر علامت ؟ باشد می توانیم تفریق انجام دهیم و اگر علامت / باشد، تقسیم خواهیم کرد و اگر علامت *باشد، ضرب خواهیم کرد.

اکنون روی دکمه ی Source کلیک کنید تا به کد خود بازگردید. متغیر فیلد زیر را در بالا و درست زیر دو متغیر دیگر اضافه کنید.

```
private char math_operator;
```

بالای کد شما باید مانند تصویر زیر باشد.

```
public class JavaCalculator extends javax.swing.JFrame {

    private double total1 = 0.0;
    private double total2 = 0.0;
    private char math_operator;

    public JavaCalculator() {
        initComponents();
    }
```

برای دریافت کاراکتر روی دکمه ای که کلیک شده بود، می توانیم متدی را تنظیم کنیم. متد زیر را به کد خود اضافه کنید.

```
private void getOperator(String btnText) {
 math_operator = btnText.charAt(0);
 total1 = total1 + Double.parseDouble(txtDisplay.getText());
 txtDisplay.setText("");
}
```

تا زمانی که متد بالا بین پرانتزهای Class و نه بین کروشه های متود دیگر، می باشد، می توانید آن را در هر جایی در کد خود قرار دهید.

ما متود getOperator را فرا خوانده ایم. این یک متد void می باشد، بنابراین هیچ مقداری گزارش نمی دهد- این متد تنها در اجرای کد همراهی می کند. بین پرانتزهای تیتر متود یک متغیر String به نام btnText داریم. این متغیر مشخصا متن مربوط به دکمه ای است که کلیک شده است.

متن مربوط به دکمه یک رشته می باشد. به هرحال عبارت های switch در جاوا نمی توانند رشته ها را کنترل کنند، بنابراین لازم است که رشته را به یک کاراکتر تغییر دهیم. این کار را می توان به وسیله ی خط زیر انجام داد.

```
math_operator = btnText.charAt(0);
```

متد charAt یک کاراکتر از یک رشته را دریافت می کند. کاراکتر مورد نظر شما بین پرانتزهای charAt قرار می گیرد. نماد ریاضی دکمه های ما در رشته همیشه در کاراکتر 0 قرار می گیرد. سپس در فیلد متغیر char که در بالای کد تنظیم کرده ایم، ذخیره می شود.

به دو خط دیگر در کد دقت کنید. این دو خط دقیقا مشابه خطوط دکمه ی به علاوه هستند و دقیقا همان کار را نیز انجام می دهند – ذخیره سازی اولین عدد در متغیری به نام .total1 اپراتور هر دکمه لازم است که این کار را انجام دهد، بنابراین وجود این دو خط در متد به جای کد مربوط به اپراتور دکمه، معنادار می باشد.

بنابراین کد btnPlus را جایگذاری کرده و دو خط زیر را از آن حذف کنید.

```
total1 = total1 + Double.parseDouble(txtDisplay.getText( ) );

        txtDisplay.setText("");
```

این دو خط را جایگزین آنها نمایید.

```
String button_text = btnPlus.getText();

        getOperator(button_text);
```

دو خط اول متن را از دکمه ی به علاوه دریافت کرده و آن را در یک رشته متغیر ذخیره می سازند. این کار به متود getOperator نیز واگذار می شود.

همان دو خط می توانند به اپراتور دیگر دکمه ها اضافه شوند و تنها نام دکمه را تغییر دهند.

به ویو design بازگردید و روی دکمه ی منها دابل کلیک کنید. برای code stub خطوط زیر را اضافه کنید.

```
String button_text = btnMinus.getText();

        getOperator(button_text);
```

(گرچه برای متغیر String از همان نام استفاده کرده ایم، به هرحال جاوا گیج نمی شود، چرا که متن هر دکمه ای برای کد مخصوص دکمه ی خود داخلی می باشد)

روی دکمه ی تقسیم خود دابل کلیک کرده و خطوط زیر را اضافه کنید.

```
String button_text = btnDivide.getText();

        getOperator(button_text);
```

در اینجا نیز کد مربوط به دکمه ی ضرب را مشاهده می کنید.

```
String button_text = btnMultiply.getText();

        getOperator(button_text);
```

اکنون که کد مربوط به اپراتور هر چهار دکمه را داریم، می توانیم با دکمه ی تساوی تطبیق دهیم.

برای دکمه ی تساوی می توان یک عبارت switch تنظیم کرد تا درک کنیم که متغیر math_operator چیست.

```
switch ( math_operator ) {

        case '+':

        break;

        case '-':

        break;

        case '/':

        break;

        case '*':

        break;

        }
```

عبارت switch برای هر کدام از اپراتورهای ریاضی یک مورد دارد: +، ـ، /، و .*هنوز هیچ کدی اضافه نکرده ایم. اما به کد مربوط به دکمه ی تساوی، نگاهی داشته باشید.

```
total2 = total1 + Double.parseDouble( txtDisplay.getText( ) ) ;

        txtDisplay.setText( Double.toString(total2) );

        total1 = 0;
```

دو خط آخر نیازی به تغییر ندارند. به هرحال خط اول می تواند در عبارت switchاستفاده شود. به یاد داشته باشید که این خط، خطی است که جمع می کند.

این خط می تواند به عنوان کد مربوط به + استفاده شود.

```
case '+':
        total2 = total1 + Double.parseDouble(txtDisplay.getText( ) );
        break;
```

اگر دکمه ی منها کلیک شده باشد، می توان به سادگی در خط بالا جمع را به منها تغییر داد.

```
case '-':

        total2 = total1 - Double.parseDouble(txtDisplay.getText( ) );

        break;
```

کد مربوط به تقسیم نیز به شکل زیر می باشد.

```
case '/':

        total2 = total1 / Double.parseDouble(txtDisplay.getText( ) );

        break;
```

کد مربوط به کاراکتر ضرب هم مانند زیر است.

```
case '*':

        total2 = total1 * Double.parseDouble(txtDisplay.getText( ) );

        break;
```

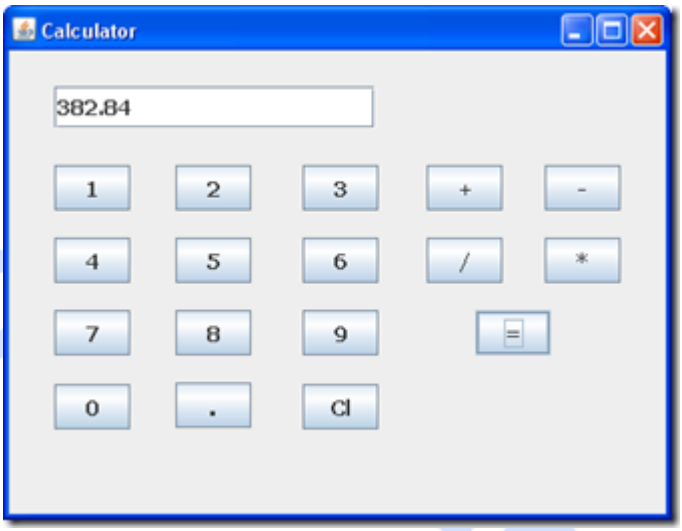
در اینجا تمام کد مربوط به دکمه ی تساوی را مشاهده می کنید.

```
switch ( math_operator ) {
  case '+':
    total2 = total1 + Double.parseDouble(txtDisplay.getText());
    break;
  case '-':
    total2 = total1 - Double.parseDouble(txtDisplay.getText());
    break;
  case '/':
    total2 = total1 / Double.parseDouble(txtDisplay.getText());
    break;
  case '*':
    total2 = total1 * Double.parseDouble(txtDisplay.getText());
    break;
}

txtDisplay.setText(Double.toString(total2));
total1 = 0;
```

زمانی که کد جدید را به دکمه ی تساوی اضافه کرده اید، ماشین حساب را اجرا کرده و آن را امتحان کنید. مورد زیر را امتحان کنید تا ببیند کار می کند یا نه.

```
58. 6 + 37.5 (answer should be 96.1)

        78 - 25.5 (answer should be 52.5)

        68 / 8 (answer should be 8.5)

        56.3 * 6.8 (answer should be 382.84)
```

اکنون یک ماشین حساب ساده دارید که می تواند عملکردهای جمع، تفریق، ضرب و تقسیم را انجام دهد.



حالا باید تمرین هایی با آبجکت های فرم داشته باشید، اجازه بدهید برنامه ی جدیدی ایجاد کنیم که از کنترل های متداول تری که روی یک فرم یافت می کنید، استفاده می کند.

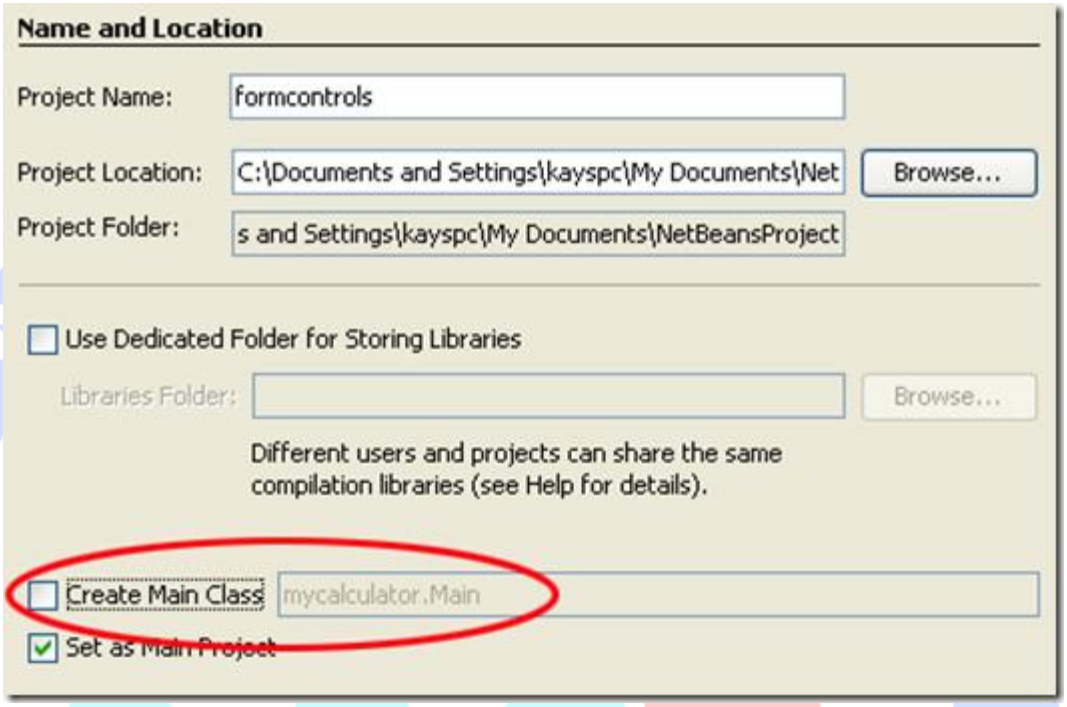
آموزش Combo Boxes ها در جاوا

در این بخش به بررسی کنترل های متداول تری می پردازیم که می توانیم به جاوا اضافه کنیم. چگونگی استفاده از کد زیر را خواهید آموخت:

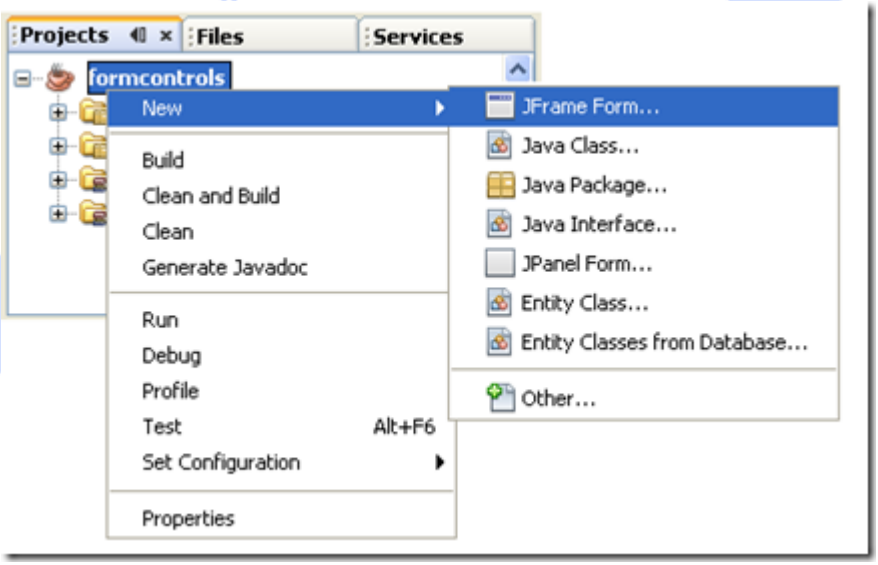
- Combo Box
- Check Box
- Radio Buttons
- Text Areas
- List Box
- Menus and Menu Items
- Open File Dialogue boxes
- Save File Dialogue boxes

که با Combo Boxes آغاز خواهیم کرد.

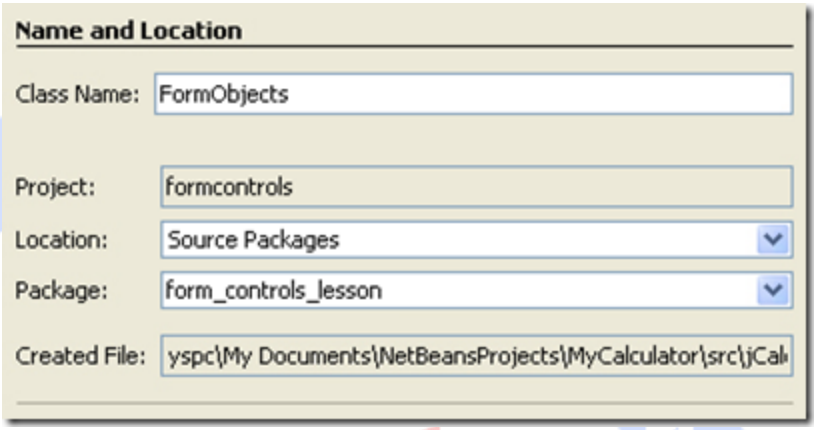
برای این مورد یک پروژه ی جدید ایجاد کنید . (Java > Application) پروژه را formcontrols بنامید و بخش "Create main class"را از حالت انتخاب در آورید:



در wizard روی دکمه ی Finish کلیک کرده تا پروژه را ایجاد کنید. اکنون با راست کلیک کردن روی نام پروژه در پنجره ی Projects و انتخابNew > JFrame Form ، یک فرم اضافه کنید:



وقتی که دیالوگ باکس ظاهر شد، FormObjectsرا با عنوان نام Class و form_controls_lesson را با عنوان نام پوشه وارد کنید:

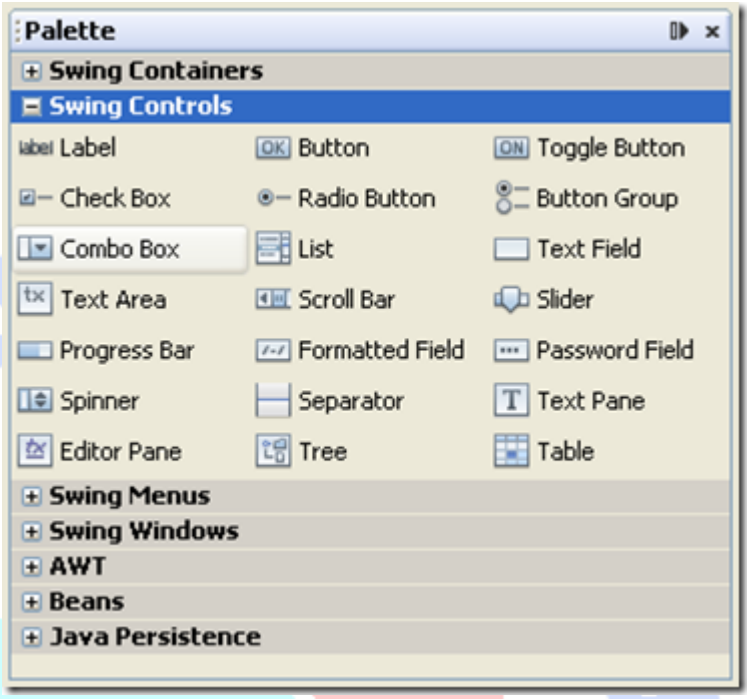


سپس گروهی با نام FormObjects خواهید داشت که در پوشه ای به نام form_controls_lesson و در پروژه ی formcontrols می باشد.

همچنین یک فرم جدید خواهید داشت که روی آن کنترل هایی را اضافه می کنید.

The JComboBox Control

combo boxلیست رو به پایینی از آیتم هایی است که می توانند توسط یک یوزر انتخاب شوند. این لیست در پالت NetBeans زیر کنترل های Swing یافت می شود:

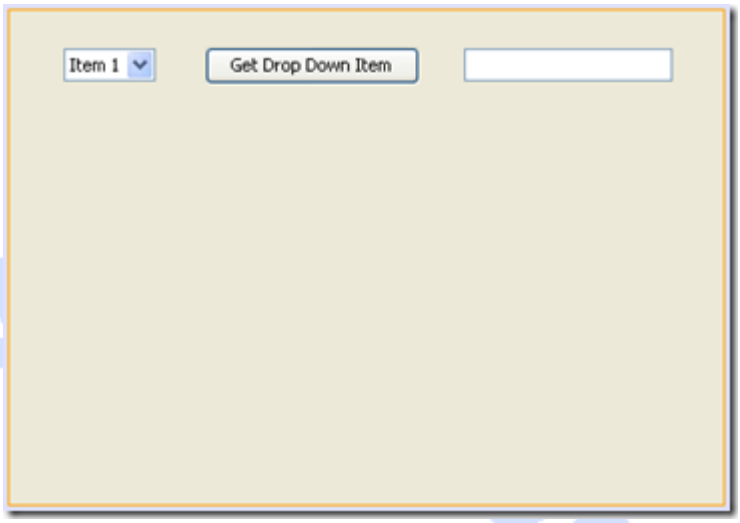


کنترل Combo Box را قرار داده و یکی را روی فرم خود درگ کنید. یک دکمه (Button)و یک Text Field (فیلد متن) را روی فرم خود درگ کنید. آنچه ما انجام خواهیم داد قرار دادن آیتم انتخاب شده از لیست در فیلد متن می باشد .این اتفاق زمانیکه دکمه کلیک شده باشد، خواهد افتاد.

روی back کلیک کرده تا به Combo Box بازگشته و آن را هایلایت کنید. راست کلیک کرده و از منوی ظاهر شده Change Variable Name را انتخاب کنید comboOne .را به عنوان نام جدید تایپ کرده و سپس روی OK کلیک کنید.

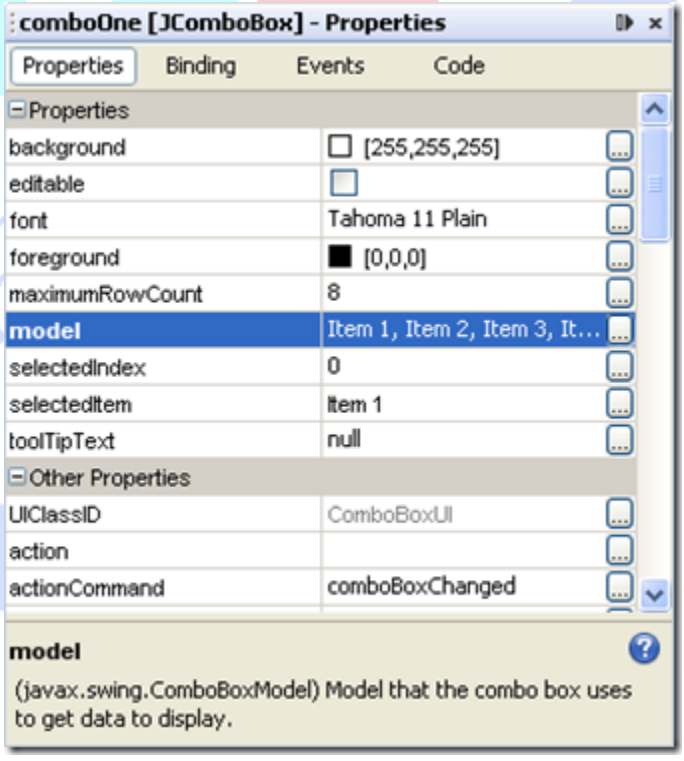
به همین طریق نام دکمه را به btnComboBox تغییر دهید. متن روی دکمه را به Get Drop Down Item تغییر دهید.

نام فیلد متن را نیز به txtComboBoxItem تغییر دهید .متن پیش فرض را حذف کرده و آن را خالی بگذارید. پس از آن فرم شما باید مانند زیر باشد:

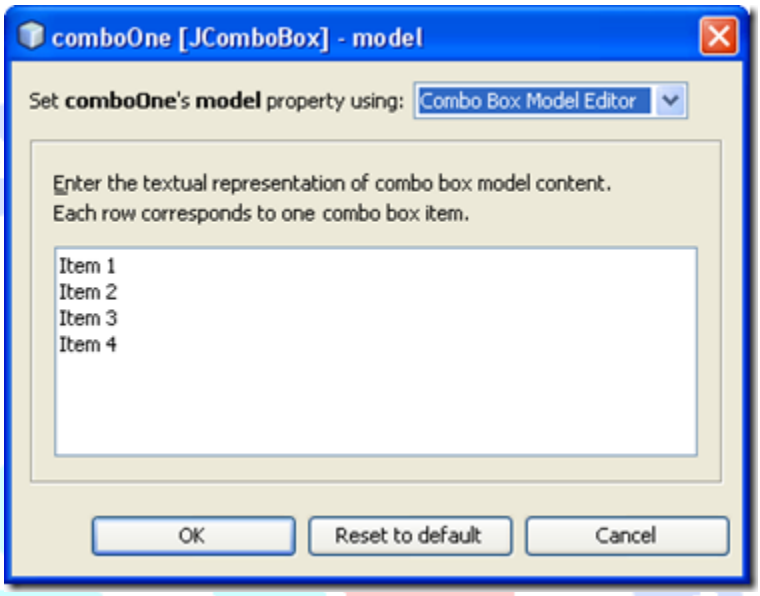


آیتم های پیش فرض در Combo Box عبارتند از Item 1 ، Item 2 و غیره. ما آیتم های خود را اضافه خواهیم کرد.

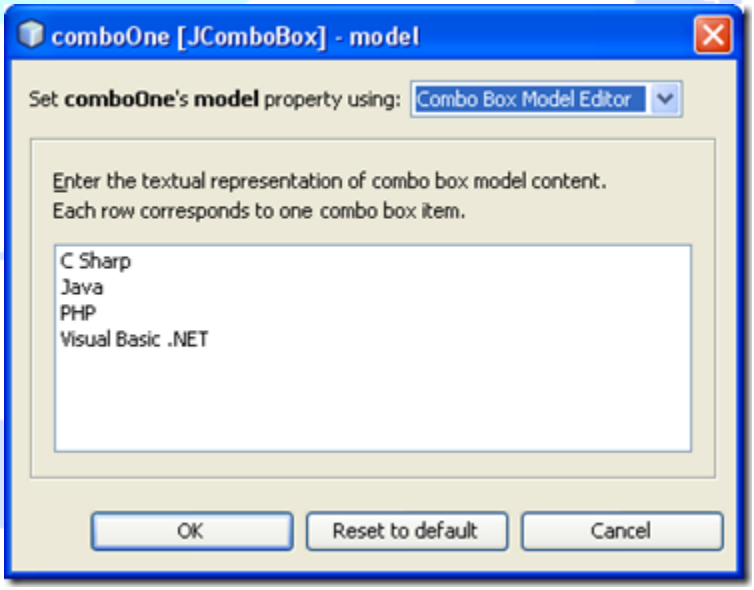
برای انتخاب روی back کلیک کرده تا به Combo Box بازگردید. اکنون به پنجره ی properties در سمت راست NetBeans دقت کنید. پراپرتی modelرا قرار دهید:



روی دکمه ی کوچک در سمت راست ردیف، دکمه ای با سه نقطه، کلیک کنید. دیالوگ باکس زیر ظاهر خواهد شد:

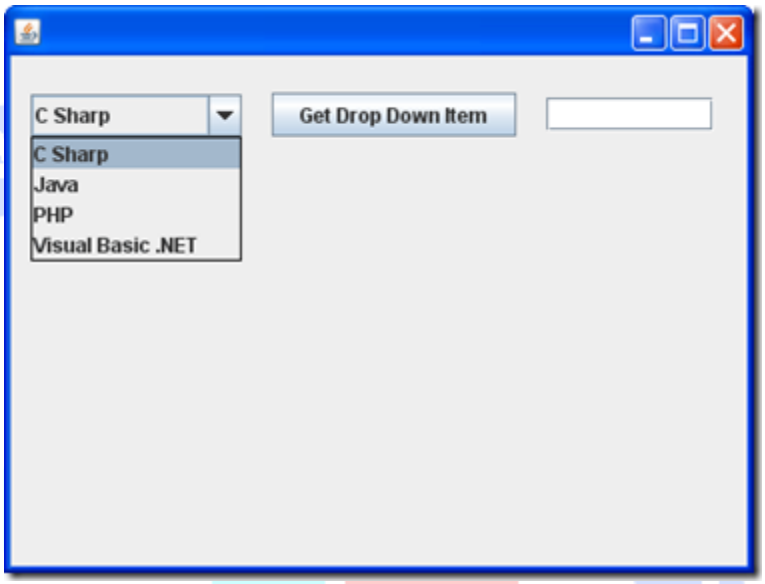


شما می توانید آیتم های بخش سفید را های لایت کرده و آنها را حذف کنید .آیتم های زیر را جایگزین کنید C : Sharp, Java, PHP, Visual Basic .NET. دیالوگ باکس شما مانند زیر خواهد بود:



وقتی تغییرات را اعمال کردید، روی OK کلیک کنید. اکنون combo box شما با آیتم های خودتان پر خواهند شد.

برنامه ی خود را اجرا کرده و آن را امتحان کنید :(وقتی از شما می خواهد تا Main Class را انتخاب کنید فقط روی OK کلیک کنید.)



برنامه را بسته و به ویو design بازگردید.

وقتی که دکمه کلیک می شود، می خواهیم که آیتم انتخاب شده در فیلد متن ظاهر شود. بنابراین روی دکمه ی خود دابل کلیک کنید تا یک code stub ایجاد کنید.

تشخیص اینکه کدام آیتم انتخاب شده است، یک متود مربوط به باکس های combo می باشد که getSelectedItemنامیده می شود. اما این متود یک آبجکت را به عنوان یک مقدار بازمی گرداند. آنچه ما می خواهیم یک متن از یک لیست می باشد. برای تبدیل یک آ Objectبه یکString ، می توانید کاری با عنوان castingانجام دهید. خطوط زیر را به code stub اضافه کنید:

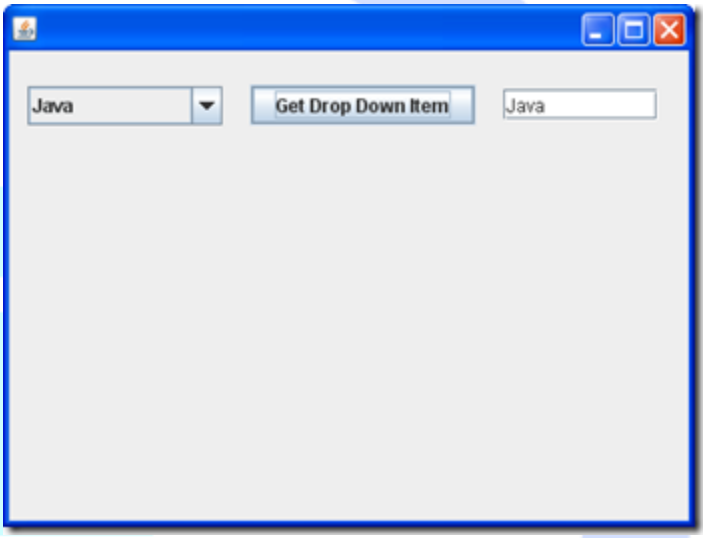
```
String itemText = (String)comboOne.getSelectedItem( );
```

بنابراین ما در حال تنظیم یک ررشته متغیر با عنوان itemText هستیم. پس از علامت تساوی از متود getSelectedItemاز comboOne استفاده می کنیم. اما دقت کنید که casting چگونه انجام می شود – با تایپ متغیر String بین پرانتزها. این درست قبل از آبجکت یا مقداری است که سعی در تبدیل (cast) آن دارید casting) .به معنای تبدیل از یک نوع متغیر به نوع دیگر می باشد(.

برای نمایش آیتم انتخاب شده در فیلد متن، فقط کافیست متن را برای فیلد متن تنظیم کنید. این خط را درست زیر خطی که دارید، اضافه کنید:

```
txtComboBoxItem.setText( itemText );
```

برنامه ی خود را مجددا اجرا کرده و آن را امتحان کنید. از لیست خود یک آیتم انتخاب کنید. سپس روی دکمه ی خود کلیک کنید. آیتمی که انتخاب کرده اید باید در فیلد متن ظاهر شود:



در حال حاضر جعبه ی combo کمی خالی به نظر می رسد. می توانید آن را با کمی رنگ و فونت های متفاوت تنظیم کنید.

برنامه ی خود را متوقف کرده و به ویو Design در NetBeans بازگردید. روی combo box کلیک کرده تا آن را انتخاب کنید. اکنون مجددا نگاهی به پنجره ی properties داشته باشید. تنظیمات زیر را امتحان کنید:

Background Colour

Foreground Colour

Font

Border

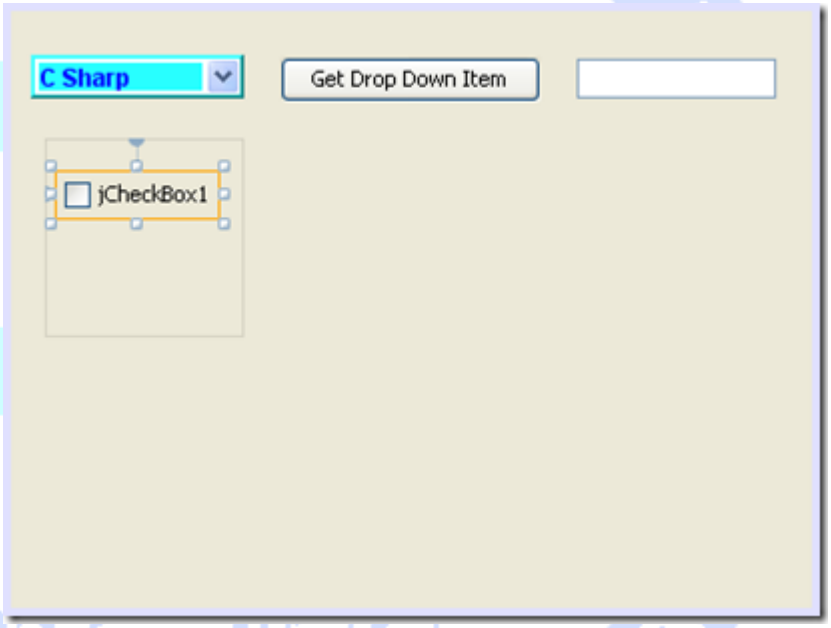
برای مدتی نیاز به اجرا در اطراف آنها دارید. برای رنگ ها RGB به نظر بسیار مناسب می باشد.

در بخش بعد چگونگی کار چک باکس های جاوا را مشاهده خواهید کرد.

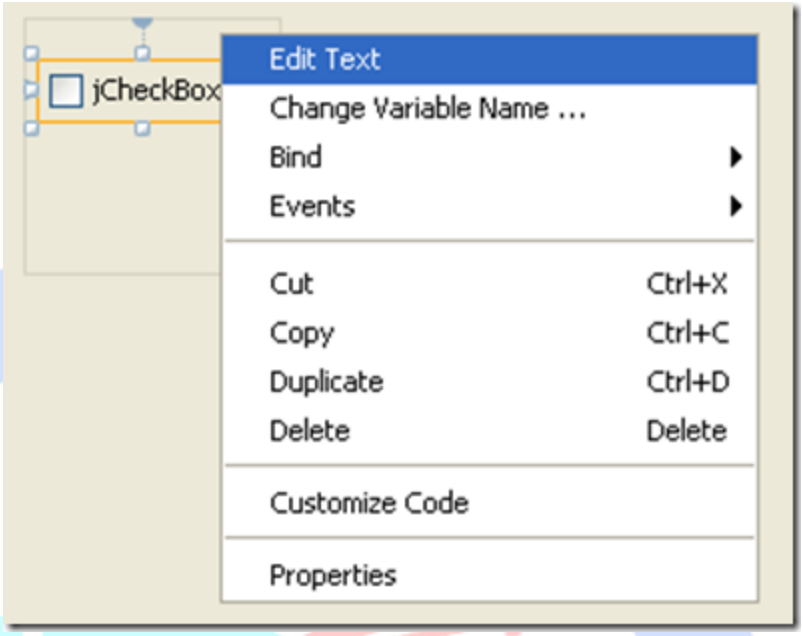
آموزش Check Boxes در جاوا

چک باکس (check box) روشی است که به یوزرها اجازه ی انتخاب کردن یا انتخاب نکردن آیتم ها را می دهد. گرچه می توانند کمی آزار دهنده ی باشند، بنابراین اضافه کردن آنها به panel فکر خوبیست. به این روش می توانید همه ی آنها را همزمان با حرکت دادن panel جابه جا کنید.

بنابراین به فرم خود یک panel اضافه کنید که در زیر Swing Containers در پالت NetBeans یافت می شود. اکنون کنترل چک باکس را جایگذاری کنید. یک چک باکس را روی panel خود درگ کنید:



متن jCheckBox1متن پیش فرض می باشد. می توانید این متن را یا در پنجره ی properties تغییر دهید و یا با راست کلیک کردن بر روی چک باکس. از منوی ظاهرشده Edit Text را انتخاب کنید. (چند مورد از آیتم های منو را در تصویر زیر لیست کرده ایم.):

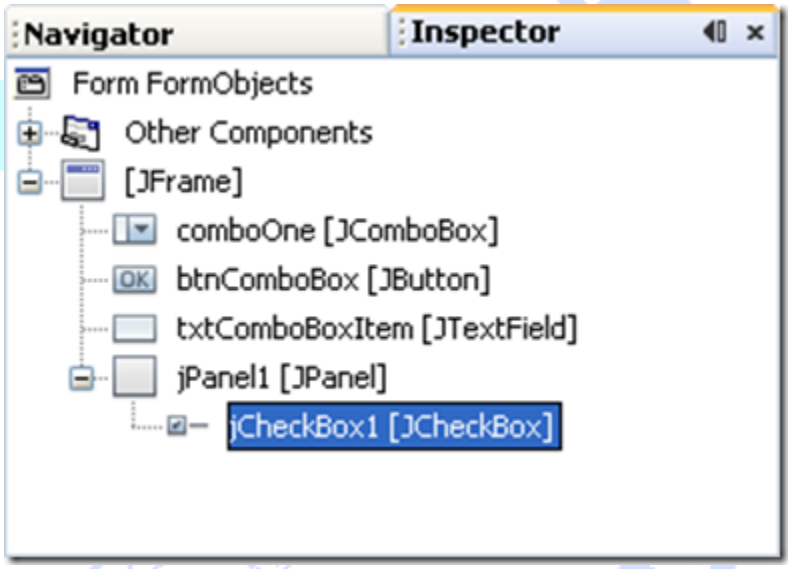


وقتی که روی Edit Textکلیک می کنید، متن پیش فرض های لایت خواهد شد:

C Sharpرا در بالای متن های لایت شده تایپ کنید:

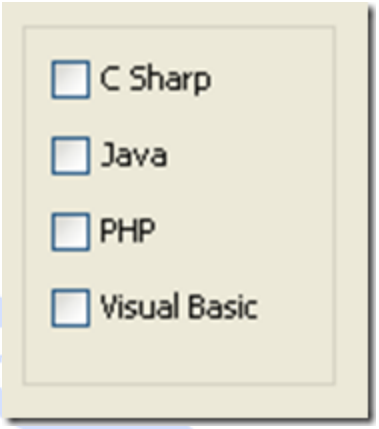
دکمه ی enter در صفحه کلید را فشار دهید، متن برای چک باکس شما تغییر خواهد کرد:

به هرحال این فقط متن را تغییر می دهد و نه نام متغیر را. همانطور که در قسمت Inspector در سمت چپ مشاهده می کنید، نام متغیر هنوز jCheckBox1 خواهد بود:



نام متغیر همان نام پیش فرض بماند. اما به یاد داشته باشید که تغییر متن یک کنترل نام متغیر آن را تغییر نخواهد داد.

اکنون که یک چک باکس به panel خود اضافه کرده اید، می توانید سه مورد دیگر نیز اضافه کنید. متن هر سه را به Java, PHP و Visual Basic تغییر دهید. سپس چک باکس شما مانند تصویر زیر به نظر خواهد رسید:

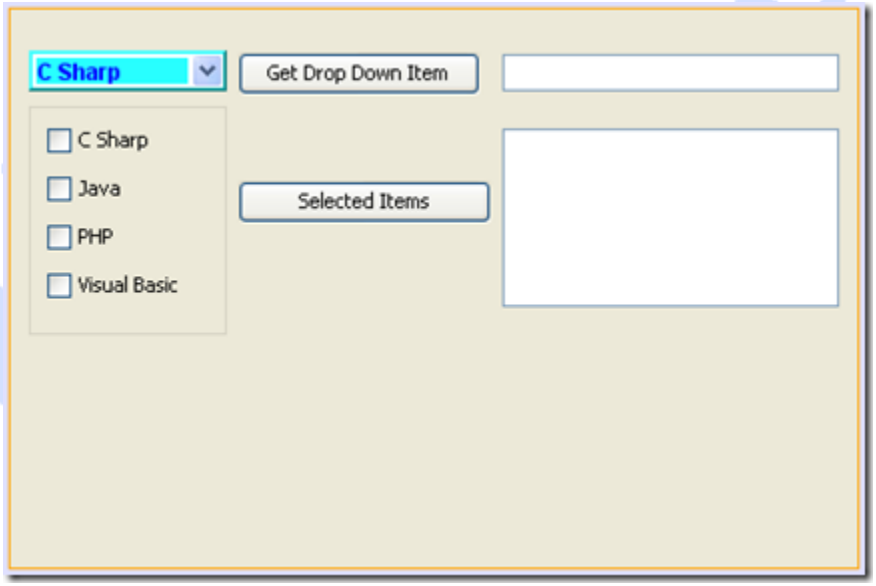


آنچه ما انجام خواهیم داد، دریافت آیتم هایی است که یک یوزر چک کرده است. برای نمایش آیتم ها، به جای استفاده از یک فیلد متن از کنترل Text Area استفاده خواهیم کرد.

بنابراین یک دکمه به فرم خود اضافه کنید. نام متغیر را به btnCheckBoxes تغییر دهید. متن روی دکمه را به Selected Items تغییر دهید.

کنترل Text Area را در پالت NetBeans جایگذاری کرده و یکی را روی فرم خود درگ کنید. نام متغیر را به taOneتغییر دهید.

وقتی کنترل های جدید خود را ردیف کردید، فرم شما باید مشابه تصویر زیر باشد:



اکنون برای کد

چک باکس های جاوا دارای یک پراپرتی به نام isSelected هستند. ما می توانیم از این پراپرتی در مجموعه ای از IF Statements عبارات شرطی (استفاده کنیم تا باکس های انتخاب شده را بررسی کنیم. اگر انتخاب شده باشند، با افزودن متن از هر چک باکس، کی توانیم یک رشته ایجاد کنیم.

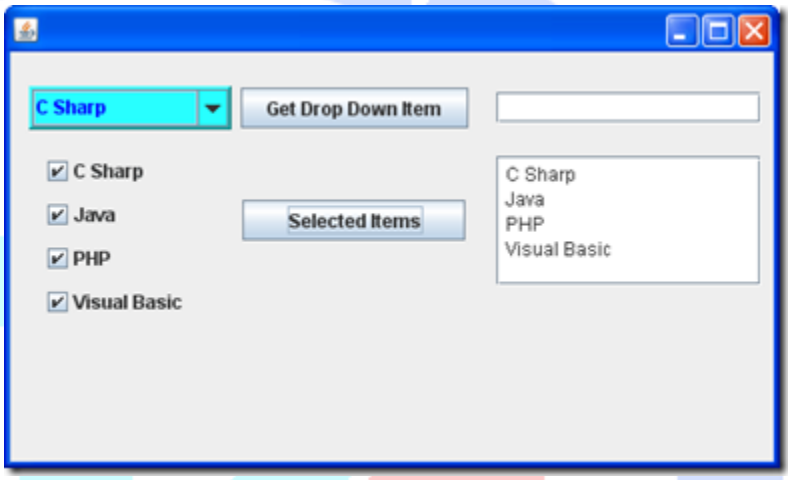
برای ایجاد یک code stub روی دکمه ی جدید خود دابل کلیک کنید. کد زیر را اضافه کنید:

```
String s1 = "";
if (jCheckBox1.isSelected()){
    s1 = s1 + " " + jCheckBox1.getText() + '\n';
}

if (jCheckBox2.isSelected()){
    s1 =  s1 + " " +  jCheckBox2.getText() + '\n';
}

if (jCheckBox3.isSelected()){
    s1 =  s1 + " " +  jCheckBox3.getText() + '\n';
}

if (jCheckBox4.isSelected()){
    s1 =  s1 + " " +  jCheckBox4.getText() + '\n';
}

taOne.setText(s1);
```

رشته ای که در حال ساخت آن هستیم s1 نامیده می شود. اگر یک چک باکس انتخاب شده باشد، متن را از آن چک باکس دریافت خواهیم کرد. که این متن همراه با کاراکتر خط جدید('\n') ، در متغیر s1 ذخیره می شود. آخرین خط از کد، متن را برای بخش text تنظیم می کند. بین پرانتزهایsetText ، متغیر s1 را داریم که رشته ای است که در حال ساخت آن می باشیم.

وقتی که تایپ کردن کد را تمام کردید، برنامه ی خود را اجرا کنید. تعدادی از چک باکس ها را انتخاب کرده و سپس روی دکمه کلیک کنید. باید متوجه شوید که آیتم هایی که انتخاب کرده اید در بخش text ظاهر شده اند:

یک یا چند تا از این باکس ها را از حالت انتخاب در آورید. تنها باکس های انتخاب شده باید در این قسمت ظاهر شوند.



در بخش بعد به Radio Buttons خواهیم پرداخت .

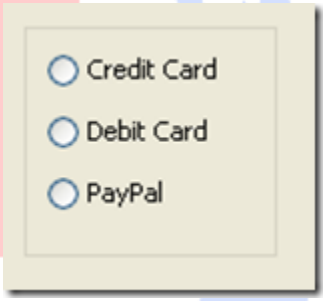
آموزش Radio Buttons ها در جاوا

دکمه های Radio معمولا برای انتخاب تنها یک آیتم از یک لیست، به جای انتخاب چند آیتم موجود، استفاده می شوند. اجازه بدهید چگونگی کار کردن آنها را بررسی کنیم.

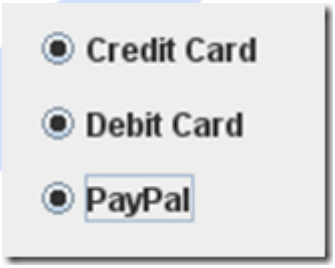
یک panel را روی فرم خود درگ و دراپ کنید. سپس کنترل Radio Button را در پالت NetBeans قرار دهید. یک دکمه ی Radio را روی پالت جدید خود دراگ کنید، که باید شبیه به تصویر زیر باشد:

متن پیش فرض برای اولین دکمه ی radio عبارت jRadioButton1 می باشد. ما از دکمه های radio خود استفاده می کنیم تا به یوزر اجازه بدهیم که یک روش پرداخت (payment method) انتخاب کند. بنابراین متن دکمه ی خود را به Credit Card تغییر دهید. (متن می تواند به همان روشی تغییر کند که در مورد چک باکس ها انجام دادید. مجددا نام متغیر همان نام پیش فرض، jRadioButton1، باقی می ماند.)

دو دکمه ی radio دیگر به panel اضافه کنید. متن را به Debit Card و PayPal تغییر دهید:



به هرحال مشکلی با دکمه های افزوده شده وجود دارد. برای مشاهده ی مشکل مجددا برنامه را اجرا کنید. اکنون یکی از دکمه های radio را انتخاب کنید .سعی کنید یک دکمه ی دیگر انتخاب کنید، مشاهده خواهید کرد که می توانید همزمان بیشتر از یک دکمه انتخاب کنید:



گرچه با این دکمه ها می خواهیم که یوزر فقط یک گزینه ی پرداخت را انتخاب کند. برای حل این مشکل، جاوا به شما اجازه ی ایجاد چیزی به نام ButtonGroupرا داده است. همانطور که از نام آن پیداست، این برنامه به

شما اجازه ی گروه بندی دکمه ها را تحت یک عنوان می دهد. سپس می توانید دکمه های radioرا به یک گروه اضافه کنید. زمانی که دکمه ها را به گروه اضافه کردید، تنها یک گزینه برای انتخاب موجود می باشد.

برای مشاهده ی چگونگی کار کردن ButtonGroup ، متد زیر را به کد خود اضافه کنید:

```
private void groupButton( ) {

ButtonGroup bg1 = new ButtonGroup( );
bg1.add(jRadioButton1);
bg1.add(jRadioButton2);
bg1.add(jRadioButton3);
}
```

هنگام انجام این کار مشاهده خواهید کرد که NetBeans به شما هشدار وجود یک مشکل را می دهد و زیر قسمت هایی از کد را با قرمز خط کشیده است. این اتفاق به این خاطر می افتد که NetBeans نمی تواند گروهی به نام ButtonGroup پیدا کند . در نتیجه نمی تواند آبجکت جدیدی از آن ایجاد کند.

برای حل این مشکل، نیاز به وارد کردن گروه مرتبط از کتابخانه ی Swing می باشد. بنابراین به بالاترین قسمت کد خود رفته و عبارت زیر را وارد کنید:

```
import javax.swing.ButtonGroup;
```

اکنون باید قسمت هایی که به رنگ قرمز خط کشیده شده بودند، از بین رفته باشند.

متد groupButton با استفاده از متود اضافه شده، دکمه های radio را به آبجکت ButtonGroup اضافه می کند:

```
bg1.add( radio_button_name );
```

برای هر دکمه یradio ، یک خط در فرم ما وجود دارد.

می توانیم از constructor متود groupButton را فرا بخوانیم. به این طریق دکمه های radio ، در هنگام بار گذاری فرم، گروه بندی خواهند شد. فراخوانی متود زیر را به constructor خود اضافه کنید:

```
/** Creates new form FormObjects */
public FormObjects() {
    initComponents();
    groupButton();
}
```

بالای پنجره ی کد شما باید مانند تصویر زیر باشد:

```
package form_controls_lesson;

import javax.swing.ButtonGroup;

public class FormObjects extends javax.swing.JFrame {

    /** Creates new form FormObjects */
    public FormObjects() {
        initComponents();
        groupButton();
    }
```

برنامه ی خود را مجددا اجرا کرده و سعی بیشتر از یک دکمه ی radio انتخاب کنید. اکنون مشاهده می کنید که می توانید فقط یک دکمه در گروه انتخاب کنید.

برای مشاهده ی دکمه ی انتخاب شده، متودی به نام isSelected وجود دارد که می توانید استفاده کنید.

یک دکمه ی معمولی به فرم خود اضافه کنید. وقتی روی این دکمه کلیک می کنیم، یک پیغام برای ما نمایش داده خواهد شد که بیان می کند کدام دکمه کلیک شده.

نام متغیر دکمه ی خود را به btnRadios تغییر دهید. پراپرتی متن را به Payment Option تغییر دهید.

اکنون روی دکمه ی جدید خود دابل کلیک کنید تا یک code stub ایجاد کنید. کد زیر را اضافه کنید:

```
String radioText = "";

if (jRadioButton1.isSelected()) {
    radioText = jRadioButton1.getText();
}

if (jRadioButton2.isSelected()) {
    radioText = jRadioButton2.getText();
}

if (jRadioButton3.isSelected()) {
    radioText = jRadioButton3.getText();
}
```

تمام کاری که در اینجا انجام می دهیم بررسی دکمه ی radio انتخاب شده می باشد. سپس متن را از دکمه ی radioدریافت کرده و آن را در متغیری به نام radioTextمرتب می کنیم.

می توانیم یک باکس پیغام داشته باشیم تا گزینه ی پرداخت انتخاب شده را نمایش دهیم. خط زیر را به پایین کد دکمه ی خود اضافه کنید، درست زیر آخرین عبارتIF :

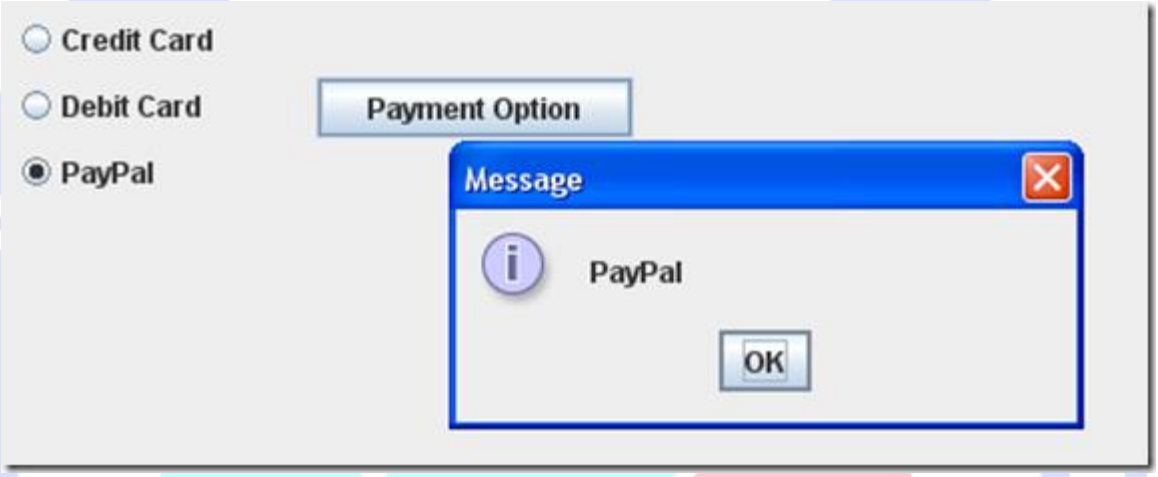
```
javax.swing.JOptionPane.showMessageDialog( FormObjects.this, radioText );
```

از آنجایی که این خط طولانی است، مجبوریم اندازه ی فونت را تغییر دهیم .اما JOptionPane را در بخش قبل مشاهده کرده اید. تنها تفاوت اولین آیتم بین پرانتزها می باشد. از آنجایی که ما از یک console استفاده می کردیم، اولین آیتم پوچ بود. اکنون:

FormObjects.this

اولین آیتم بین پرانتزها برای پنجره ای است که در آن قصد نمایش پیغام را دارید Null .به معنای بدون پنجره (no window)می باشد. عبارت FormObjects.this نیز به معنای این مولفه از گروه FormObjects ، می باشد.

برنامه ی خود را مجددا اجرا کرده و یک آیتم از radio button انتخاب کنید. سپس روی دکمه ی خود کلیک کنید. پس از آن باید صفحه ای مانند تصویر زیر را مشاهده کنید:

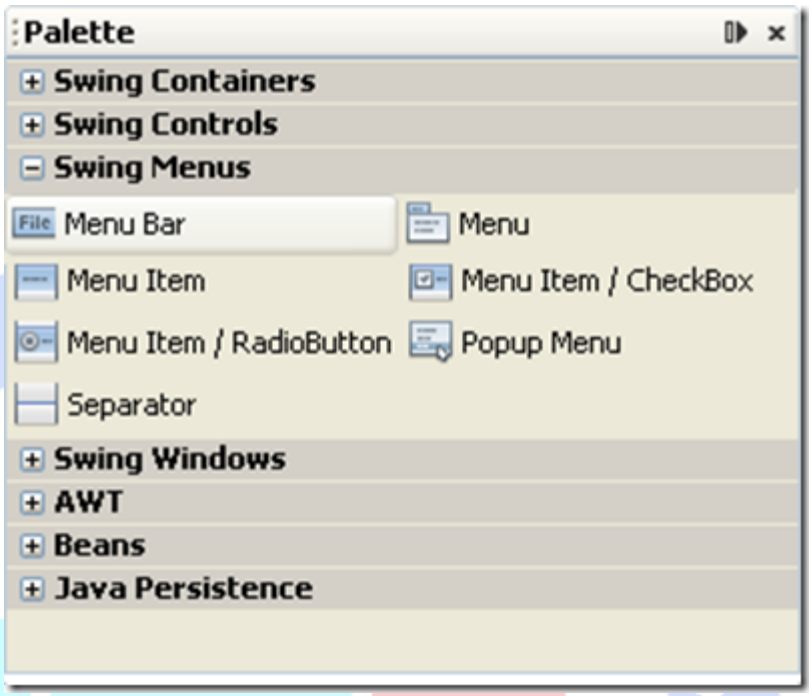


در بخش بعد چگونگی اضافه کردن منوها به فرم های جاوا را مشاهده خواهید کرد .

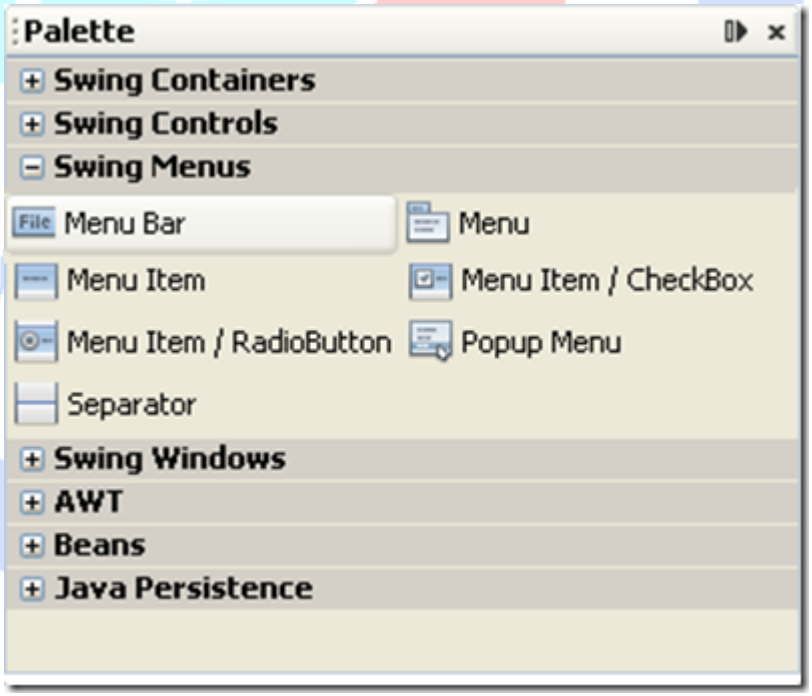
آموزش منوهای جاوا

شما می توانید به فرم های جاوای خود منوهایی اضافه کنید، مواردی مانند File، Edit، View و غیره. هر منو دارای آیتم هایی می باشد که این آیتم ها نیز دارای منوهایی می باشند.

به ویو Design بازگردید. در پالتNetBeans ، آیتم Menu Bar را قرار دهید:

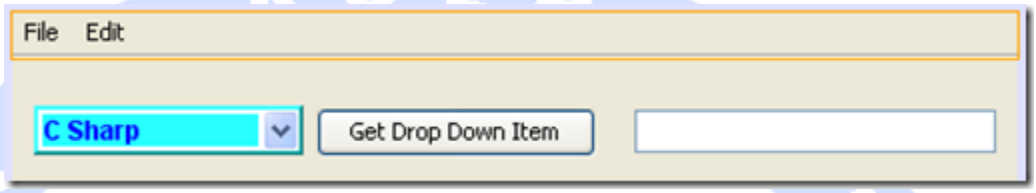


یکی را به بالای فرم خود درگ کنید. وقتی به دکمه ی ماوس اجازه ی حرکت می دهید، یک نوار منوی پیش فرض File و Edit خواهید داشت:



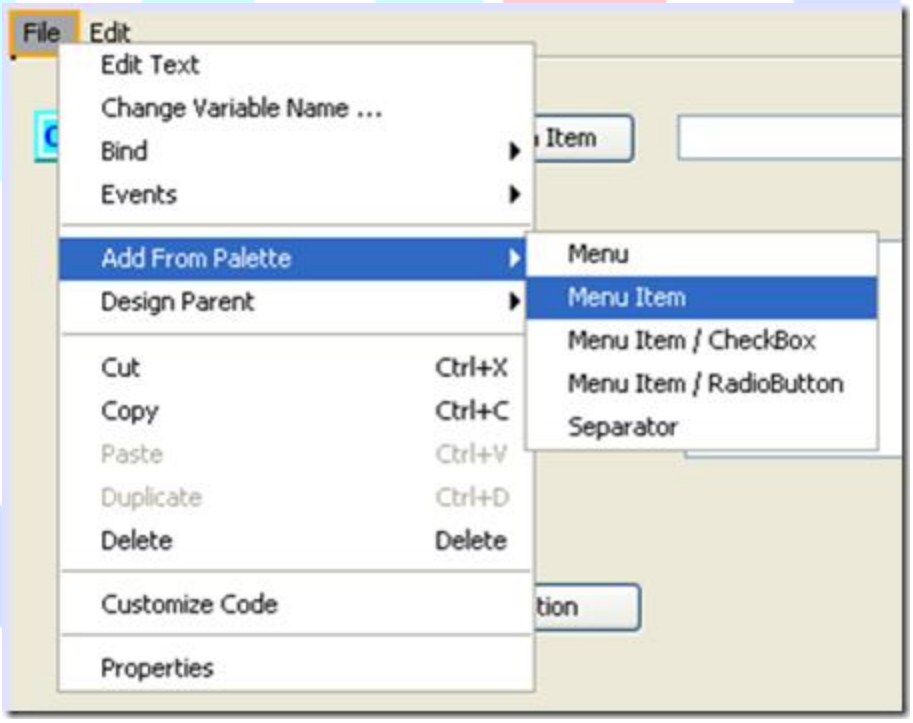
گرچه هیچ آیتم منویی به طور پیش فرض اضافه نشده است. برای اضافه کردن آیتم های خود، روی منوی File کلیک کرده تا آن را انتخاب کنید. با انتخاب آیتم منویFile ، راست کلیک کنید. یک منوی جدید ظاهر خواهد شد Add From Palette > Menu Item . را انتخاب کنید:

یک Menu Item به منوی File اضافه خواهد شد:

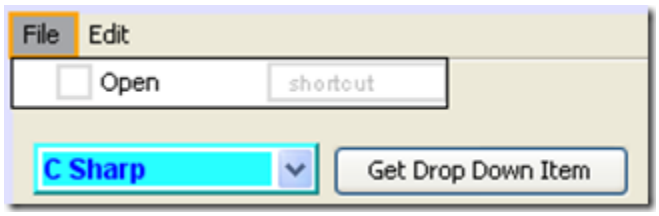


آنچه ما انجام خواهیم داد افزودن آیتم های منو برای باز کردن و یا ذخیره کردن یک فایل می باشد.

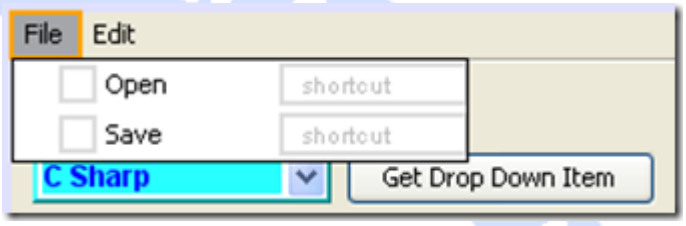
روی متن پیش فرض jMenuItem1 دابل کلیک کنید. سپس این متن های لایت خواهد شد، طوریکه می توانید روی آن تایپ کنید:



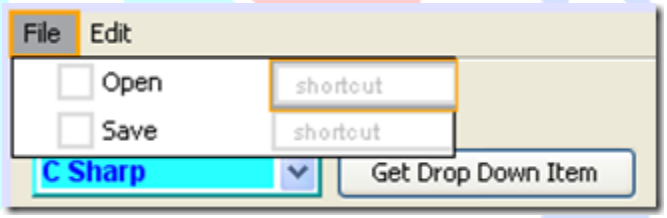
لغت Open را تایپ کرده و سپس دکمه ی enter را روی صفحه کلید فشار دهید:



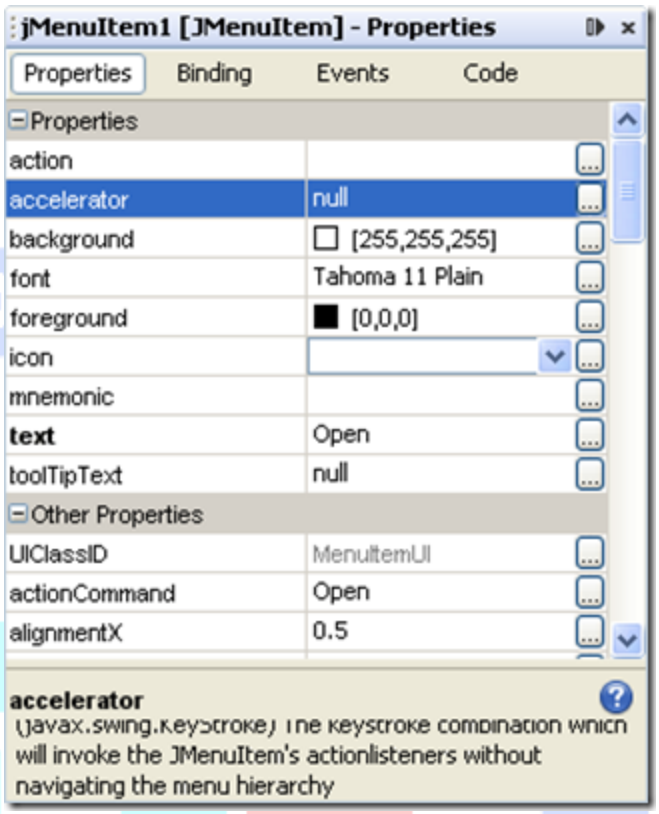
به این طریق یک آیتم دیگر اضافه کنید. این بار Save را به عنوان آیتم منو تایپ کنید:



همانطور که در بالا مشاهده می کنید، می توانید برای آیتم های منوی خود میانبرهایی (shortcuts) اضافه کنید. روی آیتم Open و سپس روی shortcut برای آن کلیک کنید:

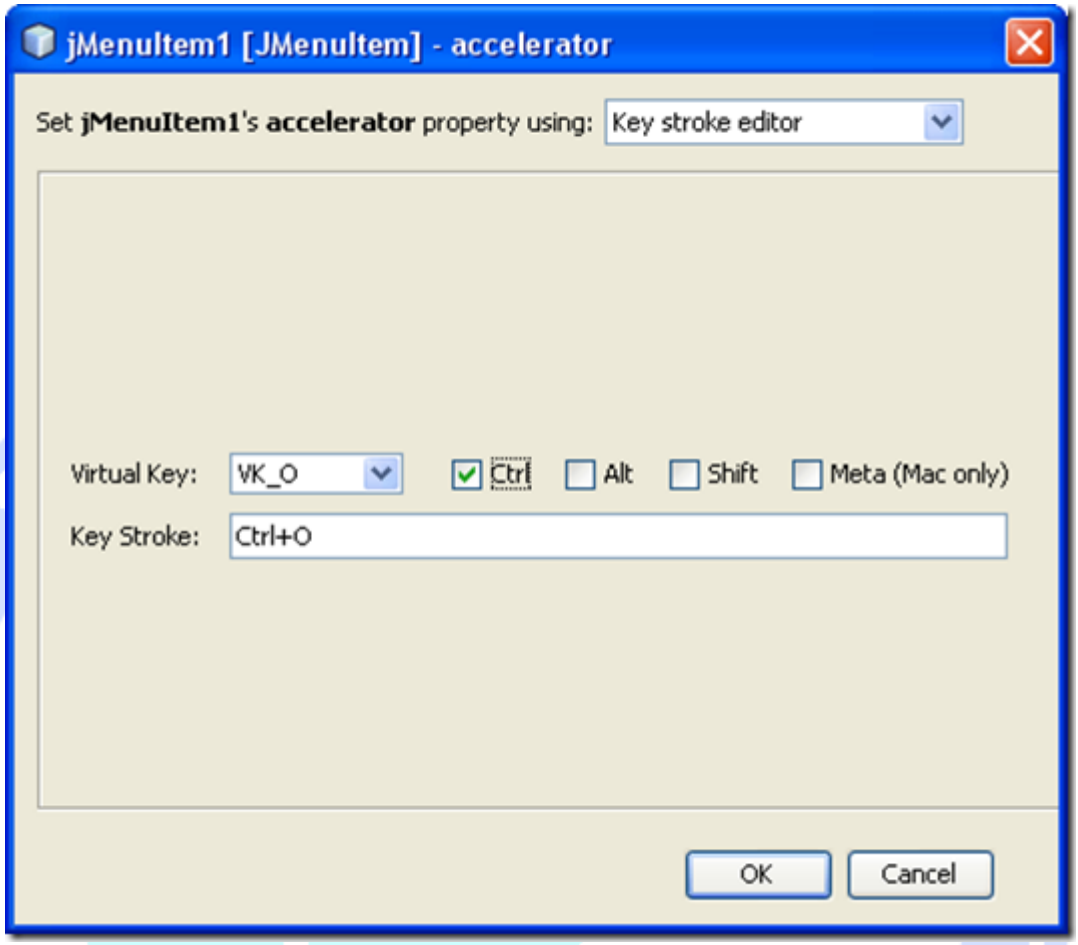


با آیتم انتخاب شده یshortcut ، نگاهی به پنجره ی properties داشته باشید:

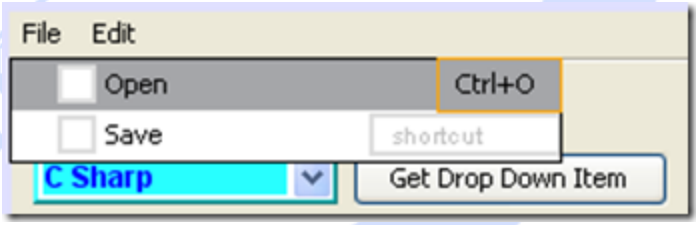


آیتم Accelerator را جایگذاری کرده و سپس روی دکمه ی کوچک در سمت راست ردیف کلیک کنید. یک دیالوگ باکس ظاهر می شود. در این دیالوگ باکس می توانید میانبر کلیدهای مورد نظر خود را تنظیم کنید. میانبر open معمولا CTRL + O می باشد.

یک O در باکس تایپ کنید که پس از آن Shift + O ظاهر خواهد شد. آیتم را Shift را از حالت انتخاب درآورده و در عوض Ctrl را انتخاب کنید:



روی OK کلیک کنید، پس از آن این میانبر به آیتم های منوی جاوا افزوده خواهد شد:



برای اینکه مشاهده کنید تمام این برنامه کار می کند یا نه، در منوی Openروی back کلیک کنید تا به حالت های لایت در آید. اکنون کلیک راست کنید. از منوی ظاهر شده Events > Action > Action Performed را انتخاب کنید. این گزینه برای آیتم منو یک code stub ایجاد خواهد کرد. خط زیر را برای کد وارد کنید:

```
javax.swing.JOptionPane.showMessageDialog( FormObjects.this, "Open" );
```

این تنها یک باکس پیغام می باشد.

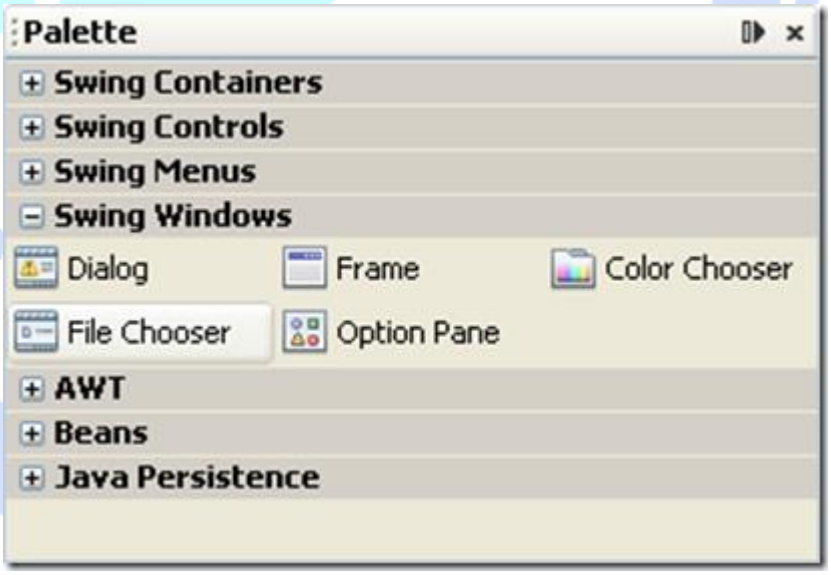
برنامه ی خود را اجرا کرده و آن را امتحان کنید. روی File > Open کلیک کنید و پس از آن باید message boxرا مشاهده کنید. روی OK کلیک کنید تا از این باکس خلاص شوید. اکنون میانبر خود را امتحان کنید. دکمه ی Ctrl روی صفحه کلید را نگه دارید و سپس حرف O را فشار دهید. مجددا منو ظاهر می شود.

برنامه را متوقف کرده و به code stub بازگردید. روی message boxکامنت بگذارید. آنچه انجام خواهیم داد، نوشتن یک کد برای نمایش دیالوگ باکس File Open می باشد. سپس یک یوزر می تواند یک فایل را برای باز شدن انتخاب کند .این مار را در بخش بعد انجام خواهیم داد .

آموزش File Choser در جاوا

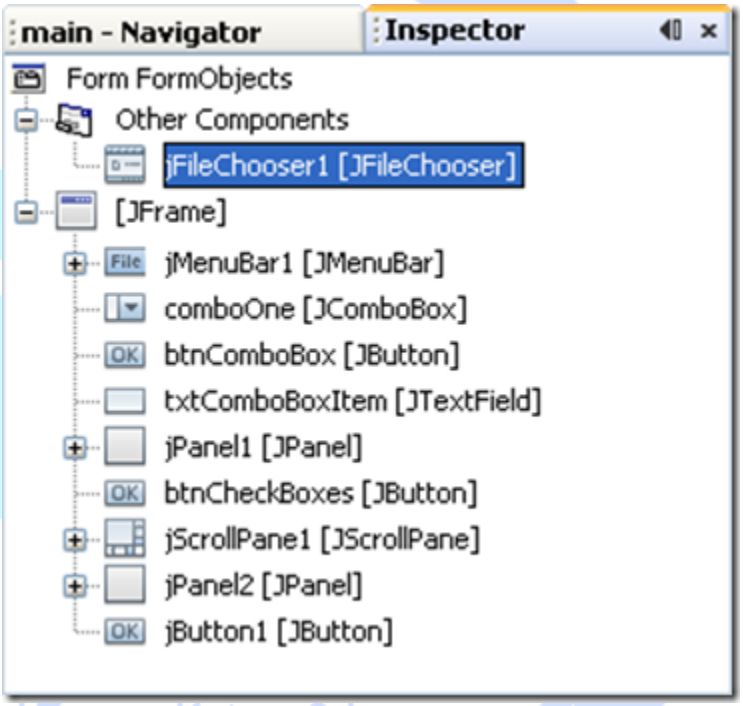
در این بخش چگونگی نمایش دیالوگ باکس های Open File را در جاوا مشاهده خواهید کرد. این کار توسط File Choserانجام می شود.

به ویو Design بازگردید. در پالت NetBeans آیتم File Chooser را قرار دهید که در زیر Swing Windowsقرار دارد:



یک File Chooser را به نزدیک فرم خود و نه در داخل آن درگ کنید و آن را درست زیر فرم و در منطقه ی سفید قرار دهید. این برنامه در واقع روی فرم ظاهر نخواهد شد، اما می توانید آن را در پنجره ی Inspector مشاهده کنید:

نام پیش فرض File Chooser در واقع jFileChooser1می باشد. روی jFileChooser1در پنجره ی Inspectorراست کلیک کنید. از منوی ظاهر شده گزینه ی Change Variable Name را انتخاب کنید. وقتی که دیالوگ باکس ظاهر می شود، dbرا به عنوان نام تایپ کنید:



روی OK کلیک کنید تا تغییر را تایید کنید. پنجره ی Inspector باید مانند تصویر زیر باشد:

اکنون یک File Chooser دارید که به پروژه اضافه شده است.

نمایش دیالوگ باکس File Chooser بسیار ساده می باشد. برای آیتم Open در منو به code stub بازگردید، جایی که message box را داشتید. اکنون خط زیر را تایپ کنید:

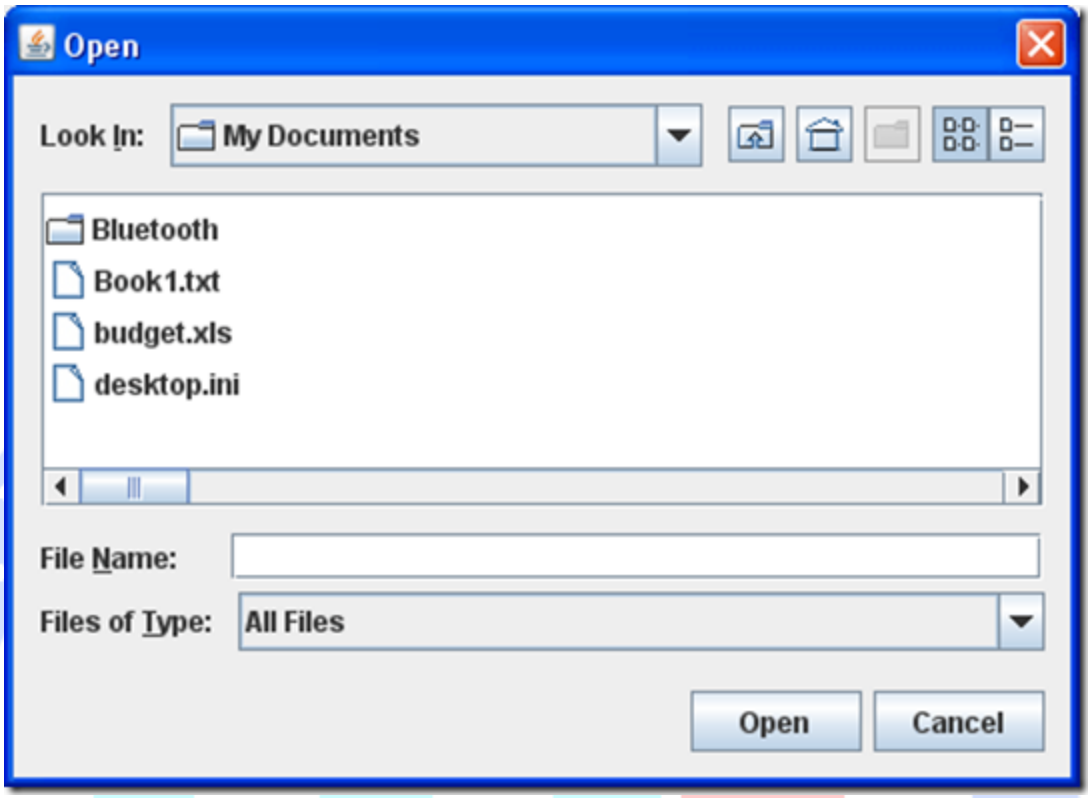
مثال

```
int returnVal = db.showOpenDialog( this );
```

File Chooser ما db نامیده می شود. ما در حال استفاده از متود ShowOpenDialog از گروه File Chooser هستیم. در بین پرانتزهای ShowOpenDialog نام پنجره ای را تایپ کنید که قرار است دیالوگ باکس را در خود داشته باشد. ما "this form" را تایپ کرده ایم.

متد ShowOpenDialog یک مقدار را گزارش می دهد. این مقدار یک عدد صحیح می باشد. این مقدار به شما می گوید که کدام دکمه در دیالوگ باکس کلیک شده است open ، cancel و غیره. ما در حال ذخیره سازی مقدار در متغیری به نام returnVal هستیم و از این مقدار در یک لحظه استفاده خواهیم کرد.

اما برنامه ی خود را مجددا اجرا کنید. روی File > Open در فرم خود کلیک کنید. یک دیالوگ باکس برای شما ظاهر خواهد شد:



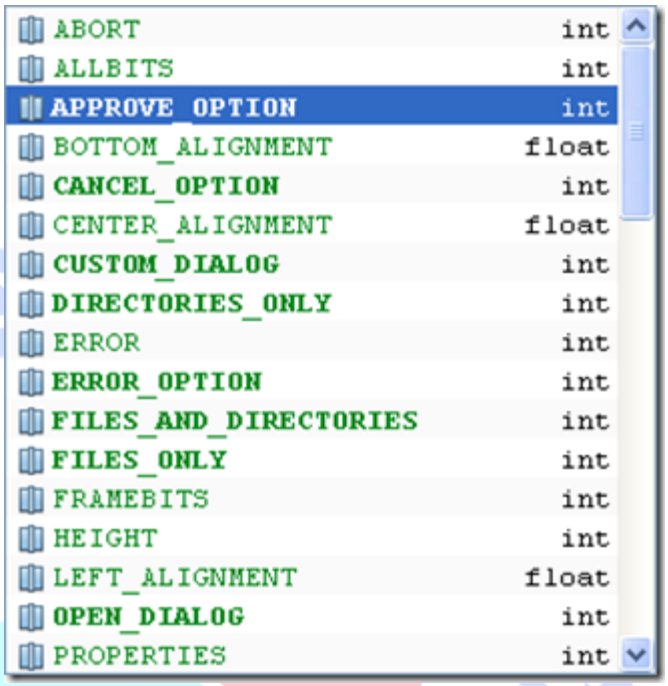
متاسفانه تنها کاری که یک باکس Open Dialogue انجام می دهد انتخاب یک فایل می باشد – در واقع این باکس چیزی را باز نمی کند. اگر می خواهید که یک فایل را باز کنید، باید که کد را خودتان بنویسید. ما این کار را به زودی انجام خواهیم داد. اما اکنون می توانیم نام و مسیر فایلی را که یک یوزر انتخاب کرده، دریافت کنیم.

ابتدا عبارت IF Statement زیر را درست زیر دو خط دیگر به کد خود اضافه کنید:

مثال

```
if (returnVal = = javax.swing.JFileChooser.APPROVE_OPTION) {
}
```

بنابراین ما در حال استفاده از Swing class به نام JFileChooser می باشیم. با این گروه می توانید بررسی کنید که کدام دکمه کلیک شده است. وقتی که یک نقطه (dot) را پس از JFileChooser تایپ می کنید، لیستی برای شما ظاهر خواهد شد:



APPROVE_OPTIONبه معنای گزینه هایی مانند دکمه های Yes یا OK میباشد .بنابراین ما در حال امتحان کردن متغیر returnVal هستیم تا هماهنگی آن را با APPROVE_OPTIONبررسی کنیم) .آیا یوزر روی OK کلیک کزد؟.(

برای رسیدن به فایلی که توسط یوزر انتخاب شده است، متودی به نام getSelectedFileوجود دارد. به هرحال این برنامه به جای یک رشته، یک آبجکت فایل بازمی گرداند. آبجکت File بخشی از گروه IO در جاوا می باشد. بنابراین خط زیر را به IF Statement اضافه کنید:

مثال

```
java.io.File file = db.getSelectedFile( );
```

بنابراین فایل انتخاب شده توسط یوزر در آبجکت فایل که file نامیده ایم، خاتمه پیدا خواهد کرد .

برای انجام کار مفید با این برنامه (برای متال باز کردن فایل)، نیاز به تبدیل آن به یک رشته می باشد:

مثال

```
String file_name = file.toString( );
```

این خط فقط از متود toString از آبجکت های File استفاده می کند. ما نتیجه را در یک متغیر جدید به نام file_nameقرار می دهیم. خط را به IF Statement خود اضافه کنید.

برای نمایش نام فایل، کامنت ها را از باکس پیغام خود حذف کنید و آن را به عنوان آخرین خط از IF Statementجابجا کنید. آخرین پارامتر بین پرانتزها را به file_name تغییر دهید:

مثال

```
javax.swing.JOptionPane.showMessageDialog(FormObjects.this, file_name);
```

از آنجایی که این خطوطانی است می توانید، می توانید یک عبارت import به بالای کد خود و زیر کدی که دارید، اضافه کنید:

مثال

```
import javax.swing.JOptionPane;
```

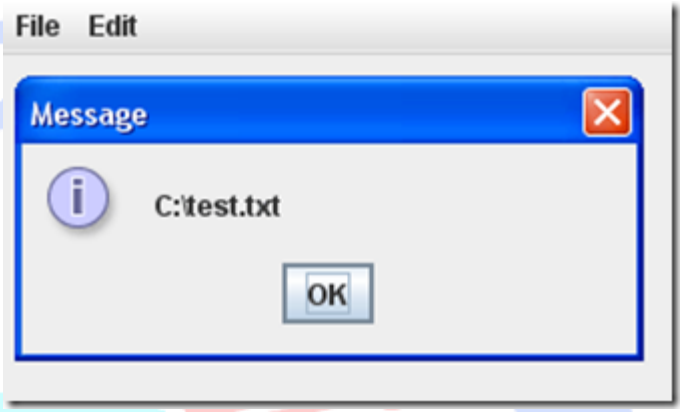
بنابراین خط مربوط به باکس پیغام می تواند فقط به شکل زیر باشد:

```
JOptionPane.showMessageDialog( FormObjects.this, file_name);
```

اکنون خواندن آن کمی آسانتر است !اما اکنون کد شما باید به شکل زیر باشد :

```
int returnVal = db.showOpenDialog(this);

if (returnVal == javax.swing.JFileChooser.APPROVE_OPTION) {
 java.io.File file = db.getSelectedFile();
 String file_name = file.toString();
 JOptionPane.showMessageDialog(FormObjects.this, file_name);
}
```

برنامه ی خود را اجرا کرده و آن را امتحان کنید. برای مشاهده ی دیالوگ باکس روی File > Open کلیک کنید. هر فایلی در کامپیوتر خود را انتخاب کرده و سپس روی Open کلیک کنید. باکس پیغام شما مانند تصویر زیر خواهد بود:



قبل از اینکه برای باز شدن فایل انتخاب شده، کد اضافه کنیم، ممکن است متوجه شده باشید که Files of Type روی دیالوگ باکس فایل Open بر روی "All files" تنظیم شده است. می توانید فایل های روی این لیست را فیلتر کنید، طوریکه یوزر می تواند فقط فایل های متن یا تصاویری با فرمت معین را باز کند (jpeg, gif, png).

برای فیلتر کردن لیست"Files of Type" ، دیالوگ باکس دارای addChoosableFileFilterمتود می باشد. اما شما نیر بین پرانتزها نیاز به آبجکت Filter دارید.

عبارت های import زیر را به بالای کد خود و درست زیر عبارت های دیگر، طوریکه کد شما طولانی نشود، وارد کنید:

مثال

```
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;
```

برای تنظیم پسوند فیلتر نام فایل، نیاز به ایجاد یک آبجکت FileFilter جدید دارید. خط زیر را درست قبل از اولین خط از کد خود (قبل از خط int returnVal)وارد کنید:

مثال

FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");

بین پرانتزهایFileNameExtensionFilter ، ابتدا نیاز به متنی دارید که در لیست Files of Type ظاهر خواهد شد. پس از یک کاما نام فایل هایی را تایپ می کنید که می خواهید نمایش داده شوند. در اینجا یک پسوند فایل مناسب اما بدون کاما نیاز است. به علامت های نقل قول دوتایی (") در بالا دقت کنید.

شما می توانید بیشتر از یک پسوند اضافه کنید. فقط کافیست یک کاما تایپ و سپس فایل هایی را که قرار است نمایش داده شوند، تایپ کنید:

FileNameExtensionFilter("Text Files", "txt", "html");

زمانی که آبجکت فیلتر را تنظیم کرده اید، می توانید از متود addChoosableFileFilter در دیالوگ باکس خود استفاده کنید:

```
db.addChoosableFileFilter( ft );
```

این خط را در کد خود درست زیر خط FileFilter اضافه کنید:

```
FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");
db.addChoosableFileFilter(ft);
int returnVal = db.showOpenDialog(this);

if (returnVal == javax.swing.JFileChooser.APPROVE_OPTION) {
  java.io.File file = db.getSelectedFile();
  String file_name = file.toString();
  JOptionPane.showMessageDialog(FormObjects.this, file_name);
}
```

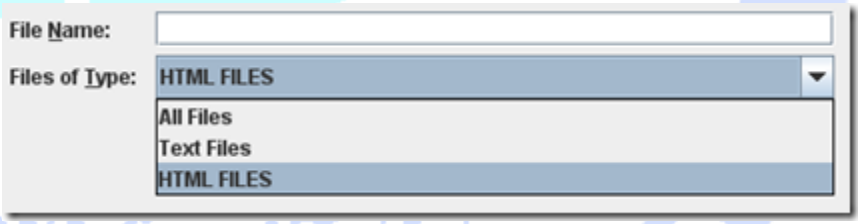
مجددا برنامه ی خود را اجرا کرده و نگاهی به دیالوگ باکس خود داشته باشید. روی پیکان در لیست رو به پایین کلیک کنید:

گزینه ی Text Files را انتخاب کنید. سپس دیالوگ باکس شما فقط فایل هایی را با پسوند txt. نمایش خواهد داد.

اگر شما خط دیگری روی لیست می خواهید) برای مثال نمایش فایل های(html ، می توانید یک آبجکت FileFilter دیگر تنظیم کنید:

```
FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");
FileFilter ft2 = new FileNameExtensionFilter("HTML FILES", "html");

db.addChoosableFileFilter(ft);
db.addChoosableFileFilter(ft2);
```

وقتی برنامه ی شما اجرا می شود، لیست Files of Type مانند زیر خواهد بود:



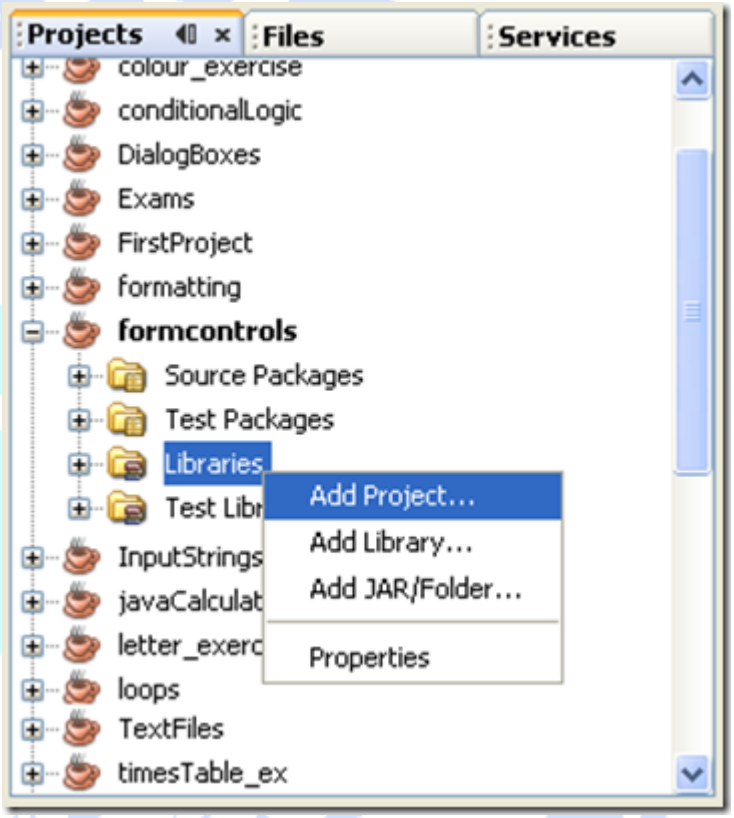
اکنون می توانیم کدی بنویسیم تا واقعا فایل را باز کنیم که این کار را در بخش بعد انجام خواهیم داد .

آموزش استفاده از دیالوگ باکس در جاوا

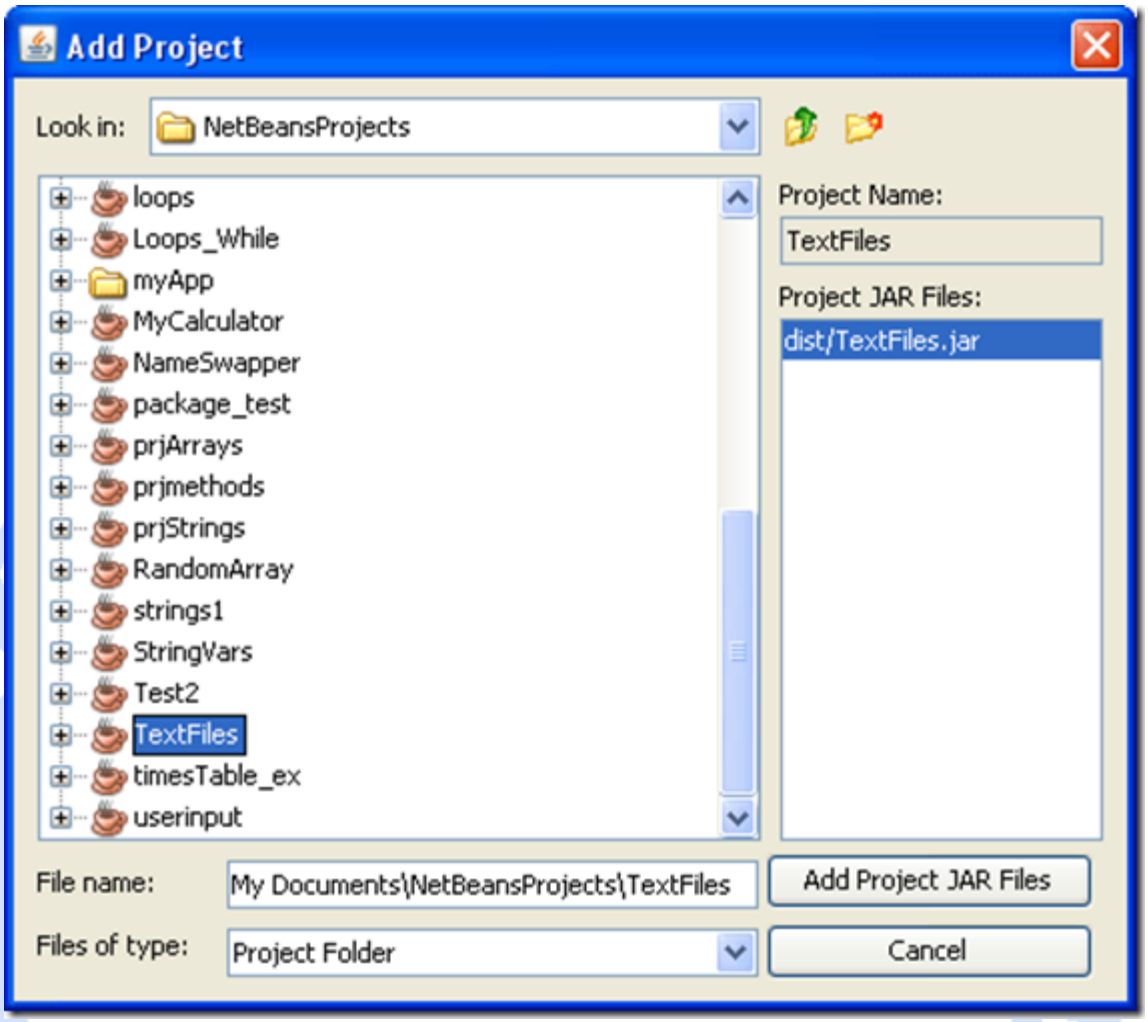
در واقع شما کدی را برای باز کردن یک فایل نوشته اید. این فرایند برای دیالوگ باکس open file تفاوتی نمی کند. و چون شما در حال حاضر گروه هایی را نوشته اید که یک فایل را باز و در آن نگارش می کنند، می توانید آنها را فقط وارد پروژه ی حاضر کنید.

برای وارد کردن فایلی که تقریبا آن را نوشته اید، نگاهی به پنجره ی properties در سمت چپ NetBeans داشته باشید. (اگر نمی توانید این پنجره را ببینید، روی آیتم منوی Window در بالای NetBeans کلیک کنید. از منوی Windowگزینه ی Properties را انتخاب کنید(.

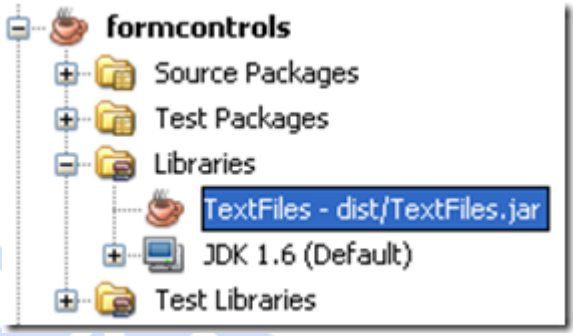
ورودی را برای پروژه ی فعلی خود باز کنید و روی آیتم libraries راست کلیک کنید:



از منوی ظاهر شده، Add Projectرا انتخاب کنید. وقتی این کار را انجام می دهید دیالوگ باکس زیر را مشاهده خواهید کرد:



مطمئن شوید که Look در باکس بالا بیانگر "NetBeans Projects" می باشد .اکنون به پایین رفته و پروژه ی TextFiles خود را جایگذاری کنید. اینجا جایی است که گروه های شما که باز هستند و در یک فایل نگارش می کنند. اکنون روی دکمه ی "Add project JAR files" کلیک کنید. دیالوگ باکس بسته شده و شما به NetBeansباز خواهید گشت. نگاهی به ورودی Libraries برای پروژه ی فعلی خود داشته باشید، مشاهده خواهید کرد که این پروژه وارد شده است:



اکنون که گروه ها را به پروژه ی خود اضافه کرده اید، می توانید یک خط import به بالای کد خود اضافه کنید:

مثال

```
import textfiles.ReadFile;
```

اگر به خاطر داشته باشید، ReadFile گروهی بود که برای باز کردن یک فایل فرا خواندیم. نام پوشه ای که با آن روبرو شدیم textfiles بود. اکنون گروه ReadFile را وارد می کنیم و می توانیم از آن آبجکت های جدیدی ایجاد کنیم.

مثال

```
ReadFile file_read = new ReadFile( file_name );
```

IF Statement خود را با افزودن کد catch... try زیر هماهنگ سازید:

```
if (returnVal == javax.swing.JFileChooser.APPROVE_OPTION) {

    java.io.File file = db.getSelectedFile();
    String file_name = file.toString();

    try {
        ReadFile file_read = new ReadFile(file_name);
        String[] aryLines = file_read.OpenFile();
        int i;
        String theText = "";
        for (i=0; i < aryLines.length; i++) {
            theText = theText + aryLines[i] + '\n';
        }

        taOne.setText(theText);
    }
    catch (java.io.IOException e) {

    }
}
```

کد درست همانی است که قبلا برای باز کردن یک فایل متن استفاده کردیم :نام فایل را دریافت کرده و یک آبجکت ReadFile جدید ایجاد کنید، متود openFileرا که ایجاد کرده ایم، فرا بخوانید و تمام خطوط را به عنوان یک ردیف (array) بازگردانده و سپس خواندن هر خط را مرور کنید. دقت کنید که همه ی خطوط را در بخش کنترل متن در فرم خود قرار دادیم.

به هرحال موردی که نادیده گرفته شده، پیغامی در بخش catch از گروه try ... catch می باشد. باکس پیغام خود را بین پرانتزهای catch حرکت دهید. بین پرانتزهای showMessageDialog یک پیغام مناسب اضافه کنید. یا می توانید از e.getMessageاستفاده کنید.

وقتی که کد را کامل کردید، آن را امتحان کنید. باید با استفاده از منوی File > Openیک فایل باز کنید. سپس فایل متن باید در فیلد متن ظاهر شود.

آموزش ذخیره فایل با استفاده از دیالوگ باکس در جاوا

اگر بخواهید دیالوگ باکس Save File را به جای Open File نمایش دهید، می توانید مجددا از File Chooserاستفاده کنید. این بار به جای showOpenDialogاز showSaveDialog استفاده کنید:

مثال

```
int returnVal = db.showSaveDialog( this );
```

مجددا این برای نمایش دیالوگ باکس کافی می باشد. لازم نیست کار دیگری انجام دهید، گرچه می توانید یک file filterنیز اضافه کنید.

مثال

```
FileFilter ft = new FileNameExtensionFilter( "Text Files", "txt" );
db.addChoosableFileFilter( ft );
int returnVal = db.showSaveDialog(this);
```

برای نوشتن یک فایل، یک عبارت import به بالای کد خود اضافه کنید:

```
import textfiles.WriteFile;
```

سپس می توانید یک آبجکت WriteFile ایجاد کنید. مجددا می توانید کد را همراه با یک بلوکtry ... catch ، در داخل IF Statement قرار دهید:

```
FileFilter ft = new FileNameExtensionFilter("Text Files", "txt");
db.addChoosableFileFilter(ft);
int returnVal = db.showSaveDialog(this);

if (returnVal == javax.swing.JFileChooser.APPROVE_OPTION) {

        java.io.File saved_file = db.getSelectedFile();
        String file_name = saved_file.toString();
    try {
        WriteFile data = new WriteFile(file_name, false);
        String alltext = taOne.getText();
        data.writeToFile(alltext);
    }
    catch (java.io.IOException e) {
        //YOUR_MESSAGE_BOX_HERE
    }
}
```

دو خط آخر در کد بالا داده را از بخش متن دریافت می کند و سپس در یک متغیر رشته به نام alltext قرار می گیرد. پس از آن می توانیم متود writeToFileرا از گروه WriteFile خود فرا بخوانیم.

یک code stub برای آیتم File > Save در منو ایجاد کنید. کد بالا را به آن اضافه کنید. برنامه ی خود را اجرا کرده و آن را امتحان کنید. در بخش textمتن جدیدی تایپ کنید. سپس در منوی خود روی File > Save کلیک کنید .وقتی که دیالوگ باکس "Save File" ظاهر می شود، نام فایل را تایپ کرده و سپس روی Save کلیک کنید. ایجاد شدن فایل خود و اینکه متنی را که تایپ کردهاید در بخش text وجود دارد، را بررسی کنید.

این مقدمه ای بود برای فرم ها و آبجکت های فرم. ما ادامه خواهیم داد. در بخش نگاهی به دیتابیس ها خواهیم کرد .

آموزش دیتابیس در جاوا

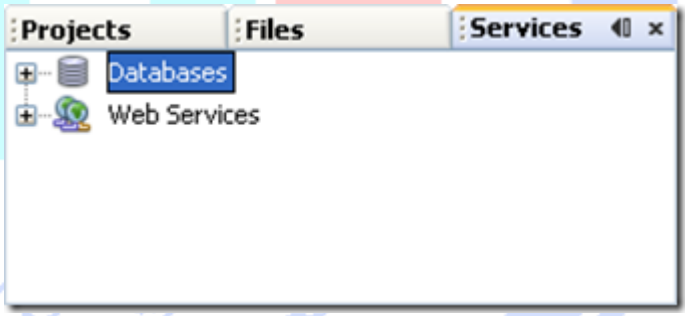
در این بخش در مورد دیتابیس های جاوا فرا خواهید گرفت. شما یک دیتابیس ساده با یک جدول ایجاد خواهید کرد و یاد خواهید گرفت که چگونه با استفاده از کد جاوا به آن متصل شوید.

در مورد جاوا و دیتابیس ها

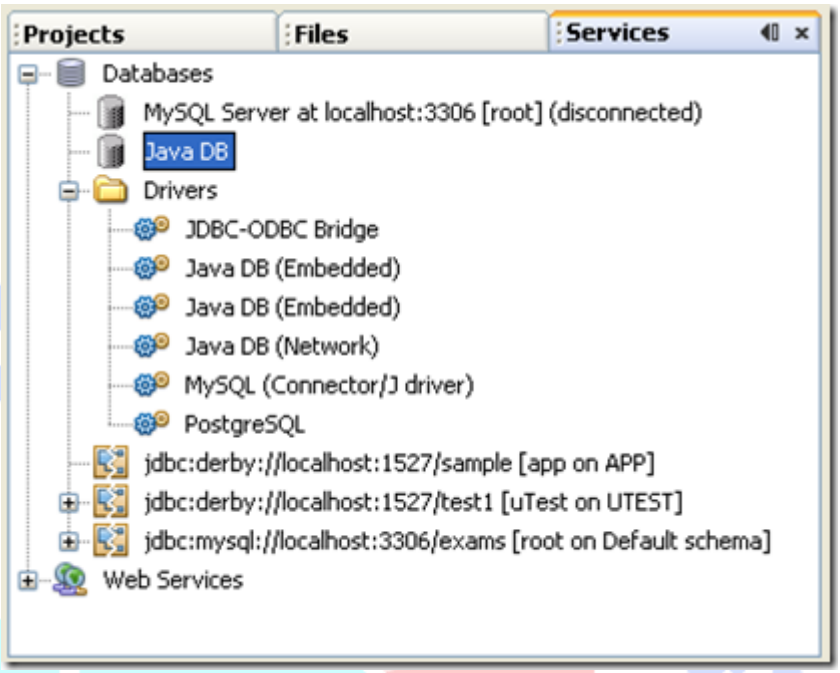
جاوا از چیزی به نام (Java Database Connectivity) JDBC برای اتصال به دیتابیس ها استفاده می کند. یک JDBC API وجود دارد که بخش برنامه نویسی است و JDBC Driver Managerکه برنامه های شما برای اتصال به دیتابیس استفاده می کنند.

JDBCبه شما اجازی اتصال به یک دامنه ی وسیع از دیتابیس ها را می دهد(Oracle, MySQLو غیره)، اما ما قصد استفاده از یک دیتابیس داخلی استفاده کنیم که از نرم افزار Java/NetBeans دریافت می کنیم. این دیتابیس Java DB نامیده می شود، ورژنی از .Apache Derby این برنامه روی یک سرور فرضی اجرا می شود که می توانید متوقف کرده و از داخل NetBeans آغاز کنید.

برای بررسی اینکه تمام موارد مورد نیاز را دارید، تب Services را در NetBeansبررسی کنید. اگر نمی توانید تب Services را مشاهده کنید، روی Windowاز منوی NetBeans کلیک کنید. از منوی Window گزینه ی Servicesرا انتخاب کنید. پس از آن صفحه ای مانند تصویر زیر مشاهده خواهید کرد:



آیتم Database را باز کنید تا آیتم Java DB و بخش Drivers را مشاهده کنید:



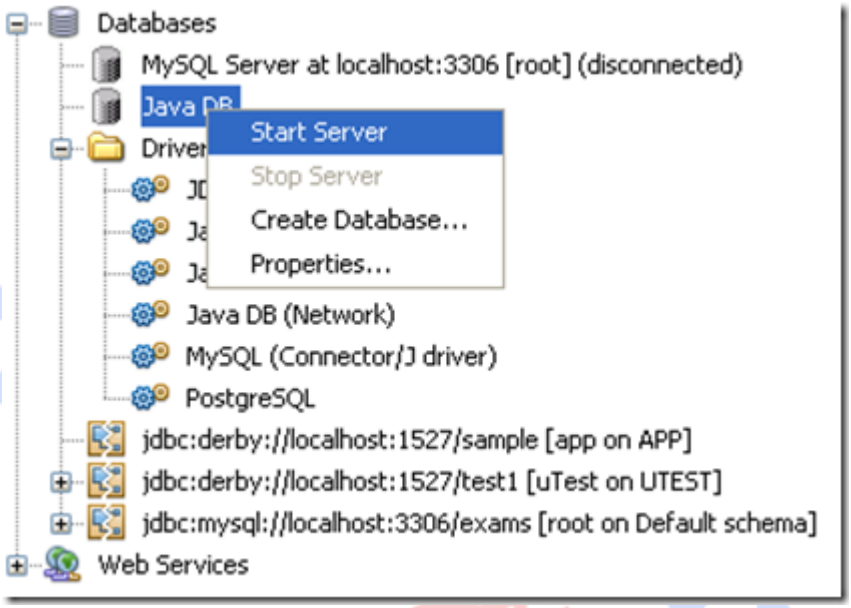
نظریه این است که سرور فرضی Java DB را آغاز کرد اید و سپس دیتابیس ها را روی سرور ایجاد و اجرا کرده اید. باید دیتابیس تنظیم شده ای به نام sample وجود داشته باشد: اما اگر چنین دیتابیسی وجود نداشت، نگران نباشید، چرا که ما دیتابیس خود را ایجاد خواهیم کرد.

در تصویر بالا سه دیتابیس وجود دارد: یکیSample ، یکی test1 و دیگری.exams

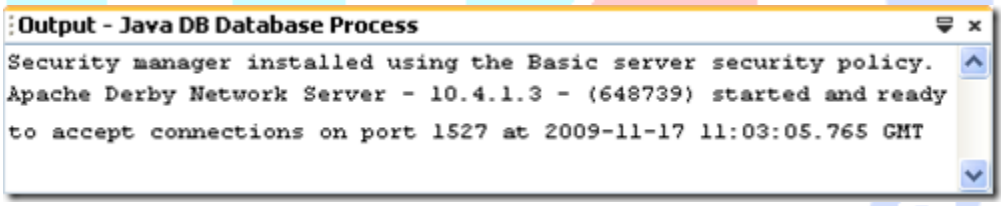
برای پروژه ی مربوط به این بخش قصد داریم که یک دیتابیس جدید تنظیم کنیم. سپس چگونگی اتصال به این دیتابیس را با استفاده از کد جاوا فرا خواهید گرفت. دیتابیسی که ما ایجاد خواهیم کرد، یک دیتابیس ساده دارای یک جدول خواهد بود و نه دارای چند جدول متصل به هم. در واقع می توانید با استفاده از Java DB چند جدول ایجاد کنید، اما لزوما نمی خواهیم مسائل را پیچیده کنیم.

آغاز Virtual Server (مرورگر فرضی)

اولین کاری که باید انجام دهیم، آغاز مرورگر می باشد. بنابراین روی Java DB راست کلیک کنید. پس از آن یک منو برای شما ظاهر خواد شد. گزینه ی Start Server را انتخاب کنید:



نگاهی به پنجره ی Output داشته باشید، چند پیغام را مشاهده خواهید کرد) :اگر یک firewall در حال اجرا داشته باشید، لازم است به Java DB اجازه دهید(.

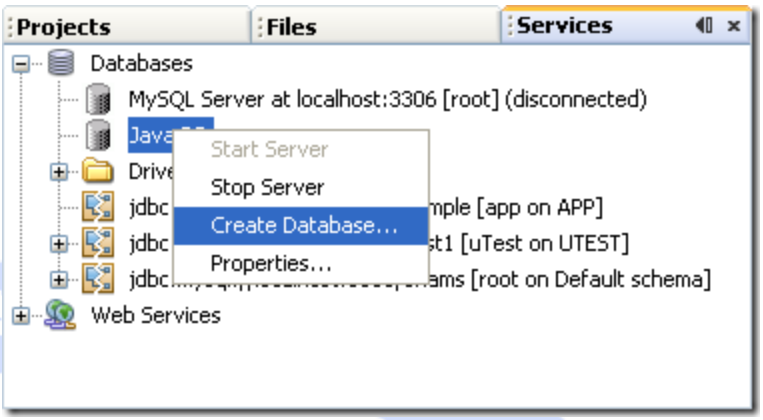


رمانی که سرور در حال اجرا می باشد، می توانید دیتابیس هایی ایجاد کنید. در بخش بعد چگونگی انجام این کار را مشاهده خواهید کرد .

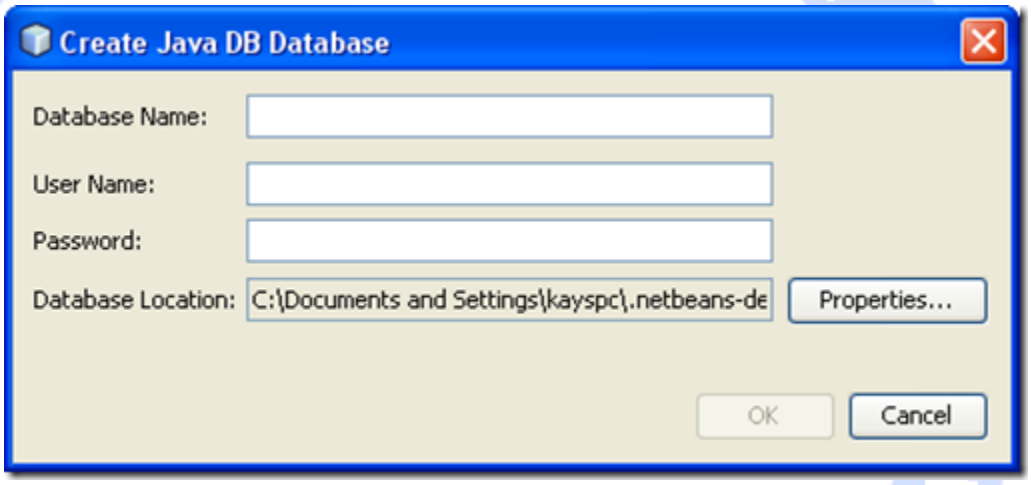
آموزش ایجاد دیتابیس با جاوا

اکنون که سرور شما آغاز به کار کرده است، می توانید جلوتر روید و یک دیتابیس ایجاد کنید.

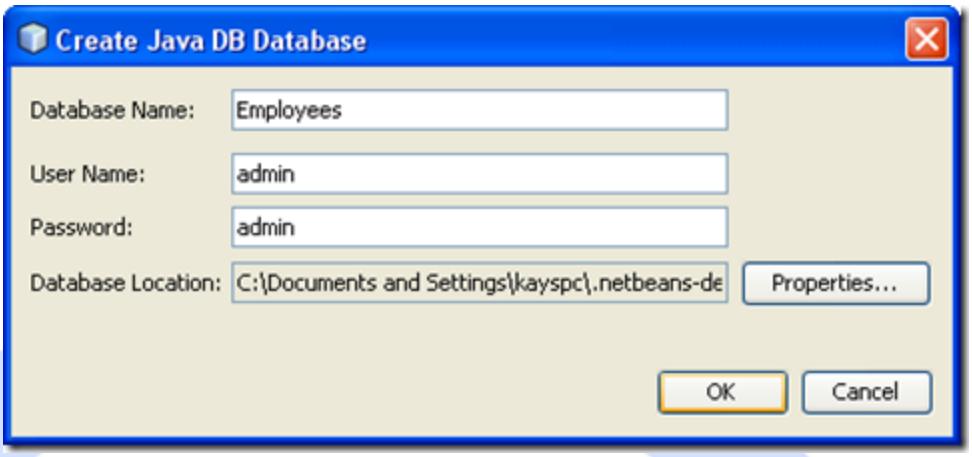
برای ایجاد یک دیتابیس جدید مجددا روی Java DB کلیک راست کنید. از منوی ظاهر شده Create Databaseرا انتخاب کنید:



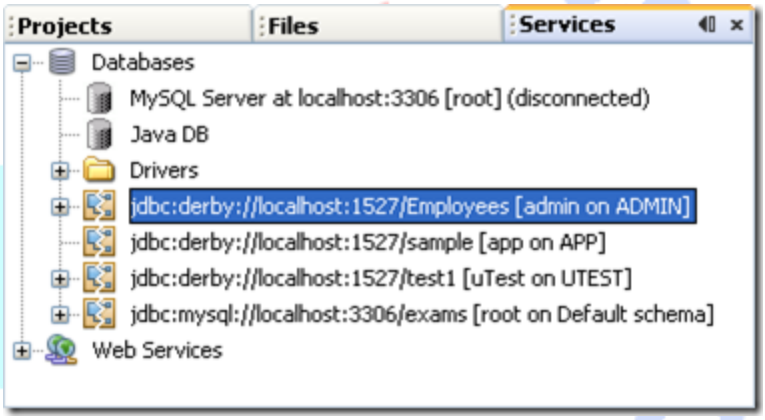
وقتی که روی Create Database کلیک می کنید، دیالوگ باکسی برای شما ظاهر خواهد شد:



در اولین باکس یک نام برای دیتابیس خود تایپ کنید. آن را Employees بنامید. هر یوزرنیم و پسووردی که می خواهید می توانید تایپ کنید. (پسووردی که ک هک کردن آن سخت باشد، نه مانند آنچه ما استفاده کرده ایم).

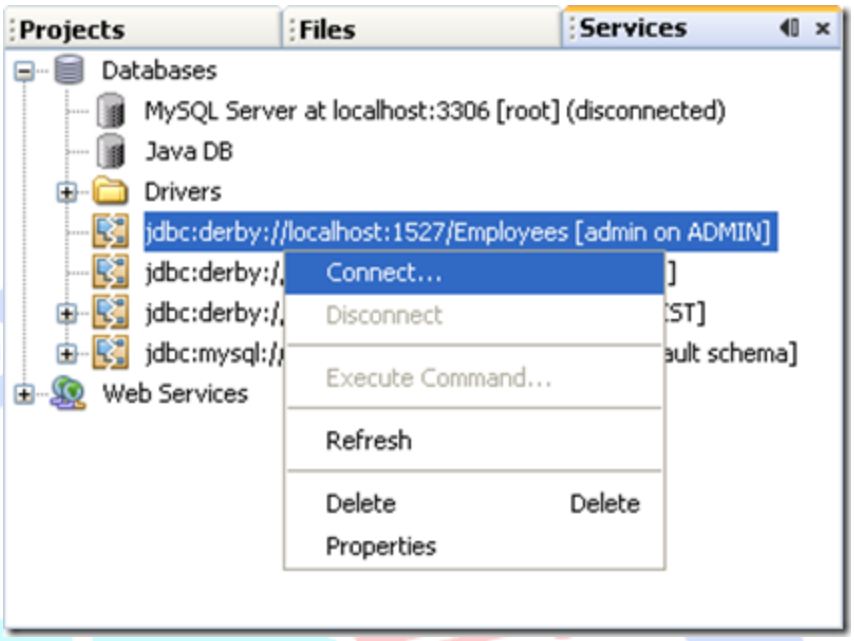


برای ایجاد دیتابیس خود روی OK کلیک کنید. این دیتابیس باید روی لیست ظاهر شود:

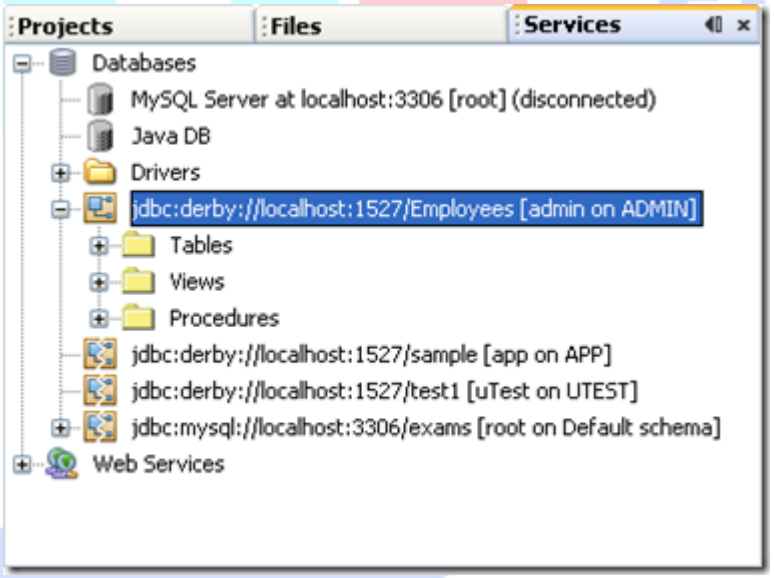


ایجاد یک جدول در دیتابیس

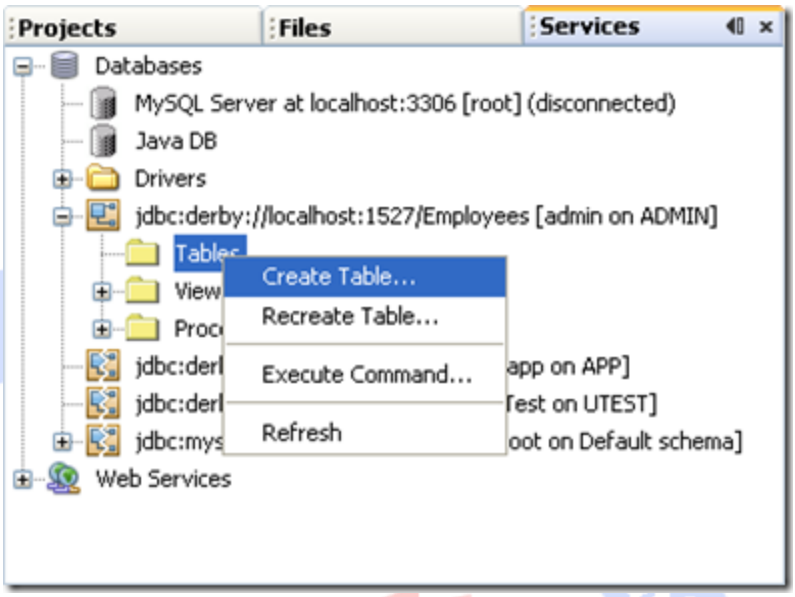
اکنون که دیتابیس ایجاد شده است، نیاز به ایجاد یک جدول در دیتابیس دارید. برای انجام این کار روی دیتابیس خود راست کلیک کنید. از منوی ظاهر شده گزینه ی Connect را انتخاب کنید:



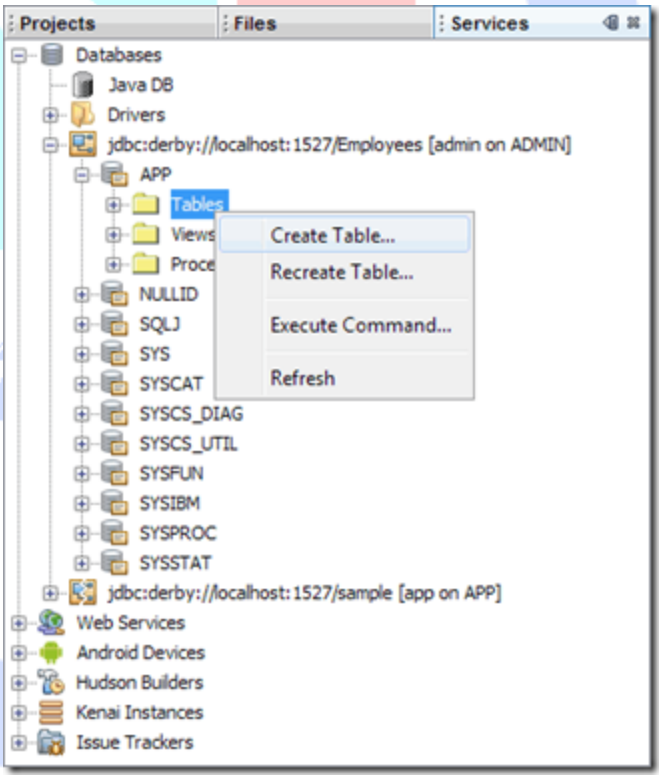
وقتی که یک اتصال برقرار می شود، چند فولدر پیش فرض برای Tables, Views و Proceduresمشاهده خواهید کرد (اگر صفحه ی شما شبیه به این نیست، پایین تر را مشاهده کنید:).



برای ایجاد یک جد.ل جدید در دیتابیس خود، روی فولدر Tablesراست کلیک کنید. از منوی ظاهر شده گزینه ی Create Table را انتخاب کنید:

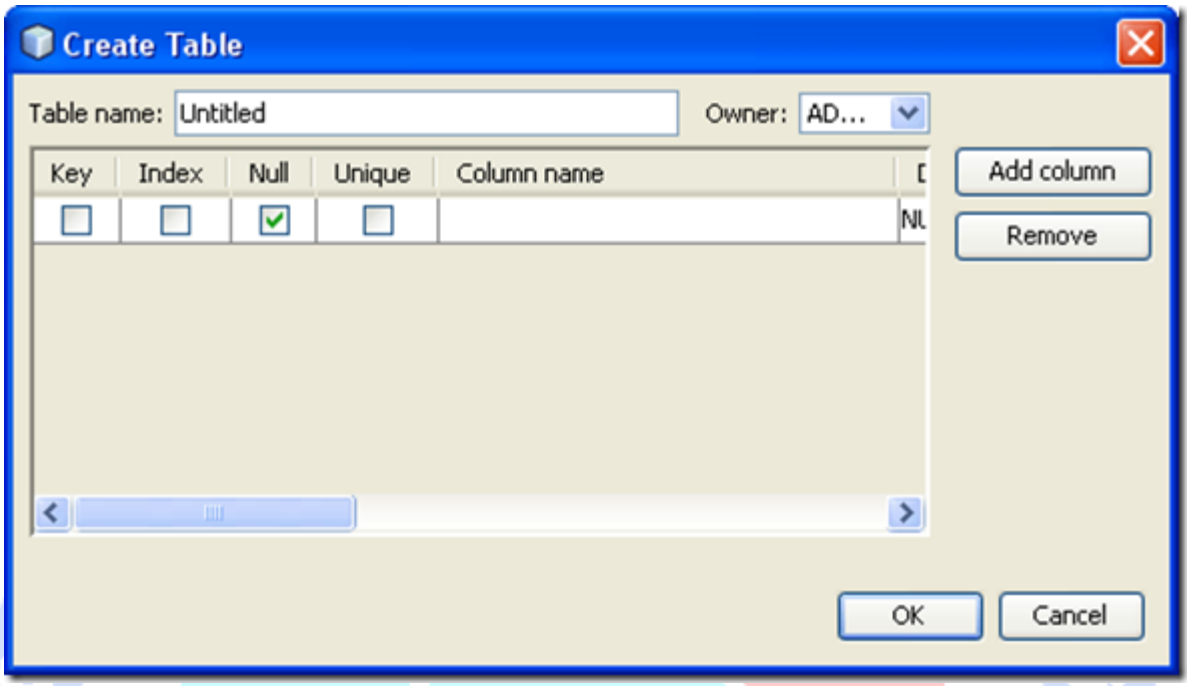


اگر سه فولدر بالا را مشاهده نکردید، در عوض صفحه ای مشابه تصویر زیر خواهید داشت:

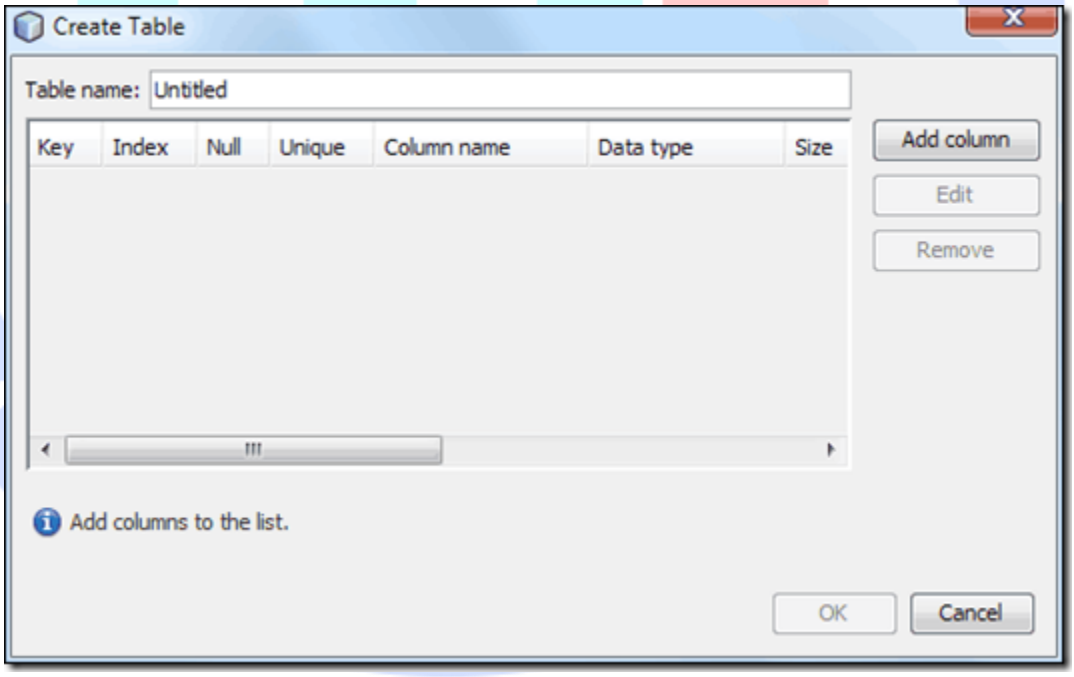


روی ورودی APP کلیک کرده و سپس روی Tables راست کلیک کنید.

وقتی روی ,Create Table کلیک کنید، یک دیالوگ باکس مشابه تصویر زیر ظاهر خواهد شد:



یا مشابه این تصویر:



از اینجا نه تنها نامی برای جدول خود تایپ خواهید کرد، بلکه ستون هایی نیز برای جاول تنظیم خواهید کرد.

در Table Name در بالا، نام پیش فرض Untitled را حذف کنید. نام جدید Workersرا برای جدول خود تایپ کنید. سپس جدولی به نام Workers خواهید داشت که در دیتابیس Employees قرار دارد.

اما از آنجایی که جدول هنوز هیچ ستونی درخود ندارد، نمی توانید روی OK کلیک کنید. ما می خواهیم که ستون هایی با نام های زیر ایجاد کنیم:

ID

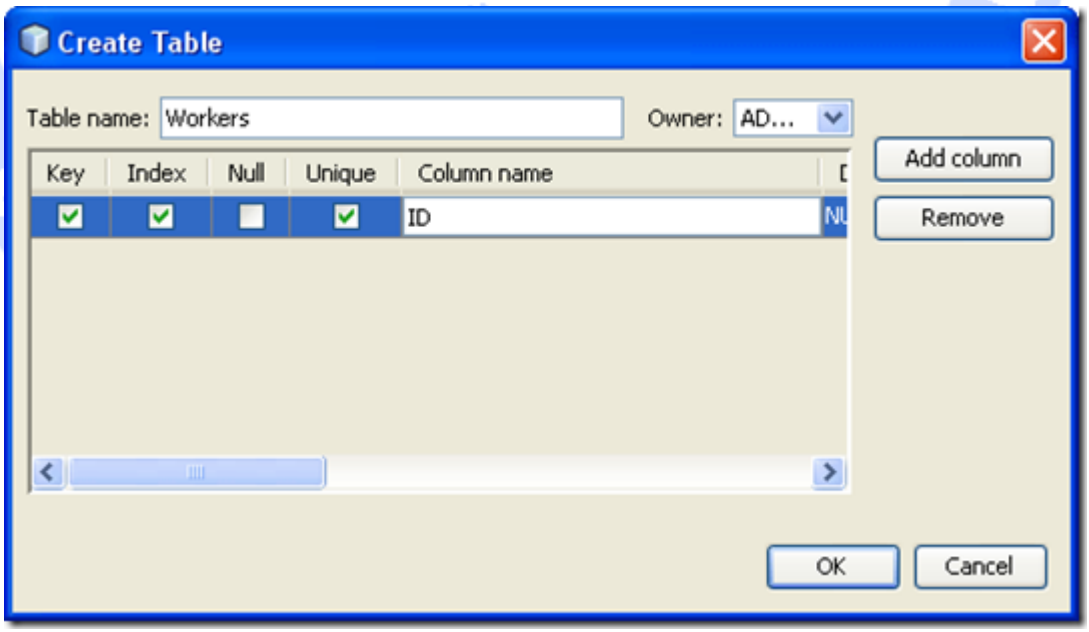
First_Name

Last_Name

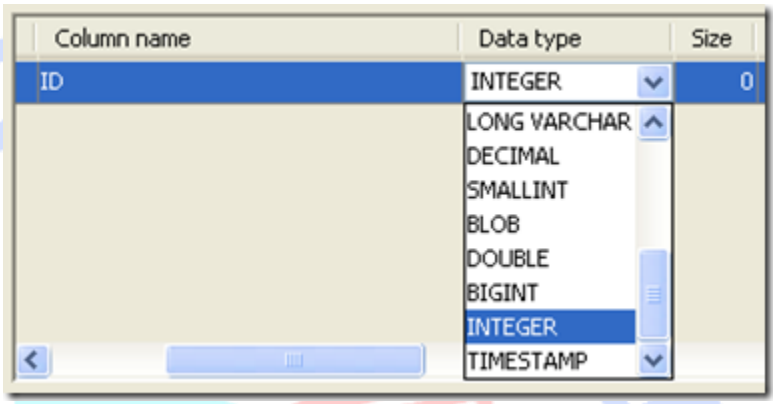
Job_Title

ستون ID یک عدد شناسایی خاص را حفظ خواهد کرد. این عدد یک ردیف را در جدول مشخص خواهد کرد. یک ستون با یک داده منحصر به فرد در این جدول با عنوان Primary Key وجود دارد. از آنجایی که این داده Primary Key (کلید اولیه) می باشد، ستون باید داده را در خود نگه دارد: ستون نمی تواند یک مقدار پوچ را در خود حفظ کند. یک مقدار پوچ یعنی هیچ اطلاعاتی در آن وجود ندارد.

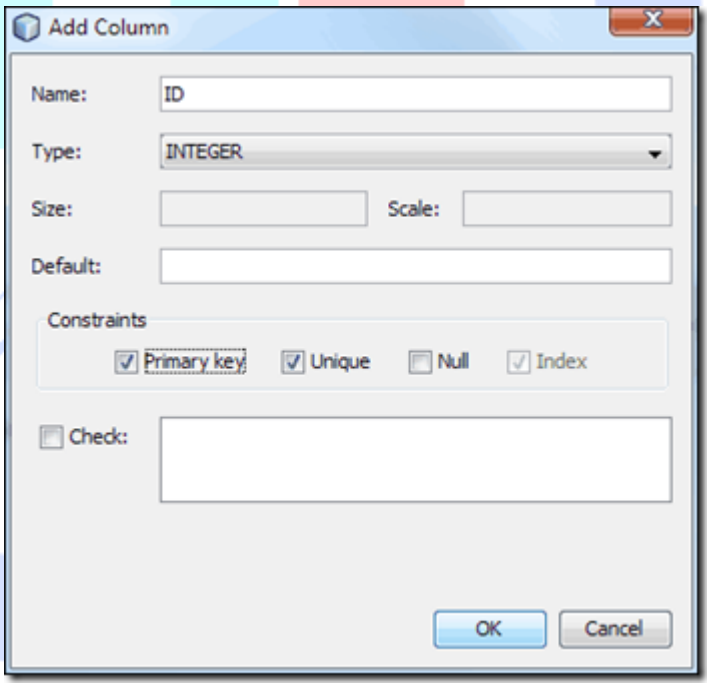
اگر دیالوگ باکس Create Table مانند مورد اول می باشد، پس در باکس مربوط به Key یک تیک قرار دهید. وقتی این تیک را قرار می دهید، چک مارک برای Indexو Uniqueنیز ظاهر خواهد شد. اکنون در بخش Column Nameیک عنوان وارد کنید ID . را تایپ کنید:



اکنون لازم است بدانید که چه نوع داده ای وارد ستون می شود. برای ستون ID، Integersرا خواهیم داشت. بنابراین اسکرول را حرکت دهید تا به Data Type برسید. روی Data Type تایپ کنید که یک لیست رو به پایین برای شما ظاهر خواهد شد. از این لیست Integers را انتخاب کنید:

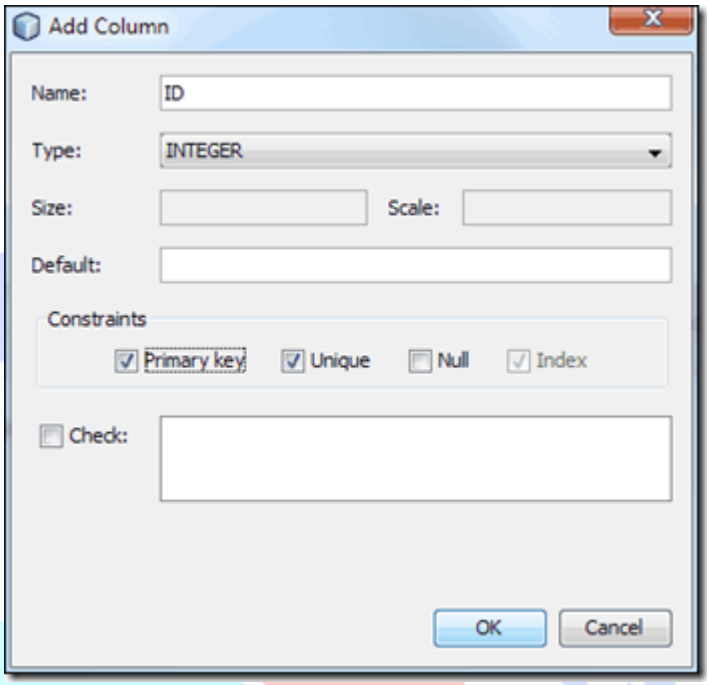


اگر دیالوگ باکس شما مشابه مورد دوم می باشد، لازم است روی دکمه ی Add Column کلیک کنید تا اولین ستون از جدول خود را اضافه کنید. پس از آن یک دیالوگ باکس دیگر ظاهر خواهد شد. مانند تصویر زیر:

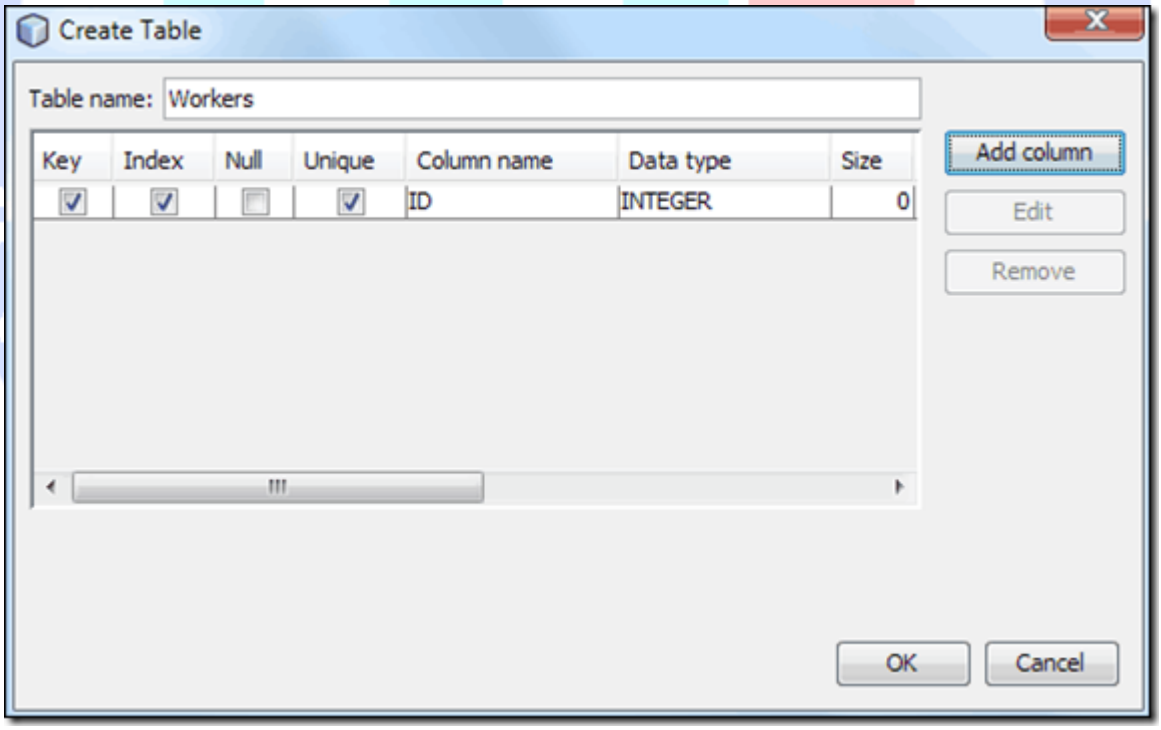


NAMEنام ستون در جدول می باشد، مانندID ، First_Nameو غیره TYPE .عبارت است از DATA TYPE, Integer, VARCHARو غیره. روی لیست رو به پایین کلیک کنید تا موارد بیشتری را مشاهده کنید.

سپس باکس های CONSTRAINTS را مانند تصویر زیر انتخاب کرده و یا از حالت انتخاب خارج کنید:



روی OK کلیک کنید که پس از آن باید به دیالوگ باکس Create Table بازگردید:



اکنون در جدول ب اندازه ی کافی فضا برای ستون ID داریم. روی دکمه ی Add Column در سمت راست کلیک کنید تا یک ستون جدید به جدول اضافه کنید. مقادیر زیر را برای این ستون وارد کنید (VARCHAR به معنای عدد متغیر کاراکترها می باشد:).

Key: Unchecked

Index: Unchecked

Null: Unchecked

Unique: Unchecked

Column Name: First_Name

Data Type: VARCHAR

Size: 20

برای سومین ستون ستون در جدول خود نیر مقادیر زیر را وارد کنید:

Key: Unchecked

Index: Unchecked

Null: Unchecked

Unique: Unchecked

Column Name: Last_Name

Data Type: VARCHAR

Size: 20

برای آخرین ستون مقادیر زیر را مشاهده می کنید:

Key: Unchecked

Index: Unchecked

Null: Unchecked

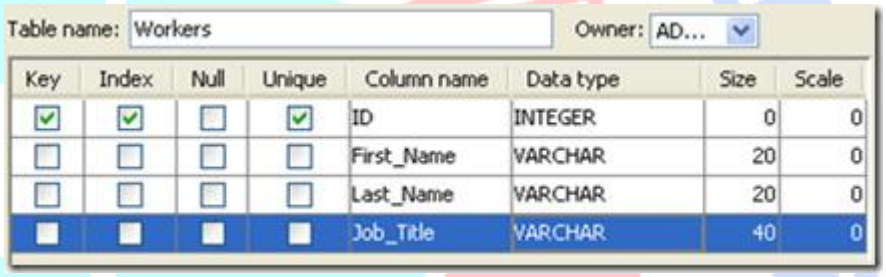
Unique: Unchecked

Column Name: Job_Title

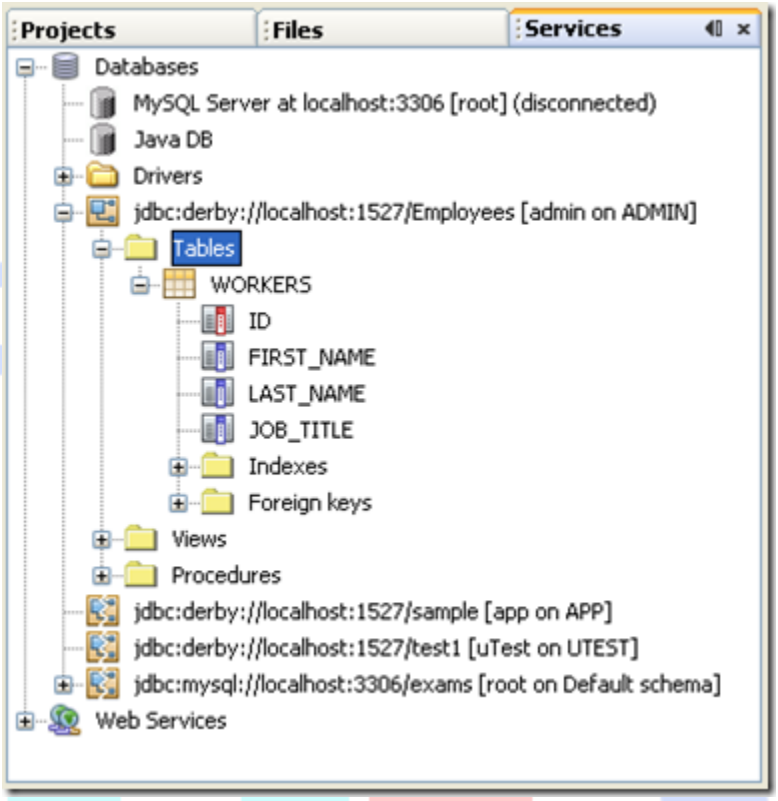
Data Type: VARCHAR

Size: 40

وقتی که کار شما به پایان رسید، دیالوگ باکس مربوط به Table باید مانند تصویر زیر باشد:



وقتی که همه ی اطلاعات را وارد کردید، روی OK کلیک کنید. سپس جدول و ستون های جدول شما ایجاد خواهند شد:

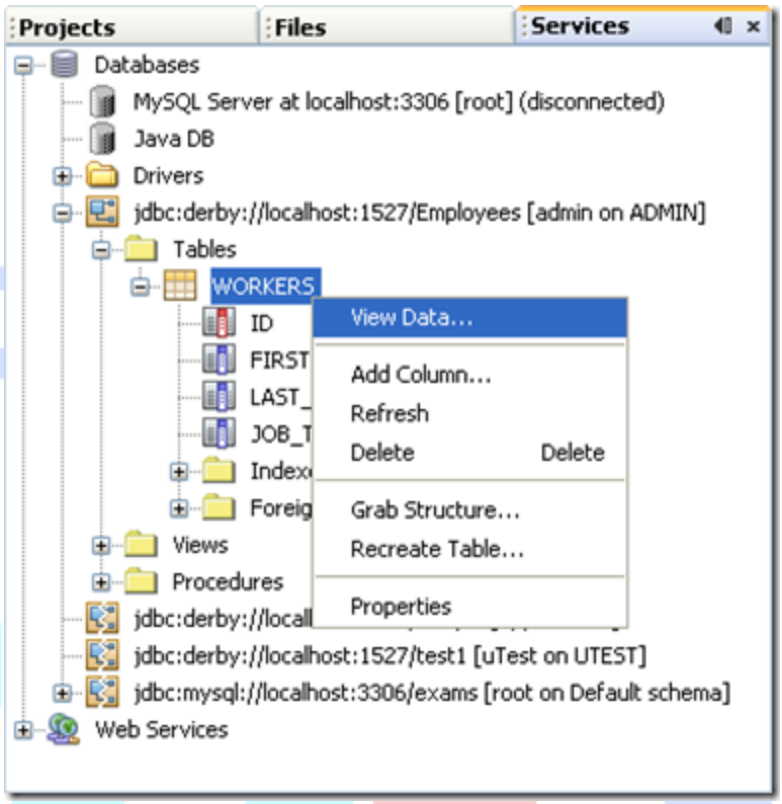


کار دیگری که باید انجام دهید، افزودن رکوردها به جدول دیتابیس می باشد که این کار را در بخش بعد انجام خواهیم داد .

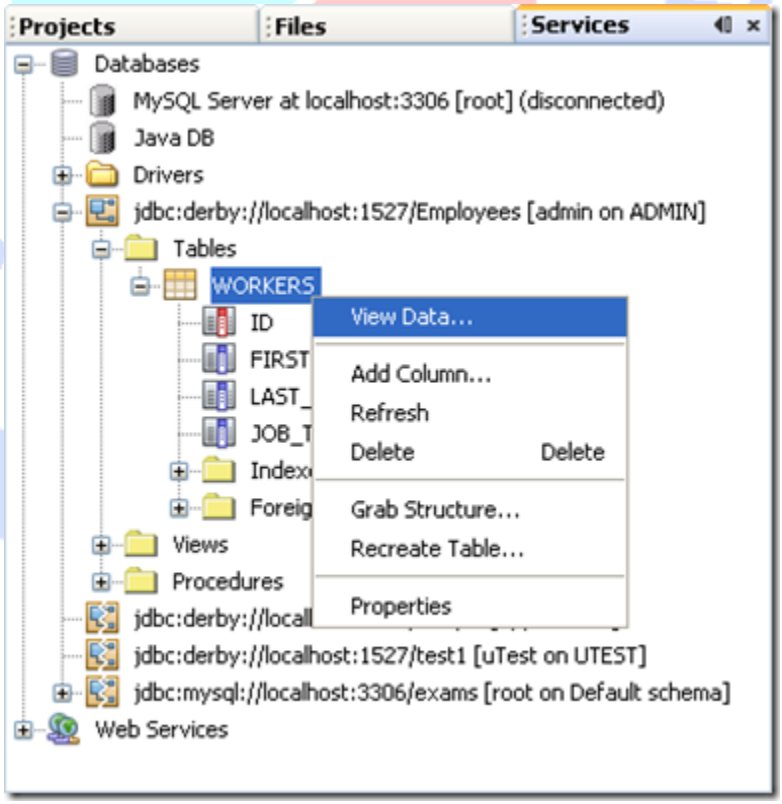
آموزش افزودن رکوردها به دیتابیس جدول جاوا

یک جدول دیتابیس مانند یک صفحه ی گسترده می باشد که دارای ستون و ردیف می باشد. هر ردیف در جدول ما دارای سلول هایی (فیلدهایی) برای یک مقدار ID، یک First Name (نام)، یک Last Name (نام خانوادگی) و یک Job Title (عنوان شغلی) می باشد. به طور مختصر چگونگی نوشتن کد مربوط به اضافه کردن ردیف های جدیدی از اطلاعات را به جدول فرا خواهید گرفت. اما می توانید از NetBeans IDE برای افزودن ردیف ها نیز استفاده کنید.

برای افزودن یک ردیف جدید به جدول خود، روی نام جدول راست کلیک کنید. از منوی ظاهر شده گزینه ی View Data را انتخاب کنید:

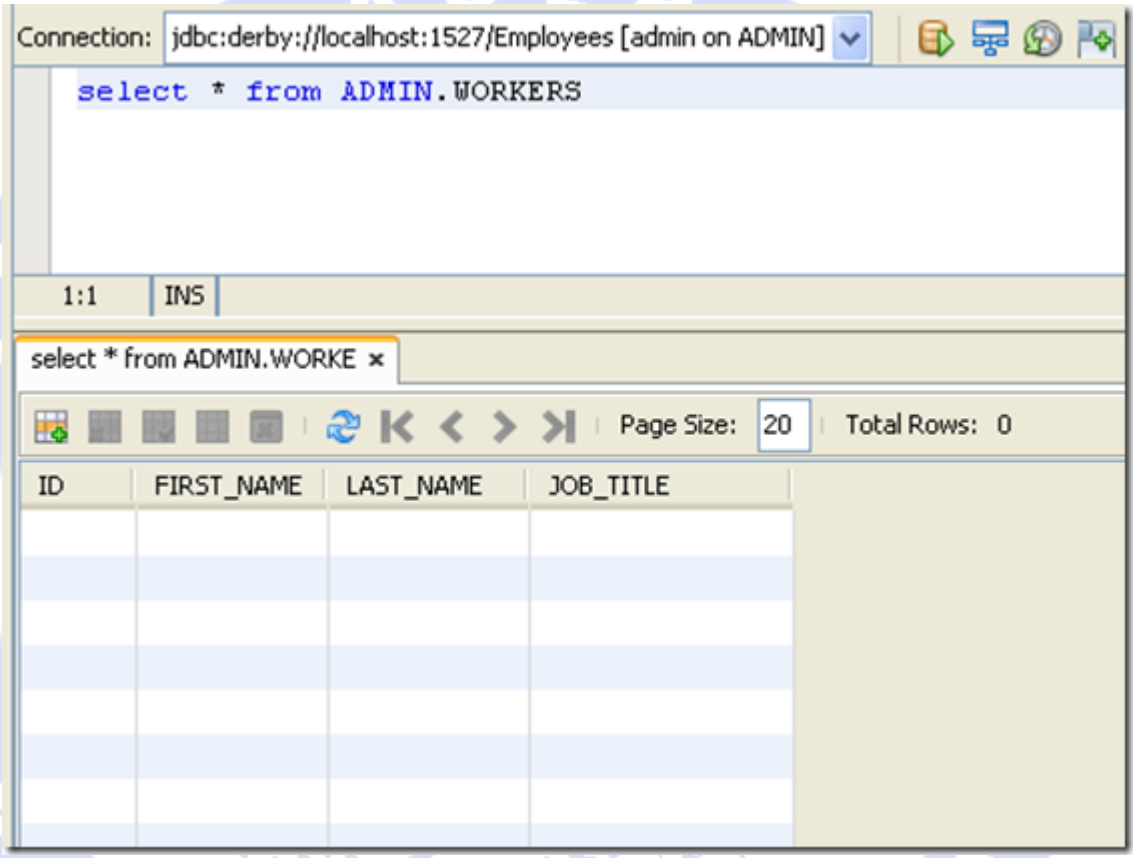


وقتی که روی View Data کلیک می کنید، یک پنجره ی جدید در NetBeans اصلی ظاهر خواهد شد:

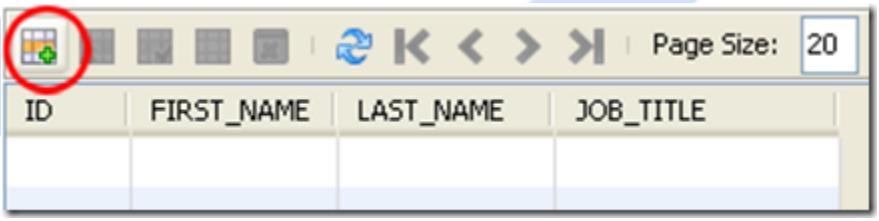


از نیمه ی پایینی پنجره برای وارد کردن ردیف های جدید جدول استفاده کنید. نیمه ی بالایی برای SQL Commands(دستورات SQL) می باشد (که به زودی پس از اتمام افزودن ردیف ها، در مورد آنها بیشتر فراخواهید گرفت.)

برای افزودن یک ردیف جدید، روی آیکنی با علامت به اضافه ی سبزرنگ در نیمه ی پایینی پنجره، کلیک کنید:



وقتی که روی آیکن ردیف جدید کلیک می کنید، یک دیالوگ باکس ظاهر می شود:



همانطور که مشاهده می کنید، باکس های متن برای هر ستون در جدول ما وجود دارد. برای ستون ID از شماره گذاری ترتیبی استفاده می کنیم که از شماره ی 1 شروع می شود. بنابراین ID ردیف دوم جدول 2 و ID ردیف

سوم 3 و غیره می باشد. اعداد شماره های ردیف ها نیستند: آنها در واقع مقادیر منحصر به فردی برای هر فیلد ID می باشند. ما می توانیم برای مقدار شماره ی ID مربوط به اولین ردیف از عدد 100 شروع کنیم. بنابراین دومین عدد 101، سومین عدد 102 و به همین ترتیب بقیه ی ردیف ها می باشند.

داده های زیر را به عنوان اولین ردیف جدول خود وارد کنید:

ID: 1

First Name: Helen

Last Name: James

Job Title: IT Manager

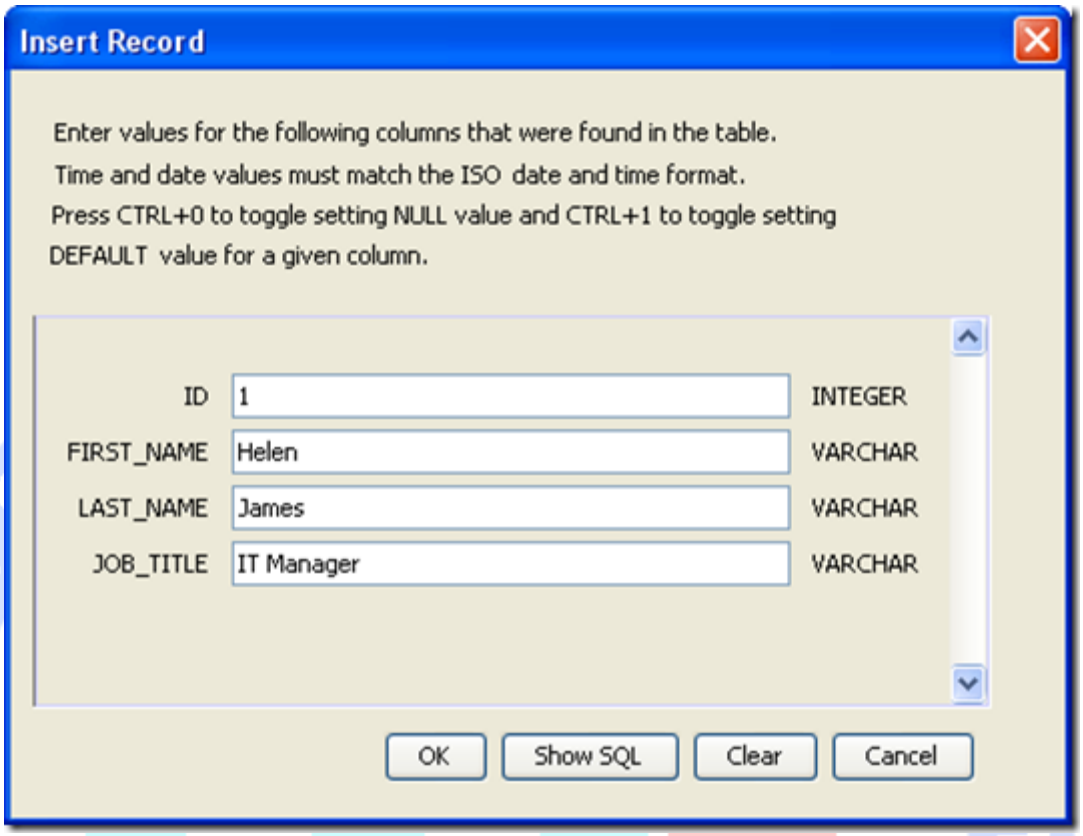
شماره شناسایی: 1

نام: هلن

نام خانوادگی: جیمز

عنوان شغلی: مدیرIT

دیالوگ باکس شما مانند تصویر زیر خواهد بود:



پس از اتمام کار روی OK کلیک کنید که به پنجره ی NetBeans باخواهید گشت. بنابراین اولین ردیف باید نمایش داده شود:

ID	FIRST_NAME	LAST_NAME	JOB_TITLE
1	Helen	James	IT Manager

سه ردیف دیگر شامل داده های زیر وارد کنید:

ID: 2

First Name: Eric

Last Name: Khan

Job Title: Programmer (عنوان شغلی: برنامه نویس)

ID: 3

First Name: Tommy

Last Name: Lee

Job Title: Systems Analyst(عنوان شغلی: تحلیل گر سیستم)

ID: 4

First Name: Priyanka

Last Name: Collins

Job Title: Programmer(عنوان شغلی: برنامه نویس)

پس از اینکه افزودن این ردیف ها به پایان رسید، پنجره ی NetBeans مانند تصویر زیر خواهد بود:

ID	FIRST_NAME	LAST_NAME	JOB_TITLE
1	Helen	James	IT Manager

در بخش بعدی کمی به فرمان های SQL خواهیم پرداخت .

آموزش فرمان های sql در جاوا

در بخش قبل رکوردهایی را در جدول دیتابیس DB جاوا ایجاد کردید. در این بخش کمی به دستورات SQL خواهید پرداخت، طوریکه می توانید این رکوردها را در یک جدول ایجاد کنید.

SQLمخفف Structured Query Language می باشد وراهی برای بررسی دیتابیس هاست. شما می توانید گزینه های بیشتری شامل انتخاب رکدودها(select) ، وارد کردن(insert) ، حذف(Delete)، آپدیت کردن رکوردها(Update) ، ایجاد جدول ها و قرار دادن جدول ها را استفاده کنید. این ابزار، ابزاری بسیار قدرتمند می باشد.

SQLاز لغات کلیدی ساده ای برای انجام کار استفاده می کند. اگر می خواهید که همه ی رکوردها را از یک جدول انتخاب کنید، لغات SELECT و FROMهمراه با نماد "all records (همه ی رکوردها) استفاده می شوند:

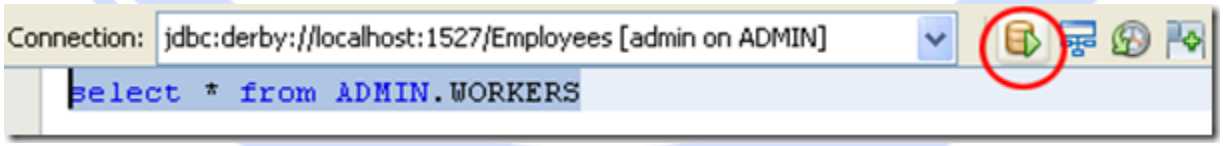
```
SELECT * FROM table_name
```

اگر نگاهی به نیمه ی بالایی پنجره ی NetBeans داشته باشید، عبارت SELECT را مشاهده خواهید کرد که تنظیم شده است :(SQL یک مورد هوشمند نیست)

```
select * from ADMIN.WORKERS
```

این عبارت بیان می کند که همه ی رکوردها را از جدولی به نام Workers انتخاب کنید. (بخش ADMIN قبل از نقطه ی مربوط بهWorkers ، موردی به نام Schemaمی باشد، که ساختار داده را توصیف می کند، به علاوه یوزرها و امتیازاتی را که دارند، تشخیص می دهد).

در NetBeans یک عبارت SQL را با کلیک کردن بر روی دکمه ی Run روی نوار ابزار، اجرا کنید:



سپس نتیجه ی مربوط به عبارات SQL در نیمه ی پایینی پجره نمایش داده می شود:

ID	FIRST_NAME	LAST_NAME	JOB_TITLE
1	Helen	James	IT Manager
2	Eric	Khan	Programmer
3	Tommy	Lee	Systems Analyst
4	Priyanka	Collins	Programmer

عبارتWHERE

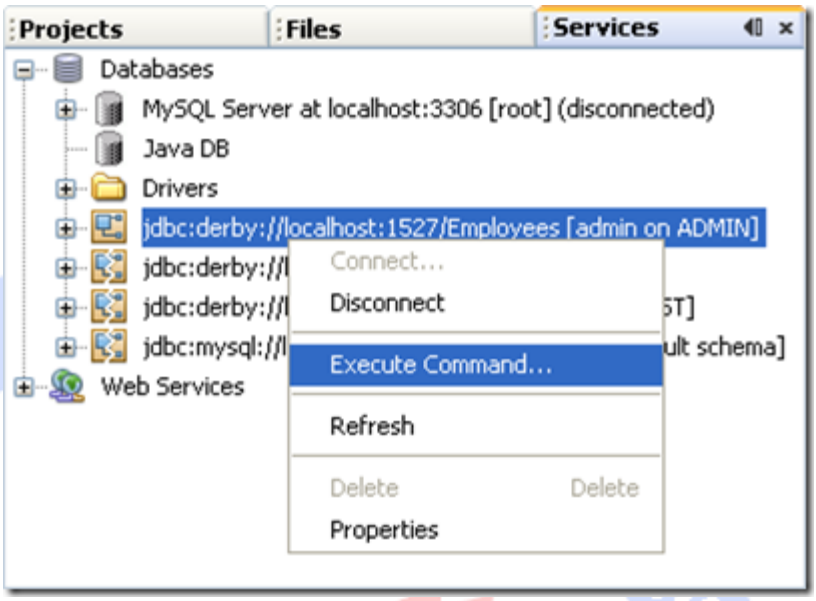
برای اینکه نتایج تحقیق خود را محدود کنید، می توانید از عبارت WHERE به همراه عبارت SELECT استفاده کنید:

```
SELECT * FROM table_name WHERE column_name=value
```

پس از لغت کلیدی WHERE نیاز به نام یک ستون از جدول خود دارید. سپس یک علامت تساوی تایپ کنید که با یک مقدار دنبال می‌شود. به عنوان مثال در اینجا یک عبارت SQL را مشاهده می کنید که تمام برنامه نویسان را به جدول ما بازمی گرداند:

```
SELECT * FROM ADMIN.WORKERS WHERE JOB_TITLE='Programmer'
```

برای اینکه این عبارت SQL را امتحان کنید، روی نام جدول خود در بخش Servicesراست کلیک کنید. از منوی ظاهر شده گزینه ی Execute Command را انتخاب کنید:



وقتی که روی Execute Command کلیک کردید، یک پنجره ی جدید ظاهر خواهد شد. عبارت SQL بالا را تایپ کرده و سپس روی آیکن RUN کلیک کنید:

نتیجه در نیمه ی پایین پنجره نمایش داده خواهد شد:

ID	FIRST_NAME	LAST_NAME	JOB_TITLE
2	Eric	Khan	Programmer
4	Priyanka	Collins	Programmer

همانطور که مشاهده می کنید، دو ردیف از query بازگردانده می شوند.

شما همچنین می توانید از لغت کلیدی LIKE با عبارت WHERE استفاده کنید .این عبارت جایگزین علامت تساوی می شود LIKE . معمولا به یک کاراکتر wildcardاستفاده می شود که کاراکتر % wildcard به معنای هر کاراکتری می باشد، به عنوان مثال وقتی که خط زیرین تنها برای یک کاراکتر استفاده می شود.

به جای علامت تساوی یا لغت کلیدی LIKE از اپراتورهای شرطی نیز می توانید استفاده کنید (بزرگتر از، کمتر از و غیره). اگر یک ستون حقوق داشته باشیم، می توانستیم تمام کارمندانی را جستجو کنیم که کمتر از 1000 در هفته دریافت می کنند:

```
SELECT * FROM ADMIN.WORKERS WHERE SALARY > 1000
```

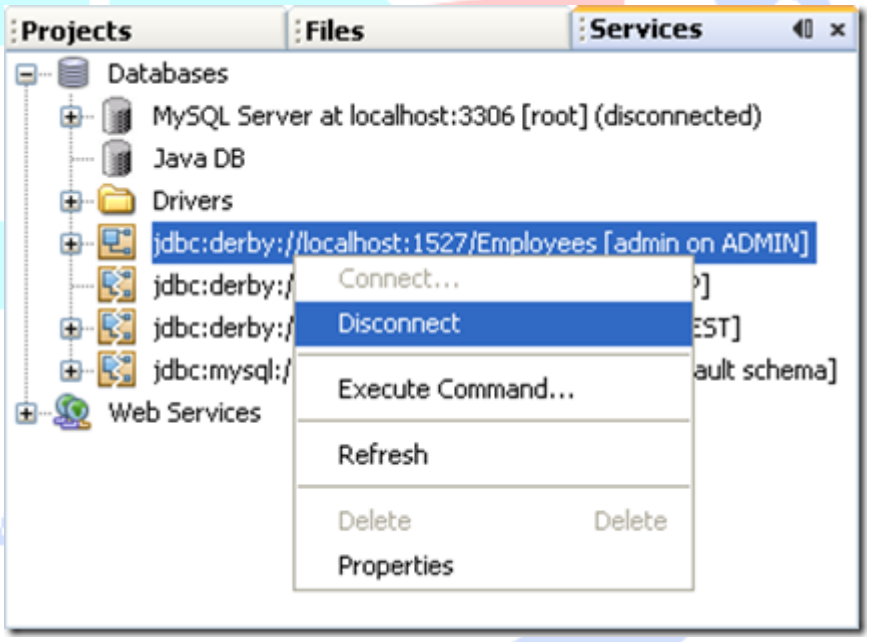
تمرین

اجرای عبارت SQL زیر را امتحان کنید:

```
SELECT * FROM ADMIN.WORKERS WHERE JOB_TITLE LIKE '%er'
```

چه تعداد نتیجه نمایش داده می شوند؟

اکنون که عبارات SQL برای شروع برنامه نویسی کافی می باشند، این مبحث را رها می کنیم. به هرحال قبل از انجام این کار، با کلیک راست کردن در بخش Services، اتصال به جدول های خود را قطع کنید:



ما مجددا به این دیتابیس و جدول متصل خواهیم شد، این بار از کد جاوا استفاده می کنیم .

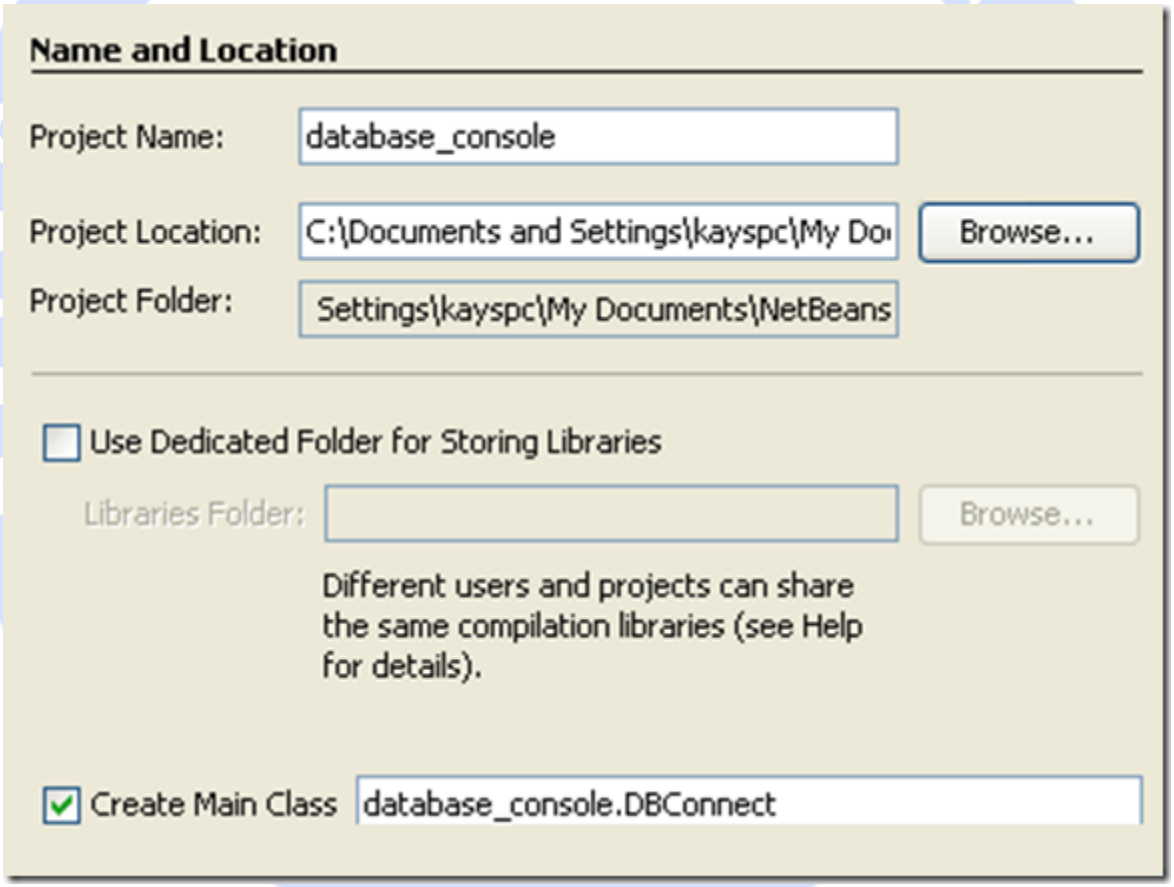
آموزش اتصال به دینابیس با کد جاوا

در بخش های بعدی یک فرم جاوا ایجاد خواهید کرد که اطلاعات را از یک دیتابیس بارگذاری می کند. این فرم دارای گزینه های Next و Previous می باشد تا یک داده را طی کنید. رکوردهای فردی نیز در فیلدهای متن

نمایش داده خواهند شد. سپس دکمه هایی نیز برای آپدیت کردن، حذف کردن و یا ایجاد یک رکورد جدید در دیتابیس اضافه خواهیم کرد.

برای آغاز و به خاطر سادگی کار، از یک پنجره ی terminal/console برای خروجی نتایج از یک دیتابیس استفاده خواهیم کرد.

بنابراین برای انجام این کار با کلیک کردن بر روی File > New Project از منوی NetBeans یک پروژه ی جدید آغاز کنید. یک Java Application (برنامه ی جاوا) ایجاد کنید. پوشه را database_console و گروه اصلی را DBConnectبنامید:



وقتی که روی Finish کلیک می کنید، کد شما اید مانند زیر باشد:

```
package database_console;

public class DBConnect {

    public static void main(String[] args) {

    }
}
```

اتصال به دیتابیس

برای اتصال به یک دیتابیس نیاز به آبجکت Connection دارید. آبجکت Connectionاز یک DriverManagerاستفاده می کند DriverManager . در نام کاربری، پسوورد و موقعیت دیتابیس شما منتقل می شود.

توجه :

این سه عبارت ورودی را به بالای کد خود اضافه کنید:

import java.sql.Connection;

import java.sql.DriverManager;

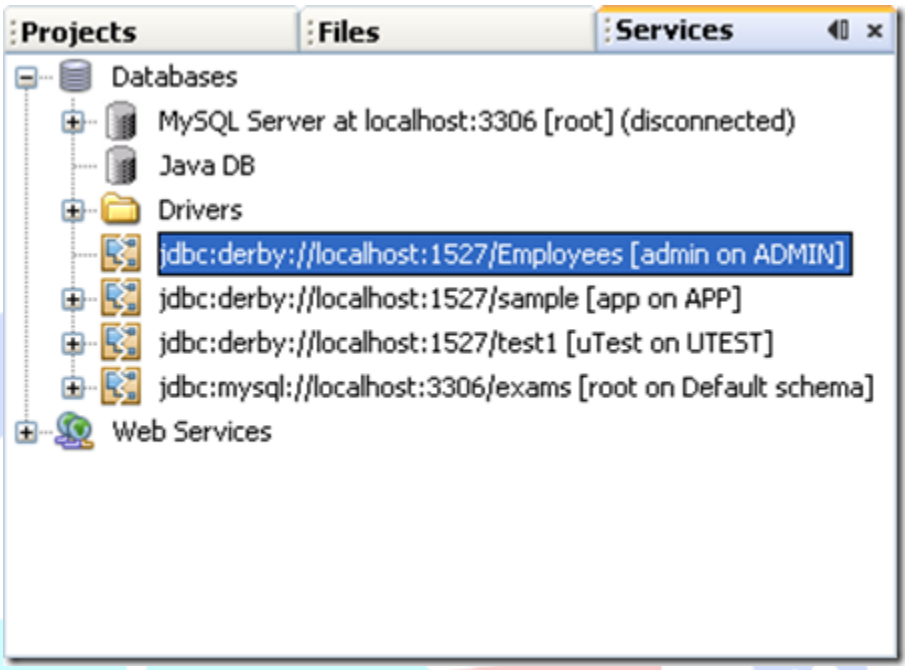
import java.sql.SQLException;

کد مربوط به اتصال یک دیتابیس مانند زیر می باشد:

```
Connection con = DriverManager.getConnection( host, username, password );
```

بنابراین DriverManager دارای متودی به نام getConnection می باشد. این برنامه نیاز به نام میزبان (که موقعیت دیتابیس شما می باشد)، یک یوزرنیم و یک پسوورد دارد. اگر یک اتصال موفق باشد، یک آبجکت Connection ایجاد می شود که آن را con می نامیم.

شما می توانید آدرس میزبان را با نگاه کردن به تب Services در سمت چپ NetBeans دریافت کنید:



آدرس هایلایت شده از دیتابیس بالا عبارت است از:

```
jdbc:derby://localhost:1527/Employees
```

اولین بخش، jdbc:derby://localhost، نوع دیتابیس و سروری است که استفاده می کنید. 1527 شماره ی پورت می باشد. دیتابیس Employees می باشد. تمام اینها می توانند در یک متغیر String قرار بگیرند:

```
String host = "jdbc:derby://localhost:1527/Employees";
```

دو رشته ی دیگر نیز برای نام کاربری و پسوورد اضافه می شوند:

```
String uName = "Your_Username_Here";
String uPass= " Your_Password_Here ";
```

این سه رشته را قبل از آبجکت اتصال وارد کنید، کد شما مانند زیر خواهد بود:

```
package database_console;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DBConnect {

  public static void main(String[] args) {

    String host = "jdbc:derby://localhost:1527/Employees";
    String uName = "admin";
    String uPass = "admin";
    Connection con = DriverManager.getConnection(host, uName, uPass);

  }
}
```

همانطور که در تصویر بالا مشاهده می کنید، زیر کد اتصال یک خط موج دار وجود دارد. دلیل این مسئله این است که در هنگام اتصال به یک دیتابیس با یک خطای خاص روبرو شدیم – خطای SQLException.

این DriverManager است که سعی دارد به دیتابیس متصل شود. اگر با شکست روبرو شود (به عنوان مثال آدرس نادرست میزبان)، یک خطای SQLException برای شما ظاهر می شود. لازم است برای رویارویی با این خطا یک کد بنویسید. در کد زیر، در عبارت try ... catch با خطا روبرو شده ایم:

```
try {

}
catch ( SQLException err ) {
System.out.println( err.getMessage( ) );
}
```

بین پرانتزهای catch یک آبجکت SQLException به نام err تنظیم کرده ایم. سپس می توانیم از متود getMessage آبجکت err استفاده کنیم.

عبارت try...catch بالا را به کد خود اضافه کنید و چهار گوشه ی خطوط اتصال کد خود را بخش try ببرید. پس از این کد شما مانند زیر خواهد بود:

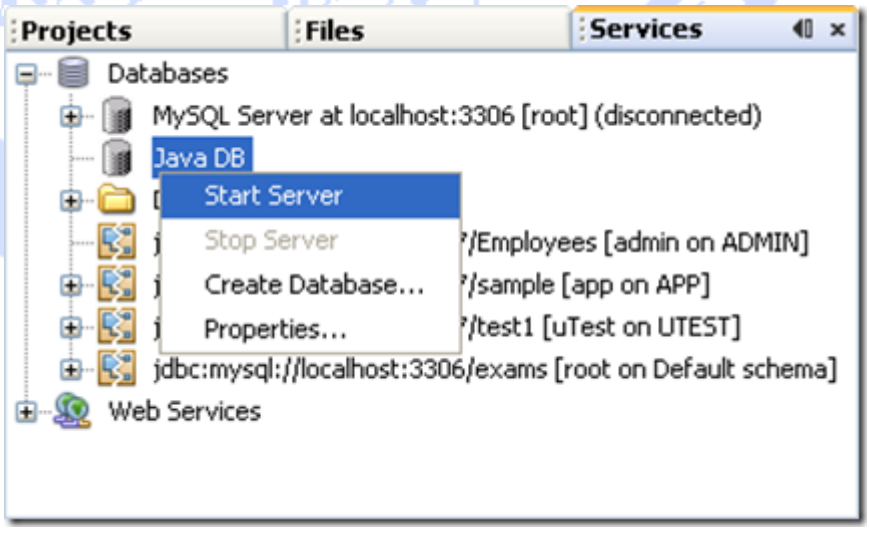
```
public static void main(String[] args) {

  try {
      String host = "jdbc:derby://localhost:1527/Employees";
      String uName = "admin";
      String uPass = "admin";
      Connection con = DriverManager.getConnection(host, uName, uPass);
  }
  catch ( SQLException err ) {
      System.out.println( err.getMessage( ) );
  }

}
```

اجرای کد خود را امتحان کرده و نتیجه را مشاهده کنید.

ممکن است در پنجره ی console خطای زیر را مشاهده کنید:

```
"java.net.ConnectException : Error connecting to server localhost on port
1527 with message Connection refused: connect."
```

اگر چنین اتفاقی افتاد به این معناست که به سرور دیتابیس خود متصل نشده اید. در این مورد در پنجره ی Service روی Java DB کلیک راست کنید. از منوی ظاهر شده Start Server را انتخاب کنید:



لازم است که مطمئن شوید که firewall هایی که دارید، اتصال به سرور را بلاک نمی کنند. یک firewall خوب فورا به شما پیغام می دهد که چیزی در تلاش است که وارد شود و از شما می پرسد که آیا اجازه ی اتصال به آن می دهید یا نه. وقتی که شما اجازه ی اتصال را می دهید پنجره ی خروجی NetBeans پیغام زیر را نمایش خواهد داد:

"Apache Derby Network Server - 10.4.1.3 - (648739) started and ready to accept connections on port 1527 at DATE_AND_TIME_HERE"

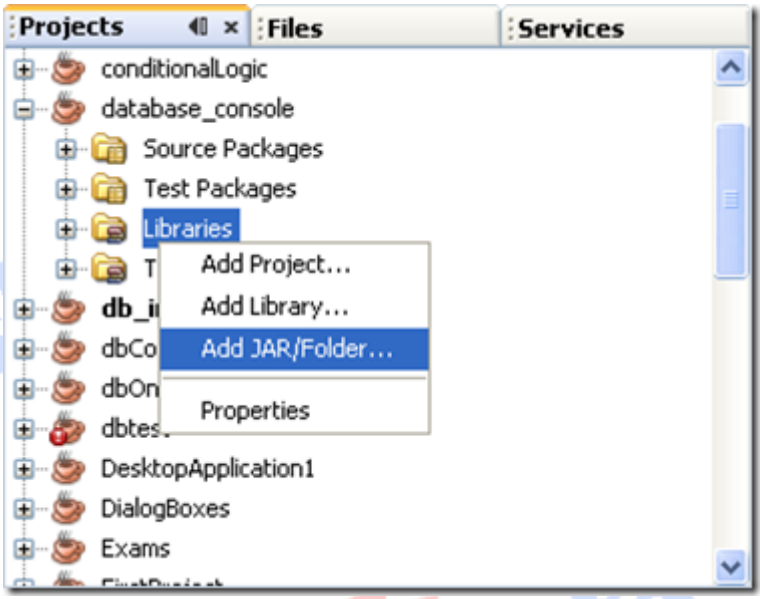
زمانیکه سرور شما آغاز به کار کرد، مجددا برنامه را اجرا کنید. دریافت یک پیغام خطای دیگر شانس خوبیست:

"No suitable driver found for jdbc:derby://localhost:1527/Employees"

دلیل ظاهر شدن این پیغام خطا این است که DriverManager برای اتصال به دیتابیس نیاز به یک Driver دارد، درایورهایی مانند Client Drivers و .Embedded Driversشما می توانید یکی از اینها را وارد کنید تا DriverManagerبتواند کار خود را انجام دهد.

روی تب Projects در سمت چپ پنجره ی Services در NetBeans کلیک کنید .(اگر نمی توانید تب Projectsرا مشاهده کنید، از نوار منو در بالای NetBeans عبارت Window > Projects را کلیک کنید).

پروژه ی خود را جایگذاری کرده و ورودی را باز کنید. روی Libraries کلیک راست کنید. از منوی ظاهر شده Add Jar/Folderرا انتخاب کنید:

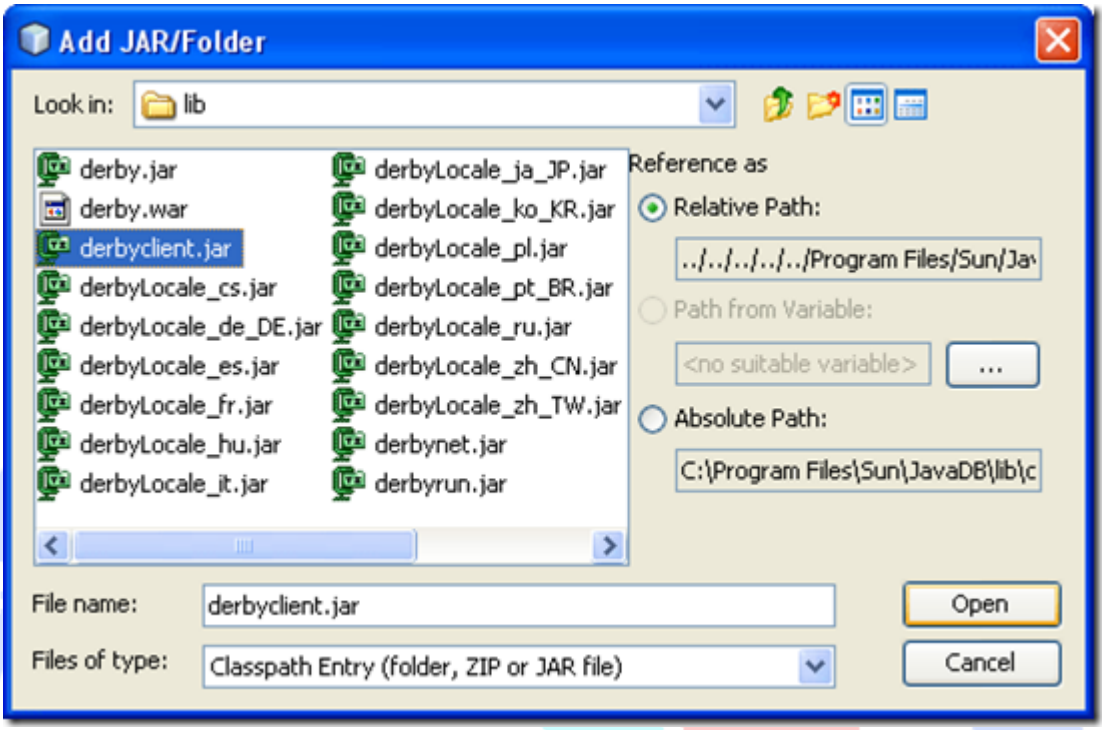


وقتی که روی Add Jar/Folder کلیک می کنید، یک دیالوگ باکس ظاهر می شود .آنچه در اینجا انجام می دهید، افزودن یک فایل Java Archive به پروژه ی خود می باشد. اما فایل JAR که اضافه می کنید، برای دربی Client Drivers (derby) می باشد. بنابراین لازم است تا این فولدر را داخلی سازید. روی یک کامپیوتر در حال اجرایwindows ، این برنامه به شکل موقعیت زیر خواهد بود:

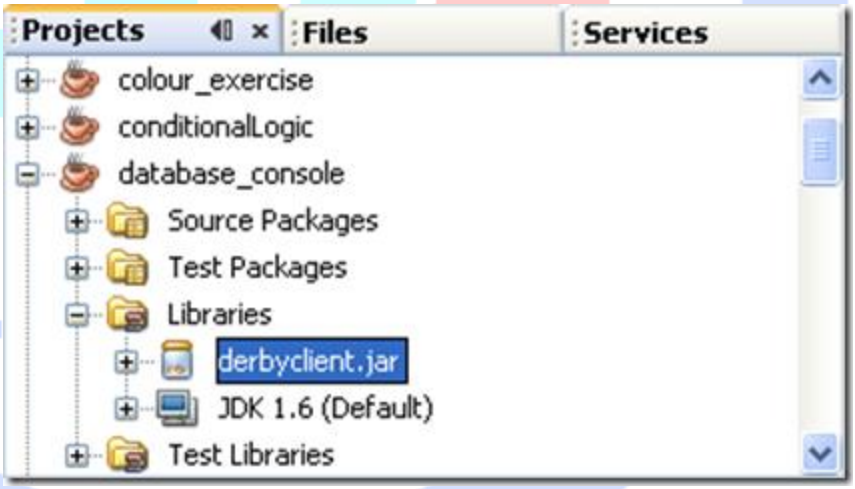
C:\Program Files\Sun\JavaDB\lib

فایلی که در حال جستجوی آن هستید، derbyclient.jarنامیده می شود. اگر نمی توانید آن را پیدا کنید، یا از یک سیستم عامل دیگر به غیر از ویندوز استفاده می کنید، می توانید فایل را جستجو کنید. به موقعیت فایل دقت کنید.

در دیالوگ باکس، فایل derbyclient.jar را انتخاب کنید:



روی open کلیک کرده و فایل به کتابخانه ی پروژه ی شما اضافه خواهد شد:



اکنون که یک درایو Client به پروژه ی شما افزوده شده است، برنامه ی خود را مجددا اجرا کنید. اکنون باید هیچ خطایی دریافت نکنید. (پنجره ی Output عبارت Run و Build Successful را بیان خواهد کرد(.

در بخش بعد آموزش دیتابیس جاوا را ادامه خواهیم داد .

آموزش اتصال جدول به دیتابیس

اکنون که به دیتابیس متصل شده اید، گام بعدی دست یابی به جدول در دیتابیس می باشد. به این خاطر نیاز به اجرای یک SQL Statement و سپس اصلاح همه ی ردیف ها و ستون هایی دارید که بازگردانده شده اند.

برای اجرای یک SQL Statement روی جدول خود، یک آبجکت Statement تنظیم کنید. بنابراین این خط را به بالای کد خود اضافه کنید:

```
import java.sql.Statement;
```

در بخش try از بلوک try ... catch خط زیر را وارد کنید (آن را درست در زیر خط Connection وارد کنید:).

```
Statement stmt = con.createStatement( );
```

در اینجا در حال ایجاد یک آبجکت Statement به نام stmt هستیم. این آبجکت به یک آبجکت Connection به همراه متود createStatment نیاز دارد.

ما نیاز به یک عبارت SQL برای آبجکت Statement هم داریم. بنابراین این خط را به کد خود اضافه کنید:

```
String SQL = "SELECT * FROM Workers";
```

در عبارات بالا همه ی رکوردها را از جدول دیتابیس به نام Workers انتخاب کنید.

می توانیم این SQL query را به متود از آبجکت Statement به نام executeQueryانتقال دهیم. آبجکت Statementکار جمع آوری همه ی رکورهایی را مه با query ما هماهنگ هستند، آغاز می کند.

به هرحال متود executeQuery همه ی رکوردها را در موردی به نام ResultSet بازمی گرداند.

قبل از اینکه این موترد را توضیح دهیم، خط زیر را به بالای کد خود اضافه کنید:

```
import java.sql.ResultSet;
```

اکنون این خط را درست در زیر خط SQL String اضافه کنید:

```
ResultSet rs = stmt.executeQuery( SQL );
```

بنابراین ResultSet به عنوان rs مطرح می شود. این آبجکت تمام رکوردهای جدول دیتابیس را در خود حفظ می کند. قبل از اینکه پیش روی کنیم، در اینجا توضیحاتی در مورد ResultSets را مشاهده می کنید.

ResultSet در جاوا

یک ResultSet راهی برای ذخیره و اصلاح رکوردهای بازگردانده شده از یک SQL query می باشد .

ResultSet ها سه نوع می باشند. نوعی که استفاده می کنید، بستگی به کاری دارد که می خواهید با داده انجام دهید:

1. آیا فقط می خواهید از طریق رکوردها پیش بروید، از ابتدا تا انتها؟
2. آیا می خواهید از طریق رکوردها به جلو و عقب بروید و همچنین تغییرات ایجاد شده در رکوردها را کشف کنید؟
3. آیا می خواهید از طریق رکوردها به جلو و عقب بروید اما به خاطر تغییرات ایجاد شده در رکوردها اذیت نشوید؟

در لیست بالا به نام TYPE_FORWARD_ONLY ResultSet ، عدد 1 را تایپ کنید .عدد در لیست یک TYPE_SCROLL_SENSITIVE ResultSet می باشد. گزینه ی سوم از ResultSet با عنوان TYPE_SCROLL_INSENSITIVE مطرح می شود.

نوع ResultSet بین پرانتزهای createStement قرار می گیرد:

```
Statement stmt = con.createStatement( );
```

از آنجایی که پرانتزها را خالی گذاشته ایم، RecordSet پیش فرض را دریافت می کنیم که TYPE_FORWARD_ONLY می باشد. در بخش بعدی از یک نوع دیگر استفاده خواهیم کرد. اما از آنها به شکل زیر استفاده کنید:

```
Statement stmt = con.createStatement( RecordSet.TYPE_SCROLL_SENSITIVE );
```

بنابراین ابتدا لغت RecordSet را تایپ کنید. بعد از یک نقطه(dot) ، نوع RecordSet مورد استفاده را تایپ کنید.

به هرحال این مسئله در همین جا تمام نمی شود. اگر می خواهید از TYPE_SCROLL_SENSITIVE یا TYPE_SCROLL_INSENSITIVE استفاده کنید، نیاز به تعیین این مسئله نیز می باشد که آیا ResultSet از نوع Read Only است یا قابل آپدیت شدن می باشد. این کار را با در ثابت داخلی شامل CONCUR_READ_ONLY و CONCUR_UPDATABLE انجام می دهید. مجددا این دو نیز بعد از لغت RecordSet قرار می گیرند:

```
ResultSet.CONCUR_READ_ONLY
ResultSet.CONCUR_UPDATABLE
```

این امر منجر به ایجاد یک کد طولانی می شود:

```
Statement stmt = con.createStatement( RecordSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
```

مسئله ی دیگری که باید در مورد ResultSets به آن عادت کنیم، موردی به نام Cursor می باشد. یک Cursor در واقع یک نشانگر به یک ردیف از جدول می باشد. وقتی که رکوردها را در یک ResultSet بارگذاری می کنید، Cursor درست به قبل از اولین ردیف در جدول اشاره می کند. سپس از متودهایی برای اصلاح Cursor استفاده می کنید. اما منظور تشخیص یک ردیف خاص در جدول می باشد.

استفاده از ResultSet

وقتی که همه ی رکوردها را در یک Results set (مجموعه از نتایج) دارید، متودهایی وجود دارند که می توانید برای اصلاح رکوردهای خود استفاده کنید . در اینجا متودهایی را مشاهده می کنید که اغلب استفاده می کنید:

نشانگر را در جدول شما به ردیف بعدی حرکت می دهد. اگر در جدول ردیف دیگری وجود نداشته باشد، مقدار False گزارش داده خواهد شد.

next	نشانگر را در جدول شما به ردیف بعدی حرکت می دهد. اگر در جدول ردیف دیگری وجود نداشته باشد، مقدار False گزارش داده خواهد شد.
Previous	نشانگر را در جدول یک ردیف به قبل بازمی گردانیم. اگر ردیف دیگری در جدول وجود نداشته باشه، مقدار False گزارش داده می شود.
First	نشانگر را به اولین ردیف در جدول حرکت می دهد.
Last	نشانگر را به آخرین ردیف در جدول حرکت می دهد.
*	نشانگر را به یک ردیف خاص در جدول حرکت می دهد. بنابراین (۵) absolute نشانگر را به ردیف شماره ی ۵ در جدول حرکت می دهد.

ResultSet دارای متودهایی نیز می باشد که می توانید برای تشخیص یک ستون خاص (field) در یک ردیف استفاده کنید. می توانید این کار را یا با استفاده از نام ستون و یا با استفاده از شماره ایندکس آن انجام دهید. برای جدول Workers ما چهار ستون تنظیم کرده ایم، که دارای نام های زیر می باشند ,ID : First_Name, Last_Name, and Job_Title. بنابراین شماره های ایندکس عبارتند از 1، 2، 3و 4.

ستون ID را برای نگهداری مقادیر صحیح (Integer) تنظیم کردیم. متودی که برای دریافت مفادیر صحیح در یک ستون انجام می دهید، getInt می باشد:

```
int id_col = rs.getInt("ID");
```

در اینجا یک متغیر صحیح به نام id_col را تنظیم کرده ایم. سپس از متود getInt از آبجکت ResultSet استفاده می کنیم که rs نامیده می شود. بین پرانتزها نام ستون را داریم که می توانستیم به جای آن از عدد ایندکس استفاده کنیم:

```
int id_col = rs.getInt(1);
```

دقت داشته باشید که عدد ایندکس دارای علامت نقل قول نمی باشد اما نام این نمادها را دارد.

برای سه ستون دیگر در جدول دیتابیس، آنها را تنظیم کردیم تا Strings (رشته ها) را حفظ کنند. بنابراین نیاز به متود getString داریم:

```
String first_name = rs.getString("First_Name");
```

یا می توانستیم از عدد ایندکس استفاده کنیم:

```
String first_name = rs.getString(2);
```

از آنجایی که ResultSet Cursor درست به قبل از اولین رکورد در هنگام بارگذاری داده اشاره دارد، لازم است که از متود بعدی برای جابه جایی به اولین ردیف استفاده کنیم. کد زیر اولین رکورد را از جدول دریافت می کند:

```
rs.next( );
int id_col = rs.getInt("ID");
String first_name = rs.getString("First_Name");
String last_name = rs.getString("Last_Name");
String job = rs.getString("Job_Title");
```

دقت کنید که rs.next در ابتدای کد قرار می گیرد. این برنامه نشانگر را به اولین رکورد در جدول حرکت می دهد.

شما می توانید برای نمایش رکورد در پنجره ی Output یک خط چاپی به کد خود اضافه کنید:

```
System.out.println( id_col + " " + first_name + " " + last_name + " " + job
);
```

اکنون کد شما باید مشابه زیر باشد (خط چاپی را کمی تطبیق دادهایم زیرا کمی بلند می باشد.):

```
try {
    String host = "jdbc:derby://localhost:1527/Employees";
    String uName = "admin";
    String uPass = "admin";
    Connection con = DriverManager.getConnection(host, uName, uPass);

    Statement stmt = con.createStatement();
    String sql = "SELECT * FROM Workers";
    ResultSet rs = stmt.executeQuery(sql);

    rs.next();
    int id_col = rs.getInt("ID");
    String first_name = rs.getString("First_Name");
    String last_name = rs.getString("Last_Name");
    String job = rs.getString("Job_Title");

    String p = id_col + " " + first_name + " " + last_name + " " + job;
    System.out.println(p);
}
catch ( SQLException err ) {
    System.out.println( err.getMessage( ) );
}
```

اگر می خواهید تمام رکوردهای جدول را بررسی کنید، می توانید از یک loop استفاده کنید. از آنجایی که متود بعدی true یا false را بازمی گرداند، می توانید از آن به عنوان شرطی برای یک while loop استفاده کنید:

```
while ( rs.next( ) ) {
}
```

در بین پرانتزهای while می توانیم rs.next را مشاهده کنیم. این امر تازمانی که نشانگر از آخرین رکورد در جدول عبور نکرده باشد، درست خواهد بود: اگر عبور کرده باشد، rs.nextمقدار false را گزارش می دهد و while loopخاتمه خواهد یافت. با استفاده ازrs.next ، نشانگر نیز همراه یک رکورد در زمان جابجا خواهد شد. در اینجا همان کد بالا را مشاهده می کنید، اما در حالیکه از while loop استفاده می کند. کد خود را برای هماهنگی تغییر دهید:

```
try {
 String host = "jdbc:derby://localhost:1527/Employees";
 String uName = "admin";
 String uPass = "admin";
 Connection con = DriverManager.getConnection(host, uName, uPass);

 Statement stmt = con.createStatement();
 String sql = "SELECT * FROM Workers";
 ResultSet rs = stmt.executeQuery(sql);

 while (rs.next()) {
     int id_col = rs.getInt("ID");
     String first_name = rs.getString("First_Name");
     String last_name = rs.getString("Last_Name");
     String job = rs.getString("Job_Title");

     String p = id_col + " " + first_name + " " + last_name + " " + job;
     System.out.println(p);
 }

 }
 catch ( SQLException err ) {
     System.out.println( err.getMessage( ) );
 }
}
```

وقتی کد بالا را اجرا می کنید، پنجره ی Output باید مورد زیر را نمایش دهد:

```
run:
1 Helen James IT Manager
2 Eric Khan Programmer
3 Tommy Lee Systems Analyst
4 Priyanka Collins Programmer
BUILD SUCCESSFUL (total time: 2 seconds)
```

اکنون که دیدگاهی در مورد چگونگی اتصال به جدول دیتابیس و نمایش رکوردها را دارید، ما ادامه خواهیم داد و با استفاده از فرم ها و دکمه ها یک برنامه ی پیچیده تر برای ورود به رکوردها خواهیم نوشت.

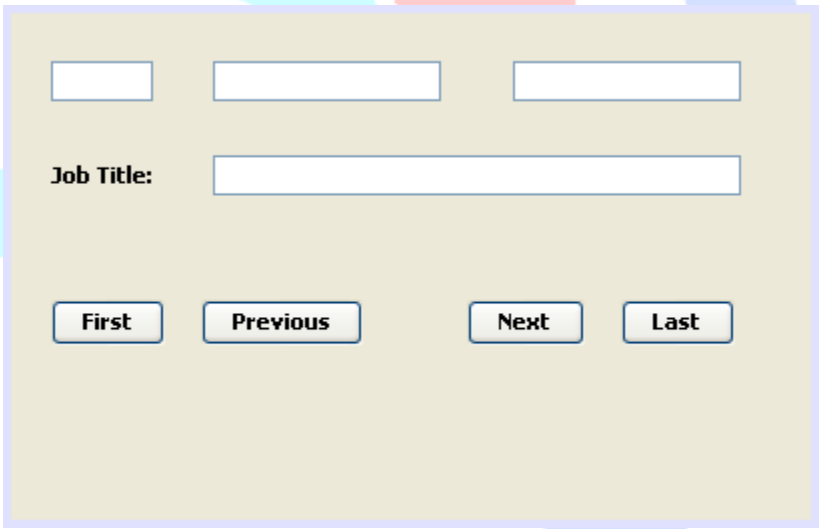
آموزش دکمه های پیمایش در جاوا

آنچه در اینجا انجام خواهیم داد، افزودن چهار دکمه به فرم می باشد. دکمه ها ما را قادر خواهند کرد تا از طریق رکوردها به جلو برویم، به عقب بازگردیم، یا وارد اولین ویا آخرین رکورد شویم.

بنابراین یک پنل جدید به فرم خود اضافه کنید. آن را بزرگ کرده و سپس دکمه ها را به پنل اضافه کنید. نام های متغیرهای دکمه ها را مانند زیر تغییر دهید:

btnNext btnPrevious btnLast btnFirst

متن روی هر دکمه را به Next, Previous, Last, First تغییر دهید. سپس فرم شما چیزی مشابه تصویر زیر خواهد بود:



حرکت به رکورد بعدی:

روی دکمه ی Next دابل کلیک کنید تا یک code stub ایجاد کنید.

شما با دکمه ی Next دو کار می توانید انجام دهید: اول بررسی اینکه آیا رکورد دیگری وجود دارد تا به آن وارد شوید و دوم اینکه اگر رکورد دیگری وجود دارد آن را در Text Fields نمایش دهید. برای این مسئله می توانیم یک IF Statement ایجاد کنیم. اما لازم است در یک بلوک try ... catch قرار گیرد .بنابراین کد زیر را به code stub مربوط به دکمه ی Next وارد کنید:

مثال

```
try {
if ( rs.next( ) ) {
}
else {
rs.previous( );
JOptionPane.showMessageDialog(Workers.this, "End of File");
}
}
catch (SQLException err) {
JOptionPane.showMessageDialog(Workers.this, err.getMessage());
}
```

IF Statement در واقع ResultSet را در یک زمان روی یک رکورد جابه جا می کند. اگر رکورد دیگری وجود نداشته باشد، یک مقدار false گزارش می شود. بخش Else در رکورد esultSet را به عقب بازمی گرداند، به این علت که نشانگر آخرین رکورد را انتقال خواهد داد.

در آکولادهای مربوط به IF Statement ، می توانیم کدی را برای نمایش رکورد در Text Fields اضافه کنیم:

مثال

```
int id_col = rs.getInt("ID");
String id = Integer.toString(id_col);
String first = rs.getString("First_Name");
String last = rs.getString("Last_Name");
String job = rs.getString("Job_Title");
textID.setText(id);
textFirstName.setText(first);
textLastName.setText(last);
textJobTitle.setText(job);
```

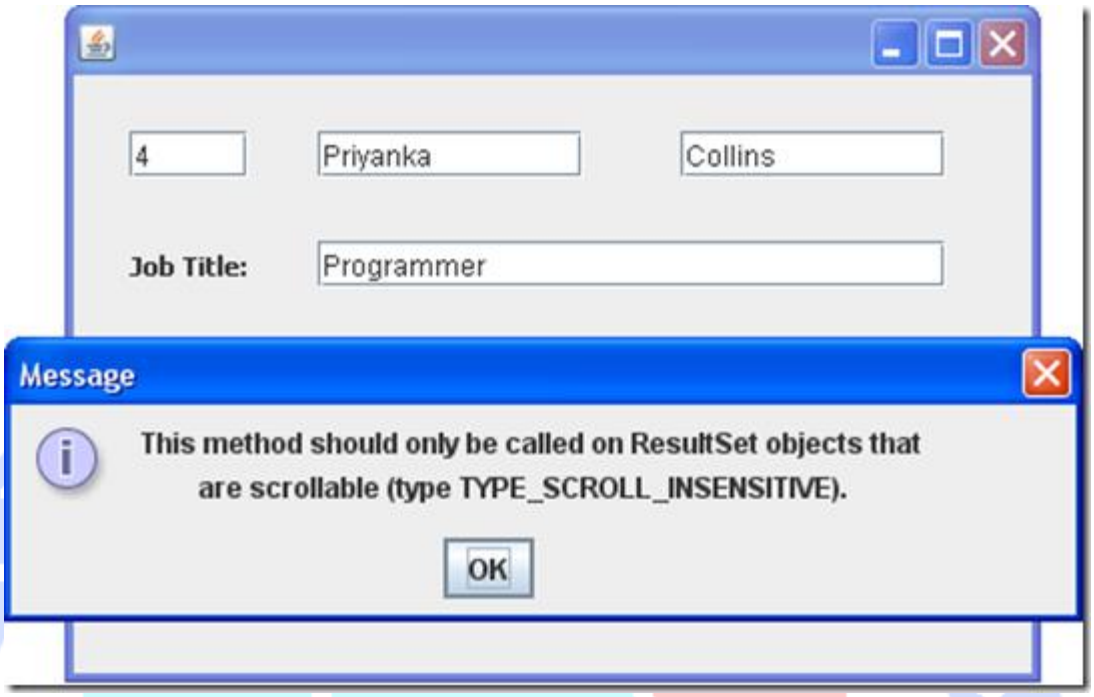
این همان کدی است که در متود DoConnect داریم. (می توانستیم یک متود جدید ایجاد کنیم، بنابراین برای جلوگیری از تکرار هر گونه کد و ساده بودن آن این کار را کردیم.)

کد مربوطبه دکمه ی Next باید مشابه کد زیر باشد:

```
try {
    if (rs.next()) {
        int id_col = rs.getInt("ID");
        String id = Integer.toString(id_col);
        String first = rs.getString("First_Name");
        String last = rs.getString("Last_Name");
        String job = rs.getString("Job_Title");

        textID.setText(id);
        textFirstName.setText(first);
        textLastName.setText(last);
        textJobTitle.setText(job);
    }
    else {
      rs.previous();
      JOptionPane.showMessageDialog(Workers.this, "End of File");
    }
}
catch (SQLException err) {
    JOptionPane.showMessageDialog(Workers.this, err.getMessage());
}
```

وقتی کد خود را اضافه کردید، برنامه ی خود را اجرا کرده و آن را امتحان کنید. اگر به کلیک کردن روی دکمه ی next ادامه دهید، همه ی داده ها را در جدول مشاهده خواهید کرد. به هرحال در اینجا مشکلی وجود دارد.

وقتی به آخرین رکورد می رسید، باید یک پیغام خطا مشاهده کنید:



مشکل این است که ما یک خط rs.previous اضافه کرده ایم. به هرحال ما از نوع ResultSet پیش فرض استفاده کرده ایم. همانطور که در آخرین بخش توضیح دادیم، این خط به شما یک ResultSet ارائه می دهد که فقط می تواند به جلو حرکت کند. ما می توانیم از نوع پیشنهاد شده در پیغام خطا استفاده کنیم.

برنامه ی خود را متوقف کرده و به پنجره ی برنامه نویسی (coding window) خود بازگردید. در متود DoConnect، خط زیر را قرار دهید:

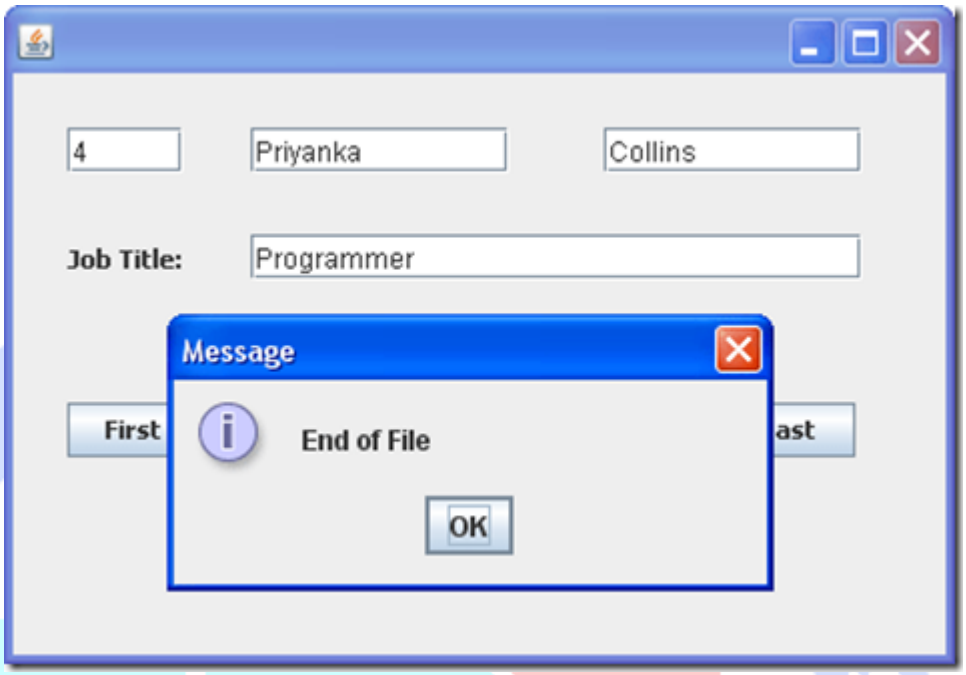
```
stmt = con.createStatement( );
```

آن را مانند زیر تغییر دهید (کد شما باید در یک خط قرار گیرد.):

```
stmt = con.createStatement(
ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_UPDATABLE );
```

اکنون نوع ResultSet به ما اجازه خواهد داد تا به عقب نیز برگردیم.

مجددا برنامه ی خود را اجرا کنید. آنقدر روی دکمه ی Next کلیک کنید تا به آخرین رکورد برسید. شما باید پیغام خطایی از بخش try از بلوک try ... catch مشاهده کنید:



در بخش بعد در مورد چگونگی بازگشتن به عقب از طریق رکوردهای دیتابیس صحبت خواهیم کرد .

آموزش move back از طریق دیتابیس جاوا

کد مربوط به دکمه ی Previous مشابه دکمه ی Next می باشد. اما به جای استفاده از rs.Next از rs.Previousاستفاده می کنید.

به صفحه ی Design برگردید و روی دکمه ی Previous دابل کلیک کنید تا یک code stub ایجاد کنید.

به جای تایپ کردن مجدد تمام کد، به سادگی کد را می توانید از دکمه ی Nextکپی و پیست کنید. سپس rs.Nextرا در عبارت IF به rs.Previous تغییر دهید rs.Previous .را در بخش ELSE به rs.Next تغییر دهید. همچنین می توانید متن پیغام خطا را از "End of File" به "Start of File"تغییر دهید.

کد شما باید مشابه زیر باشد:

```
try {
    if (rs.previous()) {
        int id_col = rs.getInt("ID");
        String id = Integer.toString(id_col);
        String first = rs.getString("First_Name");
        String last = rs.getString("Last_Name");
        String job = rs.getString("Job_Title");

        textID.setText(id);
        textFirstName.setText(first);
        textLastName.setText(last);
        textJobTitle.setText(job);
    }
    else {
      rs.next();
      JOptionPane.showMessageDialog(Workers.this, "Start of File");
    }
}
catch (SQLException err) {
    JOptionPane.showMessageDialog(Workers.this, err.getMessage());
}
```

برنامه ی خود را مجددا اجرا کنید. باید با کلیک کردن بر روی دو دکمه قادر به حرکت کردن به جلو و عقب در دیتابیس باشید .

آموزش حرکت به اولین و آخرین رکورد در جاوا

حرکت به اولین و آخرین رکوردهای دیتابیس آسان تر می باشد.

روی دکمه ی First دابل کلیک کنید تا code stub ایجاد کنید. اکنون کد زیر را اضافه کنید:

```
try {
    rs.first();
    int id_col = rs.getInt("ID");
    String id = Integer.toString(id_col);
    String first = rs.getString("First_Name");
    String last = rs.getString("Last_Name");
    String job = rs.getString("Job_Title");

    textID.setText(id);
    textFirstName.setText(first);
    textLastName.setText(last);
    textJobTitle.setText(job);
}
catch (SQLException err) {
    JOptionPane.showMessageDialog(Workers.this, err.getMessage());
}
```

اکنون نیازی به IF … ELSE Statement نداریم. تنها کاری که باید انجام دهیم حرکت نشانگر با rs.First به اولین رکورد و سپس نمایش اولین رکورد در Text Fields می باشد.

به مشابه کد زیر را به دکمه ی Last خود اضافه کنید (می توانید این کد را برای دکمه ی First نیز کپی کنید.)

```
try {
    rs.last();
    int id_col = rs.getInt("ID");
    String id = Integer.toString(id_col);
    String first = rs.getString("First_Name");
    String last = rs.getString("Last_Name");
    String job = rs.getString("Job_Title");

    textID.setText(id);
    textFirstName.setText(first);
    textLastName.setText(last);
    textJobTitle.setText(job);
}
catch (SQLException err) {
    JOptionPane.showMessageDialog(Workers.this, err.getMessage());
}
```

تنها تغییری که باید ایجاد شود، استفاده از rs.Last به جای rs.Last روی اولین خط می باشد . وقتی کد را اضافه کردید، برنامه ی خود را مجددا اجرا کنید. اکنون باید قادر به حرکت به آخرین رکورد و اولین رکورد در دیتابیس خود باشید . در بخش بعدی به چگونگی آپدیت کردن یک رکورد خواهیم پرداخت .

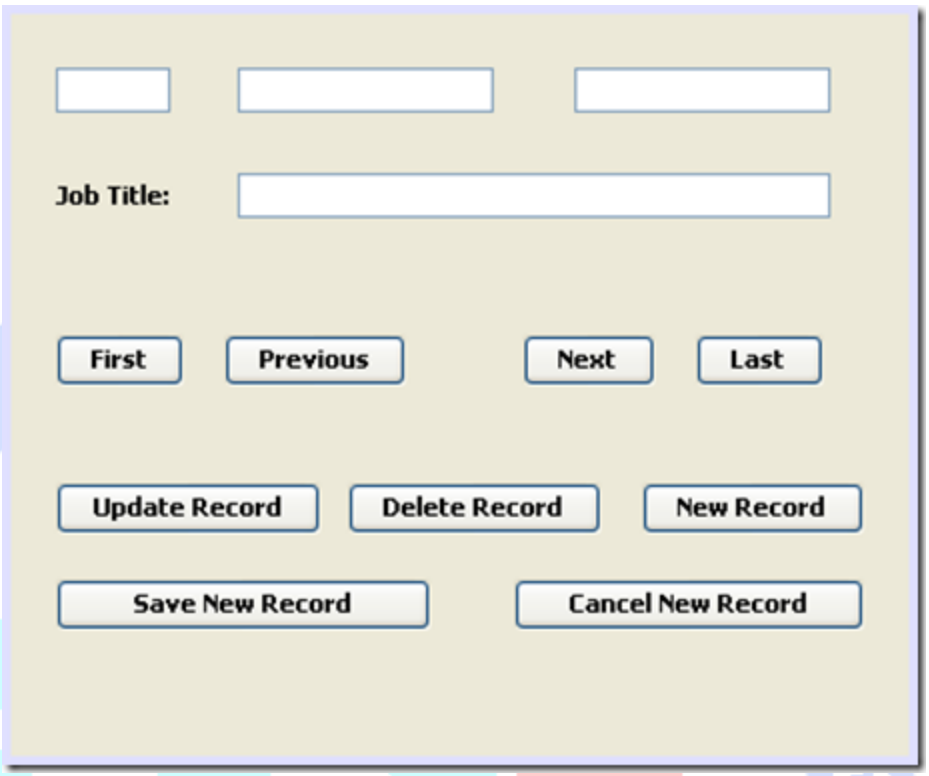
آموزش آپدیت کردن رکورد در جاوا

ResultSet دارای متودهای Update می باشد که به شما اجازه ی آپدیت کردن رکوردها، نه تنها در ResultSet، بلکه در دیتابیس تاکید شده نیز می دهد.

فرم خود را کمی بلندتر سازید. اکنون یک پنل جدید به فرم خود اضافه کنید . یک دکمه ی جدید به پنل اضافه کنید. نام پیش فرض متغیر را به btnUpdateRecord تغییر دهید. متن روی دکمه را به Update Record تغییر دهید . ما همچنین قصد داریم دکمه هایی برای ایجاد یک رکورد جدید در دیتابیس برای ذخیره کردن، کنسل کردن آپدیت ها و حذف یک رکورد، داشته باشیم. بنابراین چهار دکمه ی دیگر به پنل اضافه کنید و تغییرات زیر را اعمال کنید:

```
Button Variable Name: btnNewRecord
Button Text: New Record
Button Variable Name: btnDeleteRecord
Button Text: Delete Record
Button Variable Name: btnSaveRecord
Button Text: Save New Record
Button Variable Name: btnCancelNewRecord
Button Text: Cancel New Record
```

وقتی که کار شما انجام شد، فرم شما باید مشابه تصویر زیر باشد (گرچه برای مرتب سازی مجدد دکمه ها آزاد هستید.):



روی دکمه ی Update دابل کلیک کنید تا یک code stub جدید ایجاد ایجاد کنیم.

اولین کاری که باید انجام دهیم، دریافت متن از Text Fields می باشد:

```
String first = textFirstName.getText( );
String last = textLastName.getText( );
String job = textJobTitle.getText( );
String ID = textID.getText( );
```

به هرحال اگر بخواهیم یک فیلد ID را آپدیت کنیم، نیاز به تبدیل String به یک Integer داریم:

```
int newID = Integer.parseInt( ID );
```

آبجکت Integer دارای متودی به نام parseInt می باشد. بین پرانتزهایparseInt ، رشته ای را تایپ کنید که سعی در تغییر آن دارید.

اکنون که همه ی داده ها را از Text Fields داریم، می توانیم متودهای مرتبط آپدیت را از آبجکت ResultSet فرا بخوانیم:

```
rs.updateString( "First_Name", first );
```

تعداد کمی متودهای آپدیت وجود دارد. مورد بالا از updateString استفاده می کند. اما لازم است که شما در اینجا از جدول دیتابیس خود تایپ کنید. سه رشته (First_Name, Last_Name, Job_Title) و یک مقدار صحیح (ID) وجود دارد .بنابراین ما به سه متود updateString و یک متود updateInt نیاز داریم.

بین پرانتزهای متودهای آپدیت، نیاز به نام یک ستون از دیتابیس خود دارید (گرچه این می تواند مقدار ایندکس آن باشد.). پس از یک کاما، داده ی جایگزین را تایپ کنید. بنابراین در مثال بالا می خواهیم که ستون First_Name ار آپدیت کنیم و مقداری را در متغیری به نام first جایگزین آن کنیم.

به هرحال متودهای آپدیت فقط ResultSet را آپدیت می کنند. برای اعمال تغییرات به دیتابیس، یک فرمان updateRowرا منتشر کنید:

```
rs.updateRow( );
```

در اینجا تمام خطوط کد مربوط به آپدیت ResultSet و جدول دیتابیس را مشاهده می کنید:

```
try {
rs.updateInt( "ID", newID );
rs.updateString( "First_Name", first );
rs.updateString( "last_Name", last );
rs.updateString( "Job_Title", job );
rs.updateRow( );
JOptionPane.showMessageDialog(Workers.this, "Updated");
}
catch (SQLException err) {
System.out.println(err.getMessage() );
}
```

مجددا لازم است آن را در عبارت try ... catch قرار دهیم، درست انگار اشتباهی رخ داده است. دقت کنید که برای یک آپدیت موفق یک message box اضافه کرده ایم.

در تصویر زیر کد کامل مربوط به Update Button را مشاهده می کنید:

```
String first = textFirstName.getText();
String last = textLastName.getText();
String job = textJobTitle.getText();
String ID = textID.getText();

int newID = Integer.parseInt(ID);

try {
    rs.updateInt("ID", newID);
    rs.updateString("First_Name", first);
    rs.updateString("last_Name", last);
    rs.updateString("Job_Title", job);
    rs.updateRow();
    JOptionPane.showMessageDialog(Workers.this, "Updated");
}
catch (SQLException err) {
    System.out.println(err.getMessage());
}
```

برنامه ی خود را اجرا کرده و آن را امتحان کنید. چند داده را در یک Text Field تغییر دهید) به عنوان مثال Tommy به .(Timmy سپس روی دکمه ی Update کلیک کنید. رکورد را طی کرده و سپس به عقب بازگردید. تغییرات هنوز باید وجود داشته باشند. اکنون برنامه ی خود را بسته و مجددا آن را اجرا کنید. باید مشاهده کنید که تغییرات ثابت هستند.

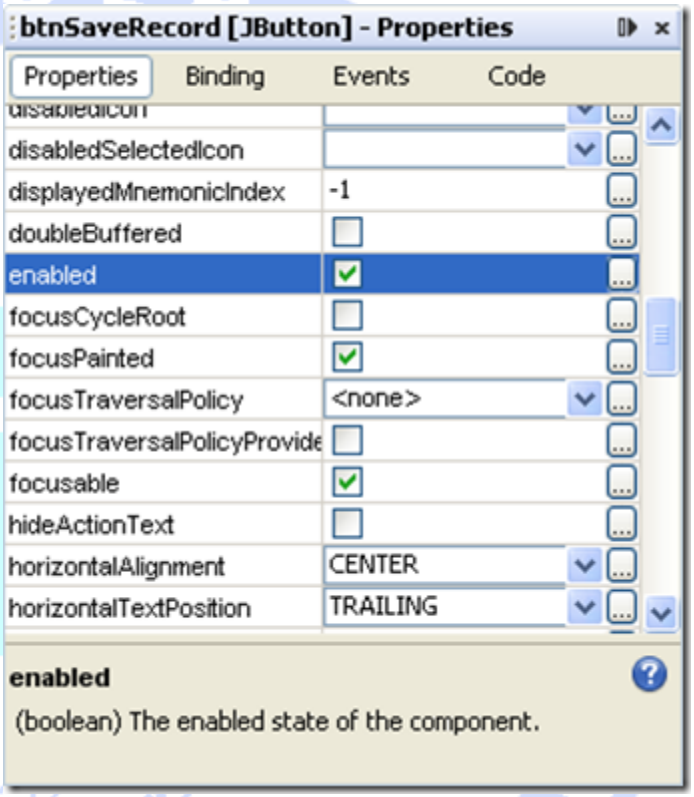
در بخش بعد به چگونگی اضافه کردن یک رکورد جدید را مشاهده خواهید کرد .

آموزش افزودن رکورد در جاوا

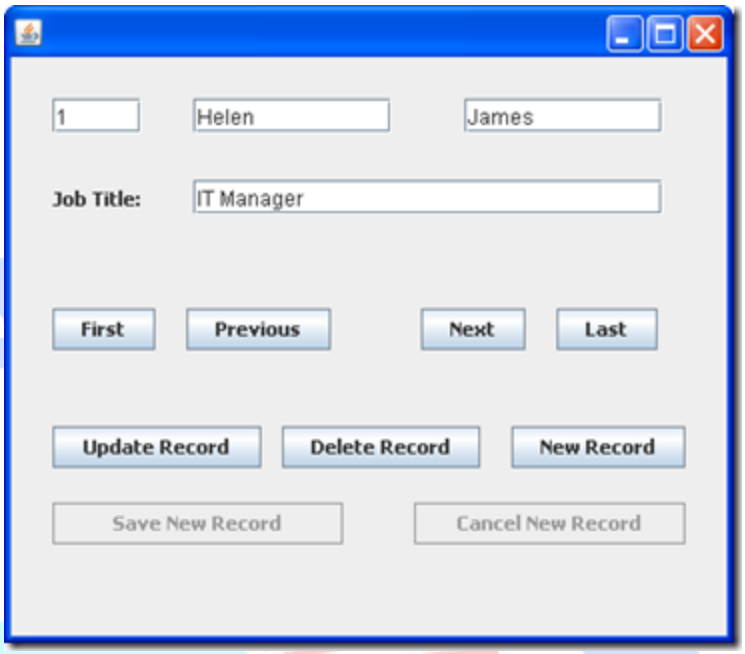
ما سه دکمه داریم که به رکوردهای جدید اشاره می کنند New Record, Save New Record :و Cancel. New Record دکمه New Record فقط Text Fields را پاک کرده و آن ها را برای داده ی جدید آماده می سازد. ما می توانیم دکمه ی New Record را غیرفعال سازیم. کار دیگری که می توانیم انجام دهیم تهیه ی یک یادداشت از رکوردی می باشد که اخیرا بارگذاری شده است. اگر یک یوزر تصمیم خود را عوض کند، می

تواند همه ی دکمه ها را با کلیک کردن بر روی دکمه ی Cancel مجددا فعال سازد. با کلیک کردن بر روی دکمه ی Save New Record ، در واقع داده را در دیتابیس ذخیره کنید.

اگر این برنامه کمی گیج کننده است، مورد زیر را امتحان کنید. روی دکمه ی Save New Record کلیک کنید تا آن را انتخاب کنید. در بخش Properties در سمت راست، پراپرتی Enabled را قرار دهید:



تیک مربوط به باکس سمت راست enabled را بردارید Save New Record .غیرفعال خواهد شد. همین کار را برای دکمه ی Cancel New Record انجام دهید .این دکمه نیز غیرفعال خواهد شد. وقتی فرم شما بارگذاری می شود، مانند تصویر زیر خواهد بود:



حتی اگر کدی برای این دو دکمه داشتید، در صورت کلیک کردن روی هرکدام از آنها هیچ اتفاقی نخواهد افتاد.

وقتی دکمه ی New Record کلیک می شود، می توانیم دکمه های زیر را غیرفعال کنیم:

First Previous Next Last Update Record Delete Record New Record

به هرحال دکمه های Save و Cancel می توانند فعال شوند. اگر یوزر روی Cancelکلیک کند، می توانیم دکمه ها را مجددا به حالت قبل بازگردانیم.

روی دکمه ی New Recordکلیک کنید تا یک code stub ایجاد کنید. خطوط زیر را به آن اضافه کنید:

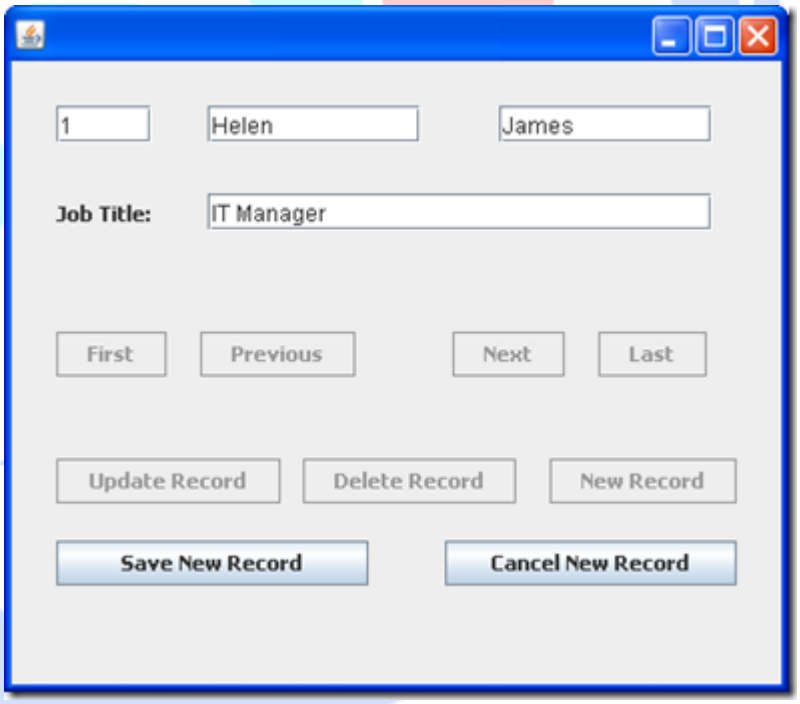
```
btnFirst.setEnabled( false );
btnPrevious.setEnabled( false ) ;
btnNext.setEnabled( false );
btnLast.setEnabled( false );
btnUpdateRecord.setEnabled( false );
btnDelete.setEnabled( false );
btnNewRecord.setEnabled( false );
btnSaveRecord.setEnabled( true );
btnCancelNewRecord.setEnabled( true );
```

بنابراین با استفاده از پراپرتی setEnabled هفت دکمه خاموش شده و دو دکمه روشن می شوند.

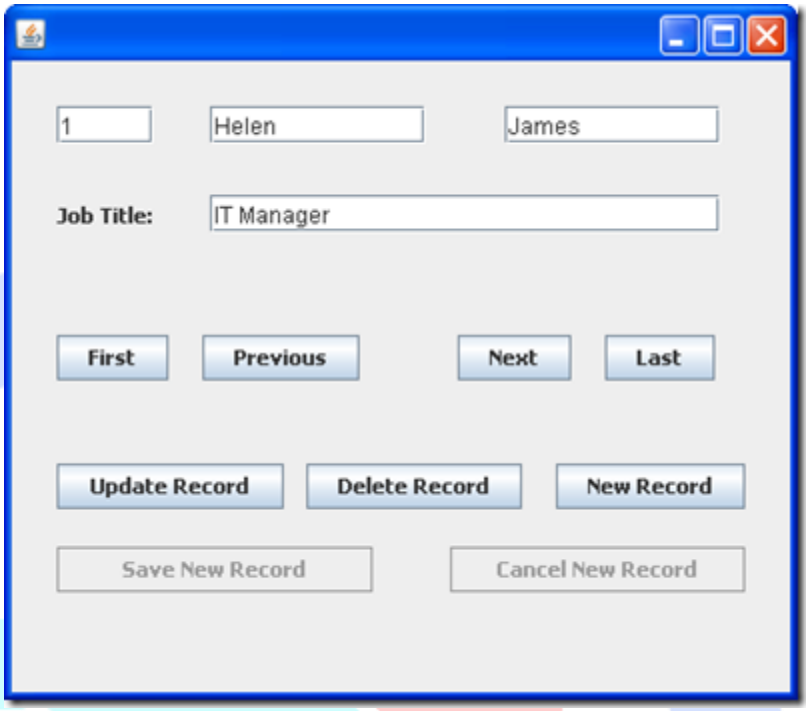
ما می توانیم عکس این کار را برای دکمه ی Cancel انجام دهیم. به ویو Designبازگردید. روی دکمه ی Cancel New Recordدابل کلیک کنید تا یک code stub جدید ایجاد کنید. موارد زیر را به آن اضافه کنید:

```
btnFirst.setEnabled( true );
btnPrevious.setEnabled( true ) ;
btnNext.setEnabled( true );
btnLast.setEnabled( true );
btnUpdateRecord.setEnabled( true );
btnDelete.setEnabled( true );
btnNewRecord.setEnabled( true );
btnSaveRecord.setEnabled( false );
btnCancelNewRecord.setEnabled( false );
```

اکنون برنامه را اجرا کرده و آن را امتحان کنید. روی دکمه ی New Record کلیک کنید که فرم شما مانند زیتصویر زیر خواهد بود:



روی دکمه ی Cancel New Record کلیک کنید که پس از آن فرم شما مانند تصویر زیر خواهد بود:



کار دیگری که باید انجام دهیم ثبت ردیفی است که اخیرا بارگذاری شده است . به عبارت دیگر ردیفی که اخیرا در Text Fields بارگذاری شده است. لزوم این کار به این خاطر است که قرار است Text Fields پاک شوند. اگر دکمه ی Cancel کلیک شده باشد، سپس کی توانیم داده ی حذف شده را مجددا بارگذاری کنیم.

متغیر عدد صحیح زیر را به بالای کد خود و درست زیر خطوط Connection, Statement و ResultSet اضافه کنید:

```
int curRow = 0;
```

بالای کد شما باید مانند تصویر زیر باشد:

```
public class Workers extends javax.swing.JFrame {

    Connection con;
    Statement stmt;
    ResultSet rs;
    int curRow = 0;

    public Workers() {
        initComponents();
        DoConnect();
    }
```

اکنون به کد New Record بازگردید.

برای درک این قضیه نشانگر به کدام ردیف اشاره می کند، متودی به نام getRowوجود دارد. این متود به شما اجازه می دهد تا شماره ی ردیفی را که نشانگر در حال خاضر روی آن است را ذخیره کنید:

```
curRow = rs.getRow( );
```

ما از شماره ی این ردیف در کد Cancel New Record استفاده خواهیم کرد.

تنها چیز دیگری که برای دکمه ی New Record باید انجام دهیم، پاک کردن Text Fields می باشد. این کار بسیار ساده می باشد:

```
textFirstName.setText("");
textLastName.setText("");
textJobTitle.setText("");
textID.setText("");
```

بنابراین ما فقط در حال تنظیم پراپرتی Text در یک رشته ی خالی می باشیم.

از آنجایی که از متود ResultSet استفاده کرده ایم، لازم است که موارد را در یک بلوک try ... catch قرار دهیم. کد مربوط به دکمه ی New Record باید مشابه زیر باشد:

```
try {
    curRow = rs.getRow();

    textFirstName.setText("");
    textLastName.setText("");
    textJobTitle.setText("");
    textID.setText("");

    btnFirst.setEnabled(false);
    btnPrevious.setEnabled(false);
    btnNext.setEnabled(false);
    btnLast.setEnabled(false);
    btnUpdateRecord.setEnabled(false);
    btnDelete.setEnabled(false);
    btnNewRecord.setEnabled(false);

    btnSaveRecord.setEnabled(true);
    btnCancelNewRecord.setEnabled(true);
}
catch (SQLException err) {
    System.out.println(err.getMessage());
}
```

برای دکمه یCancel ، نیاز به گرفتن ردیفی داریم که قبلا بارگذاری شد و داده را در Text Field قرار داد.

برای بازگرداندن نشانگر به ردیفی که قبلا به آن اشاره می کرد، می توانیم از متود absolute استفاده کنیم:

```
rs.absolute( curRow );
```

متد absolute نشانگر را به یک موقعیت ثابت در ResultSet حرکت می دهد .ما می خواهیم مقداری را که در متغیر curRow ذخیره کردیم، جابه جا کنیم.

اکنون که نشانگر در حال اشاره به ردیف درست می باشد، می توانیم داده ها را در Text Fields بارگذاری کنیم:

```
textFirstName.setText( rs.getString("First_Name") );
textLastName.setText( rs.getString("Last_Name") );
textJobTitle.setText( rs.getString("Job_Title") );
textID.setText( Integer.toString( rs.getInt("ID" )) );
```

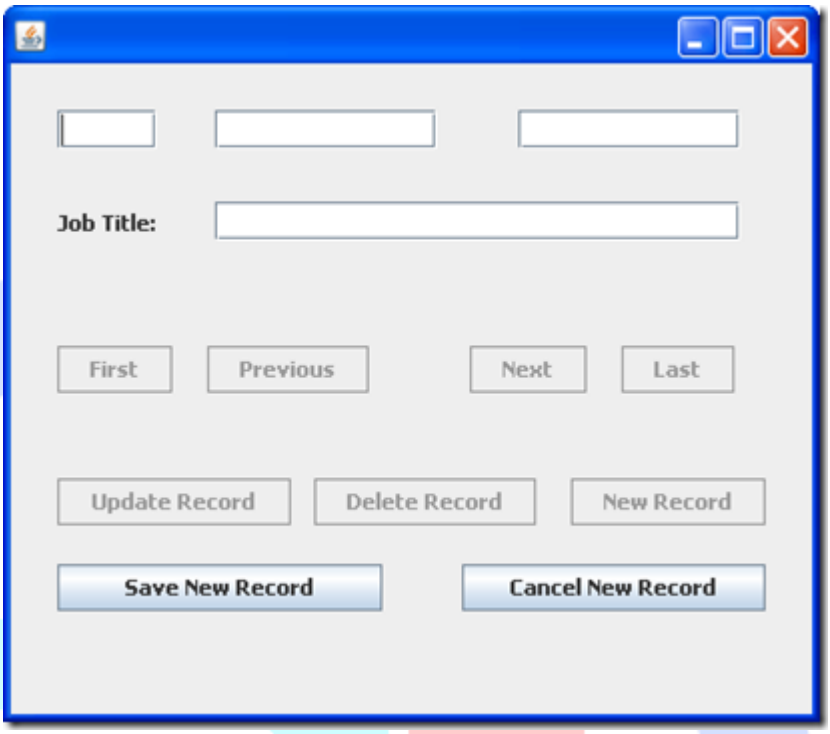
قرار دادن تمام این کد در بلوک try ... catch، کد زیر را به ما ارائه می دهد:

```
try {
    rs.absolute(curRow);
    textFirstName.setText(rs.getString("First_Name"));
    textLastName.setText(rs.getString("Last_Name"));
    textJobTitle.setText(rs.getString("Job_Title"));
    textID.setText( Integer.toString(rs.getInt("ID")) );

    btnFirst.setEnabled(true);
    btnPrevious.setEnabled(true);
    btnNext.setEnabled(true);
    btnLast.setEnabled(true);
    btnUpdateRecord.setEnabled(true);
    btnDelete.setEnabled(true);
    btnNewRecord.setEnabled(true);

    btnSaveRecord.setEnabled(false);
    btnCancelNewRecord.setEnabled(false);
}
catch (SQLException err) {
    System.out.println(err.getMessage());
}
```

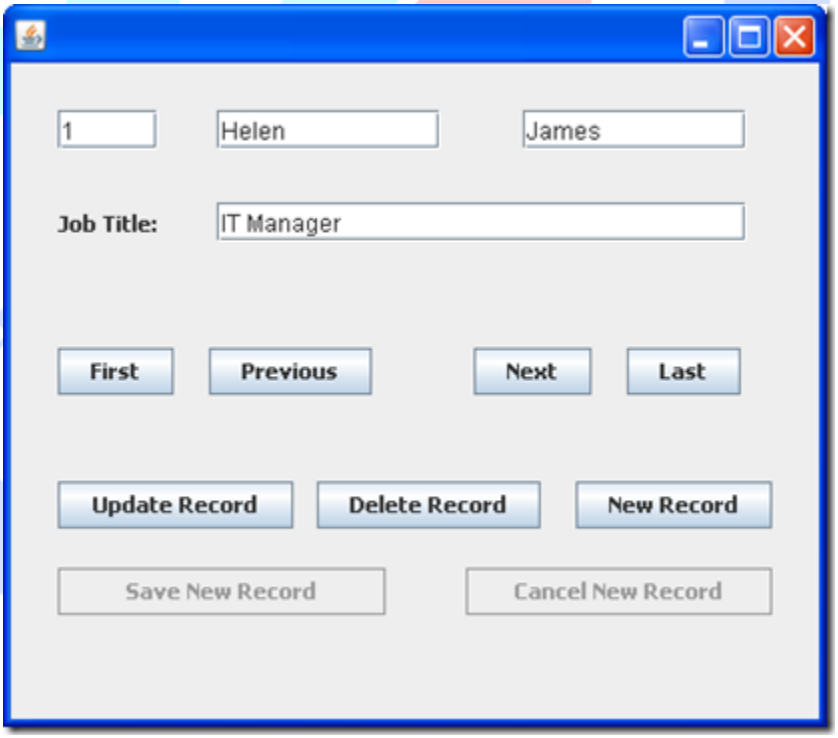
وقتی که افزودن کد مربوط به دکمه های New و Cancel را تمام کردید، برنامه ی خود را اجرا کرده و آن را امتحان کنید.

قبل از اینکه روی دکمه ی New Record کلیک کنید، فرم مانند زیر خواهد بود:

روی دکمه ی New Record کلیک کنید تا Text Fields پاک شده را مشاهده کنید:



با کلیک کردن روی دکمه یCancel ، داده را مجددا بارگذاری خواهید کرد:



اکنون که دکمه های New و Cancel تنظیم شده اند، می توانیم جلوتر برویم و داده ی وارد شده در فیلدهای متن را ذخیره کنیم.

آموزش ذخیره رکورد جدید در جاوا

قبل از اینکه بتوانید یک رکورد جدید را ذخیره کنید، باید که نشانگر را به بخشی به نام Insert Row حرکت دهید. این امر یک رکورد خالی در ResultSet ایجاد می کند. سپس داده را به ResultSet اضافه می کنید:

مثال

```
rs.moveToInsertRow( );
rs.updateInt("ID", newID);
rs.updateString("First_Name", first);
rs.updateString("Last_Name", last);
rs.updateString("Job_Title", job);
rs.insertRow( );
```

پس از افزودن داده بهResultSet ، آخرین خط یک ردیف جدید اضافه می کند.

به هرحال برای اعمال هرگونه تغییر در دیتابیس، آنچه ما انجام خواهیم داد بستن آبجکت Statement و آبجکت ResultSet می باشد. سپس می توانیم هرچیزی را مجددا بارگذاری کنیم. اگر این کار را انجام ندهیم، خطر اضافه نشدن رکورد جدید یا به ResultSet یا به دیتابیس وجود دارد. (این بستگی به نوع درایوری دارد که استفاده کرده ایم(.

برای بستن Statement یاResultSet ، باید فرمان close را انتشار دهید:

```
stmt.close( );
rs.close( );
```

کد مربوط به بارگذاری مجدد موارد، همان کدی است که در هنگام اولین بارگذاری فرم نوشتید:

```
stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);
String sql = "SELECT * FROM Workers";
rs = stmt.executeQuery(sql);
rs.next( );
```

```
int id_col = rs.getInt("ID");
String id = Integer.toString(id_col);
String first2 = rs.getString("First_Name");
String last2 = rs.getString("Last_Name");
String job2 = rs.getString("Job_Title");
textID.setText(id);
textFirstName.setText(first2);
textLastName.setText(last2);
textJobTitle.setText(job2);
```

در اینجا در واقع کار متفاوتی انجام نمی دهید: فقط همه ی رکوردها را مجددا انتخاب کرده و اولین مورد را در فیلدهای Text قرار می دهید.

در اینجا کدی را مشاهده می کنید که یک رکورد جدید در دیتابیس ذخیره می کند (واضح است که بیشترین قسمت این کد می تواند وارد متود خود شود.):

```
String first = textFirstName.getText();
String last = textLastName.getText();
String job = textJobTitle.getText();
String ID = textID.getText();
int newID = Integer.parseInt(ID);

try {
     rs.moveToInsertRow();

     rs.updateInt("ID", newID);
     rs.updateString("First_Name", first);
     rs.updateString("Last_Name", last);
     rs.updateString("Job_Title", job);

     rs.insertRow();

     stmt.close();
     rs.close();

     stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                ResultSet.CONCUR_UPDATABLE);
     String sql = "SELECT * FROM Workers";
     rs = stmt.executeQuery(sql);

     rs.next();
     int id_col = rs.getInt("ID");
     String id = Integer.toString(id_col);
     String first2 = rs.getString("First_Name");
     String last2 = rs.getString("Last_Name");
     String job2 = rs.getString("Job_Title");

     textID.setText(id);
     textFirstName.setText(first2);
     textLastName.setText(last2);
     textJobTitle.setText(job2);

     btnFirst.setEnabled(true);
     btnPrevious.setEnabled(true);
     btnNext.setEnabled(true);
     btnLast.setEnabled(true);
     btnUpdateRecord.setEnabled(true);
     btnDelete.setEnabled(true);
     btnNewRecord.setEnabled(true);
```

(مسئله ی دیگر این است که ستون ID باید منحصر به فرد باشد. به طور ایده آل، شما مسیری برای دریافت آخرین شماره ی ID و سپس اضافه کردن یک عدد به آن نوشته اید. دیتابیس های دیگر، مانندMySql ، دارای یک مقدار AutoIncrementمی باشد برای مراقبت از این موارد. فقط اطمینان حاصل کنید که مقدار ID مقداری نیست که قبلا استفاده کرده اید، در غیر اینصورت یک پیغام خطا دریافت خواهید کرد. یا مسیری برای دریافت یک ID منحصر به فرد بنویسید.)

برنامه ی خود را اجرا کرده و آن را امتحان کنید. اکنون قادر هستید که رکوردهای جدید را در دیتابیس خود ذخیره کنید.

در بخش بعدی در مورد حذف رکوردها فرا خواهید گرفت .

آموزش حذف رکورد در جاوا

حذف یک رکورد می تواند کار آسانی باشد: کافیست از متود deleteRow از ResultSet استفاده کنید:

```
rs.deleteRow( );
```

به هرحال Driver که در حال استفاده از آن هستیم، یک ردیف خالی در محل داده ای که حذف کرده، به جا می گذارد. اگر سعی کنید با استفاده از دکمه های Next و Previousبه آن ردیف حرکت کنید، فیلد متن ID دارای 0 در خود خواهد بود و همه ی فیلدهای دیگر خالی خواهدن بود.

برای حل این مشکل ابتدا یک ردیف را حذف خواهیم کرد و مجددا آبجکت Statementو آبجکت های ResultSetرا می بندیم. می توانیم همه ی داده ها را در فیلدهای متن مجددا بارگذاری کنیم. به این طریق هیچ ردیف خالی نخواهیم داشت.

در اینجا کدی برای افزودن دکمه ی Delete Record مشاهده می کنید:

```
try {
    rs.deleteRow();
    stmt.close();
    rs.close();
    stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                               ResultSet.CONCUR_UPDATABLE);
    String sql = "SELECT * FROM Workers";
    rs = stmt.executeQuery(sql);

    rs.next();
    int id_col = rs.getInt("ID");
    String id = Integer.toString(id_col);
    String first = rs.getString("First_Name");
    String last = rs.getString("Last_Name");
    String job = rs.getString("Job_Title");

    textID.setText(id);
    textFirstName.setText(first);
    textLastName.setText(last);
    textJobTitle.setText(job);
}
catch (SQLException err) {
    JOptionPane.showMessageDialog(Workers.this, err.getMessage() );
}
```

برنامه را اجرا کرده و آن را امتحان کنید. اکنون قادر هستید که رکوردها را از دیتابیس خود حذف کنید.

اکنون شما دانش پایه برای نوشتن یک برنامه دیتابیس در جاوا با استفاده از GUI را دارید. اگر مطالب این بخش را درک کردید، تبریک می گوییم .