آموزش برنامه نویسی

جاوا (JSE) در محیط نت بینز

کاری از مهدی حسن نژاد

hassannjad@gmail.com

تقدیم به :

جـــــــــــــامعه ی برنامه نویس

مقدمه

چیزی که جاوا سعی می‌کند حل کند و بسیار عالی از عهده آن بر آمده است، گسترش دادن حوزه‌های متفاوت است، می‌توانید برنامه‌های طرف کارگزار (Server) بنویسید، می‌توانید برای گوشی‌های موبایل برنامه بنویسید ، می‌توانید برنامه‌های علمی بنویسید،می‌توانید نرم‌افزار بنویسید، می‌توانید بین سیارات سفر کنید، همه کار...

جیمز گاسلینگ، پدر زبان جاوا

متاسفانه بعلت ناشناخته ماندن وتاکمی مبهم بودن این زبان حتی برای برنامه نویسان در حد متوسط گرایش به این زبان در کشور مابسیار کم رنگ می باشدو علت دیگر شاید نداشتن یک IDE یکتا وثابت برای این زبان است(اگر کمی وقت در اینترنت صرف یافتن IDE برای جاوا کنیدبه گزینه های زیادی برمی خورید Eclipse، Websphere ، JDeveloper ، Netbeans ، Intelli و... که خو گرفتند با هریک حتی برای برنامه نویسان حرفه ای وقت لازم دارد." *البته به نظر من این یک قوت است* "). من در این نوشتار قصد ندارم برتری جاوا بر بقیه زبانها را بگویم ویا تفاوت. NET با J2SE را بیان کنم بلکه آموزشی در حد مقدماتی بر این زبان خواهیم داشت و با نوشتن برنامه در محیط نت بین،با این محیط نیز اشنا خواهیم شد.

این نوشته به این امید گرداوری شده است که بتواند درارتقاء سطح علمی دوستان موثر واقع شود، اما هیچ تضمینی در مورد ان و یا درمورد مناسب بودن مطالب موجود برای اهداف خاص وجود ندارد.

امیدوارم این اموزش ،منبع فارسی خوبی برای اموزش جاوا و اشنایی با محیط نت بین باشد

بهمن ماه ۱۳۸۹

دانشگاه ازاد اسلامی- واحد سبزوار

فهرست

فصل اول – مقدمات جاوا

فصل دوم – انواع داده ها در زبان جاوا

فصل سوم – عملگرها و اولویت ها در زبان جاوا

فصل چهارم – الگوریتم ها، ساختارهای کنترلی و برنامه نویسی ساخت یافته در جاوا

فصل پنجم – اشنایی با کلاسها

فصل ششم – وراثت، بسته ها و رابطها

فصل هفتم – کلاسهای Swing

فصل هشتم – کار با MySQL در نت بینز

فصل نهم – ابزار گزارش گیری JasperReports

منابع و ماخذ

فهرست اشکال

فهرست جداول

فصل اول

مقدمات جـــاوا

۱–۱ مقدمه ای بر جاوا

در سال ۱۹۹۱ میلادی، شرکت Sun Microsystems پروژه ای تحت عنوان Green را آغاز نمود. هدف اصلی این پروژه ایجاد ابزار نرم افزاری جهت کنترل دستگاه هایی مانند set-top Box (وسیله ای جهت دسترسی به اینترنت)، PDA's (Personal Data Assistant) و... بود.

این ابزار نرم افزاری که در واقع همان زبان جاوا بود، در ابتدا با نام Oak[1] نام گذاری شد. علت این نام گذاری وجود درختان بلوط در محوطه اطراف ساختمان محل کار اعضای تیم Green بود.

تعداد اعضای تیم در آن زمان ۱۶ نفر بود که شاخص ترین آنها *James Gosling* ، *Bil Joy* ، *Patrick Naughton* بودند.(شکل ۱–۱)

شکل (۱–۱) : اعضای تیم زبان جاوا

***Green Team**. From left to right they are: Al Frazier، Joe Palrang، Mike Sheridan، Ed Frank، Don Jackson، Faye Baxter، Patrick Naughton، Chris Warth، James Gosling، Bob Weisblatt، David Lavallee، and Jon Payne.*
Missing in action: Cindy Long، Chuck Clanton، Sheueling Chang، and Craig Forrest

پس از تکمیل پروژه، مشکل بزرگی بر سر راه اعضای تیم قرار گرفت.

[1] بـلـوط

این مشکل آن بود که گروه فوق با وجود ایجاد یک تکنولوژی جدید قادر به همسو کردن بازار با اهداف خود نبودند و در نتیجه فروش محصولشان با موفقیت همراه نبود.

شکل (۱-۲) : *James Gosling*

در این زمان (حدود ۱۹۹۳) شرکت Sun حدود ۷۰ کارمند داشت. این مشکل سبب شد تا اعضای تیم و بخصوص James Gosling به فکر فرو روند که محصولی با ویژگی های Oak به چه کاری خواهد آمد؟

از ویژگی های Oak می توان به موارد زیر اشاره نمود:

Platform independent
Reliability
run media content on internet (a network with different divices)
...

پس از مدتی تحقیق و بررسی، اعضای گروه فهمیدند که با ورود به عرصه اینترنت و به کارگیری تکنولوژی ابداعی خود، به موفقیت خواهند رسید.

زیرا اینترنت در آن زمان به تازگی جایگاه ویژه خود را در بین کاربران عمومی باز کرده بود و روز به روز استفاده از آن عمومی تر می شد. (از تولد اینترنت تا آن زمان حدود ۲۰ سال می گذشت.)

Gosling در این زمینه می گوید:

همه‌ی چیزی که ما بدنبال آن بودیم، ایجاد امکانی جهت توزیع و اجرای برنامه ها تحت اینترنت بود.

اینترنت شبکه ای از کامپیوتر های مختلف از لحاظ سخت افزار و سیستم عامل می باشد.

سرانجام شرکت Sun در سال ۱۹۹۴ یک مرورگر اینترنتی با نام HotJava که توانایی اجرای apple های جاوا را داشت، بوجود آورد. (applet ها، برنامه هایی هستند که توسط مرورگر اینترنت قابل اجرا می باشند)

نکته قابل توجه دیگر آن است که Oak حدودا در همین زمان به Java تغییر نام داد.

انتخاب این نام نیز از بین یک سری کلمات منتخب و بصورت تصادفی صورت پذیرفت.

حرکت دیگری که Gosling و گروهش جهت توسعه محصول خود انجام دادند آن بود که کد جاوا را بصورت رایگان در اینترنت قرار دادند. این امر بخاطر رسیدن به دو هدف زیر صورت پذیرفت: سپس در سال ۱۹۹۵ اولین نسخه از زبان برنامه سازی جاوا (Java 1.0) در اختیار برنامه نویسان قرار گرفت.

جاوا از لحاظ syntax شبیه زبانهای قدرتمند C و C++ می باشد.

۱ -انتشار سریع جاوا

۲ -بررسی کد فوق توسط کارشناسان سرتاسر دنیا و رفع معایب موجود در آن.

این زبان تمامی ویژگی های شیءگرایی **C++** را نیز در خود مورد استفاده قرار داده است. این نکته نیز خود یکی از دلایل محبوبیت و پیشرفت سریعتر جاوا محسوب می شود.

زیرا زبانهای **C** و **C++** جزء محبوب ترین و قوی ترین زبانهای موجود محسوب می شوند و برنامه نویسان زیادی با این دو زبان چه در گذشته و چه امروزه آشنا می باشند. از این رو Gosling و گروهش برای یادگیری راحتتر و در نتیجه ترویج زبان خود در بین برنامه نویسان و همچنین قدرتمند نمودن محصول خود از ویژگی های مفید این دو زبان استفاده نمود.

علاوه بر دو زبان فوق، جاوا از خصوصیات زبان های دیگری مانند Smalltalk نیز استفاده نموده است.

با ورود نسخه اول زبان برنامه سازی جاوا و سپس نسخه های دیگر آن در سالهای بعد، این ابزار به سمت فعالیت-هایی مانند موارد زیر سوق پیدا نمود:

- On Line web stores
- Transactions Processing
- Database Interfaces
- Small platform such as cell phones، PDA،Smart Cards
- …

شعار Gosling پس از ارائه اولین نسخه جاوا این بود که:

“Write Once، Run Anywhere”

این شعار بیان کننده این مفهوم است که اگر برنامه مورد نظر خود را با زبان جاوا پیاده سازی نمایید (در هر محیطی که خواستید مانند ویندوز، لینوکس[1] و...) می توانید آنرا بدون تغییر، در سایر محیط ها اجرا کنید.

پنج هدف اصلی در زمان پیاده سازی جاو در نظر گرفته شده است:

۱ . از متدولوژی شیءگرایی در این زبان بطور کامل استفاده شود.

۲ . قابلیت اجرای برنامه های یکسان در سیستم عامل های مختلف

۳ . پشتیبانی از ویژگی های شبکه های کامپیوتری

۴ . اجرای راه دور برنامه ها بصورت کاملا امن

۵ . کاربرد راحت و ساده زبان به کمک انتخاب اجزاء مفید سایر زبانهای شیءگرا و استفاده از آن در جاوا (الگوبرداری)

اکنون با دانستن اهداف اصلی جاوا و تاریخچه آن، ممکن است سئوالی در ذهن خوانند این مقاله بوجود آید که چرا لوگوی مربوط به زبان برنامه سازی جاوا عکس یک فنجان قهوه است؟

جواب این سئوال بسیار ساده می باشد. علت انتخاب این لوگو آن است که گروه Gosling و خود او به قهوه بسیار علاقه مند می باشند. به همین خاطر لوگوی محصول خود را یک فنجان قهوه در نظر گرفته اند.

شکل (۱-۳) : لوگوی جاوا

[1] linux

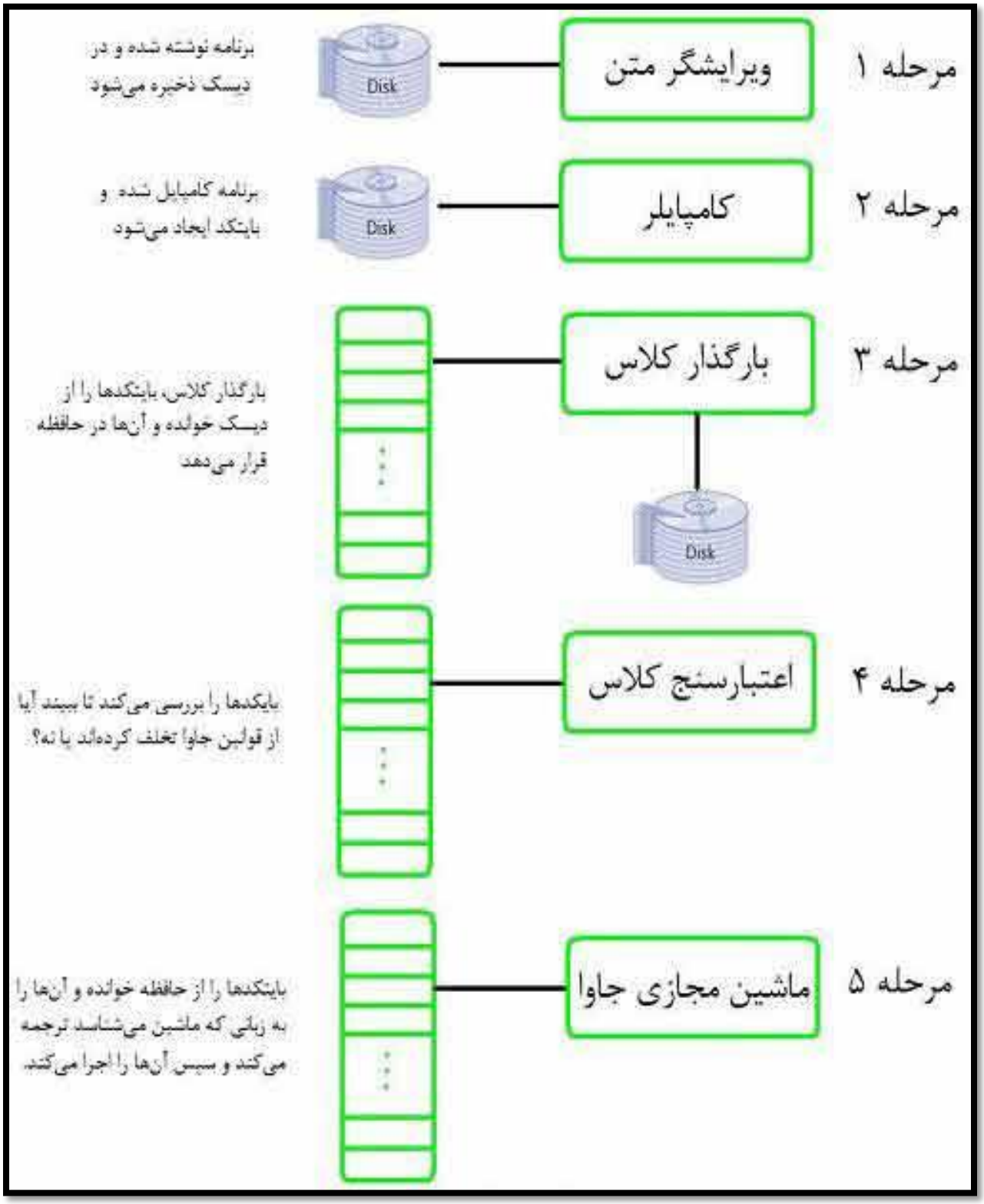
۱–۲ زبان جاوا را بهتر بشناسیم؟

زبان جاوا زبانی شیءگرا، مستقل از پلتفرم[1] و امن است که طراحی آن به گونه‌ای است که آموزش آن از C++ آسان‌تر و استفاده اشتباه از آن در مقایسه با C و C++ سخت‌تر است. شیءگرایی یکی ار شیوه‌های توسعه نرم‌افزار است. در این شیوه برنامه از ترکیب اشیا و ارتباطات بین اشیا ساخته می‌شود. درباره شیءگرایی در فصل‌های بعدی بیشتر خواهیم آموخت. مستقل بودن از پلتفرم (سکو) قابلیتی است که نرم‌افزار بدون تغییر بتواند در انواع پلتفرم‌ها اجرا شود. برنامه‌های جاوا پس از کامپایل تبدیل به «بایتکد» می‌شوند. هر ماشینی که ماشین مجازی جاوا را داشته باشد، می‌تواند «بایتکد»ها را اجرا کند. با استفاده از این ویژگی می‌توانید یک برنامه جاوا را در ویندوز نوشته و کامپایل کنید و بعد بایتکدهای تولید شده را در ویندوز، لینوکس، سولاریس و حتی در گوشی‌های موبایل مجهز به جاوا (Java Enabled) اجرا کنید. در این باره در همین فصل بیشتر خواهیم گفت.

۱–۲–۱ مراحل نوشتن و اجرای یک برنامه جاوا

در اینجا می‌خواهیم مراحل نوشتن و اجرای یک برنامه را به صورت مختصر توضیح دهیم. برای درک بهتر این مراحل به تصویر ۱-۴ دقت کنید:

[1] Platform



شکل (۱-۴) : مراحل اجرای یک برنامه

مرحله اول: برای شروع کار باید متن برنامه را بنویسید و آن را در یک فایل متنی ذخیره کنید. برای این کار می‌توانید از ویرایشگرهای متن مختلف استفاده کنید. فقط باید در هنگام ذخیره کردن فایل متن برنامه [۱] دو نکته را در نظر داشته باشید: اول این که متن برنامه را حتماً به صورت متن ساده (text) ذخیره کنید و دوم این که پسوند فایل java باشد.

مرحله دوم: در مرحله دوم کامپایلر جاوا فایل متن برنامه که در مرحله قبل تولید شده است را دریافت و آن را کامپایل می‌کند. در این مرحله کامپایلر جاوا فایلی به همان نام فایل متن برنامه و با پسوند class ایجاد می‌کند. این فایل حاوی بایتکد (دستوراتی که ماشین مجازی جاوا آن را می‌فهمد و می‌تواند آن را اجرا کند) است.

مراحل سه و چهار و پنج در هنگام اجرای برنامه جاوا اتفاق می‌افتند.

مرحله سوم: قبل از این که برنامه اجرا شود، برنامه‌ای به نام بارگذار کلاس (Class Loader) که یکی از اجزای اصلی ماشین مجازی جاوا است، اجرا شده و بایتکدها را از فایل class مورد نظر خوانده و آن‌ها را در حافظه اصلی قرار می‌دهد. در صورتی که بایتکدها نیاز به بایتکدهای کلاس‌های دیگر داشته باشند، بارگذار کلاس به تناسب سایر کلاس‌ها را نیز خوانده و آن‌ها را در حافظه قرار می‌دهد

مرحله چهارم: در این مرحله یکی دیگر از اجزای ماشین مجازی جاوا به نام اعتبارسنج کلاس [۲] فعال شده و بایتکدهایی را که در حافظه قرار گرفته‌اند بررسی می‌کند تا ببیند که آیا از قوانین جاوا تخلف کرده‌اند یا خیر؟ در صورتی که هیچ خطایی وجود نداشته باشد، برنامه برای اجرا آماده می‌شود.

مرحله پنجم: در این مرحله ماشین مجازی جاوا بایتکدهایی را که اعتبارسنجی شده و به تأیید رسیده‌اند به زبان ماشینی که بر روی آن قرار دارد ترجمه نموده و آن‌ها را اجرا می‌کند

فرایند ترجمه و اجرای برنامه‌های جاوا توسط ماشین مجازی جاوا فرایند پیچیده‌ای است

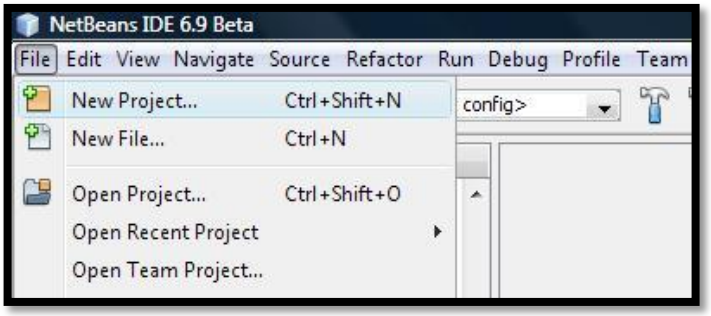
[1] Source Code [۲] Class Verifier

۱-۲-۲ تنظیم یک پروژه

برای ایجاد یک پروژه در NetBeans IDE 6.9 مراحل زیر را دنبال کنید:

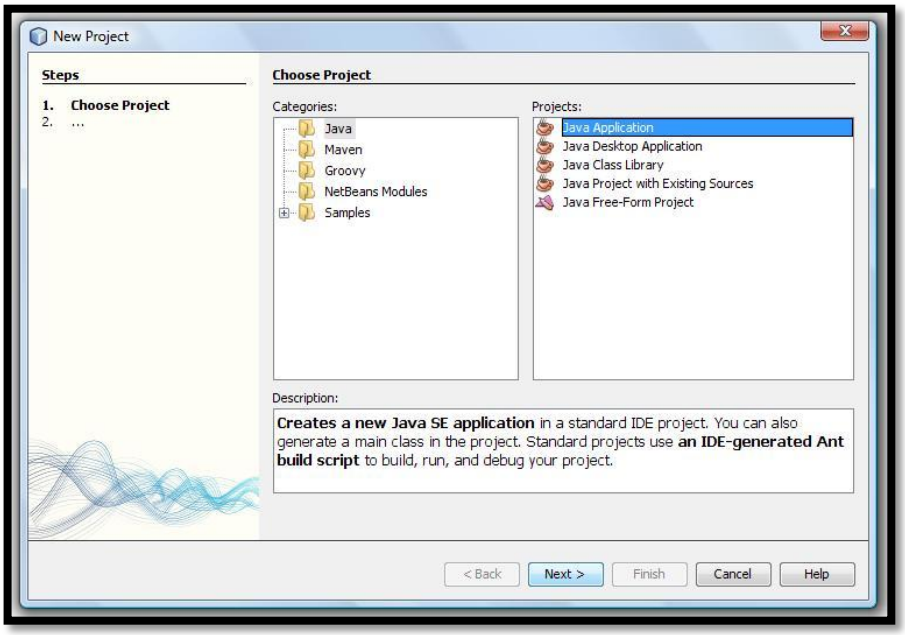
۱.برنامه jdk6.1 و Netbean6.9 را دانلود نموده و انها را نصب کنید و سپس برنامه NetBeans IDE 6.9 را اجرا کنید.

۲.در IDE از منوی File گزینه New Project را انتخاب کنید (شکل ۱-۵)



شکل (۱- ۵) : مرحله ۲ تنظیم پروژه

۳. در ویزارد New Project گزینه Java را باز کنید و از میان انواع پروژه‌های جاوا گزینه Java Application را همانند شکل زیر انتخاب کنید. (شکل ۱-۶)

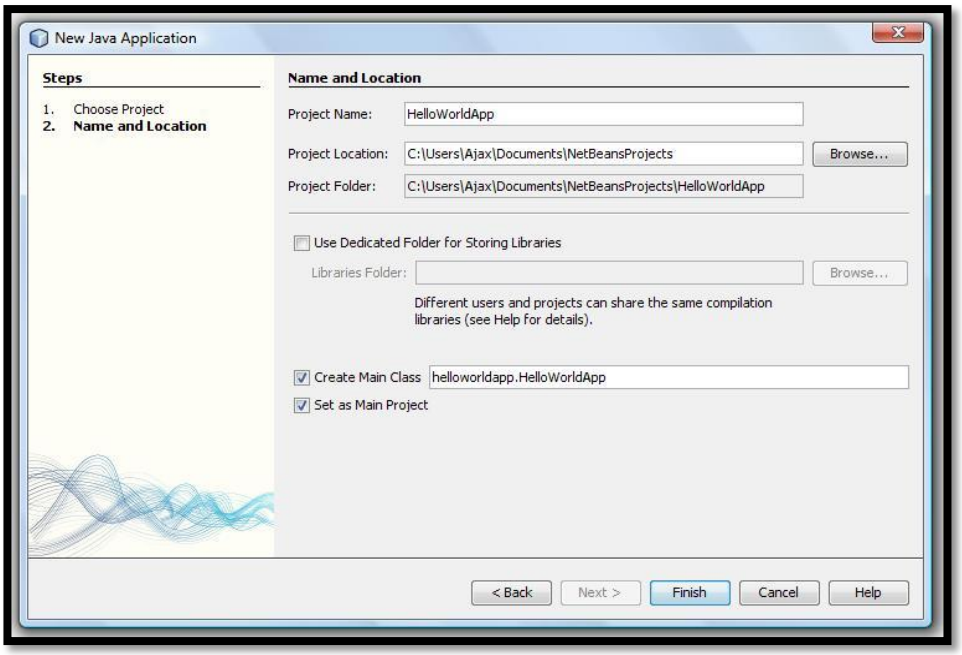


شکل (۱- ۶) : مرحله ۳ تنظیم پروژه

۴. در صفحه Name and Location از ویزارد، کارهای زیر را همانند تصویر زیر انجام دهید:

- در قسمت «Project Name» عبارت «HelloWorldApp» را وارد کنید.
- در قسمت «Create Main Class» عبارت «helloworldapp.HelloWorldApp» را وارد کنید.

«Set as Main Project» را انتخاب کنید. (شکل ۱-۷)



شکل (۱- ۷) : مرحله ۴ تنظیم پروژه

۵. بر روی دکمه «Finish» کلیک کنید.

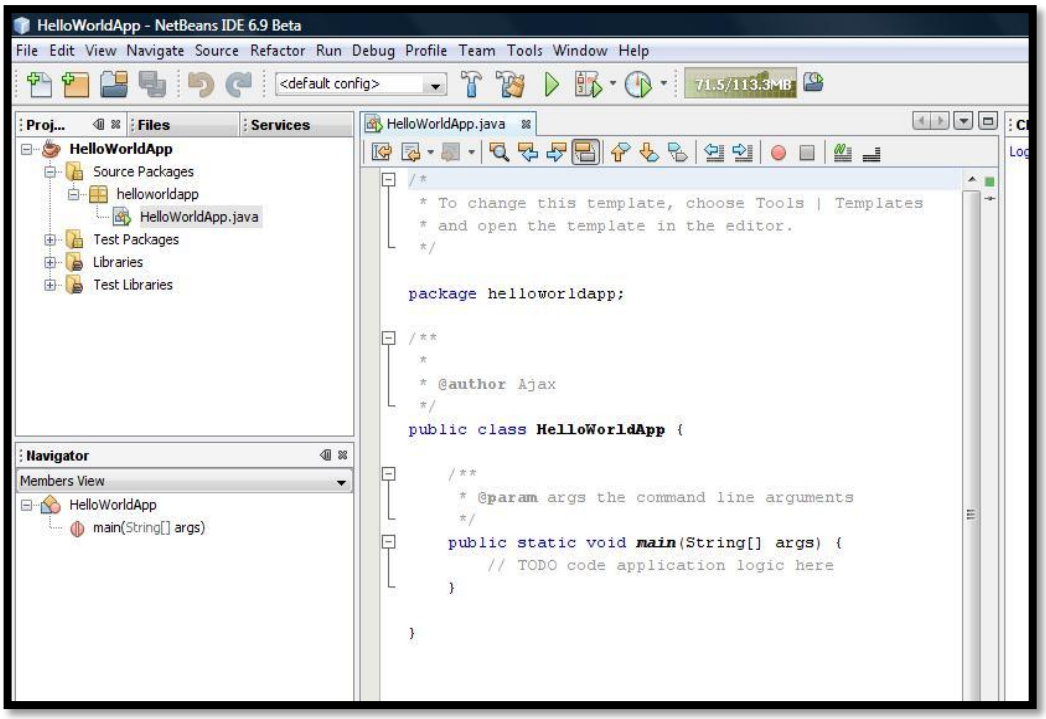
اکنون پروژه ساخته شده و در NetBeans IDE باز شده است. شما باید چندین مؤلفه (Component) را ببینید:

پنجره Projects که شامل نمایش درختی مؤلفه‌های پروژه از قبیل فایل‌های منبع، کتابخانه[1]هایی که پروژه به آن‌ها وابسته است و مانند آن‌ها است.

پنجره Editor که کدهای منبع در آن نمایش داده می‌شود، در حالی که فایلی به نام «HelloWorldApp» در آن باز است.

[1] Libraries

پنجره Navigator که به کمک آن می‌توانید به آسانی در میان اجزای کلاس انتخاب شده گردش کنید.



شکل (۱- ۸) : مرحله ۵ تنظیم پروژه

اضافه کردن کد به کدهای منبع تولید شده

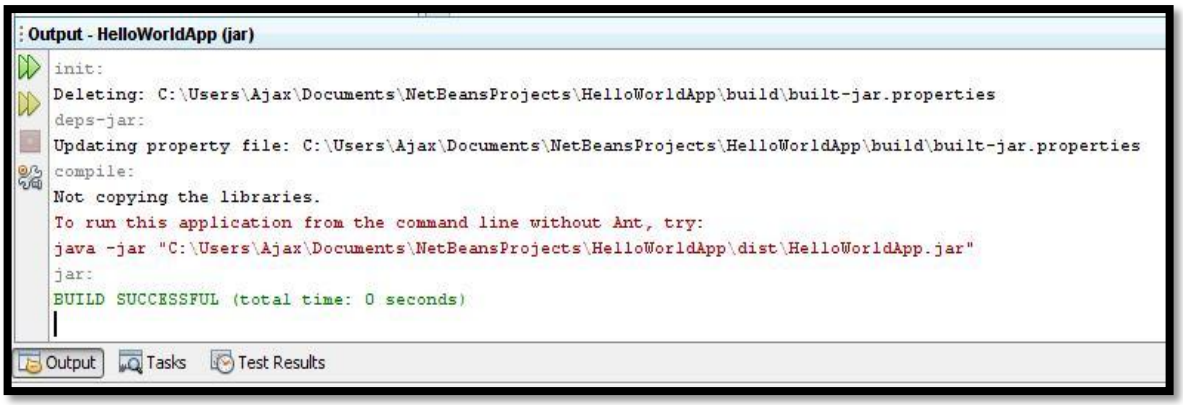
از آن جایی که شما گزینه «Create Main Class» را در ویزارد انتخاب کرده بودید، NetBeans IDE برای شما اسکلت کلاس را برای شما ایجاد می‌کند. برای نمایش پیام «Hello World» خط کد زیر را بنویسید

```
public class HelloWorldApp {
      public static void main(String[] args) {
            System.out.println("Hello World!");
      }
  }
```

حال با انتخاب File > save تغییرات فایل را ذخیره کنید.

کامپایل[1] کردن کد منبع برای کامپایل کردن برنامه خود از منوی اصلی NetBeans IDE منوی Run>Build Main Project و یا F11 را اجرا کنید. می‌توانید نتیجه اجرای Build را با انتخاب منوی Window > Output > Output ببینید.

پس از آن پنجره خروجی باز می‌شود و خروجی‌ای مشابه شکل ۱-۹ را به شما نشان می‌دهد:



شکل (۱- ۹) : کامپایل برنامه

اگر خروجی Build با عبارت «BUILD SUCCESSFUL» خاتمه یابد، به شما تبریک می‌گوییم!

با موفقیت برنامه خود را کامپایل کردید!

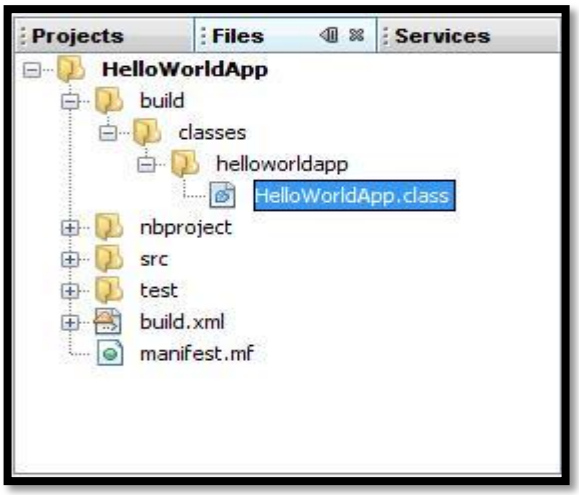
اگر خروجی build با عبارت «BUILD FAILED» خاتمه یابد، احتمالاً در کد خود یک خطای نگارشی[2] دارید. خطاهای نگارشی در پنجره خروجی (output window) به خط مربوطه در کد منبع پیوند (link) شده‌اند و با کلیک کردن بر روی آن‌ها مستقیماً به محل تولید خطا هدایت می‌شوید. پس از رفع خطا مجدداً پروژه را build کنید.

بعد از این که پروژه را build کردید، فایل‌های bytecode ساخته می‌شوند

برای مشاهده فایل .class ساخته شده، به پنجره فایل (Files) بروید و مسیر زیر را در درخت دنبال کنید

به تصویر ۱-۱۰ دقت کنید (Hello World App/build/classes/helloworldapp

[1] Compile [2] Syntax Error

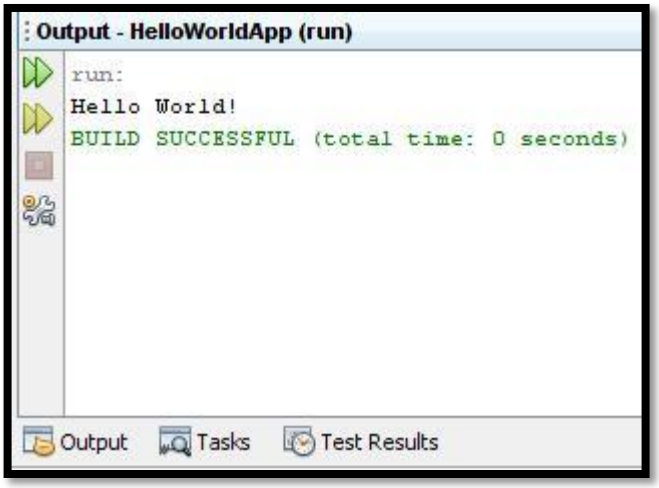


شکل (۱- ۱۰) : فایل های class.

حال که شما پروژه را build کردید، می‌توانید آن را اجرا کنید.

اجرای برنامه از نوار منوی NetBeans IDE منوی Run > Run Main Project را اجرا کنید(تصویر ۱-۱۱).

نمونه‌ای از آنچه باید ببینید را به شما نشان می‌دهد:



شکل (۱- ۱۱) : نتیجه اجرای برنامه

تبریک می‌گویم! پروژه شما کار می‌کند !

حالا شما می‌دانید که برخی از کارهای اساسی را چگونه در NetBeans IDE انجام دهید.

فصل ۲

انواع داده‌ها در زبان جاوا

۲–۱ انواع داده‌ها در زبان جاوا

در هر برنامه‌ای ما با تعدادی مقادیر سر و کار داریم. ممکن است برنامه با اعداد کار کند، یا با نویسه‌ها (کاراکترها). برای مثال فرض کنید می‌خواهیم در برنامه‌ای دو عدد مختلف را از ورودی گرفته و بر روی آن‌ها عملیات ریاضی انجام داده و نتیجه را در خروجی چاپ کنیم.

نام این برنامه را ماشین حساب ساده یا SimpleCalculator می‌نامیم:

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

حال باید برنامه را به صورتی تغییر دهیم تا بتواند دو عدد با نام‌ها و مقدارهای مختلف را ذخیره کند:

```
public class SimpleCalculator {
    public static void main(String[] args) {
        int first;
        int second;
    }
}
```

حال متد main() ما دارای دو متغیر به نام‌های first و second است.

در اصطلاح برنامه نویسی first و second را متغیر [۱] می‌نامند.

همانطور که تا به حال دیده‌اید هر متغیری شامل یک نام و یک نوع [۲] است. در مثال SimpleCalculator ما دو متغیر به نام‌های first و second که هر دو از نوع int هستند داریم.

هر متغیر علاوه بر نام و نوع، مقدار نیز دارد.

اکنون می‌خواهیم کلاس SimpleCalculator را به گونه‌ای تغییر دهیم که بتوانیم متغیرهای خود را مقداردهی کنیم:

```
import java.util.Scanner;
public class SimpleCalculator {
```

[۱] Variable [۲] type

```
    public static void main(String[] args) {
        int first;
        int second;

        Scanner input = new Scanner( System.in );

        System.out.print("Enter first integer: ");
        first = input.nextInt();

        System.out.print("Enter second integer: ");
        second = input.nextInt();
    }     }
```

فعلاً به کلاس Scanner کاری نداریم. همین قدر بدانید که برای خواندن ورودی‌های برنامه از متدهای این کلاس استفاده می‌کنیم.

پس از اجرای برنامه فوق، در کنسول خروجی عبارت Enter first integer: نمایش داده شده و شما می‌توانید جلوی آن مقدار مورد نظر خود را وارد کنید.

سپس خط بعدی با عبارت Enter second integer: نمایش داده می‌شود که می‌توانید مقدار مورد نظرتان برای متغیر second را وارد کنید:

```
import java.util.Scanner;

 public class SimpleCalculator {

     public static void main(String[] args) {
         int first;
         int second;

         Scanner input = new Scanner( System.in );

         System.out.print("Enter first integer: ");
         first = input.nextInt();

         System.out.print("Enter second integer: ");
         second = input.nextInt();

         System.out.println("first + second = " + ( first + second ) );
         System.out.println("first - second = " + ( first - second ) );
         System.out.println("first * second = " + ( first * second ) );
         System.out.println("first / second = " + ( first / second ) );
    }
 }
```

خروجی برنامه

```
Enter first integer: 10
Enter second integer: 15
first + second = 25
```

```
first - second = -5
first * second = 150
first / second = 0
```

در چهار خط آخر برنامه حاصل جمع، تفریق، ضرب و تقسیم دو عدد first و second را نمایش می‌دهیم.

هدف هر برنامه‌ی رایانه‌ای، محاسبه داده‌ها است. برای این کار هر برنامه‌ای باید بتواند داده‌ها را وارد کند، آن‌ها را پردازش کند و نتیجه را به نحو مقتضی نمایش دهد. برای انجام محاسبات بر روی داده‌ها در برنامه باید بتوانیم داده‌ها را شناسایی کنیم. باید بدانیم آن‌ها از چه نوعی هستند .همچنین باید بدانیم که در هر لحظه هر کدام از آن‌ها چه مقداری دارند. همانطور که در برنامه قبلی دیدید برای هر متغیر سه ویژگی قائل می‌شویم: نوع، نام و مقدار. الگوی کلی تعریف متغیرها (اعلان[1] نیز گفته می‌شود) در زبان جاوا به صورت زیر است:

```
Type name;
```

پس از تعریف یک متغیر باید بتوانیم مقداری را به آن نسبت دهیم. برای این کار از الگوی زیر استفاده می‌کنیم

```
Type name = Value;
```

اکنون چند مثال از تعریف و مقداردهی به متغیرها:

`int a = 5;`	**متغیری از نوع صحیح و با مقدار اولیه ۵**
`char someChar = 'A';`	**متغیری از نوع نویسه (کاراکتر) و با مقدار اولیه A**
`float PI = 3.14;`	**متغیری از نوع اعشاری و با مقدار اولیه ۳.۱۴**
`String hello = "Hello Java";`	**متغیری از نوع رشته‌ای (String) و با مقدار اولیه «Hello Java»**

جدول (۲-۱) : چند مثال از تعریف و مقدار دهی به متغیر ها

[1] Declaration

پس از آشنایی مقدماتی با متغیرها و نحوه تعریف و مقداردهی به آن‌ها آشنا شدیم، می‌توانیم فهرست کامل نوع داده‌های اولیه[1] را ببینیم :

نوع متغیر	محدوده مقادیر مجاز	توضیحات
Boolean	TrueوFalse	مناسب برای متغیرهای منطقی که همواره یکی از دو مقدار درست یا غلط دارند
Char	از صفر یونیکد تا 216-1 یونیکد	مناسب برای انواع متغیرهای حرفی. با توجه به این که متغیرهای حرفی در جاوا یونیکد ۱۶ بیتی هستند، از آن‌ها می‌توان برای کلیه حروف کلیه زبان‌ها (از جمله فارسی) استفاده کرد .
Byte	از ۱۲۸- تا ۱۲۷	مناسب برای متغیرهای عددی صحیح که در محدود مقادیر مجاز صحیح قرار دارند.
Short	از 215- تا 215-1	مناسب برای متغیرهای عددی صحیح که در محدود مقادیر مجاز صحیح (حدود منفی سی و دو هزار تا مثبت سی و دو هزار) قرار دارند.
Int	از 231- تا 231-1	مناسب برای متغیرهای عددی صحیح که در محدود مقادیر مجاز صحیح (حدود منفی دو میلیارد تا مثبت دو میلیارد) قرار دارند.
Long	از 263- تا 263-1	مناسب برای متغیرهای عددی صحیح بسیار بسیار بزرگ!
Float	استاندارد IEEE754تقریباً از - 3.4E38 تا ۳۸E۳.۴ با ۸ رقم دقت اعشاری	مناسب برای متغیرهای اعشاری با دقت خوب برای محاسبات معمولی
Double	استاندارد IEEE754 تقریباً از - 1.8E308 تا 1.8E308 با ۱۶ رقم دقت اعشاری	مناسب برای متغیرهای اعشاری با دقت بسیار زیاد برای محاسبات با دقت مضاعف
Void	---	برای متغیرهایی که هیچ نوعی ندارند (بعداً درباره نوع void بیشتر خواهیم گفت)

جدول (۲-۲) : فهرست نوع داده اولیه

۲

[1] Primitive

۲-۲-۱ مقادیر لفظی یا لیترال[1]

در جاهایی از برنامه‌های جاوا ممکن است بخواهید مستقیماً از مقادیر استفاده کنید. عبارت زیر را در نظر بگیرید:

```
int a = 5;
```

در مثال فوق عدد ۵ اصطلاحاً متغیر لفظی یا لیترال است. کامپایلر جاوا برای متغیرهای لفظی تعدادی پیش‌فرض دارد.

برای مثال متغیرهای لفظی صحیح مانند مثال فوق را به صورت پیش‌فرض از نوع int می‌شناسد. حال اگر بخواهید عدد ۵ را در یک دستور انتساب با استفاده از عملگر = به یک متغیر از نوع long نسبت دهید باید به صورت زیر عمل کنید:

```
long a = 5L;
```

حرف L که بلافاصله بعد از عدد ۵ آمده است به کامپایلر جاوا می‌فهماند که متغیر لفظی ۵ از نوع long است.

همچنین پیش‌فرض متغیرهای لفظی اعشاری در جاوا از نوع double است. دستور زیر را در نظر بگیرید:

```
float a = 3.14;
```

آیا به نظر شما این دستور کامپایل خواهد شد؟ آن را در یکی از مثال‌هایی که تاکنون نوشته‌اید امتحان کنید.

همانطور که حدس زدید و احتمالاً حدس خود را آزمودید، کامپایلر جاوا در هنگام اجرای دستور فوق خطا می‌گیرد.متغیر لفظی 3.14 را کامپایلر جاوا به صورت پیش‌فرض از نوع double فرض می‌کند.

وقتی می‌خواهد مقدار آن را در یک متغیر اعشاری از نوع float قرار دهد، دقت عدد اعشاری از نوع double باید به float کاهش یابد و این از دید کامپایلر جاوا یک خطا است.

[1] literal

برای رفع این مشکل باید به صراحت به کامپایلر جاوا بگویید که ۳.۱۴ یک متغیر لفظی اعشاری از نوع float است. برای این کار مشابه دستور زیر عمل می‌کنیم:

```
float a = 3.14F;
```

حرف F که بلافاصله بعد از ۳.۱۴ آمده است کامپایلر جاوا را مجبور می‌کند که متغیر لفظی ۳.۱۴ را از نوع اعشاری float در نظر بگیرد.

۲-۲-۲ متغیرهای رشته‌ای

همیشه متغیرهای ما از انواع عددی نیستند. در اغلب موارد، ما نیاز داریم که حروف و کلمات را در برنامه خود وارد کنیم، آن‌ها را پردازش کنیم و نتیجه‌ای را به صورت یک کلمه یا جمله نمایش دهیم .

برای مثال فرض کنید برنامه‌ای داریم که نام یک دانش‌آموز را از ورودی گرفته و نمره وی را نمایش می‌دهد. ما نیاز به متغیری داریم که بتوانیم نام دانش‌آموز را در آن ذخیره کنیم. به این نوع متغیرها رشته[1]ی گفته می‌شود. زبان جاوا دارای امکانات گسترده‌ای برای کار با رشته‌ها است که در فصول مختلف این کتاب با آن‌ها آشنا خواهیم شد.

برای تعریف یک متغیر رشته‌ای به صورت زیر اقدام کنید:

```
String name;
```

البته همانند سایر انواع متغیرها در زبان جاوا، می‌توان همزمان با تعریف یک متغیر رشته‌ای در جاوا، آن را مقداردهی اولیه نیز نمود:

```
String name = "Some text";
```

عبارت‌های رشته‌ای همواره در بین دو علامت " قرار می‌گیرند.

[1] String

۲-۳ کلاس‌های پوشاننده

تمام انواع داده که تا به حال بررسی کردیم، انواع داده اولیه[1] هستند.

همانطور که گفتیم در جاوا هر چیزی یک شیء است. انواع داده اولیه تنها چیزهایی در جاوا هستند که شیء نیستند.

البته هر کدام از انواع داده اولیه یک کلاس پوشاننده [2] دارد که در جاهایی که باید حتماً از اشیا استفاده کنیم، می‌توانیم به جای متغیرهای از نوع اولیه، از آن‌ها استفاده کنیم. برای مثال برخی از ساختمان داده‌ها در جاوا مانند Vector[3] به شما اجازه نمی‌دهد انواع داده اولیه را در آن بریزید.

برای حل این مشکل می‌توان از کلاس‌های پوشاننده انواع داده اولیه استفاده کرد. در جدول زیر فهرست این کلاس‌ها را می‌بینید:

نوع داده اولیه	کلاس پوشاننده	توضیحات و مثال‌ها	
boolean	**Boolean**	**ایجاد یک شیء Boolean با مقدار درست**	`Boolean a = new Boolean (true);`
		انتساب مقدار یک شیء Boolean به یک متغیر Boolean	`boolean b = b.booleanValue();`
char	**Character**	**ایجاد یک شیء از نوع Characterو انتساب مقدار 'A' به آن:**	`Character c = new Character('A');`
		انتساب مقدار یک شیء Character به یک متغیر از نوع:char	`char d = c.charValue();`
Byte	**Byte**	**ایجاد یک شیء از نوع Byte و انتساب مقدار ۵۰ به آن**	`Byte e = new Byte( (byte) 50 );`
		انتساب مقدار یک شیء Character به یک متغیر از نوع char	`byte f = e.byteValue();`
Short	**Short**	**ایجاد یک شیء از نوع Short و انتساب مقدار ۵۰ به آن**	`Short g = new Short( (short) 50 );`

[1] primitive
[2] Wrapper
[3] بردار

		انتساب مقدار یک شیء **Short** به یک متغیر از نوع **short**	`short h = g.shortValue();`
Int	**Integer**	ایجاد یک شیء از نوع **Integer** و انتساب مقدار ۵۰ به آن	`Integer i = new Integer(50);`
		انتساب مقدار یک شیء **Integer** به یک متغیر از نوع **int**	`int j = i.intValue();`
Long	**Long**	ایجاد یک شیء از نوع **Long** و انتساب مقدار **۵۰L** به آن	`Long k = new Long(50L);`
		انتساب مقدار یک شیء **Long** به یک متغیر از نوع **long**	`long l = k.longValue();`
Float	**Float**	ایجاد یک شیء از نوع **Float** و انتساب مقدار ۵۰ **F** به آن.	`Float m = new Float(50F);`
		انتساب مقدار یک شیء **Float** به یک متغیر از نوع **float**	`float n = m.floatValue();`
double	**Double**	ایجاد یک شیء از نوع **Double** و انتساب مقدار ۵۰ به آن	`Double o = new Double(50.0);`
		انتساب مقدار یک شیء **Double** به یک متغیر از نوع **Double**	`double p = o.doubleValue();`

جدول (۲-۳) : کلاسهای پوشاننده

متغیرهایی که نمی‌توان مقدار آن‌ها را تغییر داد:

تعریف فوق به اندازه کافی خود متناقض است.

متغیر یعنی چیزی که تغییر می‌کند، پس متغیری که نتوان مقدار آن را تغییر داد بی‌معنی است. ولی اگر بازی با کلمات را رها کنیم، می‌بینیم که در مواقع زیادی در برنامه‌ها ممکن است متغیرهایی را تعریف کنیم که تمایلی نداریم در حین اجرای برنامه، خواسته یا ناخواسته مقدار آن‌ها تغییر کند.

برای مثال فرض کنید در محاسبات ریاضی، از عدد پی (π) استفاده می‌کنیم. مقدار این متغیر در طول اجرای برنامه نباید تغییر کند.

برای این کار ما این متغیر را ثابت و نهایی تعریف می‌کنیم:

```
final double PI = 3.14;
```

پس از تعریف متغیر PI به شکل فوق دیگر نمی‌توانیم در حین اجرای برنامه مقدار آن را تغییر دهیم:

```
final double PI = 3.14;
PI = 25; // Compiler error
PI = a; // Compiler error
```

در صورت اجرای مثال فوق کامپایلر جاوا از شما خطا گرفته و برنامه را کامپایل نمی‌کند. کلمه کلیدی final برای اعلام ثابت و تغییر ناپذیر بودن یک متغیر به کار می‌رود. البته این کلمه کلیدی کاربردهای دیگری هم دارد که در جای مناسب آن‌ها را تشریح خواهیم کرد.

۲-۴ حوزه متغیرها

متغیرها فقط در همان حوزه‌ای که تعریف شوند، معتبرند. برای مثال اگر متغیری را درون یک متد تعریف کنیم،در متدهای دیگر به آن دسترسی نداریم و نمی‌توانیم مقدار آن را ببینیم یا آن را تغییر دهیم.

به مثال زیر دقت کنید:

```
public class VariableScopeTest {

     public void firstMethod() {
         int myNumber = 5;
     }

     public void someMethod () {
         myNumber = 10; //?
      }
 }
```

در این مثال متغیری به نام myNumber در متد firstMethod تعریف و مقداردهی شده است. با وجود این که هر دو متد firstMethod و someMethod هر دو متعلق به یک کلاس هستند، در متد someMethod ما هیچ دسترسی به متغیر myNumber نداریم.

در صورتی که بخواهید کلاس فوق را کامپایل کنید، کامپایلر از شما خطا می‌گیرد.

علت آن کاملاً واضح است: متغیری که در یک متد تعریف شده است، متعلق به آن متد بوده و در بیرون آن وجود ندارد. بنابراین در متدهای دیگر به آن دسترسی نداریم.

حوزه[1] متغیرها در زبان جاوا به صورت زیر تقسیم‌بندی می‌شود:

حوزه کلاس: ویژگی‌های کلاس، متغیرهایی هستند که در سرتاسر یک کلاس قابل دسترسی‌اند. برای مثال اگر یک متغیر در حوزه کلاس باشد، همه متدهای آن کلاس به آن دسترسی دارند و هر کدام از متدها که آن متغیر را تغییر دهند، مقدار آن در سایر متدها نیز تغییر می‌کند

حوزه متد: این دسته متغیرهایی هستند که در یک متد تعریف شده‌اند. این متغیرها فقط در همان متدی که تعریف شده‌اند قابل دسترسی‌اند.

حوزه بلوک: برخی اوقات متغیرها را درون حلقه‌های تکرار یا برخی دیگر از ساختارهای شرطی درون یک متد تعریف می‌کنیم. این متغیرها خارج از آن بلوک قابل دسترسی نیستند.

به‌مثال زیر دقت کنید :

```
public class VariableScopeTest {

    public void someMethod () {
        for( int i = 0 ; i < 10 ; i++ ) {
            // Do something
        }

        // Now what is i's value?
        System.out.println( "i = " + i );
    }
}
```

در مثال بالا، متغیر i در حلقه تکرار for تعریف شده است. در زمانی که می‌خواهیم مقدار i را در خروجی چاپ کنیم، با خطا مواجه می‌شویم.

علت این است که حوزه متغیر i فقط همان حلقه تکرار است و در خارج از آن دیگر متغیر i وجود ندارد.

مسأله مهمی که درباره حوزه متغیرها نیاز به توضیح دارد، مسأله همنامی متغیرها است. آیا می‌توان دو متغیر با یک نام داشت؟ بستگی دارد!

[1] Scope

اگر متغیرهای همنام در حوزه‌های متفاوت باشند، این امر امکان‌پذیر است. برای مثال اگر یک کلاس دارای یک ویژگی[1] به نام someInt باشد و یکی از متدهای آن کلاس هم یک متغیر به همان نام داشته باشد، هیچ خطایی وجود ندارد.

فقط نکته مهم در این است که متغیرهای همنام در همان حوزه‌ای که تعریف شده‌اند معتبرند. به مثال زیر دقت کنید:

```
public class VariableScopeTest {

     int someInt;

     public void someMethod () {
          int someInt;

          someInt = 5; // ?
     }
 }
```

در مثال فوق، متغیری به نام someInt در کلاس VariableScopeTest تعریف شده و جزو ویژگی‌های آن کلاس است.

در متد someMethod هم باز متغیری به نام someInt تعریف شده است.

حال اگر در متد someMethode مقدار ۵ را به متغیر someInt نسبت دهیم چه اتفاقی می‌افتد؟ آیا متغیر تعریف شده در تابع تغییر می‌کند یا ویژگی آن کلاس؟

برای پاسخ دادن به این سؤال باید به این نکته توجه داشت که اولویت با حوزه‌ای است که متغیر در آن تعریف شده است. بنابراین اگر مقدار ۵ را به someInt نسبت دهیم، متغیری که در متد فوق تعریف شده است تغییر می‌کند نه ویژگی کلاس به همان نام.

حال اگر بخواهیم در متد someMethod ویژگی کلاس به نام someInt را تغییر دهیم چه باید بکنیم؟ برای این کار کافی است از نام کلاس استفاده کنیم. به مثال زیر دقت کنید:

```
public class VariableScopeTest {

     int someInt;
```

[1] Attribute

```
    public void someMethod () {
        int someInt;
        someInt = 5; // Method variable
        VariableScopeTest.someInt = 7; // class variable
    }
}
```

۲–۵ آرایه‌ها

فرض کنید که می‌خواهید نمره‌های دانشجویان یک کلاس را گرفته و میانگین آن‌ها را محاسبه کنید. تعداد دانشجویان کلاس ۲۵ است. آیا 25 متغیر از نوع float تعریف می‌کنید؟ اگر تعداد دانشجویان کلاس ۵۰ نفر بود چه کار می‌کردید؟ برای ۲۰۰ نفر چه می‌کردید؟ همانطور که حدس می‌زنید راه‌حل، استفاده از تعداد زیادی متغیر نیست.

اگر می‌توانستیم تعداد زیادی متغیر همنوع را با یک نام ذخیره کنیم و با استفاده از یک اندیس به آن‌ها دسترسی داشته باشیم مشکل حل می‌شد.

مثلاً می‌گفتیم فهرست نمره‌های دانشجویان که شامل ۲۵ متغیر float است و بعد می‌گفتیم نمره دانشجوی اول ۱۵ و دانشجوی دوم ۱۶ و... است. به چنین نوع داده‌ای آرایه گفته می‌شود.

آرایه[1] مجموعه‌ای از متغیرها است که عنصر[2] یا جزء[3] نامیده می‌شوند و همگی از یک نوع (type) هستند. یک آرایه با چند چیز شناخته می‌شود: نام آن، تعداد متغیرهایی که نگهداری می‌کند که طول آرایه نامیده می‌شود و نوع متغیرهایی که آرایه در خود نگه می‌دارد.

بنابراین برای تعریف یک آرایه که نمره‌های دانشجویان را نگهداری کند به شکل زیر عمل می‌کنیم:

```
float[25] grades;
```

به اجزای تعریف فوق دقت کنیدfloat .نوع آرایه را مشخص می‌کند. در واقع می‌گوید که آرایه فوق عناصری از نوع floatرا نگهداری می‌کند. [۲۵] اعلام می‌کند که طول آرایه ۲۵ است.

[1] Array
[2] Element [3]Component

[] نشان دهنده آرایه است و عدد صحیحی که درون آن قرار می‌گیرد طول آرایه را مشخص می‌کند. grades هم که نام آرایه است. حال اگر بخواهیم عناصر این آرایه را مقداردهی کنیم به شکل زیر عمل می‌کنیم:

```
grades[0] = 12.3f;
grades[1] = 15.5f;
...
grades[24] = 16f;
```

همانطور که در مثال‌های بالا می‌بینید برای مقدار دهی به عناصر آرایه از نام آرایه به همراه اندیس عنصر استفاده می‌کنیم .

۲-۵-۱ آرایه ای از نوع رشته ای

برای انکه بخواهیم ارایه ای از نوع رشته داشته باشیم میتوانیم به دو صورت تعریف داشته باشیم.

```
String[] strarr1= new String[5];
String[] strarr2={"ali","reza","sadegh","hesam","mehdi"};
```

هر دو متغیر ارایه ای با طول ۵ میباشند،با این تفاوت که اگر مقادیر ارایه مشخص باشد میتوانیم از روش دوم استفاده کنیم.

میتوان توسط متد length ،که یکی از متدهای متغیر ارایه میباشد،طول ارایه را بدست اورد.

همچنین میتوان دو ارایه را بهمدیگر نسب داد.دراین صورت متغیر ارایه سمت چپ به جایی اشاره میکند که متغیر سمت راست وجود دارد.

به مثال زیر دقت کنید.

```
 public class ArrayApp {
     public static void main(String[] args) {
         int[] arr1;
         int[] arr2={15,2,3,4,5};
         arr1=arr2;
         System.out.println("length array1 : "+arr1.length);
         System.out.println("value  array1 : "+arr1[0]);
          }
}
```

خروجی برنامه

```
length array1 : 5
value  array1 : 15
```

در مثال فوق دستور println() اول طول ارایه و دومی مقدار خانه اول ارایه را نشان میدهد.

۲-۶ قواعد نامگذاری متغیرها در جاوا

نامگذاری متغیرها در جاوا دارای هیچ محدودیتی نیست

متغیرها با حروف (a-z) و (A-Z) و خط زیر (_) آغاز می‌شوند و بعد از آن هر تعداد حرف یا عدد می‌تواند بیاید .

چند مثال از نام‌های غلط برای متغیرها در جاوا:

```
int 1i; // Wrong name
String #name = "My Name"; // Wrong name
```

البته برای نامگذاری متغیرها در زبان جاوا قواعدی وجود دارد که اکثر برنامه‌نویسان جاوا آن‌ها را پذیرفته‌اند و البته به عنوان الگوی[1] صحیح نویسی در جاوا توصیه می‌شود.

قسمتی از این قواعد این‌ها است:

۱ - نام متغیرها با حرف کوچک شروع می‌شود. در صورتی که نام متغیر بیشتر از یک کلمه بود، کلمات به هم می‌چسبند و حرف ابتدایی کلمات دوم و بعد از آن با حروف بزرگ آغاز می‌شود. این شیوه نامگذاری به کوهان شتری[2] معروف است. چند مثال از نام‌های خوب و بد:

```
int number;           // Good name!
int second_number;    // Bad name!
int secondNumber;     // Good name!
String Name;          // Bad name!
String name;          // Good name!
float _average;       // Bad name!
```

۲ - متغیرهای ثابت معمولاً با حروف بزرگ نوشته شده و در صورتی که تعداد کلمات بیشتر از یک کلمه باشد، با خط زیر (_) از هم جدا می‌شوند:

```
double PI = 3.14;
String URL = "http://www.HowToProgram.ir";
String DB_URL = "jdbc:mysql://127.0.0.1:3306/testDB";
```

[1] pattern
[2] Camel case

فصل سوم

عملگرها و اولویت‌ها در زبان جاوا

۳-۱ عبارت‌ها[1]

تاکنون قطعاً عبارت‌های ریاضی فراوانی را دیده‌اید. برای مثال می‌دانید که ۲+۲ یک عبارت ریاضی است. در زبان‌های برنامه نویسی هم چنین عبارت‌هایی وجود دارند.

اگر بخواهیم دقیق‌تر باشیم باید بگوییم که هر جمله‌ای که دارای یک**ارزش** باشد، یک «**عبارت**» یا expression است که البته همانطور که حدس زده‌اید یکی از رایج‌ترین انواع عبارت‌ها، عبارت‌های ریاضی است. چندین عبارت ریاضی:

```
int x = 5;
int y = x;
int z = x * y;
```

همه جملات بالا «**عبارت**» هستند زیرا هر کدام آن‌ها دارای یک مقدار است: در خط اول مقدار متغیر لفظی ۵ به متغیری از نوع int به نام x نسبت داده می‌شود، در خط دوم مقدار متغیر x در متغیری به نام y قرار داده می‌شود و در خط سوم مقدار حاصل ضرب دو متغیر x و y در متغیر z قرار می‌گیرد. بنابراین ۵، xو x * y هر کدام یک عبارت ریاضی هستند.

مقدار یک عبارت، مقدار بازگشتی[2] نامیده می‌شود.

برای مثال اگر x متغیری از نوع int و دارای مقدار اولیه ۳ و y نیز متغیری از نوع int و دارای مقدار اولیه ۵ باشد، آنگاه مقدار بازگشتی عبارت x * y برابر با حاصل ضرب دو عدد ۳ و ۵ یعنی ۱۵ است.

[1] Expression [2] return value

به مثال زیر دقت کنید:

```
int x = 3;
int y = 5;
int z = x * y;
// If you print value of z, you will see that the value is: 15 (= 3 * 5)
System.out.println( "z value:" + z );
```

همانطور که می‌دانید در اغلب عبارت‌های ریاضی ما از نمادهایی مانند * (ضرب)، + (جمع) و مانند آن استفاده می‌کنیم. در اصطلاح برنامه نویسی به این نمادهای ویژه، عملگر [1] می‌گویند. عملگرها نمادهایی هستند که برای محاسبات ریاضی و منطقی[2] از آن‌ها استفاده می‌شود

۳-۲ عملگرهای ریاضی[3]

برای انجام محاسبات ریاضی در جاوا (java) از پنج عملگر جمع (+)، تفریق (-)، ضرب(*) ، تقسیم (/) و باقیمانده تقسیم (%) استفاده می‌کنیم.

در جدول زیر فهرست این عملگرها را به همراه مثال‌هایی از کاربرد آن‌ها خواهید دید:

نام عملگر	معنی	مثال	مقدار بازگشتی عبارت مثال
+	حاصل جمع	5 + 7	۲۱
-	حاصل تفریق	9 - 3	6
-	منفی	-3	-3
*	حاصل ضرب	3 * 5	۵۱
/	حاصل تقسیم	5 / ۳۱	5
%	باقیمانده تقسیم	8 % 3	2

جدول (۳-۱) : عملگرهای ریاضی

[1] operator
[2] logical
[3] Arithmetic

عملگرهای ریاضی را می‌توان به شیوه‌های متفاوت دسته‌بندی کرد.

یکی از انواع رایج دسته‌بندی عملگر، دسته‌بندی عملگرها بر مبنای تعداد عملوند[1] های آن‌ها است.

برای مثال چهار عمل اصلی ریاضی هر کدام دارای دو عملوند هستند. در مثال جدول فوق، ۵ و ۷ عملوندهای عبارت ۵+۷ هستند. بیشتر عملگرهای جاوا از نوع یک یا دو عملوندی هستند و فقط یک عملگر سه عملوندی در جاوا وجود دارد که در همین بخش به آن نیز اشاره می‌کنیم .

نکته۱: یکی از نکاتی که هنگام استفاده از عملگر تقسیم (/) باید مد نظر داشت، نوع متغیری است که قرار است حاصل تقسیم در آن قرار بگیرد. در صورتی که حاصل تقسیم در یک متغیر از نوع int ذخیره شود، حاصل تقسیم به بزرگترین عدد صحیح کوچکتر از نتیجه گرد می‌شود.

به مثال زیر دقت کنید:

```
int x = 16;
int y = 5;
int z = x / y;
System.out.println( "z is: " + z ); // z = 3.2 ?
```

اگر با ماشین حساب تقسیم فوق را انجام دهید، حاصل تفریق ۳.۲ نشان داده می‌شود در حالی که اگر برنامه فوق را اجرا کنید، مقدار ۳ را در خروجی خواهید دید.

علت آن هم واضح است، متغیر z از نوع int تعریف شده است و نمی‌تواند یک مقدار اعشاری را ذخیره کند.

حال در برنامه فوق تغییری می‌دهیم و آن را به صورت زیر بازنویسی می‌کنیم:

```
int x = 16;
int y = 5;
float z = x / y;
System.out.println( "z is: " + z ); // z = 3.2 ?
```

حالا فکر می‌کنید چه مقداری در خروجی چاپ خواهد شد؟ 3.2 یا ۳ ؟ امتحان کنید! بله بعد از اجرای برنامه فوق ۳.۰ را خواهید دید! اما چرا؟ این بار که متغیر z از نوع float تعریف شده بود؟

[1] operand

در توضیح این مطلب باید گفت که به علت این که هر دو عملوند عملگر تقسیم از نوع int هستند، کامپایلر جاوا حاصل تقسیم را در یک متغیر از نوع صحیح قرار داده و سپس آن را در متغیر z قرار می‌دهد. برای رفع این مشکل می‌توان از دو راه حل استفاده کرد:

تعریف عملوندهای عملگر تقسیم از نوع اعشاری (float) یا (double) یا مجبور کردن کامپایلر به انجام عمل تقسیم اعشاری.

راه حل اول را با هم می‌بینیم:

```
float x = 16.0f;
float y = 5.0f;
float z = x / y;
System.out.println( "z is: " + z ); // z = 3.2 !
```

پس از اجرای برنامه فوق، حاصل تقسیم ۳.۲ خواهد بود.

در این روش به دلیل این که هر دو عملوند عملگر تقسیم از نوع اعشاری هستند، حاصل تقسیم هم یک عدد اعشاری است.

اما اگر بخواهیم دو عدد صحیح را بر هم تقسیم نموده و حاصل تقسیم را به صورت اعشاری ببینیم چه باید کرد؟ برای چنین مواردی از روش دوم استفاده می‌کنیم. در این روش که به «**تغییر نوع**[1]» معروف است، یکی از عملوندهای عملگر تقسیم را به نوع اعشاری «تغییر نوع» داده یا اصطلاحاً آن را CAST می‌کنیم.

به مثال زیر دقت کنید:

```
int x = 16;
int y = 5;
float z = (float)x / y;
System.out.println( "z is: " + z ); // z = 3.2 !
```

[1] **Type casting**

در خط سوم این مثال قبل از انجام عمل تقسیم متغیر x را تغییر نوع داده و آن را به یک متغیر float تبدیل می‌کنیم.بعد از این تبدیل، عملگر تقسیم، به خاطر این که یکی از عملوندهای آن از نوع اعشاری است، به عملگر تقسیم اعشاری تبدیل می‌شود و حاصل تقسیم یک عدد اعشاری می‌شود.

۳-۳ عملگر انتساب

معمولاً همه با عملگر انتساب[1] آشنایی داریم.

کار اصلی این عملگر نسبت دادن مقدار عبارت سمت راست (عملوند راست) به متغیر (عملوند) سمت چپ خود است. با توجه به تعریف بالا عملیات نسبت دادن یک مقدار به یک متغیر خود یک عبارت ریاضی است زیرا دارای مقدار است. همین خاصیت باعث می‌شود که بتوانیم چندین عملگر انتساب را زنجیروار به هم متصل نموده و چندین عبارت را در یک عبارت ساده‌تر نوشت.

به مثال زیر دقت کنید:

```
int x = y = z = 10;
```

با توجه به خاصیت عملگر انتساب، عبارت فوق در زبان جاوا معتبر بوده و معنای آن این است که هر سه متغیر x و y و z دارای مقدار اولیه ۱۰ هستند. عبارت بالا معادل عبارت‌های زیر است:

```
int x;
int y;
int z = 10;
y = z; // y = 10
x = y; // x = 10
```

برای محاسبه مقدار یک عبارت انتسابی (با استفاده از عملگر انتساب)، همواره ابتدا مقدار عملوند سمت راست عبارت انتساب محاسبه شده و سپس حاصل این عبارت در متغیر سمت چپ (عملوند سمت چپ) قرار می‌گیرد.

[1] Assignment Operator

با کمک این خاصیت می‌توان عبارت‌های انتسابی به شکل زیر را نوشت:

```
int x = 5;
x = x + 10; // x = 5 + 10
```

در خط دوم مثال بالا، ابتدا عبارت x + 10 تعیین مقدار می‌شود.

از آنجایی که مقدار اولیه x برابر ۵ است، حاصل عبارت فوق برابر 10 + 5 یا ۱۵ است. سپس این مقدار در متغیر x قرار می‌گیرد. پس از آن مقدار متغیر x برابر ۱۵ است. تغییر دادن مقدار یک متغیر با استفاده از مقدار فعلی متغیر فرایند بسیار متداولی در برنامه نویسی جاوا است.

به همین منظور و برای سادگی عملیات و خوانایی بیشتر کد، چندین عملگر انتساب جدید در زبان جاوا وجود دارند که در جدول زیر به همراه مثال آمده‌اند:

عملگر	مثال	معادل
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y

جدول (۳-۲) : عمگرهای انتساب

نکته ۲: در هنگام استفاده از عملگرهای انتساب میانبر فوق، همواره باید دقت کنید. در صورتی که عبارت سمت راست پیچیده باشد، ممکن است خطاهای ناخواسته‌ای در عبارت‌های محاسباتی شما پدید آید که پیدا کردن و رفع آن‌ها معمولاً ساده نیست.

به مثال زیر دقت کنید:

```
int x = 20;
int y = 5;
...
x = x / y + 5;
x /= y + 5;
...
```

ایا گمان می‌کنید دو عبارت بالا برابر هستند؟

در عبارت بالایی ابتدا حاصل x / y محاسبه شده و سپس مقدار حاصل تقسیم با عدد ۵ جمع می‌شود و حاصل این جمع که ۹ است در متغیر x قرار می‌گیرد.

اما در عبارت دوم، ابتدا حاصل y با ۵ جمع شده سپس x بر این مجموع تقسیم می‌شود یعنی x = x / (y + 5)

۳–4 عملگرهای افزایشی و کاهشی

در برنامه نویسی جاوا بسیاری مواقع اتفاق می‌افتد که می‌خواهیم مقدار یک متغیر را یک واحد کاهش یا یک واحد افزایش دهیم.

برای این کار به سادگی می‌توانیم از هر کدام از عبارت‌های زیر استفاده کنیم:

```
int x = 20;
x = x + 1;
// or
 x += 1;
```

اما در زبان جاوا دو عملگر خاص برای این منظور به وجود آمده‌اند:

عملگر افزایش یکانی (++) و عملگر کاهش یکانی(--) .

این دو عملگر را می‌توان بدون فاصله قبل یا بعد از متغیر قرار داد تا یکی به مقدار متغیر افزوده شده یا یکی از مقدار متغیر کم شود.

به مثال‌های زیر دقت کنید:

```
int x = 20;

x++; // x = x + 1
 // or
++x; // x = x + 1
...
x--; // x = x - 1
 // or
--x; // x = x - 1
```

اگر عملگرهای افزایش یا کاهش یکانی قبل از نام متغیر قرار بگیرند، آن‌ها را عملگرهای پیش افزایش یکانی یا پیش کاهش یکانی می‌نامند.

به همین ترتیب عملگرهای افزایش یا کاهش یکانی که بعد از نام متغیر قرار می‌گیرند، عملگرهای پس افزایش یکانی یا پس کاهش یکانی نامیده می‌شوند .

تفاوت عملگر پیش افزایش یکانی و پس افزایش یکانی بسیار ظریف است.

در یک عبارت ریاضی، اگر از عملگر پیش افزایش یکانی استفاده شود، ابتدا این عملگر محاسبه شده و یک مقدار به متغیر مورد نظر اضافه می‌کند و سپس در محاسبات از این مقدار جدید متغیر استفاده می‌شود.

کارکرد عملگر پیش کاهش یکانی هم مشابه است، به این صورت که ابتدا از متغیر مورد نظر یک واحد کم شده و سپس در محاسبات از این مقدار جدید متغیر استفاده می‌شود .

عملکرد عملگر پس افزایش یکانی و عملگر پس کاهش یکانی به این صورت است که ابتدا عبارت ریاضی به صورت کامل محاسبه شده و در انتهای محاسبات به متغیر مورد نظر یک واحد افزوده شده یا یک واحد از متغیر مورد نظر کسر می‌شود.

به مثال زیر دقت کنید:

```
int x = 20;
int y;

y = x++; //(1)
 // or
y = ++x; //(2)
...
y = x--; //(3)
 // or
y = --x; //(4)
```

در عبارت (۱) ابتدا مقدار y برابر مقدار فعلی متغیر x یعنی ۲۰ می‌شود و بعد یک واحد به متغیر x اضافه می‌شود. این عبارت معادل آن است که بنویسیم:

```
//(1)
 y = x;
 x = x + 1;
```

در عبارت (۲) ابتدا یک واحد به متغیر x افزوده شده و سپس عبارت y = x ارزیابی و محاسبه می‌شود .

این عبارت معادل آن است که بنویسیم:

```
//(2)
x = x + 1;
y = x;
```

در عبارت (۳) ابتدا مقدار y برابر مقدار فعلی متغیر x یعنی ۲۰ می‌شود و بعد یک واحد از متغیر x کم می‌شود.

این عبارت معادل آن است که بنویسیم:

```
//(3)
y = x;
x = x - 1;
```

بالأخره در عبارت (۴) ابتدا یک واحد از متغیر x کم شده و سپس عبارت y = x ارزیابی و محاسبه می‌شود .این عبارت معادل آن است که بنویسیم:

```
//(4)
 x = x - 1;
 y = x;
```

۳-۵ عملگرهای مقایسه‌ای

جاوا عملگرهایی برای مقایسه متغیرها، متغیرها و متغیرهای لفظی و سایر انواع داده‌ها در برنامه دارد.

حاصل این عملگرها یک متغیر بولین[1] است که همواره دارای مقدار true یا false است.

[1] Bool

جدول زیر شامل فهرست کامل عملگرهای مقایسه‌ای جاوا و مثال‌هایی از هر کدام است:

عملگر	معنی	مثال	توضیح مثال
==	تساوی	x == 5	اگر x برابر ۵ باشد حاصل عبارت true و در غیر این صورت false است
!=	نامساوی	x != 5	اگر x برابر ۵ باشد حاصل عبارت false و در غیر این صورت true است
<	کوچک‌تر	x < 5	اگر x کوچک‌تر از ۵ باشد حاصل عبارت true و در غیر این صورت false است
>	بزرگ‌تر	x > 5	اگر x بزرگ‌تر از ۵ باشد حاصل عبارت true و در غیر این صورت false است
<=	کوچک‌تر یا مساوی	x <= 5	اگر x کوچک‌تر از ۵ یا مساوی ۵ باشد، حاصل عبارت true و در غیر این صورت false است
>=	بزرگ‌تر یا مساوی	x >= 5	اگر x بزرگ‌تر از ۵ یا مساوی ۵ باشد، حاصل عبارت true و در غیر این صورتfalse است

جدول (۳-۳) : عملگرهای مقایسه ای جاوا

یک مثال از کاربرد عملگرهای مقایسه‌ای:

```
boolean hip;
int age = 32;
// (1)
hip = (age < 25);
// (2)
hip = ( age == 32 );
// (3)
hip = (age != 32 );
```

در مثال فوق ابتدا یک متغیر boolean و سپس یک متغیر int با مقدار اولیه ۳۲ تعریف می‌کنیم. در عبارت اول، مقدار متغیر بولی hip غلط است، زیرا مقدار متغیر age کمتر از ۲۵ نیست. در واقع ۳۲ < 25 نیست.

در عبارت دوم مقدار متغیر بولی hip درست است زیرا مقدار age برابر ۳۲ است و 32 = 32 است. مقدار عبارت سوم غلط است زیرا این عبارت بررسی می‌کند که آیا متغیر age مخالف ۳۲ است یا نه؟ و چون مقدار متغیر age مخالف ۳۲ نیست، حاصل عبارت غلط می‌شود.

۳-۶ عملگرهای منطقی

عبارت‌هایی که حاصل آن‌ها یک مقدار بولی (boolean) است، می‌توانند با هم ترکیب شده و عبارت‌های پیچیده‌تری بسازند.

برای ترکیب این عبارت‌ها می‌توان از عملگرهای منطقی استفاده کرد. این عملگرها عبارتند از و (AND) ، یا (OR)، مخالف (NOT) و یای انحصاری (XOR) .

جدول زیر شامل عملگرهای منطقی جاوا و توضیحات و مثال برای هرکدام از آن‌ها است:

عملگر	معنی	مثال	توضیح مثال
&	و (AND)	A & B	اگر هر دو عبارت درست (true) باشند، مقدار بازگشتی عبارت، درست (true)و در غیر این صورت مقدار بازگشتی عبارت غلط (false) است.
&&	و (AND)	A && B	اگر هر دو عبارت درست (true) باشند، مقدار بازگشتی عبارت، درست (true)و در غیر این صورت مقدار بازگشتی عبارت غلط (false) است.
|	یا (OR)	A | B	اگر یکی از دو عبارت درست (true) باشد، مقدار بازگشتی عبارت، درست (true)و در غیر این صورت مقدار بازگشتی عبارت غلط (false) است.
||	یا (OR)	A || B	اگر یکی از دو عبارت درست (true) باشد، مقدار بازگشتی عبارت، درست (true)و در غیر این صورت مقدار بازگشتی عبارت غلط (false) است.
^	یای انحصاری (XOR)	A ^ B	فقط و فقط در صورتی مقدار بازگشتی عبارت درست (true) است که یکی از دو عبارت صحیح و دیگری غلط باشد. اگر هر دو عبارت درست (true) یا هر دو عبارت غلط (false) باشند، مقدار بازگشتی عبارت غلط (false) خواهد شد.

جدول (۳-۴) : عملگرهای منطقی جاوا

به مثال زیر دقت کنید، می‌خواهیم به دانشجویانی که نمره آن‌ها بین ۱۰ تا ۱۲ است یک اخطار کتبی بدهیم.

برای این کار باید دو عبارت شرطی را با هم ترکیب کنیم:

```
boolean warning = ( grade > 10 ) && ( grade < 12 );
```

۳-۶-۱ تفاوت & و &&

تفاوت این دو عملگر بسیار ظریف است.

جاوا برای افزایش کارایی[1] برنامه‌ها، سازوکار خاصی برای بررسی عبارت‌های منطقی دارد .اگر در یک عبارت منطقی، حاصل یکی از عملوندها غلط باشد، حاصل کل عبارت غلط می‌شود. در چنین حالتی جاوا از ادامه بررسی سایر عملوندها خودداری کرده و مقدار بازگشتی عبارت منطقی را غلط قرار می‌دهد. به مثال زیر دقت کنید:

```
float grade = 8.9f;
boolean warning = ( grade > 10 ) && ( grade < 12 );
```

جاوا هنگام پردازش عبارت منطقی خط دوم، ابتدا بررسی می‌کند که آیا grade > 10 است یا خیر. در مثال ما چون حاصل این عبارت غلط است، در نتیجه کل عبارت غلط می‌شود و جاوا سایر عبارت‌ها را بررسی نمی‌کند. در اکثر موارد این روش بسیار عالی است و نیازهای اکثر مواقع را برطرف می‌کند. اما اگر بخواهیم فارغ از این که کل عبارت درست یا غلط است، همه عبارت‌ها ارزیابی شوند چه باید بکنیم؟

به مثال زیر دقت کنید:

```
boolean winPrize = ( calculateFactorA() ) && ( calculateFactorB() );
```

اگر تابع calculateFactorA() مقدار غلط برگرداند، هیچگاه تابع calculateFactorB() فراخوانی نخواهد شد.

[1] performance

برای حل این مشکل می‌توان به صورت زیر عمل کرد:

```
boolean factorA = calculateFactorA();
boolean factorB = calculateFactorB();
boolean winPrize = ( factorA ) && ( factorB );
```

اما همانطور که می‌بینید، در این روش ما نیاز به دو متغیر اضافه داریم. همچنین تعداد خطوط مورد نیاز برای این کار از یک خط به سه خط افزایش خواهد یافت.

در موارد پیچیده‌تر از این ممکن است برنامه خوانایی و وضوح خود را نیز از دست بدهد. برای حل این مشکل، در نسخه‌های اخیر زبان جاوا عملگر & به مجموعه عملگرهای منطقی جاوا افزوده شده است. عملکرد این عملگر کاملاً مشابه && است با یک تفاوت کوچک: برای بررسی مقدار بازگشتی یک عبارت منطقی، تمام عملوندهای آن را بدون توجه به این که کل عبارت درست یا غلط خواهد شد، بررسی می‌کند.

با استفاده از عملگر & مثال قبلی ما به شکل زیر اصلاح خواهد شد:

```
boolean winPrize = ( calculateFactorA() ) & ( calculateFactorB() );
```

۳-۶-۲ تفاوت | و ||

تفاوت این دو عملگر نیز مانند تفاوت عملگرهای & و && است. جاوا در هنگام بررسی عبارت‌های منطقی شامل عملگر || برای افزایش کارایی اگر یکی از عملوندها درست باشد، از آنجایی که کل عبارت درست می‌شود، سایر عملوندها را بررسی نمی‌کند. اگر بخواهیم جاوا هنگام بررسی چنین عبارت‌هایی همه عملوندها را بررسی کند به جای || از عملگر | استفاده می‌کنیم .

توضیح ضروری :اگر از نسخه‌های قدیمی‌تر JDK مانند J2SDK 1.4 استفاده می‌کنید، اپراتورهای & و | را در اختیار ندارید! هر چه سریع‌تر نسخه جاوای خود را ارتقا دهید!

۳-۷ اولویت عملگرها

حاصل عبارت زیر چند است؟

```
int x = 6 + 4 / 2;
```

آیا مقدار متغیر x برابر ۵ است یا ۸؟

برای پاسخ دادن به این سؤال باید بدانیم که کدام عملگرها زودتر و کدام‌یک دیرتر ارزیابی می‌شوند.

اگر + زودتر از / ارزیابی بشود، حاصل عبارت ۵ و در صورتی که / زودتر ارزیابی شود، حاصل عبارت ۸ خواهد شد. همانطور که از دروس ریاضی خود به خاطر دارید، برخی عملگر دارای اولویت بالاتری نسبت به سایر عملگرها هستند و هرگاه در عبارتی قرار بگیرند، ابتدا حاصل آن‌ها محاسبه شده و سپس سایر عملگرها ارزیابی می‌شوند .

جاوا نیز برای انجام محاسبات خود از جدول اولویت عملگرها استفاده می‌کند که در آن هر عملگری دارای یک اولویت است .

جدول اولویت عملگرها در جاوا را ببینید:

عملگر	توضیح
. [] ()	از «.» برای دسترسی به متدها و متغیرهای داخل اشیا و کلاس‌ها استفاده می‌شود. از «[]» برای آرایه‌ها استفاده می‌شود. از «()» برای جداسازی عبارت‌ها استفاده می‌شود .
++ -- ! ~ instanceof	++عملگر افزایش یکانی --عملگر کاهش یکانی !نقیض (مخالف) ~ مکمل بیتی یکانی instanceof عملگری که مشخص می‌کند که شی داده شده از نوع کلاس مشخص شده است یا نه.
new (type) Expression	new: عملگری که یک نمونه از روی کلاس مشخص شده می‌سازد (type) Expression: تبدیل نوع Expression به نوع مشخص شده داخل پرانتز

ضرب، تقسیم و باقیمانده	* / %
جمع و تفریق	+ -
>> شیفت بیتی به چپ	<< >> >>>
<< شیفت بیتی به راست با علامت	
<<< شیفت بیتی به راست بدون علامت	
عملگرهای مقایسه‌ای	< > <= >=
عملگرهای تساوی و نامساوی	== !=
عملگر AND منطقی	&
عملگر AND بیتی	
یای انحصاری (XOR) منطقی	^
یای انحصاری (XOR) بیتی	
عملگر OR منطقی	|
عملگر OR بیتی	
عملگر AND منطقی	&&
عملگر OR منطقی	||
عملگر میانبر معادل if-then-else	?:
عملگرهای تساوی	= += -= *= /= %= ^=
عملگرهای تساوی	&= |= <<= >>= >>>=

جدول (۳-۵) : اولویت عملگرها در جـاوا

فصل چهارم

الگوریتم‌ها، ساختارهای کنترلی و برنامه نویسی ساخت یافته در جاوا

۴-۱ الگوریتم چیست؟

یکی از شیوه‌های تفکر سازمان‌یافته (یا ساخت یافته) برای حل مسائل استفاده از الگوریتم است.

احتمالاً می‌دانید که کلمه الگوریتم از نام ریاضیدان و منجم بزرگ ایران در قرن دوم هجری، ابوجعفر محمد بنموسی الخوارزمی، گرفته شده است .

هر مسأله‌ای را می‌توان با اجرای مجموعه‌ای از فعالیت‌ها به یک «ترتیب» مشخص حل کرد. به هر **روالی** برای حل مسائل که از

۱. فعالیت ها به معنی کارهایی که باید انجام شوند،
۲. ترتیب مشخصی برای انجام فعالیت‌هاو
۳. شرط خاتمه عملیات

ساخته شده باشد، الگوریتم می‌گویند. به عنوان مثالی از یک الگوریتم به مثال زیر دقت کنید:

مسأله: بیدار شدن از خواب و رفتن به سر کار

الگوریتم:

۱. خارج شدن از رختخواب
۲. در آوردن لباس خواب
۳. استحمام کردن
۴. لباس پوشیدن
۵. صبحانه خوردن
۶. خروج از خانه
۷. سوار تاکسی شدن (به مقصد محل کار)

چرا در الگوریتم «**ترتیب**» اجرای فعالیت‌ها به اندازه خود **فعالیت** ها مهم است؟

فرض کنید در الگوریتم بالا ترتیب اجرای فعالیت‌ها را اندکی تغییر دهیم و ترتیب اجرای فعالیت‌های ۳ و ۴ را جا به جا کنیم:

...
۳.لباس پوشیدن
۴.استحمام کردن
...

به سادگی می‌توانید حدس بزنید که چه فاجعه‌ای رخ می‌دهد !

در برنامه نویسی هم از الگوریتم استفاده می‌شود .

ساختارهای کنترلی: برنامه‌های رایانه‌ای نوشته شده با هرزبان برنامه نویسی -از جمله جاوا- از سه جزء سازنده اصلی تشکیل می‌شوند که اجرای برنامه را کنترل می‌کنند:

۱. توالی[1]
۲. انتخاب[2]
۳. تکرار[3]

توالی: توالی به معنی اجرای دستورات برنامه به صورت پشت سر هم و پیاپی است. مثالی که درباره الگوریتم زدیم، یک نمونه از ساختار توالی است. تاکنون تمام مثال‌هایی که آورده‌ایم همه از ساختار توالی پیروی می‌کنند.

به مثال زیر دقت کنید:

```
int a = 10;
int b = 10;
int c = a + b;
System.out.println( "a + b = " + c );
int d = a * b;
System.out.println( "a * b = " + d );
...
```

ترتیب اجرای برنامه در این مثال خط به خط است. ابتدا در خط اول متغیر a تعریف شده و مقداردهی می‌شود.

[1] Sequence
[2] Selection
[3] Repetition

بعد از آن در خط دوم متغیر b تعریف شده و مقداردهی می‌شود .در خط سوم متغیر c تعریف شده و با حاصل جمع a و b مقداردهی می‌شود و... این ساختار توالی نامیده می‌شود.

انتخاب: گاهی مواقع می‌خواهیم مسیر اجرای برنامه را از بین چند مسیر مختلف انتخاب کنیم. برای مثال می‌خواهیم در صورتی که نمره دانشجو بالاتر از ۱۸ بود به وی پاداش بدهیم و در صورتی که بین ۱۲ تا ۱۴ بود به وی اخطار بدهیم و در صورتی که بین ۱۰ تا ۱۲ بود، وی را جریمه کنیم.

همانطور که می‌بینید «**فعالیت**» مربوط به هر کدام از انتخاب‌های فوق با دیگری متفاوت است. به الگوریتم زیر دقت کنید:

۱. اگر نمره دانشجو بالاتر از ۱۸ بود :
به دانشجو پاداش بده
۲. اگر نمره دانشجو بین ۱۲ تا ۱۴ بود:

به دانشجو اخطار بده

۳. اگر نمره دانشجو بین ۱۰ تا ۱۲ بود:

دانشجو را جریمه کن

جاوا برای ساختار انتخاب از چندین روش مختلف استفاده می‌کند که در این بخش آن‌ها را معرفی خواهیم کرد:

- ✓ ساختارif
- ✓ ساختارif-else
- ✓ ساختار switch

تکرار: ساختارهای توالی و انتخاب برای اجرای بسیاری از الگوریتم‌ها کافی است با این حال موارد زیادی پیش می‌آید که می‌خواهیم یک فعالیت را به تعداد زیادی تکرار کنیم.

یک کار استفاده از تعداد زیادی دستورات است که به صورت پیاپی (متوالی) اجرا می‌شوند. فرض می‌کنید می‌خواهیم نامه‌های رسیده را پاسخگویی کنیم.

الگوریتم این کار به صورت زیر است:

۱. تا زمانی که نامه‌ای وجود دارد:
نامه را پاسخ بده
۲. ...

در الگوریتم فوق تا زمانی که نامه‌ای وجود دارد، فعالیت‌های ۱ و ۲ تکرار می‌شوند. این ساختار تکرار نامیده می‌شود .

جاوا برای ساختار تکرار از چندین روش مختلف استفاده می‌کند که در این فصل آن‌ها را معرفی خواهیم کرد:

- ✓ ساختارwhile
- ✓ ساختارdo...while
- ✓ ساختارfor
- ✓ ساختار foreach

۴-۲ ساختار تک انتخابی یا if

مواقعی پیش می‌آید که می‌خواهیم در صورت درست بودن یک شرط، یک دستور [1] را اجرا کنیم و اگر شرط غلط بود، بدون توجه به دستور، ادامه برنامه را اجرا کنیم. برای مثال می‌خواهیم در صورتی که نمره دانشجو بیشتر از ۱۰ بود، در کارنامه وی جمله «passed» نوشته شود:

۱. اگر نمره دانشجو بیشتر از ۱۰ است
۱-۱ بنویس «Passed»
۲. ...

در الگوریتم فوق اگر شرط درست باشد، یعنی نمره دانشجو بیشتر از ۱۰ باشد، دستور داخلی (۱-۱) اجرا می‌شود و بعد از آن دستور۲ اجرا خواهد شد، ولی اگر شرط غلط باشد، دستور(۱-۱) اجرا نشده و برنامه از دستور۲ ادامه می‌یابد.

[1] Statement

برای انتخاب تکی در جاوا از دستور if استفاده می‌کنیم:

```
if( condition )
   statement;
```

در الگوی فوق، condition هر عبارت منطقی است که مقدار بازگشتی آن true یا false باشد. statement نیز می‌تواند هر دستور جاوا باشد.

اگر تعداد دستورات داخلی دستور if بیشتر از یکی باشند از شکل زیر برای دستور if استفاده می‌کنیم:

همانطور که گفتیم، شرط[1] می‌تواندهر عبارت منطقی با مقدار بازگشتی true یا false باشد.

```
if( condition ) {
    statement1;
    statement2;
    statement3;
    ...
}
```

بنابراین می‌توان در دستور if عبارت‌های منطقی را با هم ترکیب کرد:

```
if( condition1 && ( condition2 || condition3 ) ) {
    statement;
}
```

چندین مثال از کاربرد دستورif :

```
int a = 5;
int b = 7;
int c = a + b;
if( c > 10 ) {
    System.out.println( "a + b is greater than 10!" );
}
```

در دستور if فوق، شرط این است که آیا حاصل جمع a + b از 10 بیشتر است یا نه، و از آنجایی که حاصل جمع فوق بیشتر از 10 است، شرط درست است و دستور داخلی if اجرا شده و جمله «a + b is greater than 10!» در خروجی چاپ می‌شود.

```
int a = 5;
int b = 7;
if( ( a * b ) > 40 ) {
    System.out.println( "are you sure that (a * b) is greater than 40?" );
}
```

[1] condition

در دستور if بالا، شرط بررسی می‌کند که آیا حاصل ضرب a و b بیشتر از ۴۰ است یا نه، و از آنجایی که مقدار این حاصل ضرب کمتر از ۴۰ است، شرط غلط (false) بوده دستور داخلی if اجرا نخواهد شد.

۴-۲-۱ ساختار دو انتخابی یا if-else

در دستور if اگر شرط درست بود، دستورهای داخلی if اجرا می‌شوند و در غیر این صورت، برنامه از روی این دستورها پرش می‌کند و دستورهای بعد از if را اجرا می‌کند. مواقعی پیش می‌آید که می‌خواهیم در صورت غلط بودن شرط دستور if دستور خاصی را اجرا کنیم. برای مثال به الگوریتم زیر دقت کنید:

اگر نمره دانشجو بیشتر از ۱۰ بود

بنویس **قبول**

در غیر این صورت

بنویس **مردود**

در جاوا در چنین مواردی از دستور if-else استفاده می‌کنیم:

```
if( condition )
    statement1;
 else
    statement2;
```

در الگوی فوق اگر شرط condition درست باشد، دستور statement1 اجرا خواهد شد. اما اگر شرط condition غلط باشد، دستور statement2 اجرا خواهد شد.

همانطور که پیش از این هم گفتیم اگر تعداد دستورات داخلی دستور if یا else بیشتر از یکی باشد از شکل زیر استفاده می‌کنیم:

```
if( condition ) {
    statement1;
    statement2;
    statement3;
    ...
} else {
    statement1;
    statement2;
    statement3;
    ...
}
```

مثالی از کاربرد دستور: if-else در اینجا کد جاوای همان مثالی را که به عنوان الگوریتم ذکر کردیم، می‌نویسیم:

```
if( grade > 10 ) {
    System.out.println("Passed");
 } else {
     System.out.println("Failed");
 }
```

در این مثال ابتدا شرط دستور if بررسی می‌شود، اگر شرط درست باشد دستور داخلی if اجرا شده و جمله «Passed» در خروجی چاپ می‌شود.

اما اگر شرط (grade > 10) درست نباشد، به بیان دیگر اگر grade کمتر از ۱۰ باشد، شرط if غلط می‌شود و بنابراین دستور داخلی else اجرا شده جمله «Failed» در خروجی چاپ خواهد شد.

۴-۲-۲ ترکیب دستور if و else

همانطور که دیدید، الگوی تعریف دستور if-else به شکل زیر است:

```
if( condition ) {
    statement1;
} else {
    statement2;
}
```

همانطور که گفتیم دستور statement1 و دستور statement2 می‌توانند هر دستور مجاز جاوا باشند، بنابراین می‌توان در درون دستور if و دستور else مجدداً از دستورات if و else استفاده کرد. به مثال زیر دقت کنید:

```
int a = 5;
 int b = 7;
 if( a != 5 ) {
     System.out.println("a != 5");
 } else {
     if( b == 7 ) {
         System.out.println("a = 5 and b = 7");
     }
 }
```

قطعه برنامه بالا را به دقت ببینید. همانطور که می‌بینید یک دستور if درون دستور else قرار گرفته است.

برای سادگی بیشتر می‌توان آن‌ها را ترکیب کرده و به شکل زیر نوشت:

```
int a = 5;
int b = 7;
 if( a != 5 ) {
```

```
    System.out.println("a != 5");
 } else if( b == 7 ) {
     System.out.println("a = 5 and b = 7");
 }
```

در مثال بالا ابتدا شرط دستور if بررسی می‌شود. اگر شرط if غلط باشد، آنگاه شرط else if بررسی می‌شود، اگر شرط else if درست باشد، دستورات داخلی else-if اجرا می‌شود و در غیر این صورت اجرای برنامه از اولین دستور بعد از else-if ادامه می‌یابد.

می‌توان ترکیب else-if را بیشتر هم کرد. به مثال زیر توجه کنید:

```
if( condition1 ) {
    statement1;
} else if( condition2 ) {
    statement2;
} else if( condition3 ) {
    statement3;
} else {
    statement4;
}
```

در مثال فوق اگر شرط condition1 درست باشد، statement1 اجرا خواهد شد. اگر شرط condition2 درست باشد، statement2 اجرا خواهد شد و... و اگر هیچکدام از شرط‌های فوق درست نباشند، statement4 اجرا خواهد شد .البته آخرین else اختیاری است و فقط در صورتی آن را اضافه می‌کنیم که بخواهیم در صورت غلط بودن کلیه شرط‌ها، دستورات خاصی را اجرا کنیم .

دستور if-else-if ساختار انتخاب چندتایی در جاوا است.

۴-۳ ساختار انتخاب چندتایی با switch

همانطور که در بخش قبلی گفتیم، ساختار انتخاب چندتایی در جاوا دستور if-else-if است همانطور که دیدید اگر تعداد شرط‌های این دستور زیاد باشد ممکن است پیچیدگی آن از خوانایی برنامه بکاهد. برای حل این مشکل، در جاوا ساختار انتخاب چندتایی switch معرفی شده است.

ساختار کلی دستور switch به شکل زیر است:

```
switch( statement ) {
    case value1:
```

```
        statement1;
        break;
    case value2:
        statement2;
        break;
    case value3:
        statement3;
        break;
    ...
    default:
        statementN;
        break; //Optional
}
```

هر زمان که کنترل اجرای برنامه به دستور switch می‌رسد، عبارت داخل پرانتز مقابل switch را ارزیابی می‌کند. ماحصل این عبارت باید یکی از انواع داده اولیه byte یا char یا short یا int باشد. سپس ماحاصل این عبارت را با مقادیر مقابل case های دستور مقایسه می‌کند. در صورتی که این دو مقدار با هم برابر باشند، کنترل اجرای برنامه به دستورات داخلی case منتقل شده و تارسیدن به دستور break ادامه می‌یابد.

اگر حاصل عبارت switch با هیچکدام از مقادیر case ها برابر نبود، دستورات داخلی default اجرا می‌شود. البته defaultاختیاری است و می‌توانیم آن را ننویسیم.

به مثال زیر دقت کنید:

```
int a = 5;
 int b = 10;
 switch( a + b ) {
     case 5:
         System.out.println("a + b = 5 ?");
         break;
     case 10:
         System.out.println("a + b = 10 ?");
         break;
     case 15:
         System.out.println("a + b = 15 ?");
         break;
     default:
         System.out.println("a + b = " + ( a + b ) );
         break; //Optional
 }
```

در این مثال دو متغیر صحیح از نوع int تعریف کرده‌ایم. وقتی اجرای برنامه به دستور switch می‌رسد، حاصل عبارت a + b محاسبه می‌شود و سپس این مقدار با مقادیر مقابل عبارت‌های case مقایسه می‌شود. ابتدا حاصل جمع a + b با ۵ (case اول) مقایسه می‌شود، از آنجایی که این دو عبارت برابر نیستند، این عبارت با ۱۰ (case

بعدی) مقایسه می‌شود، ولی باز هم این دو عبارت برابر نیستند، بنابراین حاصل جمع فوق با ۱۵ که مقدار case بعدی است مقایسه می‌شود، چون این دو عبارت با هم برابرند، کنترل اجرای برنامه به دستورات داخل این case منتقل می‌شود. سپس دستورات داخلی این case تا رسیدن به اولین دستور break ادامه می‌یابد.

توجه ۱: اگر دستور break را در انتهای دستورات هر case ننویسیم چه اتفاقی می‌افتد؟

به مثال زیر دقت کنید:

```
char ch = 'a';
switch( ch ) {
    case 'a':
        System.out.println("ch is a");
    case 'A':
        System.out.println("ch is A");
        break;
    default:
        System.out.println("ch is not a or A");
        break; //Optional
}
```

در دستور switch بالا، ابتدا نویسه (کاراکتر) ch با عبارت مقابل اولین case مقایسه می‌شود و چون این دو نویسه با هم برابرند، دستور داخل case اجرا شده و عبارت «ch is a» در خروجی چاپ می‌شود ولی بلافاصله دستور داخل case بعدی هم اجرا شده و عبارت «ch is A» در خروجی چاپ می‌شود .در بسیاری از موارد، مثل همین مثال، از قلم انداختن دستور break ممکن است به خطاهای منطقی (خطاهایی که از دید کامپایلر خطا نیستند ولی اجرای برنامه را مختل می‌کنند) در برنامه منجر شود. البته ممکن است از این نکته به صورت هوشمندانه‌ای استفاده کرد.

به مثال زیر دقت کنید:

```
char selection = 'a';
switch( ch ) {
    case 'a':
    case 'A':
        someMethodA();
        break;
    case 'b':
    case 'B':
        someMethodB();
        break;
    default:
        System.out.println("Unknown command.");
        break;//Optional
}
```

در این مثال، فرض بر این است که کاربر برنامه می‌تواند از بین دو گزینه a و b انتخاب کند. برای این کار کاربر باید یکی از دو حرف a یا b را وارد کند. اما ممکن است کاربر شکل بزرگ این حروف یعنی A یا B را وارد کند. با ترفندی که به کار برده‌ایم، هیچ تفاوتی بین a و A و همین طور b و B نیست و اگر کاربر a یا A را وارد کند، متد someMethodA() اجرا می‌شود. از این شیوه در آینده فراوان استفاده خواهید کرد و نمونه‌های زیادی خواهید دید!

۴-۴ ساختار تکرار با while

تاکنون الگوریتم‌های زیادی را دیده‌اید. در اغلب این الگوریتم‌ها نیازی به تکرار برخی دستورات نبوده است. ولی در اغلب مسائلی که می‌خواهیم برای آن‌ها برنامه بنویسیم، به مواردی برمی‌خوریم که نیاز داریم تعدادی از دستورات را چندین بار تکرار کنیم. برای این کار از ساختارهای تکرار که به آن‌ها حلقه یا لوپ [1] هم گفته می‌شود استفاده می‌کنیم .

فرض کنید می‌خواهید مجموع و میانگین ۱۰۰ عدد را محاسبه و در خروجی چاپ کنید. چه می‌کنید؟ آیا ۱۰۰ متغیر تعریف می‌کنید و هر بار مقدار یکی از آن‌ها را از ورودی می‌خوانید؟ یا این که یک بار دستور خواندن از ورودی را می‌نویسید و از برنامه می‌خواهید که آن را برای شما ۱۰۰ بار تکرار کند؟ مسلماً روش دوم را استفاده خواهید کرد .بنیادی‌ترین ساختار تکرار در جاوا حلقه while است. نحو[2] دستور while به صورت زیر است:

```
while( condition )
     statement;
```

و همانطور که برای سایر ساختارهای کنترلی دیدید، اگر تعداد دستورات داخلی while زیاد باشد، حتماً از {و } استفاده می‌کنیم:

```
while( condition ) {
    statement1;
    statement2;
    statement3;
...
}
```

[1] Loop
[2] Syntax

در ساختار تکرار while تا زمانی که شرط condition درست باشد، دستور یا دستورات داخلی while تکرار خواهند شد. به مثال زیر دقت کنید:

```
System.out.println( "Start of while Loop!" );
int counter = 0;
while( counter < 10 ) {
    System.out.println( "now counter is: " + counter );
    counter++;
}
System.out.println( "End of while Loop!" );
```

زمانی که اجرای برنامه به دستور while می‌رسد، مقدار اولیه متغیر counter برابر ۰ است .While، بررسی می‌کند تا ببیند که آیا شرط counter < 10 درست است یا خیر. چون این شرط درست است، عبارت داخلی while اجرا می‌شود. در داخل دستور while هر بار یکی به متغیر counter افزوده می‌شود .وقتی که مقدار counterبرابر ۱۰ می‌شود، شرط حلقه غلط می‌شود و کنترل اجرای برنامه به اولین دستور بعد از while منتقل می‌شود. خروجی قطعه کد فوق به شکل زیر است:

```
Start of while Loop!
now counter is: 0
now counter is: 1
now counter is: 2
now counter is: 3
now counter is: 4
now counter is: 5
now counter is: 6
now counter is: 7
now counter is: 8
now counter is: 9
End of while Loop!
```

نکته۱: به حلقه‌های تکرار که از یک متغیر به عنوان شمارنده استفاده می‌کنند، حلقه‌های تکرار با شمارنده می‌گویند.

در این نوع حلقه‌های تکرار، نیازمند یک متغیر شمارنده به همراه سه ویژگی زیر هستیم:

۱ - متغیر شمارنده دارای مقدار اولیه معتبر باشد .فرض کنید در مثال بالا، مقدار اولیه متغیر counter که به عنوان شمارنده حلقه به کار می‌رود، به جای ۰، برابر با ۱۰ بود. در این حالت برنامه به هیچ وجه داخل این حلقه تکرار نمی‌شد!

۲ - میزان گام افزایش یا کاهش شمارنده مشخص باشد. در هر بار اجرای حلقه، مقدار شمارنده حلقه باید تغییر کند. اگر این اتفاق نیافتد، شرط حلقه تکرار همواره درست است و ممکن است برنامه در حلقه تکرار بی‌نهایت بیافتد، حلقه تکراری که هیچ گاه از آن خارج نمی‌شود !در مثال بالا، در هر بار اجرای حلقه، به شمارنده یکی افزوده می‌شود.

۳ - شرط خاتمه حلقه تکرار، دست‌یافتنی باشد. اگر هیچ گاه شرط کنترل کننده حلقه تکرار، غلط نشود، باز هم مثل مورد بالا ممکن است در حلقه تکرار بی‌نهایت گرفتار شویم. برای مثال فرض کنید در قطعه کد بالایی، به جای این که در هر بار اجرای حلقه، یکی به counter اضافه کنیم، یکی از آن کم می‌کردیم. آنگاه هیچ وقت شرط حلقه تکرار غلط نمی‌شد و همواره این حلقه تکرار می‌شد و دستورات بعد از حلقه تکرار اصلاً اجرا نمی‌شدند!

نکته۲ : حلقه تکرار while هیچکدام از سه ویژگی گفته شده در بالا را مستقیماً کنترل نمی‌کند .بنابراین شما باید، متغیر کنترلی خود را بیرون حلقه تعریف کنید و گام افزایش یا کاهش آن را به نحو مقتضی داخل حلقه بیاورید .

۴-۴-۱ ساختار تکرار با do-while

دستور do-while از نظر مفهومی بسیار شبیه به حلقه تکرار while است.

به نحو (Syntax) این دستور دقت کنید:

```
do
    statement;
while( condition );
```

اگر تعداد دستورات داخلی do-while بیشتر از یکی باشد، از {و }استفاده می‌کنیم:

```
do {
      statement1;
      statement1;
      statement1;
      ...
 } while( condition );
```

بزرگ‌ترین تفاوت بین حلقه‌های تکرار while و do-while در این است که در حلقه‌های while شرط حلقه در ابتدای هر تکرار بررسی می‌شود ولی در حلقه do-while ابتدا یک تکرار انجام شده و سپس شرط حلقه بررسی می‌شود.

برای درک بهتر تفاوت این دو حلقه تکرار به مثال‌های زیر دقت کنید:

```
char userSelect;
do {
    userSelect = displayMenu();
    switch( userSelect ) {
        case 'a':
        case 'A':
            doMethodA();
            break;
        case 'b':
        case 'B':
            doMethodB();
            break;
        case 'q':
        case 'Q':
            sayGoodBye();
            break;
    }
} while( userSelect != 'q' && userSelect != 'Q' );
```

در قطعه کد بالا، فرض بر این است که تابع displayMenu() یک منو را به کاربر نمایش داده و از وی می‌خواهد که برای انجام یک کار، حرف a یا A و برای انجام کار دوم، حرف b یا B را وارد کند و اگر می‌خواهد از برنامه خارج شود، حرف q یا Q را وارد کند. در ابتدای اجرای برنامه، وقتی کنترل اجرای برنامه به دستور do-while می‌رسد، متغیر کنترلی حلقه تکرار، که همان userSelect است، هیچ مقداری ندارد.

اما پس از اولین تکرار در حلقه این متغیر دارای مقدار می‌شود، سپس در دستور switch این مقدار بررسی شده و اگر برابر با a یا A باشد متد doMethodA() و اگر برابر b یا B باشد، متد doMethodB() فراخوانی می‌شود. اگر کاربر q یا Q به معنی خروج از برنامه را وارد کند، متد sayGoodBye() پیغام خروج را برای وی نمایش می‌دهد. حال اولین تکرار حلقه do-while به اتمام می‌رسد و حالا شرط حلقه تکرار بررسی می‌شود. این شرط این است که متغیر userSelect دارای مقدار q و Q نباشد.

همانطور که دیدید، دستورات داخلی حلقه تکرار do-while حداقل یک بار انجام می‌شوند در حالی که در حلقه while ممکن است این اتفاق نیافتد.

نکته۳: در انتهای عبارت while در دستور do-while حتماً سمی‌کالن «;» بگذارید .

۴–۵ ساختار تکرار با for

ساختار تکرار for پرکاربردترین ساختار تکرار در زبان جاوا است. در این ساختار تکرار، بر خلاف ساختار while، متغیر شمارنده به همراه مقدار اولیه آن، میزان گام افزایش یا کاهش شمارنده و همچنین شرط خاتمه حلقه تکرار در داخل for تعریف می‌شوند و این باعث می‌شود که حلقه تکرار با for بسیار خواناتر از حلقه‌های تکرار while و do-while باشد.

به نحو حلقه for دقت کنید:

```
for( <Expression1> ; <Expression2> ; <Expression3> )
    statement;
```

همانطور که می‌بینید در تعریف حلقه تکرار با for سه عبارت در پرانتز آمده است. عبارت اول، تعریف متغیر شمارنده و مقداردهی اولیه به آن است.

عبارت دوم شرط خاتمه حلقه تکرار و عبارت سوم گام پرش شمارنده حلقه است. علت این که هر کدام از این عبارت‌ها را در <> قرار داده‌ایم، نشان از اختیاری بودن هر کدام از این عبارت‌ها است.

به مثال زیر دقت کنید:

```
for( int i = 0 ; i < 10 ; i++ )
   System.out.println( i );
```

این حلقه تکرار به زبان ساده این است: متغیر i را برابر 0 قرار بده و تا وقتی که i < 10 است، هر بار دستور زیر را اجرا کن و سپس یکی به i اضافه کن.

نکته ۴: گام پرش حلقه تکرار for پس از اجرای دستورات داخلی حلقه تکرار for اعمال می‌شود.

نکته ۵: در صورتی که تعداد دستورات داخلی ساختار تکرار forبیشتر از یکی باشد، از {و }استفاده می‌کنیم.

به مثال زیر دقت کنید :

```
int sum = 0;
for( int i = 0 ; i < 10 ; i++ ) {
    System.out.println( "i is: " + i );
    sum += i;
    System.out.println( "sum is: " + sum );
}
```

یکی از ویژگی‌های جالب ساختار تکرار for در جاوا این است که می‌توان در یک حلقه تکرار، از چندین مقدار اولیه و چندین گام پرش مختلف استفاده کرد.

به مثال زیر دقت کنید:

```
for( int i = 0 , j = 10 ; i < 10 ; i++ , j-- ) {
      System.out.println( "( " + i " , " + j + " ) " );
}
```

پس از اجرای قطعه کد فوق، خروجی زیر را خواهید دید:

```
( 0 , 10 )
( 1 , 9 )
( 2 , 8 )
( 3 , 7 )
( 4 , 6 )
( 5 , 5 )
( 6 , 4 )
( 7 , 3 )
( 8 , 2 )
( 9 , 1 )
```

همانطور که گفتیم، هر کدام از سه عبارتی که در ساختار تکرار for می‌آیند اختیاری هستند. یعنی حلقه تکرار for می‌تواند بدون تعریف متغیر شمارنده و مقداردهی به آن باشد ، یا این که حلقه for می‌تواند بدون گام پرش متغیر شمارنده باشد. البته سمی‌کالن‌ها «;» باید حتماً بیایند. به مثال‌های زیر دقت کنید:

```
int i = 0;
for( ; i < 10 ; i++ ) {
    ...
}
//--------------- OR
int i = 0;
for( ; i < 10 ; ) {
    ...
    i++;
}
```

۴-۵-۱ حلقه‌های تکرار تو در تو[1]

در هر ساختار تکرار می‌توانیم تعدادی دستور جاوا را اجرا کنیم، ولی آیا امکان دارد که در یک حلقه تکرار، یک حلقه تکرار دیگر داشته باشیم؟ قطعاً بله، البته با دقت و احتیاط! به قطعه کد زیر دقت کنید و حدس بزنید که چه کاری می‌کند:

```
for( int i = 0 ; i < 10 ; i++ ) {
     for( int j = 0 ; j < 10 ; j++ ) {
         System.out.print( ( i * j ) + "   " );
     }
  System.out.println();
```

اگر حدس زده‌اید که این برنامه جدول ضرب را تولید می‌کند درست حدس زده‌اید! نمونه خروجی برنامه را ببینید:

```
1   2   3   4   5   6   7   8   9   10
2   4   6   8   10  12  14  16  18  20
3   6   9   12  15  18  21  24  27  30
4   8   12  16  20  24  28  32  36  40
5   10  15  20  25  30  35  40  45  50
6   12  18  24  30  36  42  48  54  60
7   14  21  28  35  42  49  56  63  70
8   16  24  32  40  48  56  64  72  80
9   18  27  36  45  54  63  72  81  90
10  20  30  40  50  60  70  80  90  100
```

در برنامه فوق دو حلقه تکرار تو در تو را می‌بینید. در حلقه اول، یک متغیر شمارنده به نام i و در دومی یک متغیر شمارنده به نام j می‌بینید. ابتدا متغیر i با مقدار اولیه ۱ تعریف می‌شود.

بعد برنامه وارد حلقه تکرار داخلی می‌شود و متغبر j با مقدار اولیه ۱ تعریف می‌شود. پس از آن حلقه داخلی ۱۰ بار تکرار می‌شود و در هر بار تکرار، یک عدد به خروجی می‌رود .هر بار که تکرار حلقه داخلی تمام شود، در حلقه بیرونی، خروجی به خط بعد منتقل شده و باز مجدداً حلقه داخلی تکرار می‌شود و...

[1] Nested loop

دستور داخلی حلقه تکرار داخلی چند بار تکرار می‌شود؟ ۱۰ بار حلقه بیرونی اجرا می‌شود و در هر بار اجرای این حلقه، حلقه داخلی هم ۱۰ بار تکرار می‌شود، یعنی دستور داخلی حلقه تکرار داخلی صد بار اجرا می‌شود

نکته ۶: حلقه‌های تکرار تو در تو معمولاً برای پردازش آرایه‌های چند بعدی و ساختار داده‌های مشابه به کار می‌رود.

۴-۵-۲ دستور foreach در جاوا

معمولا این دستور در لیست ها و برای دستیابی راحتر به عناصر لیست استفاده میگردد.

به تکه کد زیر توجه کنید.

```
int[] lst={12,14,52,36,96,41};
for(int x:lst)
    System.out.print(x + ",");
```

خروجی برنامه

```
12,14,52,36,96,41
```

حلقه در هربار اجرا مقدار عنصر لیست را در متغیر x قرار میدهد.

۴-۵-۳ دستورهای break و continue

تا حدی با دستور break در دستور switch آشنا شدید. دستورهای switch و continue برای کنترل بیشتر جریان اجرای برنامه در ساختارهای انتخاب و تکرار به کار می‌روند.

بسیاری مواقع ممکن است در میانه اجرای یک حلقه تکرار از ادامه اجرای حلقه تکرار منصرف شویم. برای مثال فرض کنید در یک آرایه به دنبال یک عدد خاص می‌گردیم، به محض این که عدد مورد نظر پیدا شد، دیگر نیازی به تکرار بیهوده حلقه تکرار نیست. در این حال چه باید کرد؟ باید از حلقه بیرون آمد. برای خروج از حلقه تکرار می‌توان از دستور break استفاده کرد.

به مثال زیر دقت کنید:

```
int [] list = new int[] { 10 , 20 , 30 , 40 , 50 , 60 , 70 , 80 , 90 , 100 };

for( int x:list ) {
    if( x == 20 ) {
```

```
            System.out.println("Hooray! we found 20!");
            break;
        }
}
```

در قطعه برنامه فوق، در ابتدای هر بار اجرای حلقه، بررسی می‌کنیم که آیا عدد مورد نظر برابر ۲۰ است یا نه .اگر عدد مورد نظر برابر ۲۰ باشد، با چاپ یک عبارت، از حلقه تکرار خارج می‌شویم .

کارکرد دستور continue هم به نوعی مشابه است. این بار می‌خواهیم تا انتهای دستورات حلقه پیش نرفته و مجدداً از ابتدای حلقه، دستورات را اجرا کنیم، البته با مقادیر جدید شمارنده!

مثال زیر را ببینید:

```
float [] grades = new float[] { 18 , 9 , 11 , 16 , 6.5f , 14 , 19 , 9 };
for( int i = 0 ; i < grades.length ; i++ ) {
    if( grades[i] < 10 ) {
        continue;
    }
    System.out.println("Passed, grade is: " + grades[i] );
}
```

در حلقه تکرار فوق، اگر مقدار grades[i] کمتر از ده باشد، بقیه دستورات داخلی حلقه تکرار را ادامه نداده و مجدداً برمی‌گردیم و از ابتدای حلقه مجدداً تکرار می‌کنیم، با این تفاوت که مقدار شمارنده یکی بیشتر شده است. ماحصل این تکرار چاپ نمره‌های بیشتر از ۱۰ است.

به خروجی برنامه دقت کنید:

```
Passed, grade is: 18.0
Passed, grade is: 11.0
Passed, grade is: 16.0
Passed, grade is: 14.0
Passed, grade is: 19.0
```

نکته ۷ : کاربرد دستورات break و continue در حلقه‌های تکرار است. تنها کاربرد break در خارج از ساختارهای تکرار، در دستور switch است.

فصل پنجم

آشنایی با کلاسها

۵–۱ مقدمه

کلاسها در مرکز هسته ای جاوا جای دارند.کلاسها ساختار منطقی هس‌تند که کل زبان جاوا بر روی آن ساخته شده است چراکه شکل و ماهیت شیها را تعریف میکند.

شاید مهمترین نکته ای که در مورد کلاسها یاد گرفت آن است که نوع جدیدی از داده ها را تعریف میکنند . دادهای نوع جدید را پس از تعریف شدن میتوان بر روی شی های نوع مورد نظر بکار برد.

وقتی کلاسی را تعریف میکنید ماهیت و فرم دقیق ان معرفی میشود.

این کار با مشخص کردن داده های درون آن و روتین هایی که بر روی آن داده ها عمل میکنند ، انجام میشود.

۵–۲ تعریف کلاس

هر کلاس با استفاده از کلمه کلیدی Class تعریف میشود. کلاسها میتوانند بسیار پیچیده باشند. شکل عمومی هر کلاس در زیر نشان داده شده است.

```
Class classname {
Type instrance-variable1;     تعریف متغیر
Type instrance-variable2;
...
Type instrance-variableN;

Type methodname2(parameter-list پارامترهای تابع ){
//body of method
}

Type methodnameN(parameter-list){
//body of method
  }
}
```

داده ها یا متغیرهایی که در هر کلاس تعریف میشوند <<نمونه متغیر>> نامیده میشوند. روتینها نیز در متدها جای میگیرند و به طور کلی به متدها و متغیرهایی که در هر کلاس تعریف میشوند، اعضای کلاس گفته می‌شوند.

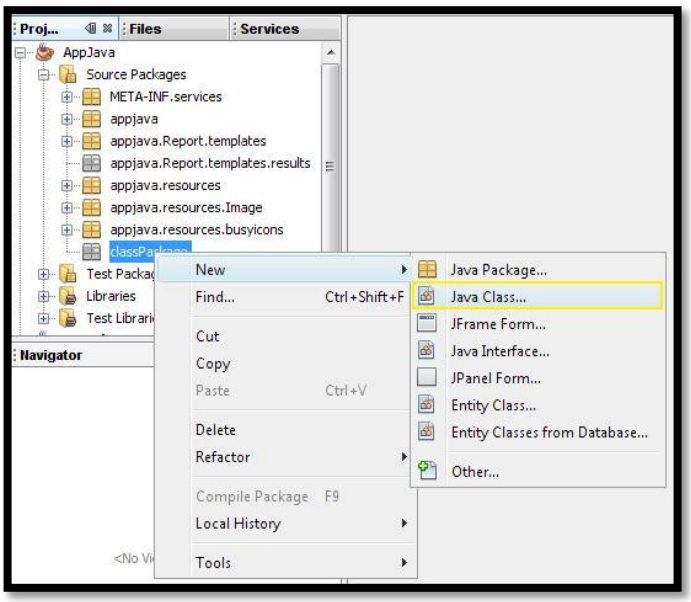
کلاسهای جاوا نیاز به متد main() ندارند. تنها زمانی چنین متدی مشخص میشود که کلاس مورد نظر، نقطه آغازین برنامهتان باشد.

۵-۲-۱ یک کلاس ساده

مطالب کلاسها را با یک مثال ساده آغاز میکنیم. برای این کار کلاسی به نام clscompute تعریف می-کنیم که دارای دو متغیر x،y میباشد.

برای اضافه کردن کلاس در نت بینز روی یکی از package های پروژه راست کلیک کرده و از منوی new گزینه-ی java class را انتخاب مینمایم.

در پنجره ظاهر شدهی New java class در قسمت Class name نام کلاس را وارد نموده و د کمه finish را کلیک نمایید (شکل ۵-۱).



شکل (۵-۱) : ایجاد کلاس

به کد زیر دقت کنید.

```
public class clsCompute {
int x;
int y;
}
```

همانطور که گفته شد هر کلاس، نوع جدیدی از داده ها را تعریف میکند.

در این مثال خاص، نوع جدیدی که ایجاد میشود clsCompute نامیده شده است.

نکته ۱ : تعریف هر کلاس جدید تنها سبب ایجاد یک الگو (Template) میشود،یک شی واقعی ایجاد نمیشود

برای آنکه یک شی clsCompute ایجاد شود، میبایست از عباراتی همچون سطر زیر استفاده کنید.

```
clsCompute myCls = new clsCompute();     ایجاد یک شي
```

پس از آنکه عبارت فوق اجرا شود، myCls به عنوان نمونهای از clsCompute ایجاد خواهد شد. از این رو، یک واقعیت <<فیزیکی>> از کلاس clsCompute ایجاد خواهد شد.

برای دستیابی به متغیرهای کلاس باید از عملگر (.) استفاده کنید. این عملگر، نام شی را به نام <<نمونه متغیر>> مرتبط میکند.

به عنوان مثال مقدار ۲۰ را به متغیر x از myCls تخصیص دهید، از عبارت زیر استفاده کنید.

```
myCls.x=20;
```

عبارت فوق برای کامپایلر مشخص میکند که مقدار ۲۰ را به نسخه ای از x که در شی myCls است تخصیص دهد.به طور کلی، از عملگر نقطه (.) برای دستیابی به نمونه متغیرها و متدهای موجود در یک شی استفاده میشود.

هرشی نسخه هی خاص خودش را از نمونه متغیرها خواهد داشت. این بدین معناست که اگر دو شی نوع clsCompute داشته باشید، هر یک از انها، نسخه های خاص خودشان را از x،y خواهند داشت.

مهم است بدانید که تغییراتی که در نمونه متغیرهای یک شی ایجاد میشوند، هیچ تاثیری بر نمونه متغیرهای شی دیگر نخواهند داشت.

۵-۳ شیوه های تعریف شی

برای دسترسی به اعضای یک کلاس ابتدا بایستی متغیری از نوع ان کلاس ایجاد شود. این متغیر سبب تعریف یک شی نمیشود، بلکه در عوض متغیری است که میتواند به یک شی ارجاع داشته باشد.

برای تعریف متغیر و اختصاص کلاس به متغیر از عملگر new استفاده میگردد.

عملگر new، حافظه ای را به طور پویا (یعنی در زمان اجرا) به شی تخصیص میدهد و نشانی آن را برمی - گرداند.این نشانی سپس در متغیر ذخیره میشود.از این رو، تمام شی های نوع کلاس در جاوا باید به طور پویا تخصیص یابند.

تعریف شی میتواند به روش های زیر صورت بگیرد.

```
clsCompute myCls=new clsCompute();   در زمان تعریف متغیر حافظه تخصیص داده میشود
```

یا

```
clsCompute myCls;
...
myCls = new clsCompute();  در برنامه حافظه به متغیر تخصیص داده میشود
```

۵-۳-۱ تخصیص متغیرهای ارجاع به شی

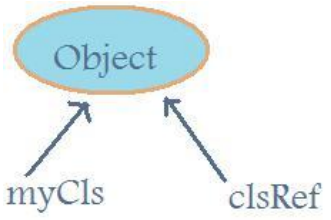
وقتی عمل تخصیص انجام میگیرد، عملکرد متغیرهای ارجاع به شی با آنچه انتظار دارید تفاوت دارد.

```
clsCompute myCls  = new clsCompute;
clsCompute clsRef = myCls;
```

پس از اجرای دستور بالا، myCls و clsRef هردو به یک شی ارجاع خواهند داشت.

تخصیص myCls و clsRef موجب تخصیص حافظه یا کپی کردن بخشی از شی اولیه نمیشود.بلکه صرفا سبب میشود که clsRef به همان شی که myCls به ان ارجاع دارد، ارجاع داشته باشد.

از این رو هرگونه تغییر در شی از طریق myCls بر شئی که clsRef به ان ارجاع دارد تاثیر خواهد گذاشت، چرا که هردو آنها یک شی هستند (شکل ۵-۲).



شکل (۵-۲) : ارجاع به شی

۵-۴ معرفی متدها[1]

کلاسها معمولا از دوچیز تشکیل میشوند:

نمونه متغیرها instance variable متدها Method موضوع متدهای بسیار گسترده است، چراکه جاوا قدرت و انعطاف پذیری زیادی را درانجا جای داده است.

شکل کلی هر متد به صورت زیر است.

```
Type name(parameter-list){
// body of method
}
```

Type نوع داده هایی را مشخص میکند که متد بازمیگرداند.Type میتواند هریک از انواع مورد بررسی قبلی باشد، از جمله انواع کلاسهایی که خودتان ایجاد میکنید.چنانچه متد چیزی را برنگرداند، type باید void باشد.

نام متد نیز به وسیله name مشخص میشود.از هر شناسه معتبری میتوانید به عنوان نام استفاده کنید، البته به غیر از مواردی که برای اقلام موجود در همان محدوده جاری استفاده شده اند.

Parameter-list فهرست زوجهایی (نوع و شناسه) است که با کاما از یکدیگر جدا میشوند.

پارامتر اساسا متغیرهایی هستند که مقدار ارگومان های ارسالی به متد را هنگام فرخوانی ان دریافت میکنند.

چنانچه متد پارامتری نداشته باشد، این فهرست خالی خواهد بود.

---

[1] method

متد هایی که نوع مقدار حاصل از فراخوانی آنها چیزی به غیر از void باشد، مقداری را با استفاده از عبارت return به روتین فراخوان باز میگردانند:

```
Return value;  مقداري است كه برگردانده ميشود value
```

برای متدهایی که هیچ مقداری را برنمیگردانند میتوان از return; استفاده کرد که سبب خروج از متد میگردد.

۵-۴-۱ متدها در کلاس

در بیشتر مواقع از متدها برای دستیابی به نمونه متغیرهای تعریف شده در کلاسها استفاده خواهد شد.در حقیقت، متدها، رابط دستیابی به بیشتر کلاسها را تعریف میکنند. این امر به ایجاد کننده کلاسها امکان می دهد تا شمای[1] ساختارهای داده ای مرتبط با کلاس را پشت سر متدهای شفاف تر پنهان نماید.

در این مثال متدی با نام areaRectangle به کلاس اضافه میشود و با استفاده از متغیر های کلاس مساحت مستطیل را بدست میاورد.

```
public class clsCompute {
int x;
int y;
public int areaRectangle(){
return x*y;
}    }
```

مانطور که مشاهده میکیند متد areaRectangle ، متدی بدون پارامتر میباشد که با استفاده از متغیر کلاس مقداری را برمیگرداند.

دونکته مهم درباره مقادیر حاصل از فراخوانی متدها وجود دارد که باید به خوبی با آنها اشنا باشید:

نوع داده های حاصل از فراخوانی متد باید با نوعی که در تعریف متد مشخص شده است سازگار باشد. به عنوان مثال، اگر نوع مقداری که یک متد باز میگرداند Boolean باشد، نمیتواند مقدار صحیحی را بازگردانید.

[1] schema

متغیر دریافت کننده مقدار حاصل از فراخوانی متد باید با نوعی که در تعریف متد مشخص شده است، سازگار باشد.

۵-۴-۲ افزودن متدهای پارامتریک

اگرچه برخی از متدها نیاز به پارامتر ندارند اما بیشتر متدها اینگونه نیستند.

پارامترها، امکان عمومیت بخشیدن به متدها را فراهم میسازد. یعنی، متدهای پارامتریک میتوانند برروی انواع دادها عمل کنند، و در شرایط نسبتا مختلف مورد استفاده قرار گیرند.

حال میتوان تابع محاسبه مساحت را اینگونه تغییر داد.

```
...
public int areaRectangle(int x,int y){
return x*y;
}
...
```

۵-۴-۵ Constructor ها[1]

جاوا این امکان را فراهم ساخته تا شیءها خودشان را به هنگام ایجاد، مقداردهی کنند. این مقداردهی خودکار، از طریق استفاده از یک constructor انجام میشود.

Constructor، یک شی را به محض ایجاد ، مقداردهی میکند. نام آن با نام کلاسی که در آن قرار دارد یکسان بوده و از نظر ساختار گرامری نیز مشابه متدهاست.

هر constructor پس از تعریف، به طور خودکار به محض ایجاد شیء فراخوانده میشود.

این وظیفه constructorها است که وضعیت داخلی یک شیء را درهمان ابتدای کار تعیین کنند(مقداردهی اولیه)، تاروتینی که نمونه ای از کلاس را ایجاد میکند،فوراً شیء قابل استفاده و مقداردهی شده ای داشته باشد.

[1] سازنده

برای مثال با استفاده از سازنده مقدار اولیه به متغیرهای y،x داده میشود و برای ایجاد کلاس مجبور خواهید بود مقداری را وارد نمایید.

```
public class clsCompute {
int x;
int y;
public clsCompute(int x,int y){  توجه داشته باشید در سازنده نوع برگشتي وجود ندارد
this.x=x;
this.y=y;
}
...
}
```

برای ایجاد کلاس شما مجبور خواهید بود مقادیر y،x را وارد نمایید.

ایجاد کلاس به صورت زیر میباشد.

```
clsCompute cls = new clsCompute(5,7);
```

۵-۴-۶ کلمه کلیدی this

گاهی اوقات متدها نیاز به ارجاع به شیئی دارند که آنها را فعال کرده است. جاوا برای فراهم ساختن این امکان، کلمه کلیدی this کرده است.

با استفاده از this در هر متد میتوان به شیء جاری ارجاع نمود. یعنی، this همیشه ارجاع به شیئی دارد که متد برای آن فعال شده است. هرجا که ارجاع به شی از کلاس جاری مجاز باشد، میتوان از this استفاده نمود.

برای درک بهتر اینکه this به چه چیزی ارجاع دارد، به کد زیر دقت کنید.

```
Public class clsCompute{
int  rf;  متغیر تعریفي کلاس
public void getRef(int rf متغیر تابع)
{
this.rf=rf;
} }
```

در کد فوق this.rf به متغیر کلاس اشاره میکند نه متغیر تابع.

۵-۴-۷ متد Finalize()

گاهی اوقات برخی از شیءها نیاز به انجام عملیات خاص پیش از بین بردن دارند.

به عنوان مثال،اگر شیئی از منابع غیر جاوا،از قبیل handle یک فایل یا فونت خاص، استفاده میکند،در ان صورت بهتر است پیش از ازاد سازی ان شیءاز ازادشدن ان منابع اطمینان حاصل نمایید.جاوا برای مدیریت این گونه شرایط، مکانیزمی به نام funalization دارد.

با استفاده از این مکانیزم میتوانید عملیات خاصی را مشخص کنید که درست پیش از ازادسازی یک شیء،تماما انجام شود.برای پیاده سازی این مکانیزم در هر کلاس،کافی است متد finalize() را تعریف نمود.

محیط اجرای جاوا این متد را هنگام بازیافت شیئی از ان کلاس فرا میخواند.

در متد finalize() باید ان عملیاتی را مشخص کنید که باید پیش از بین بردن یک شیء انجام شوند.

قسمتی که از مسئولیت بازپس گیری حافظه بلا استفاده را دارید،به طور متناوب اجرا شده و شیءهایی را جستجو میکند که دیگر ارجاعی به انها صورت نمیگیرد و بهطور غیرمستقیم نیز از طریق سایر شیءها به انها ارجاع نمیشود.درست پیش از ازادسازی هر شیء،سیستم <<زمان اجرای >> جاوا متد finalize() را برای ان شیءفرا میخواند.

شکل کلی این متد در زیر نشان داده شده است:

```
Protected void finalize(){
// finalization code here
}
```

نکته2 : کلمه کلیدی protected، مشخصه ای است که از دستیابی به finalize() توسط روتین های خارج از کلاس جلوگیری میکند

۵-۴-۸ Overload[1] کردن متدها

در زبان جاوا این امکان فراهم شده تا دو یا بیش از دو متد همنام در یک کلاس تعریف نمود، مشروط بر اینکه تعریف پارامترهای آنها متفاوت باشد در این گونه موارد گفته میشود که متدها Overload شدهاند وبه این فرایند، method overload گفته میشود.

این فرایند یکی از روشهایی است که جاوا از طریق آن پلی مورفیزم[2] پشتیبانی میکند.

وقتی متد overload شدهای فعال میشود، جاوا از انواع و یا تعداد ارگومانها برای تعیین اینکه کدا م نگارش از متد overload شده فراخوانی شده است، از این رو، متدها overload شده از جهت نوع و تعداد با یکدیگر تفاوت دارند.

اگرچه نوع مقادیری که این متدها بر میگردانند ممکن است متفاوت باشد، اما نوع مقادیر به تنهایی برای تمایز بین انها کفایت نمیکند.

وقتی که جاوا با عبارت فراخوانی این گونه متدها مو اجه میشود، متدی را اجرا میکند که پارامترهای آن با ارگومانهای مورد استفاده در عبارت فراخوانی مطابقت داشته باشد. مثال ساده زیر فرایند overload کردن را روی متد areaRectangle انجام میدهد.

```
public class clsCompute {
int x;
int y;
public int areaRectangle(int x,int y){
return x*y;
}
public int areaRectangle(){
return x*y;
}
...
}
...
clsCompute cls = new clsCompute();
cls.areaRectangle();
cls.areaRectangle(5,10);
```

[1] سربـار

[2] **چند ریختی**

همانطور که ملاحظه میکنید متد areaRectangle در کلاس فوق سربارگذاری شده است.

هنگام فراخوانی متد در برنامه، جاوا بر اساس پارامترهای ارسالی به متد تشخیص خواهد داد کدام یک را اجرا نماید. دلیل اینکه فرایند overload کردن متدها از پلی مورفیزم پشتیبانی میکند، آن است که این فرایند یکی از راههایی است که جاوا از طریق آن، مدل <<یک رابط،چند متد>> را پیاده سازی میکند.

در زبانهایی که از overload کردن پشتیبانی نمیکنند، معمولا دو یا سه نگارش از این تابع وجود دارد که نامشان قدری با یکدیگر تفاوت دارد و به عنوان مثال، تابع abs() در زبان C قدرمطلق یک عدد صحیح را برمیگرداند و labs() قدرمطلق یک عدد صحیح نوع long را برمیگرداند و fabs() نیز قدر مطلق یک مقدار اعشاری با ممیز شناور را برمیگرداند.

نکته ۳ : متدهای سازنده را نیز میتوانید همچون متدهای معمولی overload کنید. در حقیقت، در بیشتر کلاسهای مربوط به کارهای واقعی، constructorهای overload شده، نه تنها استثنا به شمار نمی آیند، بلکه کاملا معمول خواهند بود.

۵-۵ استفاده از شیءها به عنوان پارامتر

تا به حال تنها از انواع دادههای پایه و ساده به عنوان پارامتر متدها استفاده کردهایم. اما، ارسال شیءها به متد هم درست و هم متداول است.

به برنامه کوتاه زیر دقت کنید.

```
private class clsRef{
public int x;
    }
private class clsMain{
   int x;
public void getx(clsRef cls){پارامتر ارسالي متغيري از كلاس clsRef ميباشد
this.x=cls.x;
}    }
...
clsRef rf=new clsRef();
rf.x=5;
clsMain main=new clsMain();
main.getx(rf);   متغير كلاس به متد ارسال ميگردد
```

۵–۵–۱ نگاهی دقیقتر به روند ارسال پارامترها

به طورکلی، در هر زبان برنامه نویسی دو روش برای ارسال آرگومانها به یک سابروتین وجوددارد. روش نخست، Call-by-value[1] نام دارد. در این روش، مقدار آرگومان به پارامتر سابروتین کپی میشود. بنابراین، تغییراتی که در پارامتر اعمال میشوند، هیچ تاثیری بر ارگومان نخواهد داشت.

روش دوم، call-by-reference[2] نام دارد. در این روش، نشانی آرگومان (ونه مقدار آرگومان) به پارامتر ارسال میشود. از این نشانی در سابروتین برای دستیابی به خود آرگومان مشخص شده در عبارت فراخوانی استفاده میشود. این بدان معناست که تغییراتی که در پارامتر اعمال میشوند، بر آرگومان مورد استفاده برای فراخوانی سابروتین تاثیر خواهد گذاشت.

جاوا بسته به چیزی که ارسال میشود، از هر دو روش استفاده میکند. وقتی در جاوا یکی از انواع دادهای پایه را متدی ارسال میکنید، از روش call-by-value استفاده میشود. از این رو برای پارامتر دریافت کننده آرگومان رخ میدهد، بازتابی در خارج از متد نخواهید داشت.

وقتی شیئی را به متدی ارسال میکنید، وضعیت به طور چشمگیری تغییر میکند، چراکه شیءها با روش call-by-reference ارسال میشوند.

۵–۶ قابلیت بازگشت

جاوا از مسئله قابلیت بازگشتی پشیبانی میکند. منظور از قابلیت بازگشت، فرایند تعریف کردن یک چیز بر حسب خودش می باشد .قابلیت بازگشت خصوصیتی است که به متدها امکان میدهد تا خودشان را فرا بخوانند. متدی که خودش را فرابخواند، بازگشتی نامیده میشود.

یک مثال ساده برای محاسبه فاکتور یک عدد با استفاده از قابلیت بازگشتی.

```
public class factorial{
int fact(int n){
```

[1] فراخوانی با مقدار
[2] فراخوانی با ارجاع

```
    int result;
if(n==1) return 1;
result=fact(n-1)*n;
return result;
 }
  }
```

وقتی متدی خود را میخواند، فضای مورد نیاز متغیرهای محلی جدید و پارامترها در پشته تخصیص میبابد، و قسمت اجرایی متد با همین متغیرهای جدید از ابتدا اجرا میشود. با پایان رسیدن نتیجه هر عبارت فراخوانی، متغیرهای محلی قدیمی و پارامترها از روی پشته برداشته میشوند، واجرا از نقطه آغاز فراخوانی در خود متد ادامه میبابد. اصطلاحاً گفته میشود که متدهای بازگشتی همچون تلسکوپها، حرکت رو به جلو و عقب دارند.

۵-۷ کنترل دستیابی

همانگونه که میدانید <<نهان سازی[1]>>، داده ها را با روتینهایی که آنها را پردازش و مدیریت میکنند، مرتبط می‌سازد.

اما، نهان سازی خصوصیت مهم دیگری هم دارد: کنترل دستیابی از طریق نهان سازی، میتوانید کنترل کنید که کدام قسمتهای برنامه میتوانند به اعضای کلاس مورد نظر دستیابی داشته باشند. با تحت کنترل در آوردن دستیابی میتوانید از کاربرد نادرست جلوگیری کنید.

چگونگی دستیابی به هر یک از اعضای یک کلاس به وسیله <<مشخصه دستیابی[2]>> مورد استفاده برای تعریف-کردن آن تعیین میشود. جاوا مجموعهای غنی از مشخصه های دستیابی را فراهم ساخته است.

مشخصه دستیابی جاوا عبارتند از public، private، protected.

جاوا همچنین سطح دستیابی پیشفرضی را تعریف کرده است.

نکته ۴ : مشخصه دستیابی protected، زمانی اعمال میشود که وراثت مطرح باشد.

وقتی عضوی از یک کلاس به وسیله مشخصه public تعریف میشود، در آن صورت آن عضو توسط هر روتین دیگری قابل دستیابی خواهد بود.

[1] **Encapsulation**
[2] Access Specifier

وقتی عضوی از یک کلاس به وسیله مشخصه private تعریف میشود، در آن صورت تنها توسط سایر اعضای کلاس خودش قابل دستیابی خواهد بود.

برای درک بهتر موضوع به کلاس زیر توجه کنید.

```
public class clsCompute {
public int varPublic;
private int varPrivate;
public static int varStatic;
...
}
//روش دستیابی

clsCompute.varStatic=5;
-------------------------------------------------------
clsCompute cls = new clsCompute();
cls.varPublic=5;
cls.varPrivate=5;       خطا در دستیابی به متغیر
```

۵-۷-۱ متغیرهای static

عموماً، هرعضو از یک کلاس به تنها در ارتباط با شیئی از همان کلاس خودش مورد دستیابی قرار گیرد. اما، این امکان وجود دارد که بتوان اعضایی را ایجاد نمود که به تنهایی و بدون ارجاع به نمونه خاصی از کلاس، قابل دستیابی باشند. برای ایجاد اینگونه اعضا از کلمه کلیدی static در ابتدای سطر تعریف آنها استفاده کنید. وقتی عضوی از یک کلاس به صورت static تعریف میشود، دستیابی به آن پیش از ایجاد شیئی از همان کلاس، و بدون ارجاع هر گونه شی مقدور خواهد بود.

هم متدها و هم متغیرها را میتوانید، به صورت static تعریف کنید.

نمونه متغیرهایی که به صورت static تعریف میشود عملاً متغیرهای عمومی میشوند. وقتی شی هایی از کلاس یک متغیر static تعریف میشوند هیچ نسخهای از آن متغیر ایجاد نمیشود. در عوض، تمام نمونهی آن کلاس، متغیرهای static یکسانی را به اشتراک میگذارند.

متدهایی که به صورت static تعریف میشوند، چندین محدودیت دارند.

تنها سایر متدهای static را میتوانند فرا بخوانند. باید تنها با دادهای static کار کنند.

به هیچ عنوان نمیتوانند از this یا Super استفاده کنند.

۵–۸ ارگومانهای با طول متغیر[1]

J2SE5، ویژگی جدید را به جاوا افزوده است که ایجاد متدهایی را ساده میکند که نیاز به تعداد متغیری (تعداد پارامتر ناشناخته) ارگومان دارند.

این ویژگی، varags نامیده شده است و کلمات از Variable-length arguments گرفته شده است.

متدی که تعداد ارگومانهایش متغیر است، متد variable-arity یا صرفاً متد varargs نامیده میشود.

ارگومانها با طول متغیر با سه نقطه (...) مشخص میشوند.

هر متد میتواند پارامترهای <<متعارفی[2]>> ، همراه با یک پارامتر با طول متغیر داشته باشد.

به عنوان مثال، تعریف زیر کاملا قابل قبول است:

```
int fnarg(int x,int y,int z,int ... vls)
```

محدودیتهای پارامتر Varargs

۱ - پارامتر varargs باید آخرین پارامتر باشد.

۲ - تنها یک پارامتر varargs مجاز است تعریف شود.

```
void fnArg(int ... vls){
    for(int rs:vls)
        System.out.print(rs);  نمایش مقادیر ارسالی از برنامه
}
//استفاده از متد
fnArg(5,36,2,3,5,6,3,9);
```

[1] varargs
[2] standard

فصل ششم

وراثت،بسته ها و رابطها

۶–۱ وراثت

وراثت یکی از از سنگ بناهای برنامه نویسی شیگراست، زیرا امکان ایجاد طبقه بندیهای سلسله مراتبی را بوجود می‌آورد. با استفاده از وراثت، میتوانید یک کلاس عمومی بسازید که ویژگیهای مشترک یک مجموعه اقلام بهم مرتبط را تعریف نماید. این کلاس بعداً ممکن است توسط سایر کلاسها به ارث برده شود و هر کلاس برده چیزهایی را که منحصربفرد خودش باشد به آن اضافه نماید. در روش شناسی جاوا، کلاسی که به ارث برده میشود را کلاس بالا [1] مینامند. بنابراین، یک "زیر کلاس" روایت تخصصی تر و مشخص تر از یک "کلاس بالا" است. زیر کلاس،کلیه متغیرهای نمونه و روشهای توصیف شده توسط کلاس بالا را به ارث برده و منحصر بفرد خود را نیز اضافه میکند.

۶–۱–۱ مبانی وراثت

برای ارث بردن از یک کلاس، خیلی ساده کافیست تعریف یک کلاس را با استفاده از واژه کلیدی extends در کلاس دیگری قرار دهید.

برای فهم کامل مطلب وراثت، مثال ساده ای را نشان میده یم.

در این برنامه یک کلاس بالا با نام clsSuper و یک زیر کلاس موسوم به clsSub تعریف میگرد.

دقت کنید که چگونه از واژه کلیدی extends استفاده شده تا یک زیر کلاس از clsSuper ایجادشود.

```
class clsSuper{                    سوپر کلاس
int i;
int j;
void showij(){
System.out.println("i and j : "+i+" "+j);
}    }
```

[1] superclass

```
//-----------------------------------------
class clsSub extends clsSuper{          زیر کلاس
int k;


void showk(){
System.out.println("k : "+k);
}
void sumijk(){
int sum=this.i+this.j+this.k;
System.out.println("sum i+j+k : "+sum);
}    }
//                                    استفاده از کلاسها
clsSuper CSP = new clsSuper();
clsSub  CSB = new clsSub();
System.out.println("Class Super = ");
CSP.i=5;
CSP.j=7;
CSP.showij();
System.out.println("Class Sub = ");
CSB.k=9;
CSB.i=15;      وراثت از کلاس
CSB.j=10;      وراثت از کلاس
CSB.showk();
CSB.showij();  وراثت از کلاس
CSB.sumijk();
```

خروجی برنامه

```
Class Super =
i and j : 5 7

Class Sub =
k : 9
i and j : 15 10
sum i+j+k : 34
```

همانطوریکه می بینید، زیر کلاس clsSub در بر گیرنده کلیه اعضای بالای مربوط به clsSuper میباشد.

کلاس بالابودن برای یک زیر کلاس بدان معنا نیست که نمیتوان خود آن کلاس بالا را به تنهایی مورد استفاده قرار داد. بعلاوه، یک زیر کلاس میتواند کلاس بالای یک زیر کلاس دیگر باشد.

شکل عمومی اعلان یک کلاس که از یک کلاس بالا ارث میبرد، بصورت زیر است:

```
Class subclass-name extends superclass-name{
//body of class
}
```

برای هر زیر کلاسی که ایجاد میکنید، فقط یک کلاس بالا میتوانید تعریف کنید.

جاوا از انتقال وراثت چندین کلاس بالا به یک کلاس منفرد پشتیبانی نمیکند (از این نظر جاوا با C++ متفاوت است که در آن وراثت چند کلاسه امکان پذیر است). هیچ کلاسی نمیتواند کلاس بالای خودش باشد.

۶-۲ دسترسی به اعضا ی وراثت

اگرچه یک زیر کلاس در برگیرنده کلیه اعضای کلاس بالای خود میباشد،اما نمیتواند به اعضایی از کلاس بالا که بعنوان private اعلان شده اند، دسترسی داشته باشد.

به کد زیر دقت کنید.

```
 class clsSuper{      سوپر کلاس
private int i;
int j;
void showij(){
System.out.println("i and j : "+i+" "+j);
}
 }

class clsSub extends clsSuper{    زیر کلاس
...
this.i=5;  دسترسي به متغير i وجود ندارد
}
```

۶-۲-۱ کاربرد کلمه کلیدی Super

شرایطی وجود دارند که میخواهید یک کلاس بالا ایجاد کنید که جزئیات پیاده سازی خودش را خودش نگهداری کند. در این شرایط، راهی برای یک زیر کلاس وجود ندارد تا مستقیماً به این متغیرهای مربوط به خودش دسترسی داشته و یا آنها را مقداردهی اولیه نماید.

از آنجائیکه کپسول سازی یک خصلت اولیه oop است، پس باعث تعجب نیست که جاوا راه حلی برای این مشکل فراهم کرده باشد. هرگاه لازم باشد تا یک زیرکلاس به کلاس بالای قبلی خودش ارجاع نماید، اینکار با استفاده از واژه کلیدی super انجام میدهیم.

Super دو شکل عمومی دارد. اولین آن سازنده کلاس بالا را فراخوانی میکند.

دومین آن بمنظور دسترسی به یک عضو کلاس بالا که توسط یک عضو زیر کلاس مخفی مانده است، استفاده میشود.

۶-۲-۲ استفاده از Super

یک زیر کلاس میتواند روش سازنده تعریف شده توسط کلاس بالای مربوطه را با استفاده از این شکل super فراخوانی نماید:

```
Super(parameter-list);
```

در اینجا parameter-list مشخص کننده هر پارامتری است که توسط سازنده در کلاس بالا مورد نیاز باشد.

super() باید همواره اولین دستور اجرا شده داخل یک سازنده زیر کلاس باشد.

در کد زیر نحوه استفاده از super() آورده شده است.

```
class clsSuper{       سوپر کلاس
int i;
int j;
    public clsSuper(int i,int j) {
    this.i=i;
    this.j=j;
    }
}
class clsSub extends clsSuper{      زیر کلاس
int k;
    public clsSub(int i,int j,int k) {
       super(i, j);     سازنده کلاس بالا فراخواني ميگردد
        this.k=k;
    }
public showij(){
system.out.println("i j : "+super.i+" ",super.j);
}
...
}
```

نکته ۱ : کلمه کلیدی super به کلاس بالا (superclass) ارجاع میکند.

۶-۳ ایجاد یک سلسله مراتب چند سطحی[1]

میتوانید سلسله مراتبی بسازید که شامل چندین لایه وراثت بدلخواه شما باشند. کاملاً واضح است که از یک زیر کلاس بعنوان کلاس بالای یک کلاس دیگر استفاده کنیم.

[1] Multilevel

بعنوان مثال اگر سه کلاس C،B،A داشته باشیم آنگاه C میتواند یک زیر کلاس از B و یک زیر کلاس از A باشد. وقتی چنین شرایطی اتفاق می افتد، هر زیر کلاس کلیه خصلتهای موجود در کلیه کلاس بالاهای خود را به ارث میبرد. در این شرایط، C کلیه جنبه های B،A را به ارث میبرد.

به برنامه زیر توجه کنید.

```
public class A {
int var_a;
}
-----------------------------------------------------------------
public class B extends  A{
int var_b;
}
-----------------------------------------------------------------
public class C  extends  B{
int var_c;
  public C(int a,int b,int c) {
 super.var_a=a;
 super.var_b=b;
 this.var_c=c;
  }
void sum()
{ System.out.println("a+b+c : "+super.var_a+super.var_b+this.var_c);
  }
}
       استفاده از کلاس
C cls_C = new C(5, 10, 15);
cls_C.sum();
```

خروجی برنامه

```
a+b+c : 30
```

۶-۳-۱ زمان فراخوانی Constructor ها

وقتی یک سلسله مراتب کلاس ایجاد میشود، سازندگان کلاسها، که سلسله مراتب را تشکیل میدهند به چه ترتیبی فراخوانی میشوند؟ به عنوان مثال، با یک زیر کلاس تحت نام B و یک کلاس بالا تحت نام A، آیا سازنده A قبل از سازنده B فراخوانی میشود، یا بربالعکس؟

پاسخ این است که در یک سلسله مراتب کلاس، سازندگان بترتیب مشتق شدنشان از کلاس بالا به زیر کلاس فراخوانی میشوند. بعلاوه چون super() باید اولین دستوری باشد که در یک سازنده زیر کلاس اجرا میشود،

این ترتیب همانطور حفظ میشود، خواه super() استفاده شود یا نشود. اگر super() استفاده نشود آنگاه سازنده پیشفرض با سازنده بدون پارامتر هریک از زیر کلاسها اجرا خواهند شد.

برنامه زیر نشان میدهد که چه زمانی سازندگان اجرا میشوند:

```
public class A {
    public A() {
        System.out.println("A's Constructor");
    }
}
-----------------------------------------------------------
public class B extends  A{
   public B() {
        System.out.println("B's Constructor");
    }
}
-----------------------------------------------------------
public class C  extends  B{
    public C() {
        System.out.println("C's Constructor");
    }
}
       استفاده از کلاس
C cls_C = new C();
```

خروجی برنامه

```
A's Constructor
B's Constructor
C's Constructor
```

همانطوریکه مشاهده میکنید، سازندگان بترتیب مشتق شدنشان فراخوانی میشوند.

۶-۳-۲Override[1] کردن متدها

در یک سلسله مراتب کلاس، وقتی یک متد در یک زیرکلاس همان نام و نوع یک متد موجود در کلاس بالای خود را داشته باشد، آنگاه میگویند آن متد در زیرکلاس، متد موجود در کلاس بالا را لغو نموده یا از پیشروی آن جلوگیری مینماید.

[1] لغو کردن

مورد زیر را درنظر بگیرید.

```
public class A {
  void show(){
  System.out.println("Method's A");
  }
}
-----------------------------------------------------------------
public class B extends  A{
  void show(){
  System.out.println("Method's B");
  }
}
       استفاده از کلاس
B cls_B = new B();
cls_B.show();
```

خروجی برنامه

```
Method's B
```

در این مثال متد show()، Override شده است.

۶-۴ استفاده از کلاسهای مجرد[1]

شرایطی وجود دارد که میخواهید یک کلاس بالا تعریف نمایید که ساختار یک انتزاع معین را بدون یک پیاده سازی کامل از هر متدی، اعلان نماید. یعنی گاهی میخواهید یک کلاس بالا ایجاد کنید که فقط یک شکل عمومی شده را تعریف کند که توسط کلیه زیر کلاسهایش ب ه اشتراک گذاشته خواهد شد و پرکردن جزئیات این شکل عمومی بعهده هریک از زیر کلاسها واگذار میشود.

یک چنین کلاسی طبیعت متدهایی که زیر کلاسها باید پیاده سازی نمایند را تعریف میکند. یک شیوه برای وقوع این شرایط زمانی است که زیر کلاس بالا توانایی ایجاد یک پیاده سازی با معنی برای یک متد را نداشته باشد.

با تعریف متد مجرد در کلاس بالا، یک متدی بیمعنا ایجاد میشود که در واقع هیچ عملی را انجام نخواهد داد. برای معنا بخشیدن به این متد بایستی از متدهای زیر کلاس که عمل لغو متد را انجام میدهند استفاده نمود.

[1] abstract

برای اعلان یک متد مجرد،از شکل عمومی زیر استفاده نمایید.

```
abstract  type name(parameter-list)
```

برای اعلان یک کلاس بعنوان مجرد،بسادگی از واژه کلیدی abstract در جلوی واژه کلیدی class در ابتدای اعلان کلاس استفاده مینمایید.

خصوصیات یک کلاس مجرد

۱ - برای یک کلاس مجرد هیچ شیئی نمیتوان ایجاد نمود.یعنی یک کلاس مجرد نباید به طور مستقیم با عملگر new نمونه سازی شود.

۲ - همچنین نمیتوانید سازندگان مجرد یا متدهای ایستای(static) مجرد اعلان نمایید.

۳ - هر زیر کلاس از یک کلاس مجرد باید یا کلیه متدهای مجرد موجود در کلاس بالا را پیاده سازی نماید،و یا خودی بعنوام یک abstract اعلان شود.

در مثال زیر از یک کلاس و متد مجرد استفاده شده است.

```
abstract  class  A{
abstract  void callMe();
void showMessage(){
System.out.println("Abstract Class");
  }
}
-----------------------------------------------------------
public class B extends  A{
 void callMe(){     بایستي از این متد در کلاس استفاده گردد
 System.out.println("Abstract Method");
 }
}
-----------------------------------------------------------
       استفاده از کلاس
B cls_B = new B();
cls_B.callMe();
cls_B.showMessage();
```

خروجی برنامه

```
Abstract Method
Abstract Class
```

همانطور که مشاهده میکنید چونکه از متد callMe به صورت مجرد در کلاس A تعریف شده است.بایستی این متد در کلاس B تعریف گردد.

۶–۵ استفاده از Final با وراثت

واژه کلیدی final میتواند در چندین کاربرد استفاده گردد.یکی از این کاربردها،استفاده از ان در وراثت برای جلوگیری از لغو متد در کلاس بالا (superclass) میباشد.

با قراردادن کلمه کلیدی final در ابتدای متد از لغو ان جلوگیری میشود.

به قطعه زیر توجه کنید

```
class  A{
final void meth(){
System.out.println("this is a final method!!!");
  }
}
-----------------------------------------------------------
public class B extends  A{
void meth(){       Error! Can't Override
System.out.println("Illegal!");
  }
}
```

چون meth() بعنوان final اعلان شده،نمیتوان ان را در B لغو نمود.اگر چنین تلاشی بکنید،نتیجه یک خطلای compile-time خواهد بود.متدها اعلان شده بعنوان final گاهی میتوانند باعث افزایش عملکرد شوند.

۶–۵–۱ استفاده از final برای جلوگیری از وراثت

گاهی ممکن است بخواهید مانع ارث بردن از یک کلاس بشوید.

برای انجام اینکار قبل از اعلان کلاس از واژه کلیدی final استفاده نمایید.

اعلان نمودن یک کلاس بعنوان final بطور ضمنی کلیه متدهای ان نیز بعنوان final اعلان میکند. حتما میدانید که اعلان یک کلاس بعنوان هم abstract و هم final غیر مجاز است،چون یک کلاس abstract بتنهایی کامل نبوده و برای تکمیل پیاده سازیها متکی به زیر کلاسهای خود میباشد.

در اینجا یک کلاس final را مشاهده میکنید:

```
final class  A{
//...
}
```

```
-------------------------------------------------------------
public class B extends  A{  //Error! Can't subclass A
//...
}
```

همانطوریکه از "توضیحات" مشخص شده،ارث بردن B از A غیر مجاز شده،چون A بعنوان final ،اعلان شده است.

۶–۶ بسته ها[1]

قبلا نام هر مثالی از کلاس را از یک فضای نام مخصوص [2] میگرفتیم. یعنی برای هر کلاس باید یک نام منحصر به فرد استفاده میشد تا از اختلاط نامها جلوگیری شود. بدون یک راه مناسب برای مدیریت name space، پس از مدتی دچار مشکلاتی خواهید شد و مجبورید برای هر کلاس منفرد، اسامی توصیفی و مشکلی اختیار کنید. همچنین نیازمند راهی هستید تا مطمئن شوید که نامیکه برای یک کلاس انتخاب میکنید بطور منطقی منحصر بفرد بوده و با نام کلاسهای انتخاب شده توسط سایر برنامه نویسان تصادم پیدا نخواهد کرد. خوشبختانه جاوا مکانیسمی برای بخش بندی name space به قطعات قابل توجه و قابل مدیریت فراهم آورده است. این مکانیسم همان بسته یا package است. بسته هم یک روش نامگذاری و هم یک مکانیسم کنترل رویت پذیری است. میتوانید کلاسهایی را داخل یک بسته تعریف کنید که به وسیله کدهای خارج از بسته قابل دسترسی نباشند.

[1] packages
[2] name space

همچنین میتوانید اعضای کلاسی را تعریف کنید که فقط در معرض رویت سایر اعضای همان بسته قرار داشته باشند.این امر به کلاسهای شما امکان میدهد تا یک اگاهی درونی از یکدیگر داشته باند اما ان اطلاعات رابرای دیگران افشاگری نکنند.

۶-۶-۱ تعریف بسته

ایجاد یک بسته بسیار اسان است:

خیلی ساده یک فرمان package بعنوان اولین دستور در فایل منبع جاوا بگنجانید.هرکلاس اعلان شده داخل ان فایل به بسته مشخص شده تعلق خواهد داشت.

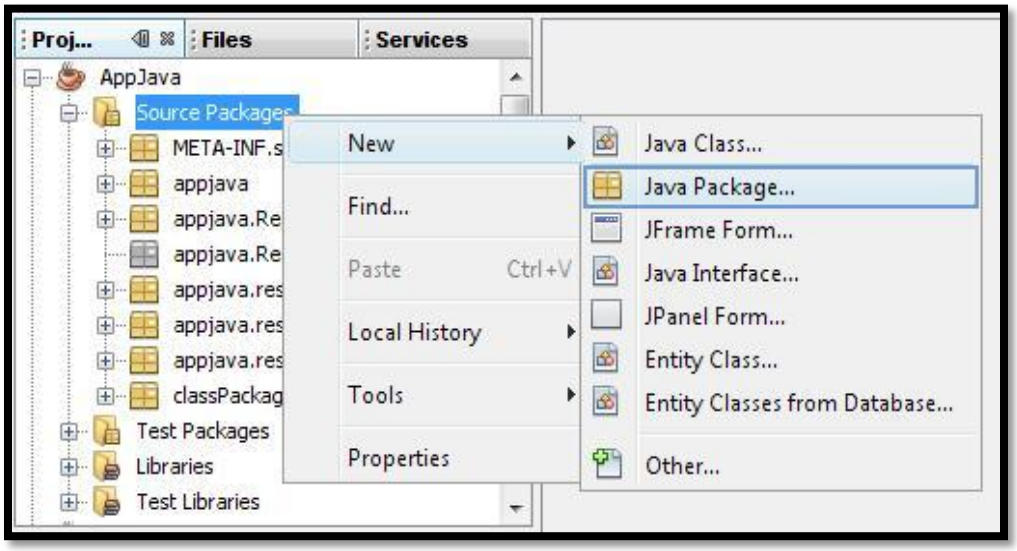
دستور package یک name space را تعریف میکند که کلاسها داخل ان ذخیره میشوند.

اگر دستور فوق را حذف کنید،اسامی کلاس در بسته پیشفرض قرار میگیرند،که هیچ اسمی ندارد.

شکل عمومی دستور package بصورت زیر است:

```
Package name-pack
```

در نت بین برای اضافه کردن یک بسته کافی است ان را به پروژه اضافه کنید(شکل ۶-۱)



شکل (۶- ۱) : اضافه کردن بسته

شما میتوانید در هر قمسمتی که خواستید بسته را اضافه نمایید.

میتوانید یک سلسله مراتب از بسته بسازید.برای انجام اینکار،خیلی ساده اسم هر بسته را از اسم بالایی اش و با استفاده از یک نقطه جدا سازید.

شکل عمومی یک دستور package چند سطحی بقرار زیر است:

```
Package name-pack1[.name-pack2][.name-pack3];
```

سلسله مراتب بسته باید در فایل سیستم توسعه جاوا منعکس شود.

بعنوان مثال یک بسته اعلان شده بعنوان

Package java.awt.image;

باید در java/awt/image، java\awt\image یا java:awt:image بترتیب روی فایل سیستم unix،windows،یا macintosh ذخیره شود.

۶–۷ رابطها[5]

با استفاده از واژه کلیدی interface،میتوانید رابط یک کلاس را از پیاده سازی ان کلاس بطور کامل مجرد نمایید.یعنی با استفاده از interface میتوانید مشخص کنید یک کلاس چکاری باید انجام دهد،اما چگونگی انرا مشخص نخواهید کرد.

رابطها از نظر قواعد صرف ونحو مشابه کلاسها هستند،اما فاقد متغیرهای نمونه هستند و متدهای انها بدون بدنه اعلان میشود.در عمل این بدان معناست که میتوانید رابطهایی تعریف کنید که درباره چگونگی پیاده سازی خود فرضیه نمیسازند.

---

[5] interfaces

هر بار که رابط تعریف شود،هر تعدادی از کلاسها میتوانند ان رابط interface را پیاده سازی نمایند.

همچنین،یک کلاس میتواند هد تعداد رابطها را پیاده سازی نماید.

برای پیاده سازی یک رابط،کلاس باید مجموعه کامل متدهای تعریف شده توسط رابط را ایجاد نماید.اما،هر کلاسی ازاد است تا جزئیات پیاده سازی خودش را تعیین نماید. با استفاده از واژه کلیدی interface، جاوا به شما امکان میدهد تا حداکثر بهره از جنبه "یک رابط و چندین متد" در چند شکلی را بدست اورید.

رابطها بمنظور حمایت از سرگیری پویای متد در حین اجرا طراحی میشوند.به طور معمول،برای اینکه یک متد از یک کلاس به کلاس دیگر فراخوانی شود،هر دو کلاس باید در زمان کامپایل حضور داشته باشند،بطوریکه کامپایلر جاوا بتواند انها را کنترل نموده و سازگاری متدها را تایید نماید.این امر بتنهایی منجر به یک محیط کلاس دهی ایستا و غیر قابل توسعه خواهد شد.

در چنین سیستمی به ناچار عملگرایی[6] در سلسله مراتب کلاس بالاتر و بالاتر میرود بطوریکه مکانیسم فوق، در دسترس زیر کلاسهای بیشتری قرار خواهد گرفت.

رابطها طراحی میشوند تا از بروز چنین مشکلاتی جلوگیری بعمل اورند.

انها قطع ارتباط بین تعریف یک متد یا مجموعه از متد و سلسله مراتب وراثت را بوجود می اورند.

از انجائیکه رابطها در سلسله مراتب متفاوتی از کلایها قرار گرفته اند،برای کلاسهایی که بر حسب سلسله مراتب کلاس مامرتط هستند،امکان پیاده سازی همان رابط وجود دارد.این نقطه ای است که قدرت واقعی رابطها را نمایان میسازد.

نکته2: رابطها،ان میزان عملگرایی لازم برای بسیاری از کاربردها را بوجود می اورند که در غیر اینصورت ناچار از متوسل شده به وراثت چندگانه در زبانی نظیر C++ خواهیم داشت .

---

[6] functionality

۶-۷-۱ تعریف نمودن یک رابط

یک رابط مشابه یک کلاس تعریف میشود.

شکل عمومی یک رابط بصورت زیر میباشد.

```
Access interface name{
Return-type method-name1(parameter-list);
Return-type method-name2(parameter-list);
Type final-varname1=value;
Type final-varname2=value;
...
Return-type method-nameN(parameter-list);
Type final-varnameN=value;
}
```

در اینجا،دسترسی(access) یا public است یا اصلا بکار نمیرود.

وقتی که مشخصگر دسترسی نگنجانیم،انگاه دسترسی پیشفرض اعمال شده و رابط فقط برای سایر اعضای بسته ای که در ان اعلان انجام گرفته،در دسترس خواهد بود.

وقتی کع بعنوان public اعلان میشود،رابط،توسط هر کد دیگری قابل استفاده خواهد بود.name همان نام رابط است و بجای ان هر شناسه معتبری را میتوان بکار برد.توجه کنید که متدهای اعلان شده دارای بدنه نیستند.انتهای انها بعد از parameter-list یک (;)قرار میگیرد و انها ضرورتا متدهای مجرد هستند.ممکن است هیچ پیاده سازی پیشفرضی برای متدهای مشخص شده داخل یم رابط وجود نداشته باشد.

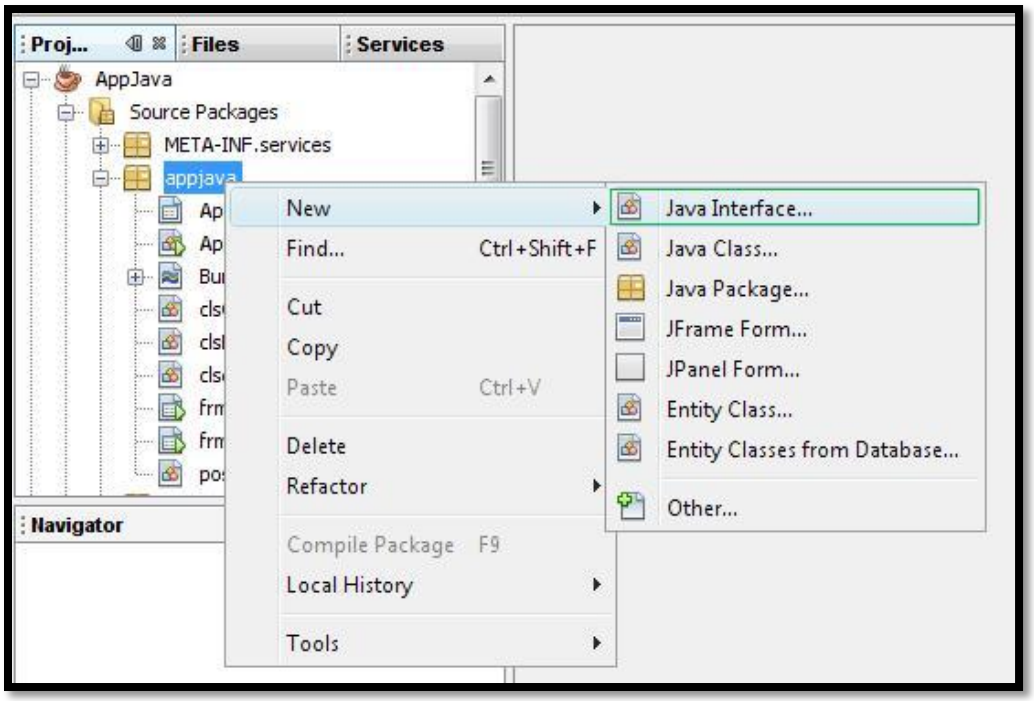
نکته3 : هر کلاسی که دربر گیرنده یک رابط باشد،باید کلیه متدها را پیاده سازی نماید.

متغیرها را میتوان داخل اعلانات رابط،اعلان نمود.انها به طور ضمنی finall و static هستند.بدیم معنی که انها توسط کلاس پیاده سازی شده قابل تغییر نیستند.انها همچنین باید با یک مقدار ثبت،مقداردهی اولیه شوند.

اگر رابط خودش بعنوان یک public اعلان شده باشد،کلیه متدها و متغیرها بطور ضمنی public خواهند بود.

۶-۷-۲ اضافه کردن Interface به پروژه

برای اضافه کردن یک رابط به پروژه،کافیست روی پروژه راست کلیک کرده و گزینه ی java Interface راازمنوی new انتخاب نمایید.(شکل ۶-۲)



شکل (۶- ۲) : اضافه کردن رابط به پروژه

در کادر ظاهرشده New java Interface،در قسمت Class name نام رابط را وارد نموده و دگمه finish را کلیک کنید.اکنون مثالی از یک تعریف رابط را مشاهده میکینید.

این مثال یک رابط ساده را تعریف میکند که دربرگیرنده متدی تحت نام callback() است که یک پارامتر تکی عدد صحیح را میگیرد.

```
public interface Callback {
void callback(int param);
}
```

۶-۷-۳ پیاده سازی رابطها

هربار رابطی را تعریف نمایید،یک یا چندین کلاس میتوانند ان رابط را پیاده سازی نمایند.برای پیاده سازی یک رابط،جمله implements را در تعریف کلاس بگنجانیدو سپس متدهای تعریف شده توسط رابط را ایجاد نمایید.

شکل عمومی یک کلاس که در بر گیرنده جمله implements است،مشابه مورد زیر میباشد:

```
Access class classname [extends superclass][implements interface [,interface]]
{
//class-body
}
```

اگر یک کلاس بیش از یک رابط را پیاده سازی نماید،رابطها را با علامت کاما(،) از یکدیگر جدا میکنیم.

اگر یک کلاس دو رابط را که متد یکسانی را اعلان میکنند،پیاده سازی نماید،انگاه ان متد یکسان توسط سرویس گردندگان هریک از رابطها استفاده خواهد شد.

متدهایی که یک رابط را پیاده سازی میکنند باید به عنوان public اعلان شوند.

همچنین تایید نوع متد پیاده سازی باید دقیقا با تایید نوع مشخص شده در تعریف interface مطابقت و سازگاری داشته باشد.

در اینجا مثال ساده ای مشاهده میکنید که رابط callback را پیاده سازی مینماید:

```
public class Client implements CallBack {
public void callBack(int p){
    System.out.println("callback called with "+p);
}
public void showMSG(){
System.out.println("Client Class ...");
}
}
```

دقت داشته باشید که callback() با استفاده از مشخصگر دسترسی public اعلان شده است.

نکته۴: وقتی که یک متد رابط را پیاده سازی میکنید، بعنوان public اعلان خواهد شد.

برای کلاسهایی که رابطها را پیاده سازی میکنند،هم مجاز و هم رایج است که اعضا اضافی برا خودشان تعریف نمایند (همانند متد showMSG())

۶–۸ دسترسی به پیاده سازیها از طریق ارجاعات رابط

میتوانید متغیرهایی را بعنوان ارجاعات شی اعلان کنید که بجای یک نوع کلاس از یک رابط استفاده نمایند. هر نمونه ای از کلاسی که رابط اعلان شده را پیاده سازی میکند را میتوان در چنان متغیری ذخیره نمود.وقتی یک متد را از طریق یکی از این ارجاعات فراخوانی میکنید،روایت صحیح برای اساس نمونه واقعی رابطی که به ان ارجاع شده،فراخوانی خواهد شد.این یکی از جنبه های کلیدی رابطهاست.

متدی که باید اجرا شود در حین اجرا بصورت پویا مورد جستجو قرار میگیرد و به کلاسها اجازه میدهد تا دیرتر از کدی که متدها را روی ان فراخوانی میکند،ایجاد شوند.این پردازش مشابه استفاده از یک ارجاع کلاس بالا برای دسترسی به یک شی زیر کلاس است.

در این مثال متد callback() را از طریق یک متغیر ارجاع رابط،فراخوانی میکند:

```
public static void main (String arg[]){
CallBack c = new Client();
c.callBack(4);
  }
}
```

خروجی برنامه

```
callback called with 4
```

دقت کنید که متغیر C طوری تعریف شده که نوع رابط callback باشد.با این وجود یک نمونه از client به ان منتسب شد.اگرچه میتوان از C برای دسترسی به متد callback() استفاده نمود،اما نمیتواند به هیچیک از اعضای کلاس client دسترسی داشته باشد.یک متغیر ارجاع رابط فقط از متدهای اعلان شده توسط اعلان interface خود،اگاهی دارد.بدین ترتیب،نمیتوان از C برای دسترسی به متد showMSG() استفاده نمود چون توسط client و نه توسط callback تعریف شده است.

۶–۸–۱ پیاده سازی نسبی[7]

اگر یک کلاس در برگیرنده یک رابط باشد،اما متد تعریف شده توسط ان رابط را کاملا پیاده سازی نکند،انگاه ان کلاس باید بعنوان abstract اعلان شود. بعنوان مثال:

```
abstract  public class Incomplete implements  CallBack {
int a,b;
void showab(){
  System.out.println("a b : "+a+" "+b);
  }
}
```

در اینجا کلاس Incomplete متد ()callback را پیاده سازی نمیکند و باید به عنوان abstract اعلان شود.هرکلاسی که از Incomplete ارث میبرد یا متد ()callback را پیاده سازی نماید و یا خودش بعنوان abstract اعلان گردد.

۶–۹ متغیرها در رابطها

میتوانید از رابطه ها برای واردکردن ثابتهای به اشتراک گذاشته شده به کلاسهای چندگانه بسادگی از اعلان یک رابط که در بر گیرنده متغیرهایی باشد که با مقادیر دلخواه مقداردهی اولیه شده باشند،استفاده نمایید. وقتی که ان رابط را در یک کلاس میگنجانید(یعنی وقتی که رابط پیاده سازی میکنید) کلیه اسامی ان متغیرها بعنوان ثابت ها در قلمرو خواهند بود.

این کار مشابه استفاده از فایل header در C++،C برای ایجاد یک رقم بزرگ از ثابتهای #defined و با اعلانات const میباشد.

---

[7] partial

اگر یک رابط در برگیرنده هیچ متدی نباشد،انگاه هر کلاسی که در برگیرنده ان رابط باشد در واقع چیزی را پیاده سازی نمیکند.مثل این است که ان کلاس متغیرهای ثابت را به name space کلاس بعنوان متغیرهای final وارد میکرده است.در مثال زیر نحوه استفاده از متغیرهادر رابط امده است.

```
public interface inter {
int yes=6;
int no=7;
}
-------------------------------------------------------------
public class clsret implements inter {
void getvalue(int i){
if(i<=6)  System.out.println("Yes : "+this.yes);
else System.out.println("No : "+this.no);
  }
}
کار با کلاس
  clsret cls = new clsret();
  cls.getvalue(5);
```

خروجی برنامه

```
Yes : 6
```

۶–۱۰ رابطه ها را میتوان گسترش داد

یک رابط با استفاده از واژه کلیدی extends میتواند از یک رابط دیگر ارث ببرد.دستور زبان مشابه کلاسهاییست، که ارث برده اند.

وقتی یک کلاسی رابطی را پیاده سازی میکند که از رابط دیگری ارث برده است،باید پیاده سازیهای کلیه متدهای تعریف شده داخل زنجیر وراثت را فراهم نماید.

مثال زیر را مشاهده کنید:

```
interface  A{
void meth1();
void meth2();
}
-------------------------------------------------------------
interface B extends A{
void meth3();
}
-------------------------------------------------------------

public class C implements B {
```

```
    public void meth1() {
     System.out.println("Implement Method1");
    }
    public void meth2() {
     System.out.println("Implement Method2");
    }
    public void meth3() {
     System.out.println("Implement Method3");
    }
}
کار با کلاس
public static void main (String arg[]){
C cls = new C();
cls.meth1();
cls.meth2();
cls.meth3();
 }
```

خروجی برنامه

```
Implement Method1
Implement Method2
Implement Method3
```

هر کلاسی که یک رابط را پیاده سازی میکند باید کلیه متدهای تعریف شده توسط ان رابط، شامل هر کدام از سایر رابطهایی که ارث برده اند را پیاده سازی نماید.

فصل هفتم

کلاسهای Swing

۷-۱ معرفی ابزارهای تولید واسط های گرافیکی در جاوا Swing،AWT

در برنامه‌های کاربردی جاوا به منظور ایجاد یک واسط گرافیکی کاربر یا از Graphical User Interface(GUI) یا از JFC ، Java Foundation Classes استفاده می‌شود.

سئوال) چرا در برنامه های کاربردی نیاز به ایجاد یک واسط گرافیکی کاربر داریم؟

برای پاسخ به این سئوال می توان به برنامه هایی که در محیط Command یا محیط تحت DOS اجرا می شوند اشاره نمود. در اینگونه برنامه ها برای انجام هر عملی باید دستورات خاص آن را تایپ نموده تا عمل مورد نظر اجرا گردد. در چنین حالتی به علت اینکه به ازای هر یک از اعمال مورد نظر باید دستور یا دستوراتی را تایپ کنیم سرعت العمل برنامه کاه‌ش می یابد و هم با یک محیط خشک و بی روح و کسل کننده رو به رو می باشبم. برای رفع این مشکل مهندسان کامپیوتر واسط گرافیکی کاربر یا GUI را ایجاد نمودند. به کمک این واسط:

«۱» محیط برنامه کاربردی از حالت خشک و بی روح به یک محیط گرافیکی زیبا تبدیل می گردد.

«۲» برای اجرای دستورات مورد نظر کافیست به کمک ماوس یا صفحه کلید فقط روی شی یا اشیاء مورد نظر کلیک نمایید. مثلا برای چاپ یک گزارش کافیست روی دکمه Print کلیک نمایید.

سئوال) واسط گرافیکی کاربرپسند « User Friendly » چگونه واسطی است؟

اگر در زمان طراحی واسط های گرافیکی کاربر، اصول طراحی این واسط ها را رعایت کنیم، نتیجه کار یک واسط گرافیکی کاربرپسند خواهد بود. به عنوان مثال می توان به سیستم عامل ویندوز اشاره نمود. دلیل اصلی موفقیت این سیستم عامل رعایت این اصول می باشد که سبب راحتی کار با امکانات آن گردیده و در نتیجه در بین کاربران محبوبیت فراوانی را کسب نموده است.

از جمله این اصول می توان به موارد زیر اشاره نمود:

«۱» انتخاب کامپوننت های مناسب برای برنامه مورد نظر.

«۲» قرار دادن کامپوننت ها در مکان مناسب.

«۳» انتخاب ترکیب رنگی، سایز و اندازه مناسب برای کامپوننت ها.

«۴» قرار دادن توضیحات کافی « راهنما » برای انجام هر عمل به گونه ای که کاربر نهایی در زمان برخورد با مشکلات خود را تنها احساس نکند.

«۵» پرهیز از دوباره کاری و انجام اعمال تکراری بصورت خودکار توسط برنامه.

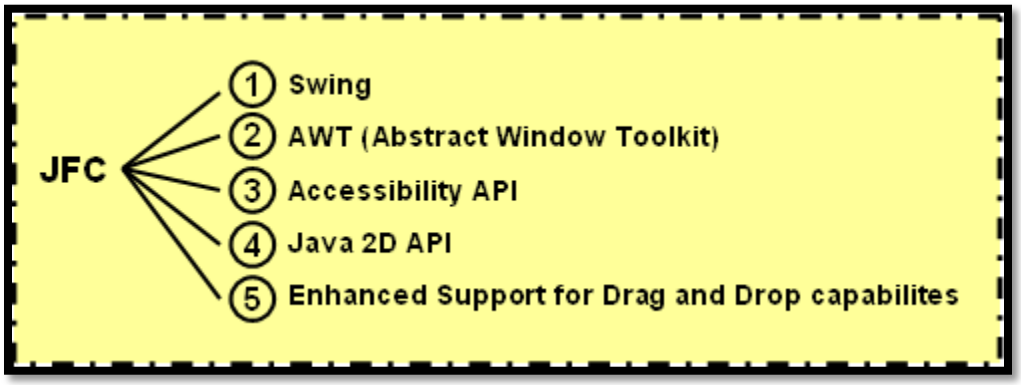
«۶» کوتاه کردن مسیر انجام عملیات مورد نظر کاربر.

«۷»...

همانطور که گفته شد برای پیاده سازی GUI ها از JFC استفاده می شود.

JFC در حقیقت یک مجموعه از کتابخانه های طراحی شده برای کمک به برنامه نویسان جاوا جهت توسعه برنامه های کاربردی تجاری و با ابعاد گسترده می باشد.

JFC از پنج بخش زیر تشکیل شده است: (شکل ۷-۱)



شکل (۷- 1) : بخشهای JFC

هر یک از این پنج بخش شامل مجموعه ای از کلاس ها جهت طراحی GUI می باشد.

مثلا برای کنترل و مدیریت رویدادهای (Event) ماوس یا صفحه کلید در واسط های کاربر، از AWT یا Swing می توان استفاده نمود و یا برای شماتیک کردن برنامه مانند پیاده سازی نمودارها، گراف ها، تصاویر و بطور کلی گرافیک دو بعدی از Java 2D می توان کمک گرفت.

در میان این مجموعه ابزارها، دو ابزار AWT و Swing برای ویژوالی کردن برنامه مورد استفاده قرار می گیرند. یعنی از هریک از دو مجموعه فوق می توان به منظور ویژوال سازی برنامه به طور مستقل از مجموعه دیگر استفاده نمود.

۷-۲ Swing چیست ؟

همانطور که می دانیم Swing قسمتی از JFC می باشد. بسته Swing شامل یک مجموعه از کامپوننت ها برای ایجاد GUI ها و اضافه کردن قابلیت محاوره ای به برنامه های کاربردی جاوا مورد استفاده قرار می گیرد. Swing شامل ابزارهای مفید و پرکاربردی مانند Table، List، Tree Controls و بسیاری ابزارهای دیگر می باشد. Swing سرنام یا مخفف عبارت خاصی نمی باشد.

پروژه Swing در اواخر سال ۱۹۹۶ شروع و در سال ۱۹۹۸ نسخه اول آن عرضه گردید. نسخه ۱.۰ این بسته شامل حدودا ۲۵۰ کلاس و ۸۰ واسط می باشد. swing 1.0 برای اجرا شدن به JDK 1.1.5 نیاز داشت.

Swing شامل ۱۸ بسته (Package) اصلی به شرح زیر می باشد:

javax.swing.text	javax.swing.plaf	javax.accessibility
javax.swing.text.html	javax.swing.plaf.basic	javax.swing
javax.swing.text.html.parser	javax.swing.plaf.metal	javax.swing.border
javax.swing.text.rtf	javax.swing.plaf.multi	javax.swing.event
javax.swing.tree	javax.swing.plaf.synth	javax.swing.colorchooser
javax.swing.undo	javax.swing.table	javax.swing.filechooser

جدول (۷- 1) : بسته های Swing

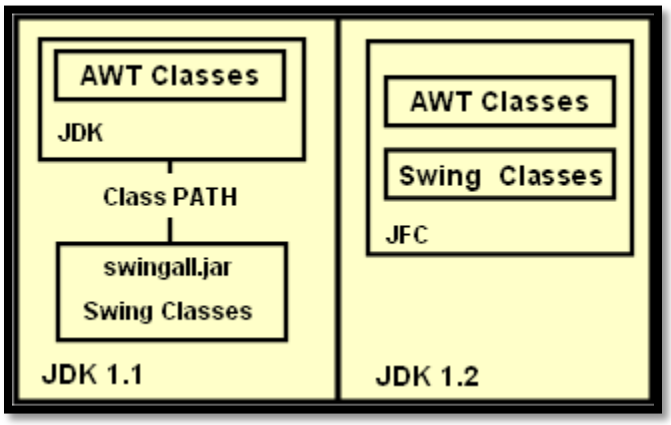
در بین بسته های فوق دو بسته ای که با رنگ زرد نشان داده شده اند بیشترین کاربرد را دارا می باشند.

سئوال) می دانیم که AWT نیز مانند Swing برای توسعه GUI ها مورد استفاده قرار می گیرد. حال سئوال اینجاست که چرا با وجود AWT به Swing نیاز می باشد؟ آیا Swing جایگزینی برای AWT می باشد؟

برای پاسخ به سئوالات فوق باید به نکات زیر دقت نمایید:

«۱» در زمان توسعه اولین نسخه جاوا، AWT نیز به همراه آن به عنوان ابزار طراحی GUI های مورد نیاز برنامه نویسان ارائه گردید. (اولین نسخه Swing در سال ۱۹۹۸ ارائه گردید) پس ابزارهای بیشتری از آن پشتیبانی می کنند. مثلا Applet هایی که با Awt نوشته می شوند در مرورگرهای بیشتری اجرا می‌شوند.

«۲» ارتباط بین AWT و Swing و JDK نسخه های ۱.۱ و ۱.۲ به بعد به صورت زیر می باشد(شکل ۷-۲)



شکل (۷- 2) : ساختار JDK 1.1 و JDK

«۳» کامپوننت های AWT اصطلاحا HeavyWeight و کامپوننت های Swing اصطلاحا LightWeight نامیده می شوند. تفاوت دو مورد فوق در این است که کامپوننت های HeavyWeight از Component Peer ها استفاده می کنند و در واقع به آنها وابسته اند. در صورتیکه کامپوننت های LightWeight به صورت مستقل از Peer Component ها عمل می کنند.

منظور از Peer Component ها در حقیقت کامپوننت های معادل کامپوننت های جاوا در سیستم عامل مورد نظر (میزبان) می باشد. می دانیم که سیستم عامل ها نیز برای نمایش واسط های گرافیکی خود از یکسری کامپوننت و متد های ویژه و خاص خود استفاده می کنند. همچنین می دانیم که بعضی از زبانهای برنامه نویسی قادر به استفاده از این اجزا می باشند. حال نکته مهم و اساسی در این بین آن است که برنامه هایی که از این کامپوننت ها و متدها استفاده می کنند در واقع به محیط فوق وابسته شده و قابلیت انتقال خود را از دست می دهند.

عمده دلیل تفاوت بین AWT و Swing وجود مشکل فوق در AWT می باشد.

در نتیجه AWT و واسطهای تولیدی با آن در واقع از Native GUI toolkit محیط ها (سیستم عامل های میزبان) خود استفاده می کنند. این امر سبب آن می شود که وقتی واسط کاربری طراحی شده در یک سیستم عامل را در سیستم عامل دیگری اجرا کنیم ممکن است با چیزی که ما در ابتدا طراحی کرده ایم متفاوت باشد.

«۴» کامپوننت های Swing کندتر از AWT بارگذاری (Load) می شوند. در عوض Swing از مدل کنترل رویداد موثرتری نسبت به AWT استفاده می کند. در نتیجه کامپوننت های Swing سریعتر از کامپوننت های مشابه در AWT اجرا می شوند. یعنی سریعتر به رویدادها واکنش نشان می دهند.

۷-۲-۱ ویژگیهای Swing

«۱» نام کامپوننت های Swing با حرف J شروع می شوند. مانند: Jtable،Jtabbed Pane

«۲» کامپوننت های Swing دارای خاصیت Tooltips می باشند.

«۳» Swing قابل حمل تر از AWT می باشد.

«۴» کامپوننت های Swing از قابلیت Undo/Redo پشتیبانی می کنند. مثلا به کمک خاصیت Undo یک عنصر حذف شده از Jtable را می توان باز گرداند و یا یک متن تغییر یافته را در JtextField به حالت قبل بازگرداند.

«۵» Swing شامل کامپوننت های جدید و مفیدی مانند Jtree، JprogressBar ، Jtable، JInternalFrame و... می باشد.

«۶» پشتیبانی از خاصیت Look and Feel یا به اختصار F&L یا LnF توسط Swing که یکی از ویژگی های بسیار مهم در طراحی واسط ه توسط Swing می باشد. توسط این قابلیت، برنامه ها در یک محیط جدید (سیستم عامل دیگر) بدون نیاز به کامپایل مجدد (Run-time) کامپوننت های خود را تنظیم می کنند.
این تنظیمات میتواند به یکی از دو فرم زیر باشد:

۱. به گونه ای عمل کنیم تا واسط طراحی شده در تمامی سیستم عامل ها به یک صورت نمایش داده شوند.(بدون تغییر در ظاهر اجزا)

۲. به گونه ای عمل کنیم تا کامپوننت های واسط طراحی شده در هر سیستم عامل مشابه کامپوننت های سیستم عامل مورد نظر تغییر شکل دهند.

۷-۳ معرفی تکنیک های مدیریت لایه بندی یا چیدمان کامپوننت ها

یکی از بحث های مهم در ساختن یک Graphical User Interface یا یک (GUI) یا به زبان ساده تر یکی از مهم ترین و پر کاربرد ترین مسأله ای که در هنگام ایجاد یک فریم با آن رو برو هستیم، مدیریت لایه بندی یا چیدمان کامپوننت‌های آن فریم (Layout) است.

اینکه بتوانیم فریمی بسازیم که در آن کامپوننت‌های مختلف در جاهای متفاوت در فریم قرار گیرند، به زیبایی و کارایی برنامه ما می افزاید.

در میان کلاس های Swing و AWT، هشت مدل مختلف لایه بندی وجود دارد که می توان به فراخور نیاز هریک از آن ها را بکار برد یا حتی می توان ترکیبی از آنها را بکار برد.

این مدل های لایه بندی عبارتند از:

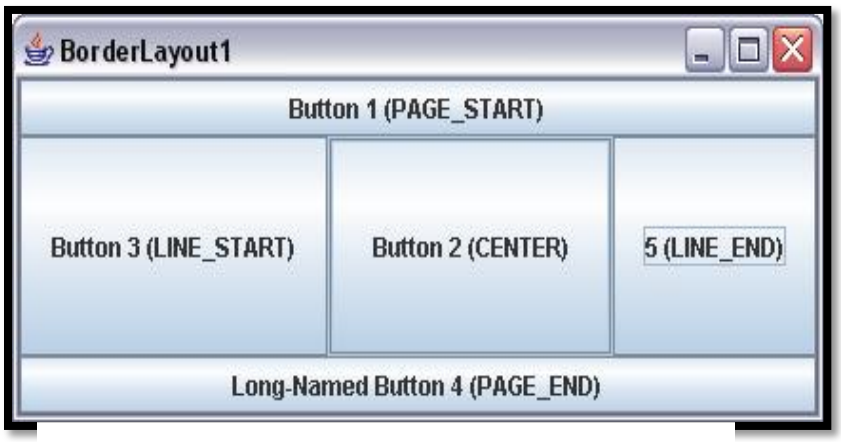
BorderLayout
BoxLayout
CardLayout
FlowLayout

GridBagLayout
GridLayout
GroupLayout
SpringLayout

۷-۳-۱ مدل لایه بندی BorderLayout

BorderLayout یکی از مدل های لایه بندی است که فریم را به پنج قسمت بالا، پایین، چپ، راست و مرکز تقسیم می کند.به عنوان نمونه، برنامه BorderLayout1.java را اجرا نمایید و نتیجه را با تصویر۷-۳ مقایسه نمایید:

این مدل یکی از کلاس های موجود در بسته java.awt می باشد.



شکل (۷- 3) : نتیجه برنامه borderLayout1

نکته ۱ : BorderLayout، لایه بندی پیش فرض برای JApplet و JFrame می باشد.

۷-۳-۱-۱ ایجاد لایه بندی BorderLayout

برای لایه بندی یک فرم در این روش، فقط کافیست که ابتدا فرم را بصورت پنج ناحیه در نظر بگیرید. سپس هریک از اجزا را در محل مورد نظر خود قرار دهید. انجام این عمل بسیار ساده می باشد.

برای انجام این کار بصورت زیر عمل می نماییم.

```
JFrame frame = new JFrame("BorderLayout1");
container= frame.getContentPane();
container.setLayout(new BorderLayout());
JButton button = new JButton("Button 1 (PAGE_START)");
container.add(button, BorderLayout.PAGE_START);
button = new JButton("Button 2 (CENTER)");
button.setPreferredSize(new Dimension(150,100));
container.add(button, BorderLayout.CENTER);
button = new JButton("Button 3 (LINE_START)");
container.add(button, BorderLayout.LINE_START);
```

```
button = new JButton("Long-Named Button 4 (PAGE_END)");
container.add(button, BorderLayout.PAGE_END);
button = new JButton("5 (LINE_END)");
container.add(button, BorderLayout.LINE_END);
```

نکاتی در مورد کدهای بالا:

– به طور کلی برای لایه بندی کردن هر کانتینر با هر مدل لایه بندی، دو راه وجود دارد:

یک راه استفاده از متدسازنده آن مدل لایه بندی است.

```
container= frame.getContentPane();
container.setLayout(new BorderLayout());
```

یا راه دیگر تعریف آن مدل به صورت مستقیم و استفاده از آن است.

```
container= frame.getContentPane();
FlowLayout layout = new FlowLayout ();
container.setLayout(layout);
```

حتی یک شئ JPanel هم می تواند به این دو شیوه لایه بندی شود:

```
JPanel panel = new JPanel(new BorderLayout());
```

نکته دوم در تکه کد بالا، نحوه قرار دادن یک کامپوننت در یکی از پنج ناحیه می باشد.

به عنوان نمونه به کمک عبارت زیر، دکمه ای را در ناحیه وسط فرم اضافه می کنیم.

```
container.add(button, BorderLayout.CENTER);
```

نکته ۲ : بجز کلمه کلیدی CENTER دوازده کلمه کلیدی دیگر نیز برای تعیین جای یک کامپوننت وجود دارد. از جمله این موارد می توان به PAGE_START ، PAGE_END، LINE_START و... اشاره نمود.

مسأله دیگری که در کد های بالا وجود دارد، درباره ی الگوی قسمت اول می باشد:

```
add(component, BorderLayout.CENTER)
```

گاهی می توان کد بالا را به صورت زیر نیز نوشت:

```
add(BorderLayout.CENTER, component)
```

اما هیچ گاه خطای زیر را مرتکب نشوید:

```
add("Center", component)
```

۷-۳-۱-۲ ایجاد فضای خالی میان کامپوننت ها

یکی دیگر از مسائلی که در این نوع لایه بندی وجود دارد، قرار دادن فاصله میان این پنج ناحیه است. متد زیر این کار را انجام می دهد :

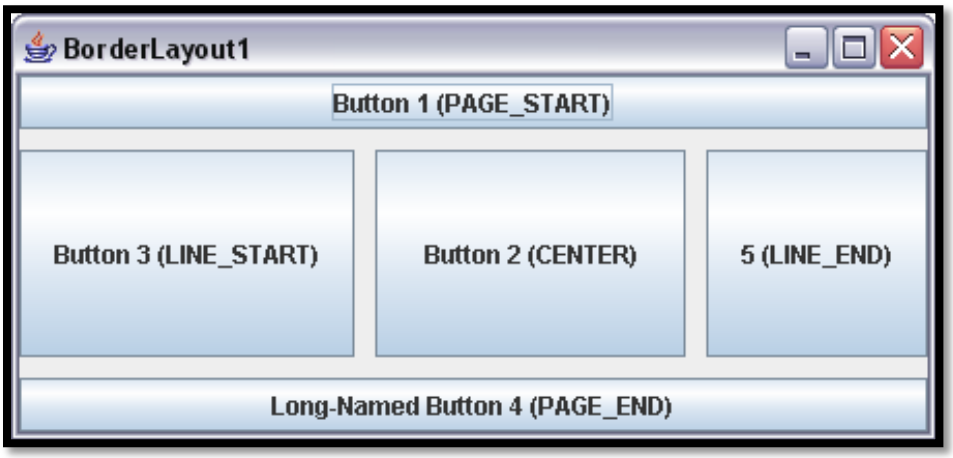
```
BorderLayout(int horizontalGap, int verticalGap)
```

به صورت زیر می توان از این کد استفاده کرد:

```
...
JFrame frame = new JFrame("BorderLayout1");
container= frame.getContentPane();
container.setLayout(new BorderLayout(10,10));
...
```

بوسیله این کد که درمتد سازنده قرارمی گیرد، میان هر کامپوننت می توان فضایی به اندازه ۱۰ پیکسل قرار داد.

تصویر ۷-۴ خروجی برنامه را که در آن تغییر فوق اعمال شده است، نمایش می دهد.



شکل (۷- 4) : خروجی برنامه BorderLayout

همچنین می توان از طریق دو متد زیر نیز عمل فوق را انجام داد:

1- setHgap(int)
2- setVgap(int)

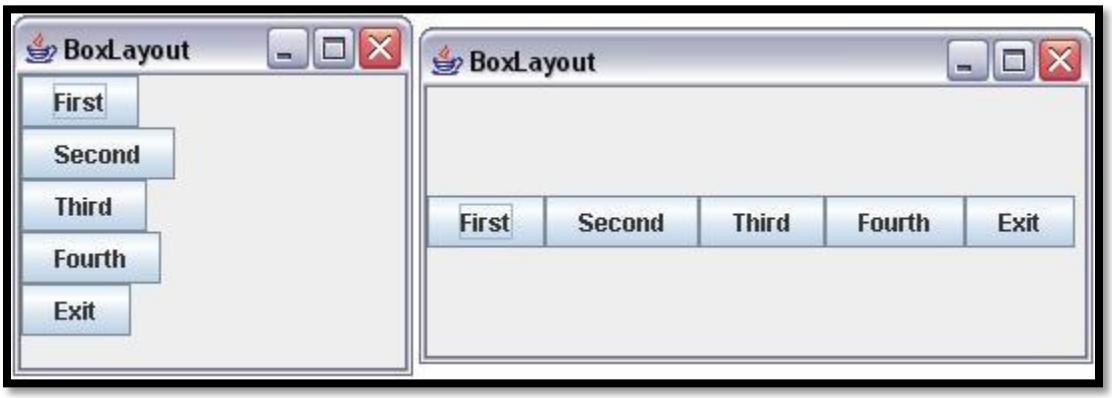
بوسیله این دو متد می توان فاصله افقی یا عمودی میان کامپوننت ها ایجاد کرد. برای این منظور باید قطعه کد زیر را به برنامه اضافه کنیم:

```
...
JFrame frame = new JFrame("BorderLayout1");
container= frame.getContentPane();
BorderLayout layout= new BorderLayout();
container.setLayout(layout);
layout.setHgap(10);
...
```

۷-۳-۲ مدل لایه بندی BoxLayout

یکی دیگر از انواع مدلهای لایه بندی در فریم های جاوا، مدل لایه بندی BoxLayout می باشد. برای شروع کار بهتر است ابتدا برنامه VerticalBox.javaرا اجرا نموده و خروجی آن را بررسی نمایید.

این مدل لایه بندی، یکی از کلاس های بسته javax.swing می باشد.



شکل (۷- 5) : خروجی برنامه BoxLayout

همانطور که می بینید این مدل لایه بندی قادر است کامپوننت ها را در یک ردیف یا یک ستون قرار دهد.

۷-۳-۲-۱ ایجاد لایه بندی BoxLayout

برای ایجاد این نوع لایه بندی، به صورت زیر عمل می کنیم:

```
...
JFrame frame = new JFrame("BoxLayout");
container =frame.getContentPane();
BoxLayout boxlayout = new BoxLayout(container, BoxLayout.Y_AXIS);
container.setLayout(boxlayout);
button = new JButton("First");
container.add(button);
```

```
button = new JButton("Second");
container.add(button);
button = new JButton("Third");
container.add(button);
...
```

کد های بالا قسمتی از برنامه قبل می باشد. در این بخش نحوه لایه بندی از نوع BoxLayout نمایش داده شده است. در تکه برنامه بالا کانتینر بصورتی لایه‌بندی شده است که کامپوننت ها بصورت عمودی و در یک ردیف قرار بگیرند.

«تصویر ۷-۴» حال اگر بخواهیم فرم پس از لایه بندی بصورت تصویر سمت راست شود، بجای عبارت قرمز رنگ مشخص شده در تکه کد فوق، از عبارت AXIS_X.BoxLayout استفاده می نماییم.

اگر بخواهیم کد بالا را به صورت زیر تغییر دهیم، برنامه اجرا نخواهد شد و در خروجي اعلام خطاخواهد کرد:

```
...
        JFrame frame = new JFrame("BoxLayout");
        BoxLayout boxlayout = new BoxLayout( frame , BoxLayout.Y_AXIS);
        frame.setLayout(boxlayout);
Error Message
Exception in thread "main" java.awt.AWTError: BoxLayout can't be shared
at javax.swing.BoxLayout.checkContainer(BoxLayout.java:415)
at javax.swing.BoxLayout.invalidateLayout(BoxLayout.java:202)
Error Message

         button = new JButton("First");
        frame.add(button);
...
```

اما چه دليلي باعث مي شود كه برنامه اعلام خطا كند؟
همانطور كه در قسمت فريم ها گفته شد،
كامپونت ها بطور پيش فرض، درون قاب Content Pane فريم قرار مي گيرند. اما در كد بالا، بجاي اينكه كامپونت ها را درون اين قاب (Content Pane) قرار دهيم در فريم اصلي قرار داده ايم كه مجاز به انجام اين كار نيستيم.

۷-۳-۲-۲ نکاتی در مورد سایز، مکان و Border کامپونت ها در BoxLayout

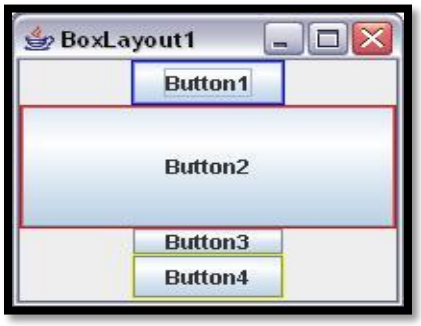
وقتی یک BoxLayout، کامپونت ها را به صورت عمودی یا افقی لایه بندی می کند،

ابتدا برنامه BoxLayout1.java را اجرا نمایید. سپس سعی کنید تا فرم را تغییر اندازه دهید .

خواهید دید که بجز دکمه دوم، اندازه سایر دکمه ها تغییر نخواهد کرد.

کامپوننت ها در سایز عادی خودشان یا Preferred در کانتینر قرار می‌گیرند.

حالت اگر اندازه فرم تغییر



حالت پیش فرض و بدون تغییر اندازه

شکل (۷- ۶) : خروجی برنامه BoxLayout1.java

تکه کد های زیر قسمتی از این برنامه را نشان می دهد:

```
...
Button2 = new JButton("Button2");
Button2.setAlignmentX(Component.CENTER_ALIGNMENT);
Button2.setMaximumSize(new Dimension(Short.MAX_VALUE,Short.MAX_VALUE));
frame.add(Button2);

System.out.println(Button2.getMinimumSize());
System.out.println(Button2.getMaximumSize());
System.out.println(Button2.getPreferredSize());

Button2.setBorder(BorderFactory.createCompoundBorder(
BorderFactory.createLineBorder(Color.RED),Button2.getBorder()));

Button3 = new JButton("Button3");
Button3.setAlignmentX(Component.CENTER_ALIGNMENT);
Button3.setPreferredSize(new Dimension(Short.SIZE,Short.SIZE));
frame.add(Button3);
...
```

در عبارت فوق چند نکته مهم وجود دارد:

۱- در این برنامه برای تعیین اندازه کامپوننت ها از سه متد زیر استفاده شده است:

1-setMaximumSize
2-setMinimumSize
3-setPreferredSize

متد اول برای تعیین حداکثر و متد دوم جهت تعیین حداقل ابعاد کامپوننت مورد نظر بکار می رود.

اما در متد سوم اندازه کامپوننت توسط مدیریت لایه بندی (Manager Layout) یک کانتینر که وظیفه ترسیم یک کامپوننت را بر عهده دارد، تعیین می شود.

محاسبه ابعاد کامپوننت در این روش به عواملی چون سایز فونت داخل کامپوننت ، سایز متن داخل یک کامپوننت ، تصویری که ممکن است درون یک کامپوننت قرار بگیرد، نوع و اندازه Border ی که ممکن است برای یک کامپوننت در نظر گرفته شود و... بستگی دارد.

نکته ۳ : توجه نمایید که بهتر است از این متد برای تعیین ابعاد کامپوننت استفاده شود.

روشهای تغییر سایز یک کامپوننت

a. در روش اول می توان از متد با الگوی set****Size استفاده کرد:

```
comp.setMinimumSize(new Dimension(50, 25));
comp.setPreferredSize(new Dimension(50, 25));
comp.setMaximumSize(new Dimension(Short.MAX_VALUE,Short.MAX_VALUE));
```

b. در روش دوم می توان متد با الگوی get****Size به صورت زیر ابعاد کامپونت را تعیین نمود:

```
public Dimension getMaximumSize() {
    size = getPreferredSize();
    size.width = Short.MAX_VALUE;
    return size;  }
```

۲- استفاده از کلاس Short:

این کلاس دارای چندین مقدار ثابت می باشد.

از جمله این ثوابت، دو ثابت MAX_VALUE، MIN_VALUE می باشد.

a - مقدار MAX_VALUE یا حداکثر مقدار، ثابتی با ارزش 2^{15}-1 می باشد.

b- مقدار MIN_VALUE یا حداقل مقدار، ثابتی با ارزش 2^{15} - می باشد.

۳ -استفاده از سه متد زیر برای نمایش ابعاد تعیین شده در هر یک از سه متد توضیح داده شده در مورد شماره یک.

```
System.out.println(Button2.getMinimumSize());
System.out.println(Button2.getMaximumSize());
System.out.println(Button2.getPreferredSize());
```

۴ -کامپوننت هایی که در لایه بندی BoxLayout قرار می گیرند به صورت پیش فرض از چپ به راست قرار می گیرند. برای اینکه بتوان آنها را درمرکز فریم قرار داد، بصورت زیر عمل می نماییم:

```
Button1.setAlignmentX(Component.CENTER_ALIGNMENT);
```

علاوه بر آنچه که در آرگمان متد setAlignmentX وجود دارد

می توان حالات دیگری هم برای قرار دادن کامپوننت ها درون متد استفاده کرد ، از جمله این موارد Component.ALIGNMENT_RIGHT ،Component.ALIGNMENT_LEFT و... می باشند.

۵- نکته آخر در این برنامه نحوه تعیین Border می باشد.

برای انجام این کار از عبارت زیر کمک می گیریم.

```
Button2.setBorder(BorderFactory.createCompoundBorder(
BorderFactory.createLineBorder(Color.RED),Button2.getBorder()));
```

۷-۳-۲-۳ ایجاد فضای خالی میان کامپوننت ها

یکی از نکات مهم در این مدل لایه بندی آن است که تمامی کامپوننت های موجود در فرم، بدون هیچ فاصله ای در کنار یکدیگر قرار می گیرند. برای رفع این مشکل باید به کمک راه کارهایی بین کامپوننت ها، فاصله مورد نیاز را ایجاد نماییم.

برای قرار دادن فضای خالی میان کامپوننت ها دو راه وجود دارد:

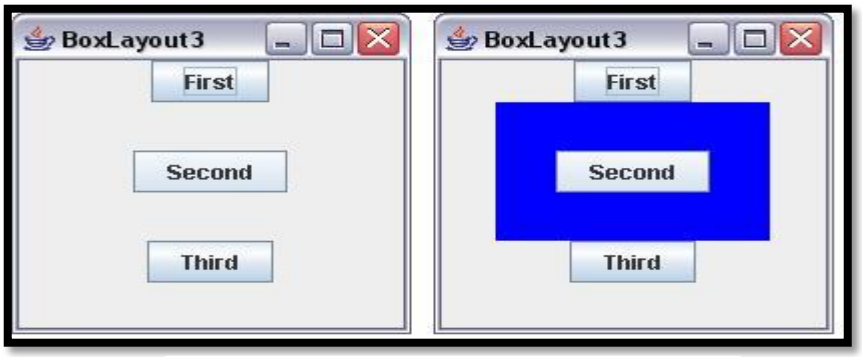
۱- راه اول این است که از Border میان کامپوننت ها استفاده نماییم.

در این روش به کمک Border هر یک از کامپوننت ها میزان فضای خالی مورد نیاز را بوجود می آوریم. تکه کد زیر قسمتی از این برنامه می باشد که در آن کامپوننت Button بوسیله استفاده از تکنیک Border، از دو Button دیگر جدا شده است:

```
...
button = new JButton("First");
button.setAlignmentX(Component.CENTER_ALIGNMENT);
container.add(button);
button = new JButton("Second");
button.setAlignmentX(Component.CENTER_ALIGNMENT);
container.add(button);
Color color = container.getBackground();
button.setBorder(BorderFactory.createCompoundBorder(
```

```
BorderFactory.createMatteBorder(30,30,30,30,color),button.getBorder()));
button = new JButton("Third");
button.setAlignmentX(Component.CENTER_ALIGNMENT);
container.add(button);
...
```

تصویر ۷-۷ خروجی این برنامه را نشان می دهد. همانطور که در این تصویر نشان داده شده است می توانید علاوه بر ابعاد Border، رنگی را نیز برای آن تعیین نمایید.



شکل (۷- ۷) : خروجی برنامه Border

شکل بالا- چپ خروجی اصلی برنامه را نشان می دهد.

در حالی که شکل دیگر برای ملموس تر شدن نتیجه نشان داده شده است. برای ایجاد شکل سمت راست، باید در قطعه کدی که معرفی شد، بجای کلمه color در آرگمان پنجم متد createMatteBorder از واژه Color.BLUE استفاده کنیم.

۲- روش دیگر ایجاد فاصله میان کامپوننت ها، استفاده از کامپوننت های نامرئی است.

در این شیوه برای ایجاد فاصله از کلاس Box و زیر کلاس آن با نام Box.Filler بهره می گیریم. Box.Filler کامپوننت شفافی است که رنگ نمی پذیرد و برای ایجاد فاصله بکار می رود. جدول زیر نشان می دهد که این کلاس چگونه کار می کند.

نوع ایجاد فضا	چگونگی ایجاد محدودیت	متد

`Box.createRigidArea(size)`		`rigid area`	
`Box.createHorizontalGlue()`		افقی	`Glue`
`Box.createVerticalGlue()`		عموی	
`Box.createHorizontalStrut(int width)`		افقی	`Strut`
`Box.createVerticalStrut(int height)`		عمودی	
`new Box.Filler(minSize،prefSize،maxSize)`	همانند قبل	`custom Box.Filler`	

جدول (۷- 2) : عملکرد کلاس Box

اکنون به بررسی هریک از متدهای بالا می پردازیم:

۱- Rigid Area: زمانی از این مدل استفاده می شود که بخواهیم یک سایز fixed میان دو کامپوننت به وجود بیاوریم. این متد در دو جهت پهنا و ارتفاع اقدام به قرار دادن فضای خالی میان کامپوننت ها می کند.

همچنین این مدل می تواند برای لایه بندی افقی و عمودی استفاده شود.

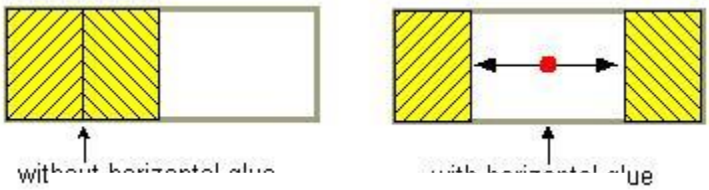
قطعه کد زیر این کار را انجام می دهد که در این قطعه کد فضایی به‌اندازه ۵ پیکسل میان دو کامپوننت در پهنا قرار می گیرد.

در این حالت کامپونت هایی در بالا و پایین یکدیگر وجود ندارند پس نیازی هم برای تعیین ارتفاع فضای خالی بالا و پایین کامپوننت ها نخواهیم داشت و مقدار صفر را برای آن در نظر خواهیم گرفت.(شکل ۷-۸)

```
container.add(firstComponent);
container.add(Box.createRigidArea(new Dimension(5,0)));
container.add(secondComponent);
```

شکل (۷- ۸) : فضای خالی کامپوننت ها

۲- Glue: در این شیوه محدودیتی در اینکه دقیقاً چه مقدار فاصله میان کامپوننت ها قرار دهیم نداریم و با تغییر سایز کانتینری که کامپوننت ها درون آن هستند، فاصله میان کامپوننت ها هم تغییر می کند. این مسأله باعث می شود که این شیوه در مقایسه با تکنیک Strut و Area Rigid قابل انعطاف **پذیر** باشد. معمولاً توصیه می شود که از HorizontalGlue در لایه بندی افقی و از VerticalGlue در لایه بندی عمودی استفاده شود.



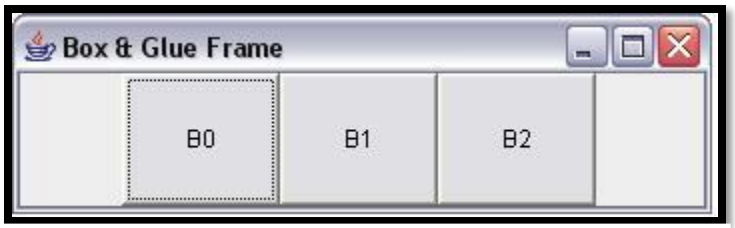
شکل (۷- ۹) : فاصله میان کامپوننت ها

```
container.add(firstComponent);
container.add(Box.createHorizontalGlue());
یا
container.add(Box.createVerticalGlue());
container.add(secondComponent);
```

تصویر ۷-۱۰ خروجی برنامه GlueSample.java را نمایش می دهد. در این برنامه نحوه پیاده سازی این تکنیک قابل مشاهده می باشد.

شکل (۷- ۱۰) : خروجی برنامه GlueSample.java

برنامه HBoxWithGlue.java برنامه ی دیگری است که نحوه استفاده از این تکنیک را نمایش داده است. تصویر ۷-۱۱ خروجی این برنامه می باشد.



شکل (۷- ۱۱) : خروجی برنامه HBoxWithGlue.java

۳- Strut: این شیوه هم فضای ثابتی برای ایجاد فاصله میان کامپوننت ها ایجاد می کند. نکته مهمی که در این روش وجود دارد این است که باید در یک مدل لایه بندی افقی از HorizontalStrut و در یک لایه بندی عمودی از VerticalStrut استفاده شود زیرا هنگامی که از این مدل در Panel های تو در تو استفاده می‌کنیم مشکل به وجود می آید، معمولاً توصیه می شود که به جای آن از Rigid Area استفاده شود.

```
container.add(firstComponent);
container.add(Box.createHorizontalStrut());
```

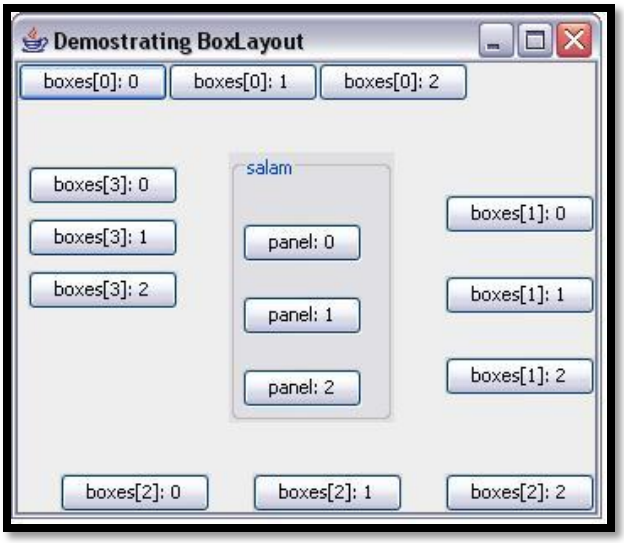
یا

```
container.add(Box.createVerticalStrut ());
container.add(secondComponent);
```

۴- Custom: می توان بوسیله ی این شیوه کامپوننتی ایجاد کرد که به اندازه دلخواه ما دارای سایز ماکسیمم ، مینیمم و مرجع باشد. به کد زیر دقت کنید:

```
 container.add(firstComponent);
Dimension minSize = new Dimension(5, 100);
Dimension prefSize = new Dimension(5, 100);
Dimension maxSize = new Dimension(Short.MAX_VALUE, 100);
container.add(new Box.Filler(minSize, prefSize, maxSize));
ontainer.add(secondComponent);
```

اکنون برنامه BoxLayout2.java را اجرا نمایید. این برنامه همه تکنیک های قبل را بصورت یکجا در خود جمع کرده است. خروجی این برنامه بصورت زیر می‌باشد.(شکل ۷-۱۲)



شکل (۷- ۱۲) : خروجی برنامه BoxLayout2.java

۷-۳-۳ مدل لایه بندی CardLayout

یکی دیگر از مدل های لایه بندی، CardLayout است. این مدل لایه بندی زیر کلاسی از AWT می باشد و هنگامی بکار می رود که بخواهیم با تغییر حالت یک یا چند کامپوننت، وضعیت کامپوننت های دیگر موجود در فریم را تغییر دهیم.

به عنوان مثال با انتخاب یکی از آیتم های ComboBox، دسته ای دیگر از کامپوننت ها روی فریم ظاهر شوند.

قبل از هر چیز برنامه CardLayoutDemo.java را اجرا بگیرید و نتیجه را با آن چه مشاهده می کنید مقایسه نمایید:

شکل (۷- ۱۳) : خروجی برنامه CardLayoutDemo.java

همان طور که می بینید با تغییر آیتم ComboBox کامپوننت های روی فریم تغییر می کنند.

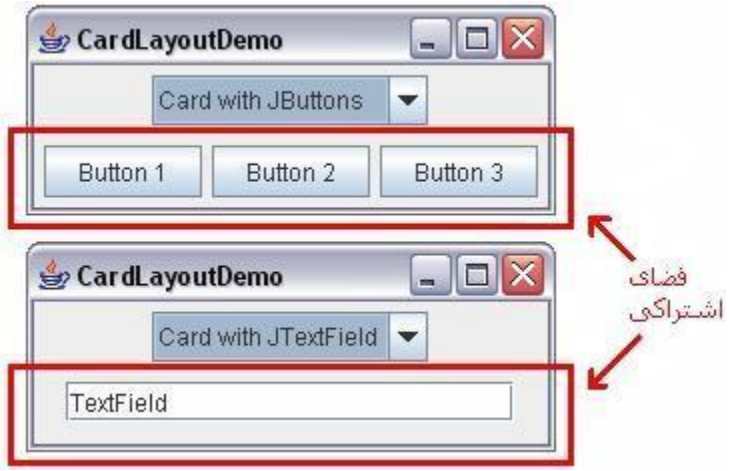
۷-۳-۳-۱ یک CardLayout چگونه کار می کند؟

در لایه بندی به روش CardLayout،کامپوننت های مورد نیاز فریم، بر روی دو یا چند **کانتینر «مثل** JPanel» قرار می‌گیرند. سپس مدل لایه بندی CardLayout کانتینرهای موجود را مدیریت می کند. یعنی به یک کامپوننت کاری نداشته و کار خود را بر اساس یک گروه از کامپوننت ها یا یک کانتینر انجام می دهد.

کانتینرها معمولاً فضای مشخصی از فریم را میان خود به اشتراک می گذارند و بصورت چند صفحه که روی هم قرار گرفته باشند، در آن فضا قرار می‌گیرند.

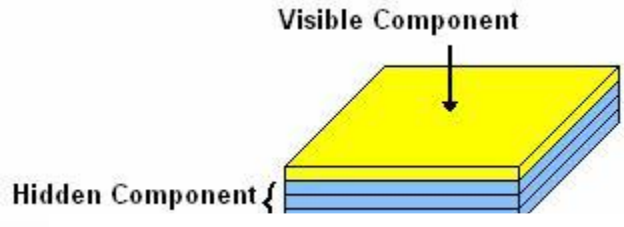
در مثال CardLayoutDemo که از آن اجرا گرفته شد، روی یک شئ از کلاس JPanel کامپوننت JTextField و روی شئ دیگری از این کلاس، سه کامپوننت JButton قرار گرفته است.

فضای اشتراکی هم همان قسمت پایین فریم است. با تغییر حالت آیتم ComboBox، کامپوننت های شئ JPanel «یعنی همان کانتینر ها» در فضای اشتراکی جای خود را عوض می کنند و با انجام این تغییر جا در هر مرحله، کامپوننتهای موجود در آن کانتینر نمایش داده می شوند.(شکل ۷-۱۴)



شکل (۷- ۱۴) : فضای اشتراکی اشیاء

شکل ۷-۱۵ چگونگی قرار گرفتن کانتینر ها را بصورت صفحات روی هم نشان می دهد. در این حالت کامپوننت های روی کانتینر بالایی آشکار بوده و کامپوننت های سایر کانتینرها که در لایه های زیرین قرار دارند، پنهان می باشند.



شکل (۷- ۱۵) : چگونگی قرار گرفتن کانتینرها

۷-۳-۳-۲ چگونه یک فریم را بر اساس مدل CardLayout لایه بندی کنیم؟

در این قسمت با ذکر چند مثال ساده، نحوه لایه بندی بر اساس مدل CardLayout را مورد بررسی قرار می دهیم.

«مثال شماره۱: **CardLayoutwithButton1.java** » در این مثال دو کانتینر از نوع JPanel ایجاد شده است که رنگ زمینه یکی زرد و رنگ زمینه دیگری قرمز می باشد. همچنین در قسمت پایین فریم دو دکمه ایجاد شده است.

حال اگر برنامه فوق را اجرا نمایید خواهید دید که با کلیک کردن بر روی هر یک از دو دکمه موجود، رنگ فریم تغییر می کند.

این تعویض رنگ نشان دهنده آن است که در زمان کلیک کردن بر هر یک از دکمه ها، کانتینر بالایی، زیر کانتینر پایینی قرار می‌گیرد. تصویر ۷-۱۶ خروجی این برنامه را نمایش می دهد.

شکل (۷- 16) : خروجی برنامه CardLayoutwithButton1.java

برای ایجاد این مدل لایه بندی ابتدا کانتینر هایی به صورت زیر تعریف می کنیم:

```
...
      Container container;
      CardLayout layout;
      JPanel CardPanel;
      JPanel SubPanel1,SubPanel2;
      JPanel ButtonPanel;
...
```

سپس در متد سازنده برنامه کد های زیر اضافه می کنیم:

```
public CardLayoutwithButton1() {
        JFrame frame = new JFrame("CardLayout with Button1");
        container = frame.getContentPane();
        container.setLayout(new BorderLayout());

        layout = new CardLayout();
        CardPanel = new JPanel();
        CardPanel.setLayout(layout);

        SubPanel1= new JPanel();
        SubPanel1.setBackground(Color.YELLOW);
        SubPanel2= new JPanel();
        SubPanel2.setBackground(Color.BLUE);
        ...
}
```

همانطور که در بدنه این متد می بینید، دو کانتینر SubPanel1 و SubPanel2 قرار است بصورت دو لایه روی هم قرار بگیرند.

پس باید ابتدا یک کانتینر به عنوان کانتینر اصلی با مدل CardLayout لایه بندی شود. سپس دو کانتینر فوق به آن اضافه گردند. قسمت Bold شده در کد فوق نشان دهنده کانتینر اصلی می باشد.

در قسمت بعدی باید دو زیر پنلی که در کد های بالا رنگ های شان را تغییر دادیم، یعنی SubPanel1 و SubPanel2 را به CardPanel اضافه کنیم :

```
...
        CardPanel.add(SubPanel1, "1" );
        CardPanel.add(SubPanel2, "2" );
...
```

قسمت بسیار مهم در کد های بالا، قسمتی است که با رنگ قرمز نشان داده شده است. هنگامی که این دو زیر پنل به CardPanel اضافه می شوند، حتماً باید توسط رشته دلخواهی « "۱" و "۲" » مشخص شوند. در حقیقت این دو رشته به عنوان **شناسه** این دو پنل تعیین می شوند و در هنگام فراخوانی هر یک از این زیر پنل ها،به عنوان نام آنها مورد استفاده قرار می گیرند.

۷-۳-۴ مختصری در مورد Event Handling

تا اینجا یک قسمت مهم از کار ما به پایان رسید. قسمت دیگر، کارهای مربوط Event Handling است.

اگر بخواهیم به طور خاص در مورد Handling Event صحبت کنیم، می توانیم بگوییم که هر اتفاقی که روی فریم ما می افتد مانند کلیک کردن روی یک دکمه، تغییر در آیتم های یک ComboBox، تغییر در حالت یک CheckBox و... یک رویداد یا Event را تولید می کند. بوسیله این رویداد تولید شده می توان مدیریت و انجام بسیاری از کار هایی که نیاز است در برنامه انجام شوند را انجام داد.

معمولاً برای استفاده از رویدادها باید واسط یا Interface ی که مربوط به رویداد مورد نظر می باشد و همچنین متدهایی که به واسط مورد نظر، مربوط می شوند «Abstract Methods» را بکار گیریم.

به تکه کد زیر از برنامه CardLayoutwithButton1.java دقت کنید:

```
...
   public class CardLayoutwithButton1 implements ActionListener{
...
```

همان طور که در عبارت فوق مشخص است، در این برنامه از واسط ActionListener در کلاس اصلی برنامه استفاده شده است.

در این برنامه رویداد تولیدشده کلیک روی button است.

اما برای اینکه رویداد دکمه های این برنامه را به تنها متد این واسط که در ادامه به شما معرفی خواهیم کرد، متصل کنیم باید از کد های زیر استفاده نماییم :

```
...
button1 = new JButton("Button 1");
button1.addActionListener(this);
button2 = new JButton("Button 2");
button2.addActionListener(this);
...
```

به وسیله متد addActionListener و کلمه کلیدی this، هر رویدادی که توسط این دو دکمه ایجاد می شود توسط متد زیر دریافت می گردد:

```
...
     public void actionPerformed(ActionEvent e) {
         if(e.getSource()== button1)
               layout.show(CardPanel,"1");
         else if(e.getSource()== button2)
               layout.show(CardPanel,"2");
```

```
    }
...
```

متد actionPerformed تنها متد واسط ActionListener است که یکی از پر کاربرد ترین متد هاست و رویدادهای تولید شده توسط کامپوننت هایی همچون JButton را پشتیبانی می کند. این متد به وسیله آرگومان ورودی خود که از نوع ActionEvent است،

می توان کامپوننتی که رویدادی را ایجاد کرده را شناسایی کرده و رویداد فوق را مدیریت نماید.

اما در مورد کد های درون این متد :

۱. if(e.getSource()== button1) : جهت شناسایی کامپوننتی است که رویداد تولید شده مربوط به آن می باشد. بوسیله متد getSource و پارامتر e این مسأله تشخیص داده می شود.

۲. layout.show(CardPanel"،"1"): نمایش یکی از زیر پنل های موجود در CardPanel. مسأله مهم دراین کد همان رشته شناسه است که حتماً باید با آن چه قبلاً توضیح داده شد، یکسان باشد.

در حقیقت این کد می گوید زیر پنلی از CardPanel را نشان بده که با شناسه "۱" به CardPanel اضافه شده است.

قسمت Event Handling را به گونه دیگر نیز می توان پیاده سازی نمود.

همانطور که گفتیم در روش قبل در اعلان کلاس اصلی برنامه باید متعلقاتی اضافه می کردیم، اما در این روش نیازی به این کار نداریم.

به قطعه کدهایی که از برنامه CardLayoutwithButton2.java انتخاب شده است دقت کنید.

تنها تفاوت این برنامه با برنامه قبل، بخش Event Handling می باشد.

```
...
  public class CardLayoutwithButton2{
    ...
     CardPanel.add(SubPanel1,"1");
     CardPanel.add(SubPanel2,"2");
      button1 = new JButton("Button 1");
      button2 = new JButton("Button 2");
      button1.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
               layout.show(CardPanel,"1");
          }
```

```
        });
        button2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
              layout.show(CardPanel,"2");
            }
        });
    ...
```

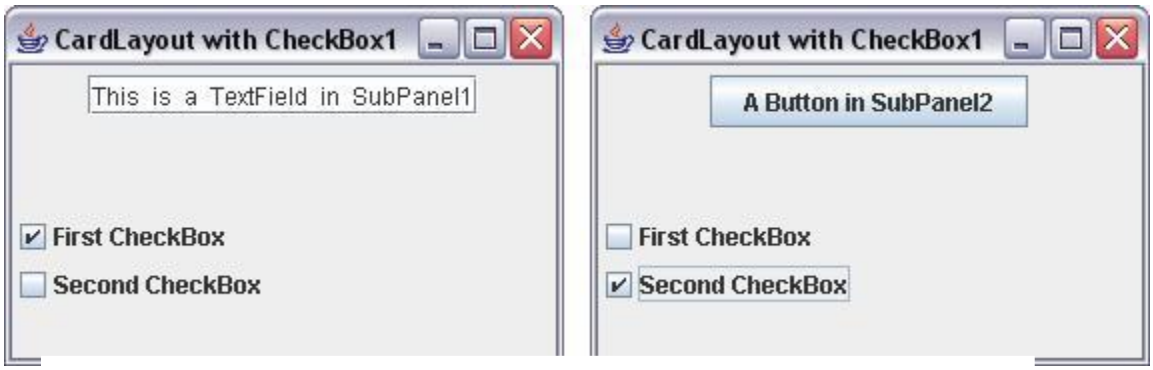
به قسمت هایی که با رنگ قرمز مشخص شده است دقت کنید. این شیوه که به شیوه استفاده از "*تکنیک کلاس های داخلی*" مرسوم است، کامپوننت ها را مستقیماً به متد actionPerformed متصل می کند.

همانطور که در کدها دیده می شود دیگر از پارامتر e استفاده نشده است و فقط قسمت نمایش پنل ها در متد آمده است.

«مثال شماره۲: CardLayoutwithCheckBox1.java »

قبل از هر چیز ابتدا برنامه را اجرا بگیرید. در این برنامه از کامپوننت JCheckBox برای تغییر حالت استفاده شده است.

این برنامه هم دقیقاً مانند آنچه که در مثال قبل دیدیم عمل می کند. یعنی مطالب مربوط به قسمت لایه بندی و قسمت Event Handling همان است که در مثال شماره ۱ گفته شد.



شکل (۷- 17) : خروجی برنامه CardLayoutCheckBox1.java

کامپوننت JCheckBox نیز مانند JButton از واسط ActionListener و به تبع آن از متد actionPerformed استفاده می کند. در این برنامه رویداد تولید شده تغییر حالت CheckBox می باشد.

به تکه برنامه زیر دقت کنید: این بخش نحوه اضافه کردن کامپوننت به صورت مستقیم با استفاده از متدسازنده کامپوننت ها به زیر پنل ها را نمایش می دهد.

```
SubPanel1.add(new JTextField("This is a TextField in SubPanel1"));
        SubPanel2.add(new JButton("A Button in SubPanel2"));

        checkBox1= new JCheckBox("First CheckBox");
        checkBox1.addActionListener(this);

        checkBox1.setSelected(true);

        checkBox2= new JCheckBox("Second CheckBox");
        checkBox2.addActionListener(this);

         ButtonGroup group = new ButtonGroup();
         group.add(checkBox1);
         group.add(checkBox2);
        ...
```

به قسمت **Bold** شده کد بالا نگاه کنید.

از کلاسی به نام ButtonGroup که زیر کلاسی از swing است استفاده شده است. کامپوننت هایی که به شئ این کلاس اضافه می شوند، با هم عمل می کنند. یعنی در یک لحظه تنها می توان یکی از آن ها را استفاده کرد. اگر از این کلاس در برنامه استفاده نکنیم،برنامه هیچ خطایی نمی گیرد اما هر دو CheckBox را می توان انتخاب نمود.

«مثال شماره۳: **CardLayoutwithComboBox.java** »

در این برنامه از یک ComboBox برای تغییر پنل های موجود در فریم استفاده شده است. در این مثال روی هر پنل یک تصویر قرار داده شده است و با تغییر آیتمهای ComboBox پنل تغییر می کند و در نتیجه تصاویر مختلفی بر فریم ظاهر می شود. این برنامه برخلاف دیگر برنامه هایی که تاکنون دیدیم از واسط دیگری برای گوش دادن به رویداد هایی که در فریم اتفاق می افتد، استفاده می کند.

بنابراین متدی هم که رویدادهای فریم را می گیرد نیز تغییر می کند رویدادی که در این برنامه اتفاق می افتد، تغییر آیتم های موجود در ComboBox است. واسطی که در این برنامه از آن استفاده می شود ItemListener و متد مربوط به آن itemStateChanged است.

قبل از توضیح بیشتر بهتر است برنامه را یک بار اجرا بگیرید. دقت کنید که قبل از آن تصاویر مربوطه را در فولدر پکیج اصلی برنامه و در کنار فولدر هایی چون build و src قرار دهید

شکل (۷- 18) : خروجی برنامه CardLayoutwithComboBox.java

عبارت زیر نشان دهنده آن است که در این برنامه قرار است تا رویدادهای کامپوننتی مانند ComboBox مدیریت شود.

```
...
  public class CardLayoutwithComboBox implements ItemListener{
...
```

برای قرار دادن تصاویر در پنل ها، از شئ کلاس JLabel استفاده شده است.

```
...
        SubPanel1= new JPanel();
        ImageIcon icon1 = new ImageIcon("Federer.jpg");
        JLabel label1= new JLabel(icon1);
        SubPanel1.add(label1);

        SubPanel2 = new JPanel();
        ImageIcon icon2 = new ImageIcon("Nadal.jpg");
        JLabel label2= new JLabel(icon2);
        SubPanel2.add(label2);
...
```

پس از این مرحله نوبت به تعریف ComboBox و آیتم های آن می رسد.

```
...
      String comboBoxItems[] = {"Federer","Nadal","Murray","Roddick"};
      JComboBox comboBox = new JComboBox(comboBoxItems);
      comboBox.setEditable(false);
      comboBox.addItemListener(this);

      CardPanel.add(SubPanel1,"Federer");
      CardPanel.add(SubPanel2,"Nadal");
      CardPanel.add(SubPanel3,"Murray");
```

```
        CardPanel.add(SubPanel4,"Roddick");
...
```

همان طور که دیده می شود شئ کلاس JComboBox یعنی comboBox برای گوش کردن رویداد هایش توسط متد itemStateChanged، از متد addItemListener و کلمه کلیدی this استفاده می کند.

توجه ۱ : همچنین همان طور که دیده می شود پنل های مختلف دارای شناسه هم نام با نام آیتم های comboBox هستند، این مسأله ایست بسیار مهم. زیرا با تغییر این نام هاست که رویداد متفاوت تولید می شود.

علاوه بر متد show که در مثال های قبل به کرات از آن استفاده کردیم، متد های دیگری لایه بندی CardLayout نیز وجود دارند که در مواردی برای ما سودمندند.

از جمله متد ها :

first: «برای قراردادن اولین کانتینر روی دیگر کانتینر ها»،

Last: «برای قراردادن آخرین کانتینر روی دیگر کانتینر ها»،

Previous: «برای قراردادن کانتینر قبلی روی دیگر کانتینر ها» و

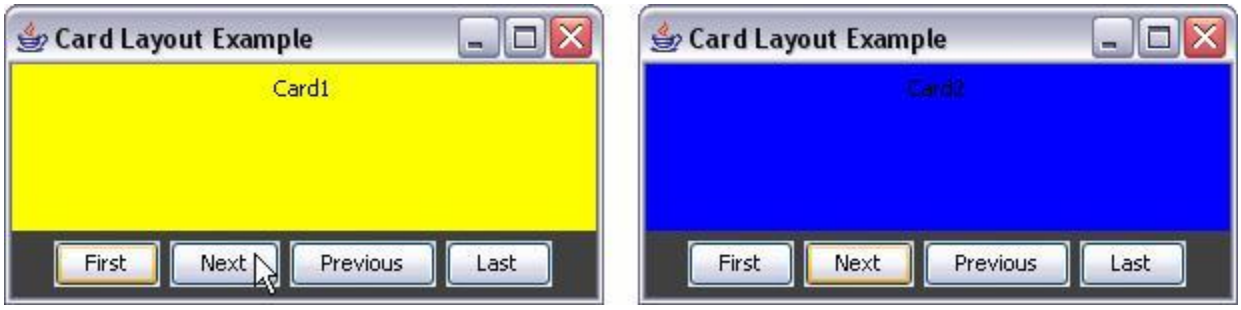
Next: «برای قراردادن کانتینر بعدی روی دیگر کانتینر ها»

برنامه CardLayoutExp.java نحوه استفاده از این متدها را نمایش داده است.

این برنامه از چهار پنل تشکیل شده است که هریک دارای رنگی جداگانه اند.

با فشاردادن هر یک از این دکمه ها، پنل مربوطه تغییر کرده و رنگ جدیدی برای ما در صفحه ظاهر می شود.

تصویر ۷-۱۹ خروجی این برنامه را نمایش می دهد.



شکل (۷- 19) : خروجی برنامه CardLayoutExp.java

۷-۳-۵ مدل لایه بندی FlowLayout

یکی از ساده ترین مدل های چیدمان کامپوننت ها در فریم، مدل لایه بندی FlowLayout می باشد که به صورت پیش فرض برای کانتینرهای JPanel مورد استفاده قرار می گیرد.

این مدل زیر کلاسی از AWT است.

برای شروع کار بهتر است ابتدا برنامه FlowLayout1.java را اجرا بگیرید.

خروجی این برنامه بصورت زیر می باشد. (شکل ۷-۲۰)

شکل (۷- 20) : خروجی برنامه FlowLayout1.java

۷-۳-۵-۱ ایجاد مدل FlowLayout

متن برنامه فوق بصورت زیر می باشد.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class FlowLayout1 {
   Container container;
   public FlowLayout1(){
         JFrame frame = new JFrame("FlowLayout1");
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         container = frame.getContentPane();
         container.setLayout(new FlowLayout());

container.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);

         frame.add(new JButton("Button 1"));
         frame.add(new JButton("Button 2"));
         frame.add(new JButton("Button 3"));
         frame.add(new JButton("Long-Named Button 4"));

         frame.add(new JButton("5"));
         frame.setSize(580,80);
         frame.setVisible(true);
   }
   public static void main(String[] args) {
```

```
            new FlowLayout1();
    }
}
```

همانطور که در کد بالا مشاهده می نمایید، فریم برنامه توسط متد setLayout و مدل FlowLayout لایه بندی شده است.

```
container.setLayout(new FlowLayout());
```

این عبارت را بصورت زیر نیز می توان نوشت:

```
FlowLayout flowLayout =new FlowLayout();
container.setLayout(flowLayout);
```

۷-۳-۵-۲ بعضی از خصوصیات FlowLayout

این مدل لایه بندی کامپوننت های مختلف را در حالت پیش فرض، به ترتیب از چپ به راست و در صورت انتخاب کاربر از راست به چپ و تا جائیکه کانتینر فریم فضا داشته باشد، قرار می دهد.

در غیر این صورت در خط بعدی کامپوننت ها را قرار می دهد.این مدل لایه بندی کامپوننت ها را بر اساس Preferred Size آنها در فریم قرار می دهد.

اگر پنجره برنامه بالا (شکل ۷-۲۰) را پس از اجرا تغییر اندازه دهیم، هر تعداد از کامپوننت ها که در یک خط جا نشوند به خط بعد منتقل می شوند.(شکل ۷-۲۱)

شکل (۷- 21) : خروجی برنامه FlowLayout1.java

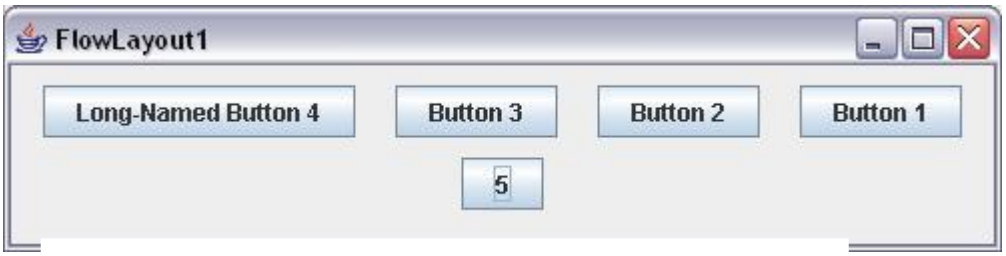
همانطور که گفته شد در این مدل می توان ترتیب چینش کامپوننت ها را در فریم تعیین نمود. برای انجام این کار بر اساس ترتیب مورد نظر یکی از دو عبارت زیر را به برنامه اضافه می نماییم.

```
container.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
container.setComponentOrientation(ComponentOrientation.LEFT_TO_RIGHT);
```

موضوع قابل بحث در این مدل آن است که می توان بین کامپوننت ها بر اساس نیاز فاصله ایجاد نمود. همچنین موقعیت چینش کامپوننت ها را نیز می توان تعیین نمود. به عبارت زیر دقت نمایید:

```
container.setLayout(new FlowLayout(FlowLayout.CENTER,20,10));
```

همان طور که می بینید، آرگومان اول متد سازنده به مکان کامپوننت ها در کانتینر، آرگومان دوم به فاصله افقی میان کامپوننت ها و آرگمان سوم هم به فاصله عمودی میان کامپوننت ها اشاره دارد. اگر عبارت فوق را به برنامه قبل اضافه نمایید، خروجی برنامه بصورت شکل ۷-۲۲ خواهد شد.



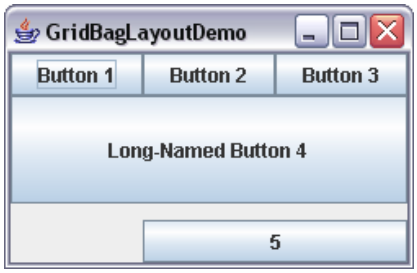
شکل (۷- 22) : خروجی برنامه FlowLayout.java

برای انجام عمل قبل بصورت زیر نیز می توان عمل نمود:

```
FlowLayout flow = new FlowLayout();
flow.setHgap(20);
flow.setVgap(10);
flow.setAlignment(FlowLayout.CENTER);
container.setLayout(flow);
```

۷-۳-۶ لایه بندی GridBagLayout

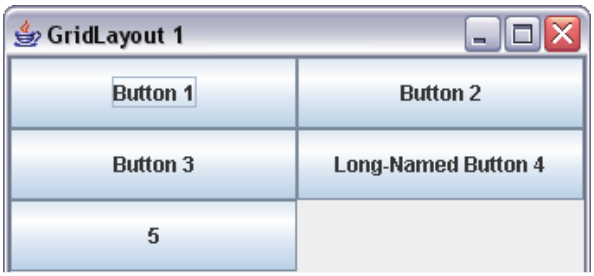
این مدل هم انعطاف پذیر و هم پیچیده است. در این مدل شبکه ای از سلول های نامتقارن داریم (هر سلول دارای طول و عرض متفاوت بادیگری) که کامپوننت ها می توانند دربیش از یک سلول قرار گیرند. (شکل ۷-۲۳)



شکل (۷- 23) : لایه بندی GridBagLayout

۷-۳-۷ لایه بندی GridLayout

این تکنیک، شبکه ای یکسان از سلول ها تشکیل می دهد که هرکامپوننت در یکی ازسلول ها قرار می گیرد.

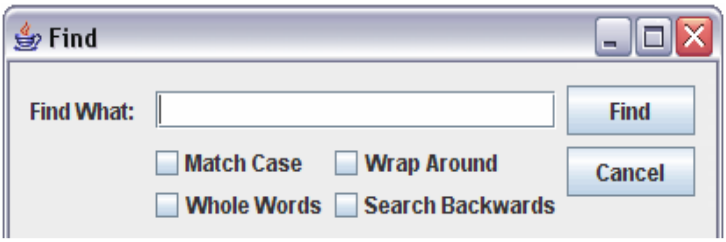


شکل (۷- 24) : لایه بندی GridLayout

۷-۳-۸ لایه بندی GroupLayout

این نوع لایه بندی از لایه بندی های توسعه داده شده برای ایجاد یک GUI است. این نوع لایه بندی، از لایه های عمودی و افقی به صورت جداگانه ای استفاده می کند.

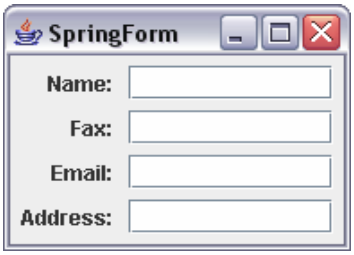
هر چند هر کامپوننت باید دوبار در لایه بندی قرار داده شود.(شکل ۷-۲۵)



شکل (۷- 25) : لایه بندی GroupLayout

۷-۳-۹ لایه بندی SpringLayout

این مدل نیز مدلی انعطاف پذیر است. در این مدل می توان کنترل دقیقی بین روابط میان لبه های کامپوننت تحت کنترل این مدل لایه بندی بوجود آورد.(شکل ۷-۲۶)

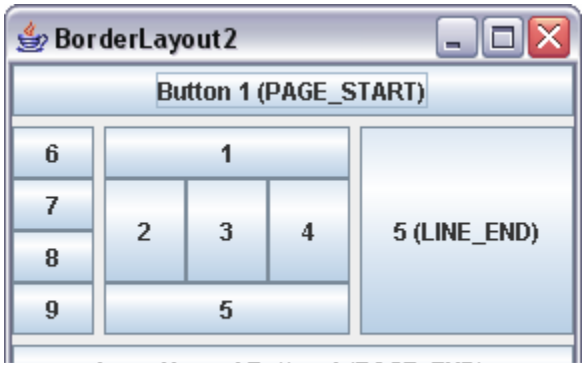


شکل (۷- 26) : لایه بندی SpringLayout

۷-۳-۱۰ استفاده از BorderLayout و دیگر مدل های لایه بندی برای هریک از ناحیه ها

در این بخش تکنیکی مورد بحث قرار گرفته است که به وسیله آن می توان هریک از پنج ناحیه را با استفاده از یک روش لایه بندی کرد.

برای انجام این کار پس از لایه بندی فریم برای بار اول با استفاده از BorderLayout، در هر ناحیه یک JPanel اضافه می کنیم. بعد هر JPanel را با استفاده از تکنیک لایه بندی مورد نظر خود، لایه بندی مجدد می کینم. تصویر ۷-۲۷ خروجی برنامه BorderLayout2.java می باشد. در این برنامه قسمت مرکزی بوسیله BorderLayout و قسمت سمت چپ نیز بوسیله مدل لایه بندی BoxLayout لایه بندی شده است.



شکل (۷- 27) : خروجی برنامه BorderLayout2.java

به چگونگی ساخت این فریم دقت کنید:

```
...
        JFrame frame = new JFrame("BorderLayout2");
        frame.getContentPane();
```

```
        frame.setLayout(new BorderLayout(5,5));
        JButton button = new JButton("Button 1 (PAGE_START)");
        frame.add(button, BorderLayout.PAGE_START);
...
```

لایه بندی قسمت مرکزی فریم اصلی:

```
    JPanel panel=new JPanel(new BorderLayout());
    JButton internalbutton1 = new JButton("1");
    JButton internalbutton2 = new JButton("2");
    JButton internalbutton3 = new JButton("3");
    JButton internalbutton4 = new JButton("4");
    JButton internalbutton5 = new JButton("5");
    panel.add(internalbutton1,BorderLayout.NORTH);
    panel.add(internalbutton3,BorderLayout.CENTER);
    panel.add(internalbutton2,BorderLayout.WEST);
    panel.add(internalbutton4,BorderLayout.EAST);
    panel.add(internalbutton5,BorderLayout.SOUTH);
```

اگر در متد سازنده JPanel نوع لایه بندی را تعیین نکنیم، JPanel به صورت مدل **FlowLayout** لایه بندی می شود. زیرا JPanel به صورت پیش فرض از مدل لایه بندی **FlowLayout** استفاده می کند.

حال JPanel لایه بندی شده را، درناحیه مرکزی فریم اصلی قرار می دهیم.

```
frame.add(panel,BorderLayout.CENTER);
```

لایه بندی ناحیه چپ فریم اصلی:

```
JPanel panel2 = new JPanel();
BoxLayout boxLayout =new BoxLayout(panel2,BoxLayout.Y_AXIS);
panel2.setLayout(boxLayout);
JButton internalbutton6 = new JButton("6");
JButton internalbutton7 = new JButton("7");
JButton internalbutton8 = new JButton("8");
JButton internalbutton9 = new JButton("9");
panel2.add(internalbutton6);
panel2.add(internalbutton7);
panel2.add(internalbutton8);
panel2.add(internalbutton9);
```

حال JPanel فوق را در ناحیه چپ فریم اصلی اضافه می کنیم.

```
frame.add(panel2, BorderLayout.LINE_START)
```

کد اجرایی برنامه های استفاده شده در این مبحث در محتوای بسته و در پوشه ی layers وجود دارد

۴-۷ تعیین حد و مرز کامپونت های swing توسط Border ها

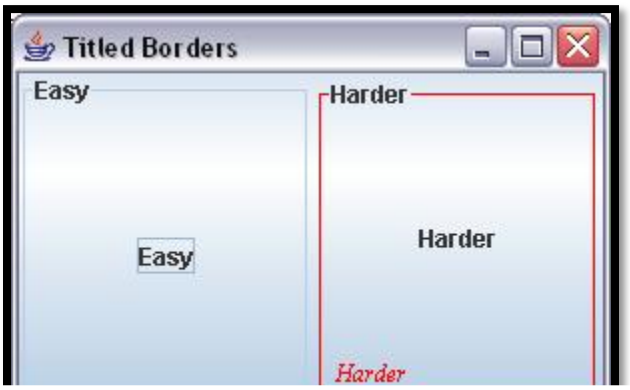
هر یک از اعضای خانواده JComponent، دارای یک یا چند نوع Border می باشد.

یک Border در واقع کلاسی می‌باشد که در کامپونت‌ها جهت مشخص کردن حاشیه و حد و مرز کامپونت مورد نظر بکار گرفته می شود. Border ها یکی از اجزاء بسیار مفید در هنگام کار با کامپونت ها محسوب می شوند. کاربردهای Border عبارتند از:

۱- Border با رسم خطوط به دور کامپونت مورد نظر، **حد و مرز** آن شی را مشخص می کند.

۲- به کمک Border ها می توان فضای خالی یا Gap مورد نیاز بین کامپونت های موجود در یک فریم را ایجاد نمود.

۳- به کمک Border ها می توان برای هر کامپونت یک عنوان یا Title تعیین نمود.



شکل (۷- 28) : Border ها برای کامپونت

4- استفاده از انواع Border ها در یک فریم، زیبایی خاصی به آن فریم می بخشد و در نتیجه یک برنامه کاربر پسند یا User Freindly ایجاد می گردد.

نکته ۴ : در جاوا برای هر کامپونتی که از کلاس JComponent ارث بری کرده باشد، می توان Border تعیین نمود.

نکته ۵ : به طور کلی، اگر قصد تنظیم Border برای کامپونت های استاندارد swing بجز JLabel و JPanel را دارید، توصیه می شود که کامپونت های خود را درون یک JPanel قرار داده و برای JPanel فوق Border تعیین نمایید.

۷-۴-۱ بررسی روش های ایجاد Border

قبل از بررسی روش های ایجاد Border ذکر این نکته ضروری است که برای ایجاد یک Border دور یک JComponent مانند یک JButton یا کامپوننت های دیگر این مجموعه در تمامی روش ها، از متد setBorder استفاده می‌نماییم.

این متد یک آرگومان ورودی دارد که تعیین کننده نوع Border می باشد. به عبارت زیر دقت نمایید:

```
JButton button1 = new JButton();
button1.setBorder(Border);
```

روش هایی که در ادامه بررسی می نماییم، در واقع روش های ایجاد Border و ارسال آن به عنوان آرگومان ورودی متد setBorder می باشد.

۱- استفاده از کلاس BorderFactory

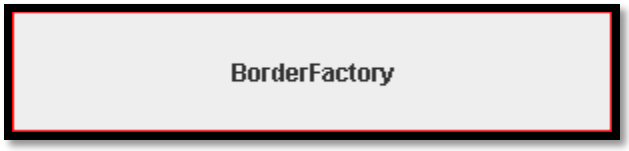
کلاس BorderFactory کلاسی است که سازنده اغلب Border های استاندارد می باشد.

این کلاس عضوی از بسته javax.swing می باشد.

برای ایجاد Border در این روش بصورت زیر عمل می کنیم.

```
label1.setBorder(BorderFactory.createLineBorder(Color.RED));
```

عبارت فوق قسمتی از برنامه MainClass5. java می باشد. خروجی این برنامه شکل ۷-۲۹ می باشد.



شکل (۷- 29) : خروجی برنامه MainClass5.java

کلاس Borderfactory دارای حدودا ۲۳ نوع Border می باشد.

در عبارت بالا نوع Border مورد استفاده createLineBorder بوده که خطی با رنگ مورد نظر کاربر به دور کامپوننت رسم می نماید.

دقت کنید که عبارت بالا را می توان به صورت زیر نیز نوشت:

```
Border border = BorderFactory.createLineBorder(Color.RED);
label1.setBorder(border);
```

در این حالت باید javax.swing.border.Border در ابتدای برنامه import شود.

۲- استفاده از کلاس MatteBorder

این کلاس عضوی از بسته javax.swing.border می باشد. توسط کلاس فوق قسمتی از فضای کامپوننت مورد نظر برای نمایش یک کادر رنگی دور آن استفاده می شود. برای استفاده از این کلاس به صورت زیر عمل می کنیم:

```
Border solidBorder = new MatteBorder( 20, 15, 20, 15, Color.RED);
JButton solidButton = new JButton("Download Me");
solidButton.setBorder(solidBorder);
```

چهار عدد ورودی در این متد میزان فضای مورد استفاده برای رسم کادر دور کامپوننت می باشند که دو عدد ۲۰ ضخامت کادر در بالا و پایین کامپوننت و دو عدد ۱۵ ضخامت کادر در طرف چپ و راست کامپوننت می باشند.

عبارت فوق قسمتی از برنامه ColorMatteBorder.java می باشد. خروجی این برنامه شکل ۷-۳۰ میباشد.

شکل (۷- 30) : خروجی برنامه ColorMatteBorder.java

3- استفاده از کلاس CompoundBorder یا ایجاد Border ترکیبی این تکنیک کمی پیچیده تر از دو تکنیک قبلی می باشد.

در روش با کمک کلاس CompoundBorder که عضوی از بسته javax.swing.border می باشد، قصد داریم Border ی ایجاد نماییم که ترکیبی از چندین Border دیگر باشد.

برای این کار ابتدا Border های مورد نیاز خود را به تعداد دلخواه ایجاد می نماییم. سپس دو تا دوتا و با کمک کلاس CompoundBorder، آنها را بصورت یک border واحد تبدیل می نماییم.

سپس border حاصل را با یکی دیگر از Border تکی، با کمک کلاس فوق ترکیب کرده و این بار یک Border که ترکیبی از سه Border است را بوجود می آوریم.

این عمل را به تعداد مورد نیاز انجام می دهیم. برای انجام این عمل بصورت زیر اقدام نمایید:

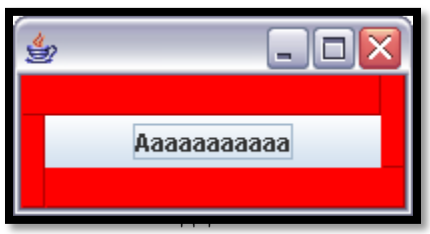
```
Border lineBorder = LineBorder.createBlackLineBorder();
Border bevelBorder = BorderFactory.createRaisedBevelBorder();
Border bevelLineBorder = new CompoundBorder(bevelBorder, lineBorder);
```

عبارت فوق قسمتی از برنامه ACompoundBorder.java می باشد. خروجی این برنامه شکل ۷-۳۱ میباشد.

شکل (۷- 31) : خروجی برنامه ACompoundBorder.java

۴- استفاده از Border های ایجاد شده توسط کاربر یا CustomBorder

یکی دیگر از روش های ایجاد Border، ایجاد Border های است که توسط کاربرایجاد می شود. قبل از هرگونه توضیحی برنامه CustomBorderDemo.java را اجرا نمایید. خروجی این برنامه شکل ۷-۳۲ می باشد.



شکل (۷- 32) : خروجی برنامه CustomBorderDemo.java

کد برنامه فوق بصورت زیر می باشد:

```
class SimpleBorder implements Border {
  ...
public SimpleBorder() {
   this.top = 20;
   this.left = 12;
   this.bottom = 20;
   this.right = 12;
   this.color = Color.RED;
}

public void paintBorder(Component c, Graphics g, int x, int y,_
                        int width, int height) {
   Insets insets = getBorderInsets(c);
   if (color != null)
          g.setColor(color);
g.fill3DRect(0,0, width-insets.right, insets.top, true);
g.fill3DRect(0, insets.top, insets.left,height-insets.top, true);
g.fill3DRect(insets.left, height-insets.bottom, width-insets.left,_
             insets.bottom, true);
   g.fill3DRect(width-insets.right, 0, insets.right,height-insets.bottom,
true);
  }
   ...
}

public class CustomBorderDemo {
   public static void main(String[] a) {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton button = new JButton("Aaaaaaaaaaa");
        button.setBorder(new SimpleBorder());
       ...
   }
}
```

در این برنامه یک کلاس با نام SimpleBorder وجود دارد که Border مورد نظر توسط این کلاس ساخته می شود. این کلاس از واسط Border در خود استفاده می نماید. با استفاده از این واسط در کلاس، متدی در بدنه کلاس با نام **paintBorder** افزوده می شود که در عمل Border درون آن ایجاد می گردد.

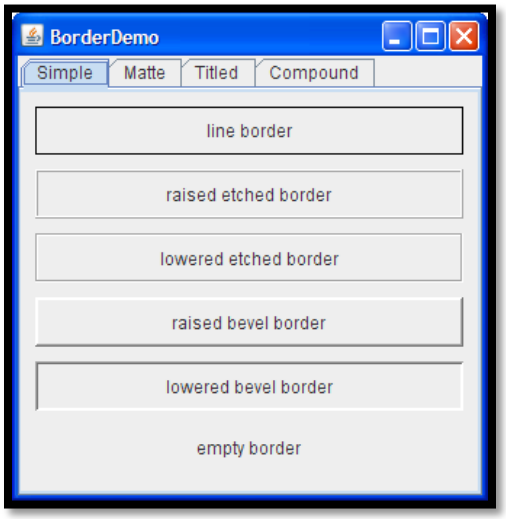
در نهایت زمانی که قصد استفاده از Border فوق را در کامپوننت های دیگر داشته باشیم، بصورت زیر عمل می نماییم.

```
button.setBorder(new SimpleBorder());
```

توجه ۶: برای ایجاد Border در برنامه های مختلف جاوا، متد ها یا کلاس های متعددی وجود دارد که به علت تعدد آنها نمی توان در این مبحث به تمامی آنها پرداخت.

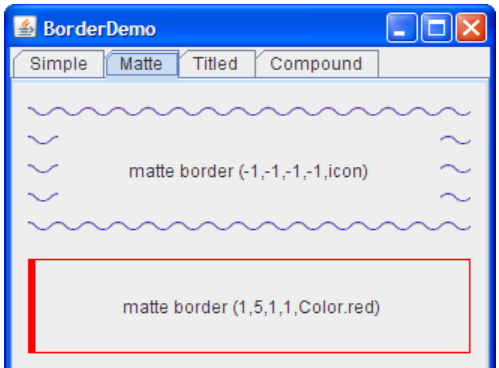
در پایان نمونه برنامه مفید دیگری را قرار داده ایم تا با بررسی آن، نکات مختلف دیگری را در این زمینه بدست آورید. نام این برنامه BorderDemo.java می‌باشد.

این برنامه از چهار صفحه تشکیل شده است. در صفحه اول این برنامه (شکل ۷-۳۳) چند نمونه از Border های ساده مورد استفاده قرار گفته است.



شکل (۷- 33) : خروجی برنامه BorderDemo.java

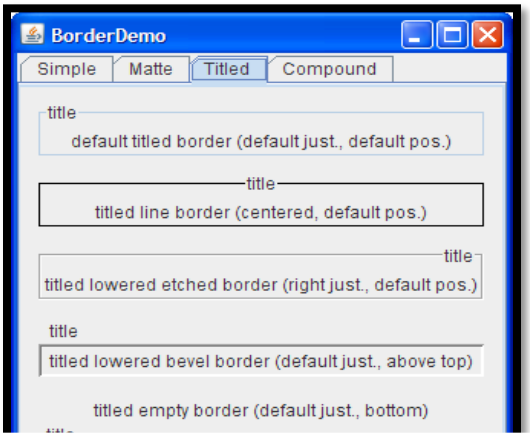
در صفحه دوم (شکل ۷-۳۴) چند نوع matte Border مورد استفاده قرار گرفته است. در این بخش علاوه بر استفاده از رنگ برای رسم Border، از یک فایل کوچک تصویری نیز استفاده شده است. نکته مهم در این بین آن است که اندازه فایل تصویر مورد استفاده نباید در حدی باشد که در حین اجرای برنامه، تاثیر نامطلوبی بر ظاهر برنامه ایجاد کند.



شکل (۷- 34) : خروجی برنامه BorderDemo.java

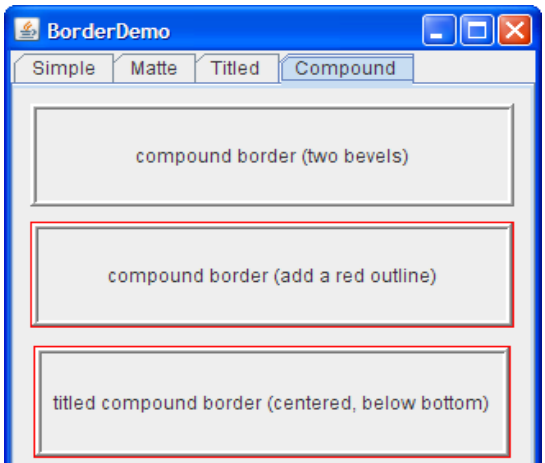
در صفحه سوم (شکل ۷-۳۵) از titled Border استفاده شده است. همانطور که در تصویر زیر مشخص است، در این نوع Border عنوانی در Border قرار داده می شود که توسط آن می توان یک توضیح کوچک یا یک عنوان به کامپوننت مورد نظر اختصاص داد.

نکته دیگر در تصویر محل قرار گیری عنوان مورد نظر در Border می باشد.



شکل (۷- 35) : خروجی برنامه BorderDemo.java

صفحه چهارم(شکل ۷-۳۶) نیز نحوه پیاده سازی Compound Border یا Border های ترکیبی را نشان می دهد.



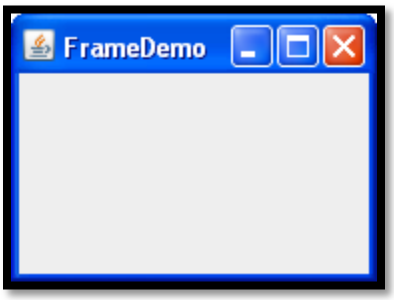
شکل (۷- 36) : خروجی برنامه BorderDemo.java

کد اجرایی برنامه های استفاده شده در این مبحث در محتوای بسته و در پوشه ی border وجود دارد

۷-۵ کلاس JFram

۷-۵-۱ JFrame چیست؟

همانطور که می دانید، مهمترین عنصر و به عبارت دیگر نقطه شروع طراحی یک برنامه ویژوالی در جاوا، ایجاد یک فرم یا فریم یا پنجره می باشد که سایر اجزا برنامه، بر روی آن قرار می گیرد.



شکل (۷- ۳۷) : کلاس JFram

یک Frame، نمونه ای از کلاس JFrame می باشد که همانند

یک ظرف «Container»، دیگر کامپوننت های Swing مانند jCheckBox،JTextField و... را در خود نگهداری می کند.

یک Frame پنجره ای است که دارای نوار عنوان «Title bar»، دکمه هایی برای تغییر سایز و بستن فریم و امکانی برای قراردادن آیکون در نوار عنوان است. Frame ها از لحاظ ویژوالی یا بصری، در بالاترین سطح کلاس های swing قرار دارند.

۷-۵-۲ ایجاد یک فریم

برای تشکیل یک فریم می توان به دو شیوه عمل کرد:

اولین روش ارث بردن کلاس اصلی برنامه از کلاس JFrame است. برای انجام اینکار بصورت زیر عمل می نماییم:

```
package JFrame;
import javax.swing.*;
public class Frame extends JFrame{
     public Frame(){
              super("FrameDemo");
              setVisible(true);
              setSize(200,200);      }

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

دومین روش تعریف شئ ای از کلاس JFrame به صورت زیر می باشد:

```
public static void main(String[] args) {
             Fram frm=new Frame();
                    }}
```

۷-۵-۳ اضافه کردن عنوان به یک فریم

همانطور که در قسمت قبل دیدیم، به دو صورت می توان یک فریم را ایجاد نمود. بسته به اینکه از کدام روش استفاده کنیم، دو شیوه متفاوت برای تعیین عنوان یک فریم وجود دارد.

اگر فریم توسط روش اول ایجاد شده باشد، می توان به وسیله به کارگیری متد super() در متد سازنده کلاس، عنوان فریم را تعیین نمود:

```
 public class Frame extends JFrame{
   public Frame(){
      super("FrameDemo");
      ...
```

```
    }
}
```

اگر در کلاسی که از یک کلاس دیگر ارث برده است، از متد super استفاده نمایید، در واقع متد سازنده کلاس پدر را مقداردهی کرده اید. برای درک بهتر این نکته، در اینجا کلاس برنامه یعنی کلاس Frame از کلاس JFrame ارث بری می کند و به وسیله متدsuper آن را مقدار دهی کرده است. این مقدار دهی سبب تعیین عنوان فریم می شود.

حال اگر توسط روش دوم یک فریم را ایجاد کرده باشید، می توانید به سادگی در متد سازنده، همان زمان که شئ کلاس را تعریف می نمایید، برای فریم نیز یک عنوان قرار دهید. به عبارت زیر توجه کنید:

```
JFrame frame = new JFrame("FrameDemo");
```

ویا اینکه شی را تعریف و بعد عنوان را تنظیم نماییم.

```
Frame frm = new Frame();
frm.setTitle("FramDemo")
```

۷-۵-۴ تعیین اندازه یک فریم

نکته مهم دیگر در ایجاد یک فریم، تعیین اندازه آن فریم می باشد. برای انجام این کار دو روش وجود دارد. اول آنکه بصورت مستقیم و با کمک متد setSize اندازه فریم مورد نظر تعیین نمایید. به عبارت زیر توجه کنید:

```
...
setSize(200,200);
...
```

دومین روش آن است که اندازه فریم را بر اساس اندازه کامپوننت های درون آن تعیین کنید. یعنی بررسی کنید که برای قرار گرفتن کامپوننت های مورد نیاز در فریم، چقدر فضا نیاز است. برای انجام چنین کاری می توان از متد pack بهره برد.

نکته مهم در این متد آن است که متد فوق تمامی بررسی های لازم را بصورت خودکار انجام داده و اندازه فریم را بر اساس محتویات آن تعیین می کند. برای استفاده از این متد بصورت زیر عمل می کنیم:

```
...
  pack();
...
```

دقت کنید که در زمان استفاده از متدهای قبل، اگر در متد سازنده برنامه از این متدها استفاده می کنید نیازی به ذکر نام فریم مورد استفاده نمی باشد. ولی اگر در بیرون از متد سازنده برنامه « مثلا در متد main» قصد استفاده از این متدها را دارید، باید بصورت زیر عمل نمایید:

```
  ...
frame.pack();
  ...
```

۷-۵-۵ تعیین محل نمایش فریم در صفحه نمایش

به برنامه زیر دقت نمایید:

```
import javax.swing.JFrame;
public class CreatingAWindow {
   public static void main(String[] args) {
      JFrame aWindow = new JFrame("This is the Window Title");
      int windowWidth = 400; // Window width in pixels
      int windowHeight = 150; // Window height in pixels
      aWindow.setBounds(50, 100, windowWidth, windowHeight);
      aWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      aWindow.setVisible(true);
   }
}
```

برای تعیین محل نمایش یک فریم در جاوا، می توان از متد setBound استفاده نمود. این متد دارای چهار آرگومان ورودی می باشد.

دو آرگومان اول مختصات نقطه ای از صفحه نمایش است که می خواهید فریم در آنجا نمایش داده شود. دو آرگومان بعد نیز، اندازه فریم مورد نظر می باشد.

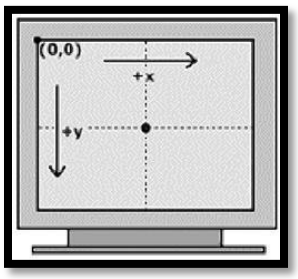
چگونه فریم خود را در وسط صفحه نمایش قرار دهیم؟

بطور پیش فرض JFrame خود را در گوشه بالا - سمت چپ صفحه نمایش قرار می دهد، ولی در اغلب موارد نیاز است تا فریم در وسط صفحه نمایش قرار گیرد.

روش اول:

نکته کلیدی در این مساله آن است که ابتدا سعی کنیم تا نقطه وسط صفحه نمایش را در حالت جاری « از لحاظ رزولیشن » بدست آوریم. برای انجام این کار از متد getCenterPoint استفاده می کنیم.

این متد با استفاده از مختصات محیط کاری جاری در صفحه نمایش، نقطه وسط محیط را به عنوان خروجی باز می گرداند. این متد یک خروجی از نوع شی Point که حاوی مختصات طول و عرض نقطه وسط محیط کاری است را باز می گرداند.



شکل (۷- ۳۸) : صفحه نمایش

برای استفاده از این متد، عبارت زیر را در برنامه خود استفاده نمایید:

```
 Point center =
GraphicsEnvironment.getLocalGraphicsEnvironment().getCenterPoint();
```

نکته دیگر آن است که این متد جز کتابخانه java.awt.GraphicsEnvironment می باشد که در ابتدای برنامه باید import گردد.حال که توانستیم نقطه وسط صفحه نمایش را بدست آوریم، با یک محاسبه ساده می توانیم فریم خود را در وسط محیط کاری قرار دهیم.برای انجام این کار باید نقطه گوشه بالا-سمت چپ را بدست آوریم. می دانید که این نقطه محل شروع رسم فریم ما توسط جاوا می باشد. تابع setBounds تابعی است که چهار آرگومان ورودی دارد. دو تای اول، مختصات نقطه گوشه بالا-سمت چپ را مشخص می کند و دو آرگومان بعد، ابعاد فریم را تعیین می کند. به عبارت زیر توجه کنید:

```
frmWindow.setBounds(  center.x - windowWidth / 2   ,  center.y -
windowHeight /2   , windowWidth,windowHeight);
```

همانطور که می بینید قسمت پر رنگ در عبارت فوق، محاسبه مورد نیاز برای یافتن نقطه گوشه بالا-سمت چپ « محل شروع ترسیم فریم توسط جاوا »، می‌باشد.برای درک بهتر مطالب فوق، در ادامه یک برنامه ساده آورده شده است که با اجرا آن می توانید نکات فوق را براحتی درک نمایید.

```
import java.awt.GraphicsEnvironment;
import java.awt.Point;
import javax.swing.JFrame;
public class CenteringaWindow {
   public static void main(String[] args) {
      JFrame frmWindow = new JFrame("This is the Window Title");
      Point center =
GraphicsEnvironment.getLocalGraphicsEnvironment().getCenterPoint();
      int windowWidth = 400;
      int windowHeight = 150;

      // تعیین مکان و اندازه فرم
      frmWindow.setBounds(center.x - windowWidth / 2, center.y - windowHeight
/ 2, windowWidth,
      windowHeight);
      frmWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frmWindow.setVisible(true); // نمایش فرم
   }
}
```

خروجی این برنامه فرمی را در وسط صفحه نمایش خواهد داد.

در بعضی شرایط ممکن است حالتی پیش آید که محاسبه فوق خیلی دقیق نباشد. راه حلی که در چنین شرایطی پیشنهاد می شود آن است که ابتدا با کمک متد getMaximumWindowBounds ابعاد محیط کاری که در حال حاضر به عنوان محیط کاری مجاز در اختیار برنامه ما قرار دارد را بدست آوریم.

```
GraphicsEnvironment.getLocalGraphicsEnvironment().getMaximumWindowBounds();
```

خروجی این متد شی ای از نوع Rectangle یا چهار ضلعی می باشد که به کمک آن می توان بصورت دقیق تری نقطه وسط محیط کاری را محاسبه نمود.

روش دوم

در این روش نیز اصول کار مانند روش قبل می باشد، اما از متد متفاوتی استفاده شده است. متد مورد استفاده برای انجام این کار، متد getToolKit میباشد. این متد که یکی از متد های کلاس ToolKit واقع در بسته java.awt است، می تواند با سیستم عامل ارتباط برقرار کرده و طی یک پرس و جو اندازه صفحه نمایش را

بدست آورد. در ادامه برنامه‌ای آورده شده است که در آن نحوه کار با این متد و در نتیجه قرار دادن فریم در وسط صفحه نمایش ذکر شده است.

```
import java.awt.Dimension;
import java.awt.Toolkit;
import javax.swing.JFrame;

public class SizingWindowswithToolkit {
   public static void main(String[] args) {
      JFrame frmWindow = new JFrame("This is the Window Title");
      Toolkit theKit = frmWindow.getToolkit();
      Dimension wndSize = theKit.getScreenSize();
      frmWindow.setBounds(wndSize.width / 4, wndSize.height / 4,
      wndSize.width / 2, wndSize.height / 2);

      frmWindow.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frmWindow.setVisible(true);
   }
}
```

۷–۵–۶ چگونه ظاهر یک JFrame را از لحاظ گرافیکی تغییر دهیم؟

بطور پیش فرض، دکوراسیون یا ظاهر یک پنجره در جاوا مانند پنجره ها یا فرم های معمول سیستم عامل محل اجرای برنامه جاوا می باشد. به عنوان نمونه به تصاویر زیر دقت کنید:

شکل (۷– ۳۹) : ظاهر پنجره در حالت MS-Windows

شکل (۷– 40) : ظاهر پنجره در حالت Motif

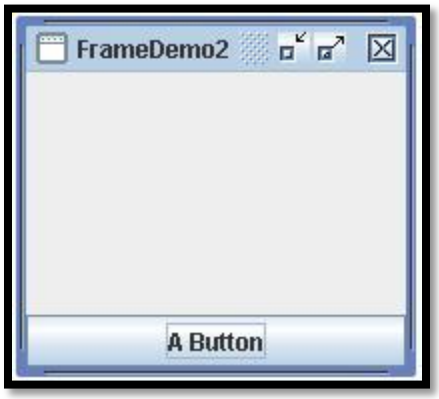
شکل (۷- 41) : ظاهر پنجره در حالت Java

این تصاویر هر سه حاصل اجرای یک برنامه با ظاهر های متفاوت می باشد.

۱ -در این قسمت سعی داریم تا فریمی ایجاد کنیم که به عنوان نمونه دکمه‌های MAX، MIN و Close آن از لحاظ گرافیکی تغییر کند و یا اصلا فریم مورد نظر هیچ گونه حالت گرافیکی نداشته باشد و...

قبل از شروع کار به تصاویر زیر دقت کنید:

شکل (۷- 43) : فریم در حالت پیشفرض



شکل (۷- ۴۲) : فریم باحاشیه SOUTH

شکل (۷- ۴۴) : فریم بدون حاشیه

این سه تصویر، هر سه اجرای یک برنامه اما با دکوراسیون های متفاوت می باشد. تصویر ۷-۴۳ حالت پیش فرض بوده و هر برنامه جاوا در ابتدا به این صورت می‌باشد

برای تغییر ظاهر فریم به صورتی که شبیه تصویر ۷-۴۲ شود، نیاز است تا عبارت زیر به کد برنامه اضافه شود.

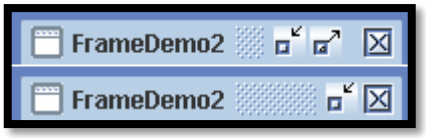
```
public FrameDemo2a() {
     JFrame.setDefaultLookAndFeelDecorated(true);
     JFrame frame = new JFrame("FrameDemo2");
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     JButton button = new JButton("A Button");
     frame.getContentPane().add(button, BorderLayout.SOUTH);
     frame.setSize(200,180);
     frame.setVisible(true);
  }
```

برای تغییر ظاهر فریم به صورتی که شبیه تصویر سمت راست شود، نیاز است تا عبارت زیر به کد برنامه اضافه شود.

```
public FrameDemo2a() {
     JFrame frame = new JFrame("FrameDemo2");
     frame.setUndecorated(true);
     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
     JButton button = new JButton("A Button");
     frame.getContentPane().add(button, BorderLayout.SOUTH);
     frame.setSize(200,180);
     frame.setVisible(true);
  }
```

۲- گام بعدی در تنظیمات فریم ها، جلوگیری از تغییر اندازه پنجره ها می باشد. در اکثر موارد اجزا درون فرم به گونه ای می باشند که نیاز به تغییر اندازه فرم نبوده و همیشه از لحاظ ابعاد ثابت می باشند. در چنین حالتی بهتر است امکان تغییر اندازه فریم را غیر فعال نماییم. برای انجام این کار عبارت زیر را به کد برنامه اضافه می نماییم.

```
JFrame frame = new JFrame( "Not resizable" );
frame.setResizable( false );
```



شکل (۷- ۴۵) : نوار عنوان فریم، قبل و بعد از غیر فعال

۳- سومین مرحله در تنظیمات ظاهر یک فریم در جاوا، انتخاب نوع قاب پنجره می باشد. قبل از هر چیز به تصاویر ۷-۴۶ دقت کنید:

شکل (۷- ۴۶) : قاب پنجره های فریم

تصاویر ۷-۴۶ نشان دهنده قابهای استاندارد جاوا می باشد که یک برنامه نویس می تواند بر اساس نیاز خود یکی از آنها را انتخاب نماید. برای انتخاب هر یک از قاب های فوق باید عبارت زیر را به برنامه اضافه نمایید:

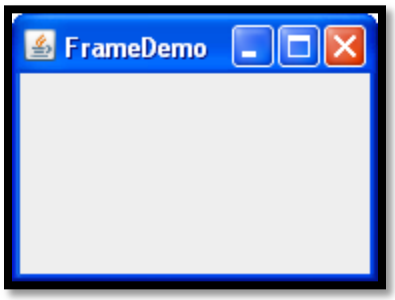
```
JFrame frame = new JFrame("FrameDemo2");
frame.setUndecorated(true);
frame.getRootPane().setWindowDecorationStyle(JRootPane.ERROR_DIALOG);
```

برای انتخاب هریک از قاب های فوق، بجای قسمت قرمز رنگ در عبارت بالا کافیست عبارت داخل قاب مورد نظر خود را قرار دهید.

۷–۵–۷ چگونه آیکون JFrame را تغییر دهیم؟

از نکات موثر در زمان طراحی و پیاده سازی یک برنامه کاربردی در جاوا و سایر زبان های برنامه نویسی، ایجاد واسط های گرافیکی کاربر پسند یا freindly User می باشد. برای طراحی چنین واسط یا به عبارت ساده تر، فرم ها یا فریم هایی بکارگیری نکات مختلفی ضروری می باشد.

یکی از این موراد مهم استفاده از فایل های تصویری کوچک یا همان آیکون ها در برنامه و در فریم های مختلف آن می باشد. یعنی با توجه به محتویات یک فرم، می توان آیکونی متناسب با آن به فرم مورد نظر اضافه کرد. مزیت انجام چنین کاری آن است که کاربر با توجه به تصویر ۷-۴۷ و بدون مطالعه محتویات آن می تواند بطور حدودی از وظایفی که بر عهده فرم فوق می باشد، آگاه گردد.



شکل (۷- ۴۷) : ایکن فریم

پیش فرض در برنامه هایی که برای آنها آیکونی تعیین نشده باشد، آیکون معروف جاوا که همان فنجان قهوه باشد، به عنوان آیکون فریم مورد استفاده قرار می گیرد. اما اگر کسی بخواهد آیکون دیگر را بجای این آیکون قرار دهد می تواند یکی از دو روش زیر را پیاده سازی نماید. البته توجه نمایید که در هر دو روش زیر،

باید در نهایت از متد setIconImage استفاده نمایید. آرگومان ورودی این متد یک فایل تصویری می باشد «همان فایل آیکون» که با فراخوانی این متد توسط فریم مورد نظر بجای آیکون فریم قرار می گیرد.

```
frame.setIconImage(image);
```

دو روشی که در ادامه مورد بحث قرار می گیرد در واقع روشهای فراخوانی و ارسال فایل تصویری مورد نظر به متد فوق را مورد بررسی قرار می دهد.

۱. استفاده از یک فایل تصویری با فرمت GIF یا PNG

فرمت های مورد استفاده برای آیکون در جاوا معمولا دو فرمت معروف gif و png می باشند. در این روش فایل آیکون ما از قبل آماده می باشد و فقط کافیست که ما آن را در برنامه خود فراخوانی کنیم. برای انجام چنین کاری می توان یکی از دو تکنیک زیر را مورد استفاده قرار دهید.

روش اول:

اگر از محیط NetBeans برای اجرای برنامه زیر استفاده می کنید، باید فایل آیکون مورد نظر را در پوشه classes و در کنار فایل MainClass.class قرار دهید.

```
import java.awt.Image;
import java.awt.Toolkit;
import javax.swing.JFrame;

public class MainClass {
    public static void main(String[] args) {
        JFrame f = new JFrame("FrameIcon");
        Image im = Toolkit.getDefaultToolkit().getImage("middle.gif");
        f.setIconImage(im);
        f.setSize(100, 100);
        f.setLocation(200, 200);
        f.setVisible(true);
     }  }
```

خروجی برنامه پس از اجرا شکل ۷-۴۸ می باشد:

شکل (۷- ۴۸) : خروجی برنامه تغییر ایکن فریم

روش دوم:

اگر از محیط NetBeans برای اجرای برنامه زیر استفاده می کنید، باید فایل آیکون مورد نظر را در پوشه اصلی پروژه یعنی در کنار پوشه های src و build و... قرار دهید. خروجی این روش نیز مانند تصویر قبل می باشد.

ویا اینکه به برنامه یک package اضافه نموده و تصاویر را در ان اضافه نموده و نهایتا به صورت مقابل ادرسی دهی میکنیم:

```
String pth=”/MyPack/mypicture”;
```

به کد زیر دقت کنید:

```
import java.awt.*;
import javax.swing.*;
public class FrameDemo3 {
   public FrameDemo3(){
        JFrame frame = new JFrame("FrameDemo4");
        frame.setIconImage(getFDImage());
        frame.setSize(200,180);
        frame.setVisible(true);
   }
protected static Image getFDImage() {
        java.net.URL imgURL = FrameDemo2.class.getResource("middle.gif");
        if (imgURL != null) {
                return new ImageIcon(imgURL).getImage();
        } else {
                return null;
       }  }
   public static void main(String[] args) {
        new FrameDemo3();
   }}
```

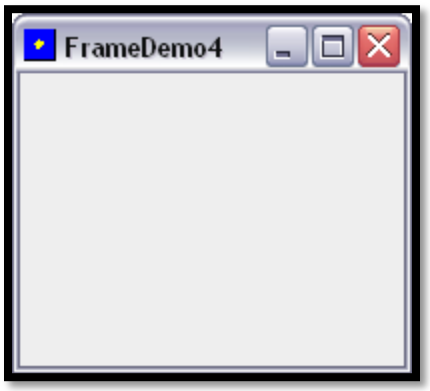
۲. ایجاد یک آیکون در برنامه با کمک تکنیک بافرینگ

تفاوت این روش با دو روش قبل در آن است که ما به کمک برنامه نویسی و با استفاده از تکنیک بافرینگ اقدام به ایجاد یک آیکون می نماییم و از فایل تصویری خاصی کمک نمی گیریم. در برنامه زیر به کمک توابع گرافیکی جاوا، یک آیکون ساده که شامل یک مربع آبر رنگ و یک بیضی زرد رنگ در وسط آن است، ایجاد می کنیم.

به کد زیر خوب توجه کنید:

```
import java.awt.*;
import javax.swing.*;
import java.awt.image.BufferedImage;

public class FrameDemo {
   public FrameDemo(){
        JFrame frame = new JFrame("FrameDemo4");
        frame.setIconImage(createFDImage());
        frame.setSize(200,180);
        frame.setVisible(true);
   }
protected static Image createFDImage() {
     //Create a 16x16 pixel image.
 BufferedImage bi = new BufferedImage(16, 16, BufferedImage.TYPE_INT_RGB);
        //Draw into it.
        Graphics g = bi.getGraphics();
        g.setColor(Color.BLUE);
        g.fillRect(0, 0, 15, 15);
        g.setColor(Color.YELLOW);
        g.fillOval(5, 5, 5, 5);
        g.dispose();
        return bi;
   }

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

خروجی این برنامه بصورت شکل ۷-۴۹ می باشد:



شکل (۷- ۴۹) : خروجی برنامه تکنیک بافرینگ

۷–۵–۸ کنترل دکمه Close یا ضربدر بالای فریم

استفاده از متد setDefaultCloseOperation() سبب می شود تا زمانی که کاربر بر روی دکمه close بالای فریم کلیک کند، فریم بسته می شود.

این متد دارای چهار آرگومان ورودی می باشد که عبارتند از:

(JDialog استفاده شوند):

1- DO_NOTHING_ON_CLOSE

با ارسال این آرگومان به متد فوق، وقتی کاربر بر روی دکمه close کلیک کند، هیچ اتفاقی نمی افتد و فرم بسته نخواهد شد.

2- HIDE_ON_CLOSE

این آرگومان، مقدار پیش فرض متد فوق می باشد. با انتخاب این مقدار در واقع زمانی که کاربر دکمه close را کلیک می کند، فریم بسته نمی شود بلکه مخفی می شود «Hide» و می توان در مراحل بعد دوبار آن فریم را به کمک متدهای دیگر نمایش داد.

3- DISPOSE_ON_CLOSE

این مقدار سبب بسته شدن فریم می شود. علاوه بر بستن فریم مورد نظر، تمامی منابع سیستم که در اختیار فریم است نیز آزاد شده و به سیستم باز گردانده می شوند.

4- EXIT_ON_CLOSE

با ارسال این آرگومان به متد، برنامه خاتمه می یابد. این آرگومان معادل دستور زیر عمل می کند.

system.exit(0);

قطعه کد زیر باعث ایجاد فریمی با قابلیت بسته نشدن ان میگردد.

```
JFrame frm=new JFrame("No Closing");
frm.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
frm.show();
```

۷-۵-۹ بررسی ساختار فریم ها در جاوا و لایه بندی فریم

ساختار یک فریم در جاوا را می توان از دو دیدگاه مورد بررسی قرار داد.

۱ -دیدگاه ظاهر فریم

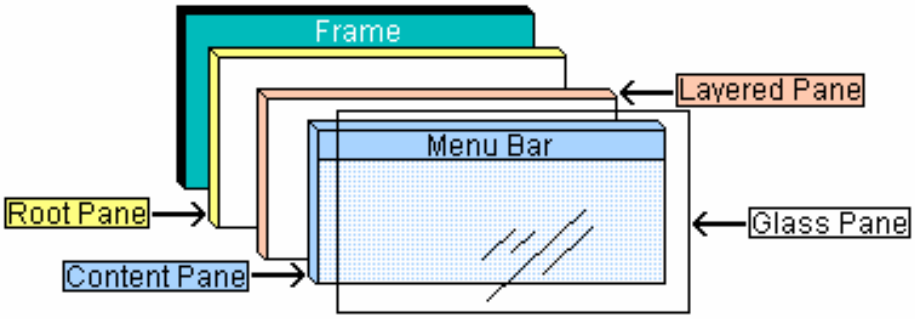
۲ -دیدگاه لایه بندی فریم

۱ - دیدگاه ظاهر فریم

یک فریم از لحاظ ظاهری از یکسری دکمه، عنوان فریم، آیکون فریم و حاشیه یا Border تشکیل شده است. این بعد از ساختار فریم ها نیاز به توضیح زیادی ندارد و در سایر مقالات به اندازه کافی به آن پرداخته شده است.

۲- دیدگاه لایه بندی فریم

قبل از هرگونه توضیحی به دقت به شکل ۷-۵۰ توجه نمایید.



شکل (۷- ۵۰) : دیدگاه لایه بندی فریم

یک فریم در ظاهر دارای یک لایه می باشد ولی همانطور که می بینیم این نظریه درست نبوده و یک فریم دارای چهار لایه اصلی زیر می باشد.

1- Root Pane 2- Layered Pane 3- Content Pane 4- Glass Pane

اولین نکته بسیار مهم، با توجه به مطالب بالا، آن است که نمی توان مانند AWT به صورت مستقیم سایر کامپوننت های Swing را به JFrame اضافه نمود.

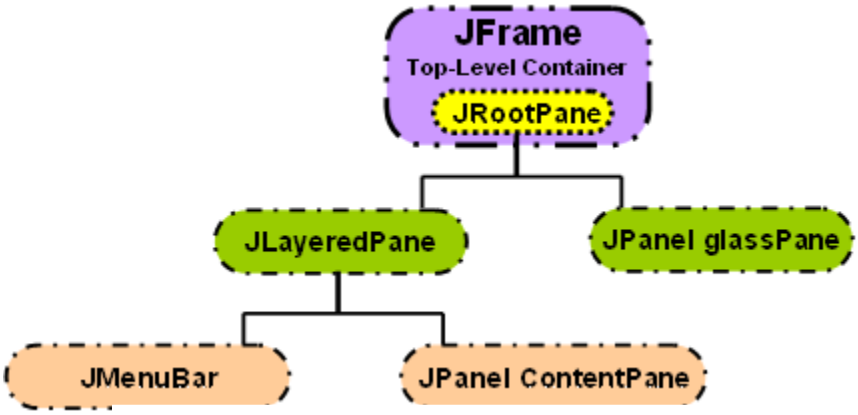
در JFrame ها، معمولا کامپوننت های swing به Content Pane اضافه می‌شوند و بطور مستقیم به خود JFrame اضافه نمی شوند.

در واقع یک فریم از چند قاب یا Pane مجزا تشکیل شده است و بهتر است که در هنگام اضافه کردن سایر کامپوننت های دیگر به آن، مشخص کنیم که قرار است کامپوننت مورد نظر به کدام قاب اضافه شود.

برای درک بهتر این قضیه کمی عمیق تر به این نکته می پردازیم. یک JFrame در واقع یک کانتینر سطح بالا یا top-level container می باشد. این نوع کامپوننت ها دارای یک نمونه از کامپوننت JRootPane می باشند.

خود دو کامپوننت JPanel glassPane و JLayeredPane می باشد.

اگر باز عمیق تر نگاه کنیم JLayeredPane از دو کامپوننت JMenuBar و JPanel contentPane تشکیل شده است.(شکل ۷-۵۱)



شکل (۷- ۵۱) : ساختار درختی قابهای فریم

پس به طور خلاصه و در یک جمله می توان گفت که کامپوننت های Swing به صورت مستقیم به JFrame اضافه نمی شوند بلکه به یکی از قاب های آن اضافه می شوند. در ادامه به کمک دو برنامه نشان خواهیم داد که چگونه می توان در قاب های مختلف یک فریم، کامپوننت های swing را اضافه نمود.

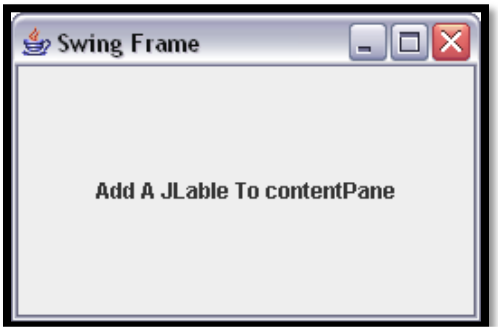
اضافه کردن کامپوننت های swing به content pane

به کد زیر دقت کنید

```
import javax.swing.JFrame;
import javax.swing.JLabel;

public class TJFrame {
   public static void main(String[] args) {
      JFrame frame = new JFrame("Swing Frame");
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JLabel label = new JLabel("Add A JLable To contentPane", JLabel.CENTER);
```

```
        frame.getContentPane().add(label);
        frame.setSize(350, 200);
        frame.setVisible(true);
    }}
```

خروجی این برنامه بصورت شکل ۷-۵۲ می باشد.



شکل (۷- 52) : خروجی برنامه اضافه کردن کلاسهای Swing به Content Pane

اضافه کردن کامپوننت های swing به glass pane

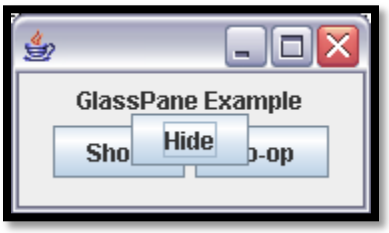
در این برنامه سه دکمه وجود دارد که دوتای آنها به contentPane و سومی به glassPane اضافه شده است. همانطور که در اولین شکل این مقاله می بینید، glassPane اولین لایه بوده و روی سایر لایه ها قرار دارد. اگر برنامه زیر را اجرا نمایید و به خروجی آن دقت کنید، خواهید دید که دکمه ای که به glassPane اضافه شده ا ست روی دو دکمه دیگر قرار دارد و این امر نشان دهنده بحث لایه بندی در فریم ها می باشد.

به کد زیر دقت کنید:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class MainClass3 {
   public static void main(String[] args) {
      JFrame f = new JFrame();
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      final JPanel p1 = new JPanel();
      p1.add(new JLabel("GlassPane Example"));
      JButton show = new JButton("Show");
      p1.add(show);
      p1.add(new JButton("No-op"));
      f.getContentPane().add(p1);
      final JPanel glass = (JPanel)f.getGlassPane();
      glass.setVisible(true);
      glass.setLayout(new GridBagLayout());
      JButton glassButton = new JButton("Hide");
```

```
    glass.add(glassButton);
    f.setSize(180, 100);
    f.setVisible(true);

    show.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
          glass.setVisible(true);
          p1.repaint();
       }
    });
    glassButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
          glass.setVisible(false);
          p1.repaint();
       }
    });
}}
```

خروجي شكل ۷-۵۳



شكل (۷- 53) : خروجی برنامه اضافه کردن کلاسهای Swing به Glass Pane

۷-۵-۱۰ کلاس JLayeredPane و ارتباط ان با Jfram

با توجه به مطالب قبل دریافتیم که کلاس JLayeredPane بخش از JRootPane می باشد. در واقع این JLayeredPane فراتر از یک لایه بوده و کانتینری است که نقش مدیریت مجموعه‌ای از لایه ها و کامپوننت های موجود در آنها را بر عهده دارد. نحوه قرار گیری لایه ها در این بخش با توجه به اولویت آنها تعیین می شود. به عبارت دیگر هر لایه دارای یک شماره می باشد و در نتیجه هر لایه ای که شماره آن کوچک تر باشد، بالاتر از سایر لایه ها قرار می گیرد.

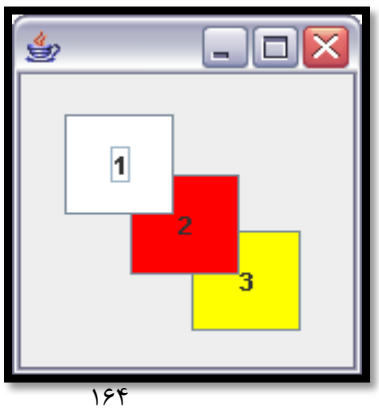
در هنگام اضافه کردن کامپوننت ها به فریم، اگر بخواهیم تعیین کنیم که هر کامپوننت در کدام لایه قرار گیرد، کافیست از متد زیر که دارای دو آرگومان ورودی می‌باشد، استفاده نماییم.

```
JFrame f = new JFrame();
JLayeredPane lp = f.getLayeredPane();
lp.add(button, new Integer(1));
```

همانطور که می بینید در تکه کد قبل یک دکمه به لایه شماره ۱ افزوده شده است. برای درک بهتر، برنامه زیر را با دقت بررسی نمایید. خروجی برنامه بصورت زیر می باشد.

```
import javax.swing.*;
import java.awt.Color;
public class SimpleLayers {
   public static void main(String[] args) {

      // Create a frame & gets its layered pane
      JFrame f = new JFrame();
      JLayeredPane lp = f.getLayeredPane();
      // Create 3 buttons
      JButton top = new JButton("1");
      top.setBackground(Color.white);
      top.setBounds(20, 20, 50, 50);
      JButton middle = new JButton("2");
      middle.setBackground(Color.red);
      middle.setBounds(50, 50, 50, 50);
      JButton bottom = new JButton("3");
      bottom.setBackground(Color.yellow);
      bottom.setBounds(78, 78, 50, 50);
      // Place the buttons in different layers
      lp.add(middle, new Integer(2));
      lp.add(top, new Integer(3));
      lp.add(bottom, new Integer(1));
      // Show the frame
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      f.setSize(160, 180);
      f.setVisible(true);
   }
}
```

خروجی برنامه (شکل ۷-۵۴):



شکل (۷- 54) : خروجی برنامه کلاس JLayeredPane

همانطور که در تصویر ۷-۵۴ مشاهده می کنید، سه دکمه با رنگ های سفید، قرمز و زرد به ترتیب در سه لایه مختلف «لایه ۱، لایه ۲، لایه ۳» قرار داده شده اند. با اجرای برنامه متوجه خواهید شد که سه دکمه روی یکدیگر قرار دارند. حال سعی کنید که شماره لایه دو تا از دکمه ها را یکی کنید و نتیجه را با وضعیت فعلی مقایسه نمایید. دقت کنید که حتما نیاز نیست شماره لایه ها از یک شروع شده و به ترتیب نیز باشند. بلکه می توانید از هر شماره دیگر دلخواه نیز شرو کرده و فقط باید شماره ها متفاوت باشند. مثلا می توانستیم به جای سه شماره فوق به ترتیب ۱۰ ، ۲۰ و ۳۰ قرار دهیم.

۷-۶ کلاسهای JTextField و JPasswordField

jTextField وJPasswordField یکی دیگر از زیرکلاسهای Swing میباشد.که یک فضا برای ورود اطلاعات به برنامه کاربرد فراهم میباشد.

شکل (۷- 55) : خروجی برنامه JTextField

به تکه کد زیر دقت کنید.

```
JTextField field = new JTextField(10);
JPasswordField fieldPass = new JPasswordField(10);
```

روش فوق نحوه ی ایجاد یک textField و passwordField است، با مقدار اولیه ۱۰ که تعداد کراکترهای داخل ان را نمایش میدهد

۷–۶–1 سازنده های مهم کلاس

سازنده	توضیحات
`Field=new JTextField(WIDTH);`	حداکثر طول **field** را مشخص میکند
`Field=new JTextField(TOOL_TIP_TEXT_KEY);`	مقدار پیشفرض که میتواند در داخل **field** باشد را تعیین میکند
`Field=new JTextField( TOOL_TIP_TEXT_KEY، WIDTH);`	این سازنده ترکیبی از دو سازنده فوق میباشد که برای تعیین طول و مقدار پیشفرض استفاده میگردد.

جدول (۷– 3) : سازندگان مهم کلاس JTextField وJPasswordField

از جمله سازنده ها و متدهای مهم اها میتوان به موارد زیر اشاره نمود.

۷–۶–2 متدهای مهم کلاس

متدها	توضیحات
`field.setText(TOOL_TIP_TEXT_KEY);`	متنی را در داخل **field** قرار میدهد
`field.setColumns(WIDTH);`	توسط این متد میتوان طول **field** را تعیین نمود
`String field.getText();`	مقدار داخل **field** را برمیگرداند
`Char[] fieldPass.getPassword();`	برای گرفتن پسورد ورودی میباشد
`field.setSize(WIDTH، WIDTH);`	اندازه **field** را تعیین میکند
`field.setEditable(true);`	تعیین میکند ایا **field** قابل ویرایش باشد یا خیر؟
`field.setFont(Font f);`	میتوان فونت استفاده شده در **field** را تعریف نمود.
`field.addActionListener(Listener);`	یک اکشن به **field** اضافه میکند

جدول (۷– 4) : متد های مهم کلاس JTextField وJPasswordField

به مثال زیر توجه کنید(شکل ۷-۵۵ خروجی برنامه زیر میباشد)

```
import javax.swing.*;
public class SimpleJTextFieldExample {
```

```
  public static void main(String[] args) {
    JTextField field = new JTextField(20);

 ایجاد یک اکشن برای تکس باکس
field.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e){
            System.out.println(field.getText());
         }
      });

    JFrame frame = new JFrame( );
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane( ).add(field); // Adds to CENTER
    frame.pack( );
    frame.setVisible(true);
  }
}
```

۷–۷ کلاس JTextArea

JTextArea یکی دیگر از زیر کلاسهای Swing میباشد.برای ایجاد یک فضای باز بهتر برای ورود متن میتوان از JTextArea استفاده نمود.

برای ایجاد یک ناحیه متنی به صورت زیر عمل میکنیم.

```
JTextArea txtArea = new JTextArea("Text Area");
```

شکل (۷– 56) : خروجی برنامه JTextArea

تکه کد فوق یک ناحیه متنی با مقدار اولیه "Text Area" ایجاد میکند.

از جمله سازندگان و متدهای مهم این کلاس میتوان به موارد زیر اشاره نمود.

۷–۷–۱ سازندگان مهم کلاس

سازنده	توضیحات

متن پیشفرض txtArea را تعیین میکند	`txtArea = new JTextArea(text);`
ناحیه متنی با تعیین تعداد خطوط (row) و تعداد حروف در هر خط(columns) ایجاد میکند	`txtArea = new JTextArea(rows، columns);`
همانند سازنده فوق عمل میکند با این تفاوت که متن پیشفرض را تعیین میکند	`txtArea = new JTextArea(text، rows، columns);`

۷-۷-۲ متد ها

جدول (۷- ۵) : سازندگان مهم کلاس JTextArea

متدها	توضیحات
`txtArea.setText(text);`	متنی را در txtArea نمایش میدهد
`string txtArea.getText();`	متن txtArea را برمیگرداند
`txtArea.setColumns(columns);`	حداکثر تعداد حرف در هر خط را تعیین میکند
`txtArea.setRows(rows);`	حداکثر تعداد خطوط txtArea را مشخص میکند
`txtArea.setFont(font);`	فونت ناحیه متنی را تنظیم میکند
`txtArea.setSelectedTextColor(Color.*);`	رنگ پشت زمینه ی متن انتخابی را تعیین میکند
`int txtArea.getLineCount();`	تعداد خطوط ناحیه متنی را برمیگرداند

جدول (۷- 6) : متدهای مهم کلاس JTextArea

به مثال زیر توجه کنید.(شکل ۷-۵۶ خروجی برنامه زیر میباشد)

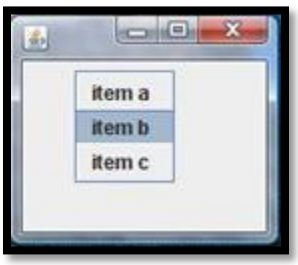
```
// JTextArea Example
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
class JTextAreaExample {
   public static void main(String[] args) {
      JFrame frame = new JFrame();
      Container cp = frame.getContentPane();
      cp.setLayout(new FlowLayout());
      JLabel lab = new JLabel("لطفا متن را وارد کنید");
      cp.add(lab);
      final JTextArea txtArea = new JTextArea("Type here",10,20);
      cp.add(txtArea);
      JButton btn = new JButton("نمایش");
      cp.add(btn);
      btn.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            String s = txtArea.getText();
            System.out.println("mesage="+s); }
```

```
      });
      frame.setSize(300,250);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

این برنامه یک ناحیه متنی در وسط فرم و یک دگمه در انتهای فرم قرار داده،با کلیک روی دگمه متن txtArea نمایش داده میشود

۷–۸ کلاس JPopupMenu

Popupmenu ها منوهای کوچکی هستند که مثلا با راست کلیک کردن بر روی برنامه میتوان انها را انتخاب کرد و اصولا یکسری shourt cut را در خود جای میدهد.



شکل (۷– 57) : خروجی برنامه JPopupMenu

برای درک بهتر موضوع به مثال زیر دقت کنید

```
// JPopupMenu Example
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

class JPopupMenuExample {
متغیری از نوع popupmenu به صورت سراسری ایجاد میگردد
   static JPopupMenu menu = new JPopupMenu("Popup");
متدی برای دادن یک اکشن به زیر منو و اضافه کردن ان به منو
   static void addItem(final String s) {
      JMenuItem item = new JMenuItem(s);
      item.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            System.out.println(e.getActionCommand()+" selected");}
      });
      menu.add(item);
   }
-------------------------------------------------------------------------
   public static void main(String[] args) {
      JPanel panel = new JPanel();
در این قسمت پنل ایجاد شده را روی کلیک موس در حالت listen قرار داده تا به محض
کلیک روی پنل،منو در مختصات x,y نمایش داده شود.
```

```
panel.addMouseListener(new MouseAdapter(){
          public void mousePressed(MouseEvent e) {
             if(e.isPopupTrigger())
                menu.show(e.getComponent(),e.getX(),e.getY());
          }
          public void mouseReleased(MouseEvent e) {
             if(e.isPopupTrigger())
                menu.show(e.getComponent(),e.getX(),e.getY());
          }  });
چندین ایتم به منو اضافه میشود.
       addItem("item a");addItem("item b");addItem("item c");
       JFrame frame = new JFrame();
       Container cp = frame.getContentPane();
       cp.add(panel);
       frame.setSize(200,100);
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frame.setVisible(true);
}        }
```

شکل ۷-۵۷ خروجی برنامه فوق میباشد.

در برنامه فوق تابعی با نام addItem ایجاد شده است که برای تنظیم کردن Action هر زیر منو و اضافه کردن ان به منوی اصلی مورد استفاده قرار میگیرید.

در درون برنامه main پنلی اضافه شده است و متد addMouseListener از پنل برای گوش دادن به کلیک موس و نمایش یا پنهان سازی منوی اصلی با کلیک کردن کاربر روی پنل تنظیم شده است

۷-۹ کلاس JLable

Jlabel یکی دیگر از زیر کلاسهای Swing میباشد که یک متن ثابت را در روی صفحه نمایش خواهد داد. برای ایجاد یک برچسب به صورت زیر عمل میکنیم.

```
JLabel lbl = new JLabel("Defualt Text");
```

شکل (۷- 58) : خروجی برنامه JLable

تکه کد فوق یک برچسب با مقدار اولیه "Defualt Text" ایجاد میکند.

از جمله سازندگان و متدهای مهم این کلاس میتوان به موارد زیر اشاره نمود.

۷-۹-۱ سازندگان مهم کلاس

سازنده	توضیحات
`Lbl = new JLabel(text);`	متن نمایشی برچسب را تعیین میکند
`Lbl = new JLabel(image);`	میتوان تصویری را در برچسب نمایش داد
`lbl=new JLabel(image، horizontalAlignment);`	این سازنده تصویر برچسب و جهت نمایش دادن ان را تعیین میکند
`lbl=new JLabel(text، horizontalAlignment);`	متن و جهت نمایش ان را تعیین میکند
`Lbl = new JLabel(text، image، horizontalAlignment);`	این سازنده نیز ترکیبی از موارد فوق میباشد که متن و تصویر و نیز جهت ان را تنظیم مینماید

جدول (۷- 7) : سازندگان مهم کلاس JLabel

۷-۹-۲ متد های مهم کلاس

متدها	توضیحات
`lbl.setText(text);`	متنی را در lbl نمایش میدهد
`lbl.setSize(width، height);`	اندازه ی شی lbl را تعیین میکند
`lbl.setIcon(image);`	توسط این متد تصویری رادر برچسب نمایش میدهیم
`lbl.setFont(font);`	فونت برچسب را تنظیم میکند
`lbl.setBackground(Color.*);`	پشت زمینه برچسب تعیین میشود
`lbl.setForeground(Color.*);`	رنگ قلم برچسب را میتوان تغییر داد
`String lbl.getText();`	متن نمایشی lbl را برمیگرداند
`int lbl.getText().length();`	طول متن lbl را برمیگرداند

جدول (۷- 8) : متدهای مهم کلاس JLabel

به مثال زیر توجه کنید(شکل ۷-۵۸ خروجی برنامه زیر میباشد)

```
import javax.swing.*;

public class SimpleJLabelExample {
  public static void main(String[] args){

 ImageIcon ico = new ImageIcon("ادرس تصویر");
   JLabel lbl = new JLabel("Picure Member ...",ico,0);

    JFrame frame = new JFrame( );
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane( ).add(lbl); // Adds to CENTER
    frame.pack( );
    frame.setVisible(true);
  }
}
```

۷–۱۰ کلاس JButton

JButton یکی دیگر از زیر کلاسهای Swing میباشد.JButton یک دگمه در interface ایجاد میکند که توسط ان User متواند با کلیک بر روی ان یک Action را انجام دهد.

شکل (۷– 59) : خروجی برنامه JButton

برای ایجاد یک دگمه به صورت زیر عمل میکنیم.

```
JButton btn = new JButton("Click...");
```

تکه کد فوق یک دگمه با برچسب "Click..." ایجاد میکند.

مهمترین سازندگان و متدهای کلاس JButton موارد زیر میباشد.

۷–۱۰–۱ سازندگان مهم کلاس

سازنده	توضیحات

متن نمایشی برچسب دگمه را تعیین میکند	`Btn = new JButton(text);`
متغیری از نوع **Action** را دریافت کرده و بر اساس اکشن **btn** عمل میکند	`Btn = new JButton(action);`
تصویری را روی دگمه نمایش میدهد	`Btn = new JButton(image);`
متن و تصویری را روی دگمه نمایش میدهد	`Btn = new JButton(text،image);`

۷-۱۰-۲ متدهای

جدول (۷- ۹) : سازندگان مهم کلاس **JButton**

متدها	توضیحات
`btn.setAction(Action);`	اکشن یا عمل دگمه را تعیین میکند
`btn.setText(text);`	متن نمایشی برچسب دگمه را تعیین میکند
`btn.setFont(font);`	فونت برچسب **btn** را مشخص میکند
`btn.setEnabled(bool);`	با **true** بودن ورودی این متد،دگمه فعال خواهد بود
`btn.setSize(width، height);`	اندازه **btn** را تنظیم میکند
`btn.addActionListener(ActionListener);`	یک **ActionListener** به دگمه اضافه میکند

جدول (۷- ۱۰) : متدهای مهم کلاس **JButton**

به مثال زیر توجه کنید(خروجی شکل ۷-۵۹):

```
import javax.swing.*;

public class SimpleJLabelExample {
  public static void main(String[] args){


 ImageIcon ico = new ImageIcon("ادرس تصویر");
JButton btn = new JButton("...کلیک کنید ",ico);
    JFrame frame = new JFrame( );
    frame.setTitle("JButton");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane( ).add(btn); // Adds to CENTER
    frame.pack( );
    frame.setVisible(true);
  }
}
```

۷–۱۰–۳ اضافه کردن Action به دکمه

در مثال قبلی ما یک دکمه ایجاد کردیم ولی این دکمه ی با کلیک بر روی آن هیچ کاری انجام نمی دهد ، حالا بایستی عملی انجام دهیم تا بتوانیم روی Action دگمه کنترل (handle) داشته باشیم.

زمانی که کاربر با یک کامپوننت در ارتباط باشد یک اتفاقی ممکن است رخ دهد، برنامه ی بایستی آمادگی رخ دادن ان اتفاقی را داشته باشد، در صورتی که روی اون اتفاق listen شود.

برای ایجاد یک Action به صورت زیر عمل میکنیم.

```
ActionListener act = new ActionListener() {
         public void actionPerformed(ActionEvent e) {
          //            function code
         }
      };
```

برای استفاده از کلاس فوق بایستی کتابخانه زیر را import کنید

```
import java.awt.event.*
```

به کد زیر دقت کنید.

```
// ActionListener Example
import java.awt.event.*;
import javax.swing.*;

class ActionListenerExample{
   public static void main(String[] args) {

      JFrame jframe = new JFrame();
      JButton btn = new JButton("Click ...");

      ActionListener act = new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            System.out.println("action performed");
         }
      };

      btn.addActionListener(act);
      jframe.getContentPane().add(btn);
      jframe.pack();
      jframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      jframe.setVisible(true);
   }
}
```

همانطور که در کد فوق مشاهده میکنید یک Action بانام act ایجاد شده است که اضافه نمودن ان به دگمه،هرگاه روی دگمه کلیک شود پیغام "action performed" نمایش داده خواهد شد.

۷–۱۱ کلاس JCheckBox

JCheckBox یکی دیگر از زیر کلاسهای Swing میباشد.

jcheckbox ها توانایی انتخاب شده به دو صورت را دارند.

۱ -به صورت JRadioButton که در هر لحظه فقط یکی انتخاب خواهد شد.

۲ -قابلیت انتخاب چندین گزینه

شکل (۷– ۶۰) : خروجی برنامه JCheckBox

برای ایجاد یک jcheckbox به صورت زیر عمل میکنیم.

```
JCheckBox chk = new JCheckBox("Checkbox");
```

تکه کد فوق یک checkbox با برچسب "Checkbox" ایجاد میکند

مهمترین سازندگان و متدهای کلاس JCheckBox موارد زیر میباشد.

۷–۱۱–۱ سازندگان مهم کلاس

توضیحات	سازنده
متن نمایشی برچسب را تعیین میکند	`chk = new JCheckBox(text);`
اکشن صورت گرفته توسط chk را تعیین میکند	`chk = new JCheckBox(Action);`
تصویری را در کنار chk قرار میدهد	`chk = new JCheckBox(image);`
تصویری را نمایش داده و اگر selected = true باشد chk در حالت انتخاب قرار میگیرد	`chk = new JCheckBox(image، selected);`
متن برچسب و تصویر chk را تنظیم مینماید	`chk = new JCheckBox(text، image);`
متن برچسب و تصویر و در حالت انتخاب chk را مشخص میکند	`chk = new JCheckBox(text، image، selected);`

جدول (۷- ۱۱) : سازندگان مهم کلاس JCheckbox

۷-۱۱-۲ متد های مهم کلاس

توضیحات	متدها
متن برچسب chk را تغییر میدهد	`chk.setText(text);`
با bool == true شی checkbox در حالت انتخاب قرار میگیرد	`chk.setSelected(bool);`
توسط این متد میتوان تصویر chk را تغییر داد	`chk.setIcon(image);`
با bool == true شی checkbox فعال میباشد	`chk.setEnabled(bool);`
یک Action میتوان به chk اضافه نمود	`chk.addActionListener(ActionListener);`
ارایه ای از موارد انتخابی chk را برمیگرداند	`Object[]  chk.getSelectedObjects();`
نشان میدهد ایا chk انتخاب شده است یا خیر	`jCheckBox.isSelected();`

جدول (۷- ۱۲) : متدهای مهم کلاس JCheckbox

توجه ۲ : معمولا به JCheckBox ها tickBox نیز گفته میشود

به کد زیر دقت کنید.(شکل ۷-۶۰ خروجی برنامه زیر میباشد)

```
import java.awt.event.*;
import javax.swing.*;
class JCheckBoxExample {
 public static void main(String[] args) {
  JFrame frame = new JFrame();
  frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
  Container cp = frame.getContentPane();

  Box box = new Box(BoxLayout.Y_AXIS);
  cp.add(box);
  box.add(new JLabel("ورزش مورد علاقه خود را انتخاب نمایید"));
  String[] sports = {"Football", "Rugby", "Cricket", "Badminton", "Baseball"};

 JCheckBox[] chk = new JCheckBox[sports.length];   ارايه اي از چک باکسي

حلقه براي مقداردهي کردن به هر چک باکس واضافه کردن اکشن به انها

      for (int i = 0; i < sports.length; i++){
         chk[i] = new JCheckBox(sports[i]);          تنظيم برچسب چک باکس
         box.add(cba[i]);
         chk[i].addItemListener(new ItemListener(){

   public void itemStateChanged(ItemEvent e){
               JCheckBox jCheckBox = (JCheckBox)e.getSource();
               if(jCheckBox.isSelected())
                  System.out.println(jCheckBox.getText() + " selected");
               else
                  System.out.println(jCheckBox.getText() + " deselected");
            }
         });
      }
      frame.pack();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

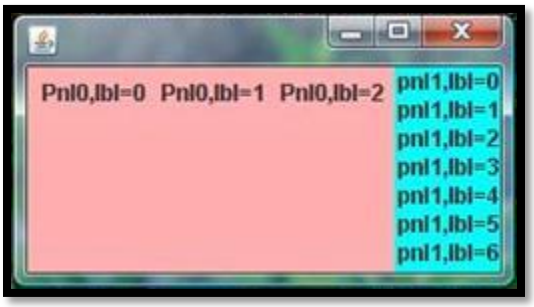
در برنامه از یک ارایه ای از نوع checkbox استفاده شده است که در یک حلقه ی for هر یک از checkbox ها مقداردهی شده و Action ی به انها اضافه میگردد

در مثال پیش ActionListener کلاسی بود که یک action به دگمه اضافه میکرد ولی در این مثال برای اضافه کرده یک action به checkbox ازکلاس ItemListener استفاده میشود و همچنین تابعی که action های این کلاس را کنترل میکند به صورت زیر میباشد.

```
public void itemStateChanged(ItemEvent e){}
```

۱۲-۷ کلاس JPanel

JPanel یکی از زیر کلاسهای JComponent و تمام خصوصیات انرا دارا میباشد و میتواند به container هایی همچون JFram اضافه بشود.



شکل (۷- 61) : خروجی برنامه JPanel

نکته ۷ : JComponent خود یک زیر کلاس از Container است در نتیجه JPanel هم میتواند در داخل Container ها اضافه شود

برای ایجاد یک پنل به صورت زیر عمل میکنیم.

```
JPanel pnl = new JPanel();
```

تکه کد فوق یک پنل ایجاد میکند

مهمترین سازندگان و متدهای کلاس JPanel موارد زیر میباشد.

۷-۱۲-۱ سازندگان مهم کلاس

سازنده	توضیحات
`Pnl = new JPanel(layoutManager)`	نوع لایه بندی پنل را مشخص میکند (در بخش لایه بندی به این معقوله پرداخته شده است)

جدول (۷- ۱۳) : سازندگان مهم کلاس JPanel

۷-۱۲-۲ متدهای مهم کلاس

متد	توضیحات
`pnl.setLayout(layoutManager);`	لایه بندی پنل را تغییر میدهد
`pnl.setBounds(Rectangle);`	اندازه و محل قرارگیری پنل را مشخص میکند

pnl.setVisible(bool);	بادرست بودن **bool** پنل نمایش داده میشود
pnl.setEnabled(bool);	با درست بودن **bool** پنل فعال خواهد شد.
pnl.setFont(font);	باعث تغییر فونت **pnl** خواهد شد.
pnl.setBackground(Color.*);	رنگ پشت زمینه پنل را تغییر میدهد

جدول (۷- ۱۴) : متدهای مهم کلاس JPanel

به مثال زیر توجه کنید.(شکل ۷-۶۱ خروجی برنامه زیر میباشد)

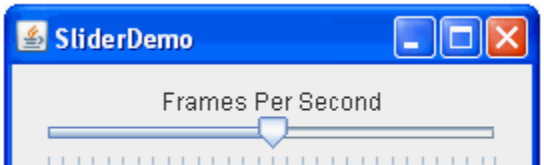
```
// JPanel Example
import java.awt.*;
import javax.swing.*;
class JPanelExample {
این مثال دو پنل ساخته و به هر یک تعدادي برچسب اضافه و در نهایت پنلها را به
 فرم اضافه میکند
   public static void main(String[] args){
      JPanel jpanel_0 = new JPanel();
      jpanel_0.setBackground(Color.pink);
      for(int i=0;i<3;++i) jpanel_0.add(new JLabel(" Pnl0,lbl="+i));

JPanel jpanel_1 = new JPanel(new GridLayout(0,1));
      jpanel_1.setBackground(Color.cyan);
      for(int i=0;i<7;++i) jpanel_1.add(new JLabel("  pnl1,lbl="+i));
ایجاد فرم و اضافه کردن دو پنل به کانتینر فرم
      JFrame frame = new JFrame();
      Container cp = frame.getContentPane();
      cp.add(jpanel_0,BorderLayout.WEST);
      cp.add(jpanel_1,BorderLayout.EAST);

      frame.pack();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
}
```

۷-۱۳ کلاس JSliders

شی Jslider یک کلاس از کتابخانه swing میباشد.

کامپوننت JSlider به کاربر این اجازه رو میدهد تا یک مقدار عددی در یک بازه داشته باشد

استفاده از این شی در برنامه و با ترکیبی از اشیاء دیگر کلاس Swing میباشد.به عنوان مثال با تغییر نوار لغزنده بتوان مکان یک شی را تغییر داد.



شکل (۷- 62) : شکل JSlider

تعریف کلاس JSlider به صورت زیر میباشد

```
JSlider jsl = new JSlider(0, 100, 20);
```

تکه کد فوق یک نوارلغزنده با حداقل صفروحداکثر۱۰۰و مقدار اولیه ۲۰ ایجاد میکند

۷-۱۳-۱ سازندگان مهم کلاس

سازنده	توضیحات
`Jsliders()`	کلاس **Jslider** افقی و با محدوده ی حداقل ۰ و حداکثر ۱۰۰ ایجاد میکند
`JSlider(int min، int max)` `JSlider(int min، int max، int value)`	شی **Jslider** افقی با محدوده ی حداقل و حداکثر مشخص شده ایجاد میکند ارگومان سوم مقدار اولیه شی را مشخص میکند
`JSlider(int orientation)` `JSlider(int orientation، int min، int max، int value)`	شی **Jslider** با جهت مشخص شده (افقی یا عمودی) ایجاد میکند. **JSlider.HORIZONTAL** (افقی) **JSlider.VERTICAL** (عمودی) شی **Jslider** با جهت و محدوده ی حداقل و حداکثر و مقدار اولیه شی ایجاد میکند.
`JSlider(BoundedRangeModel)`	متغیری از نوع **BoundedRangeModel** ایجاد کرده و توسط این متغییر میتوان جهت و محدوده و نیز مقدار اولیه شی را مشخص نمود و به شی **JSlider** نسبت داد.

۷–۱۳–۲ متد های مهم کلاس

جدول (۷- ۱۵) : سازندگان مهم کلاس JSlider

	توضیحات
`void setValue(int)` `int getValue()`	متد **setValue** مقدار اولیه شی را مشخص میکند و باعث تغییر نوار لغزنده میگردد. متد **getValue** برای بدست اوردن مقدار شی استفاده میگردد. خروجی این متد نوع عددی میباشد
`void setOrientation(int)` `int getOrientation()`	متد **setOrientation** جهت شی را مشخص میکند. **Jslider.HORIZONTAL** جهت افقی و **Jslider.VERTICAL** جهت عمودی شی را تنظیم میکند. متد **getOrientation** جهت شی را (۰–افقی،۱–عمودی) به صورت عددی برمیگرداند
`void setInverte(boolean)` `boolean getInverted()`	متد **setInverted** تعیین میکند شی لغزنده در حالت افقی مقدار حداکثر در سمت چپ قرار گیرد و یا در حالت عمودی در پایین قرار گیرد. متد **getInverted** تعیین میکند شی به صورت معکوس میباشد یا خیر

`void setMinimum(int)`
`int getMinimum()`
`void setMaximum(int)`
`int getMaximum()`
	متد **setMinimum** مقدار حداقل بازه ی شی را مشخص میکند.
	متد **getMinimum** مقدار حداقل بازه را برمیگرداند
	متد **setMaximum** مقدار حداکثر بازه ی شی را مشخص میکند.
	متد **getMaximum** مقدار حداکثر بازه را برمیگرداند.

`void setMajorTickSpacing(int)`
`int getMajorTickSpacing()`
`void setMinorTickSpacing(int)`
`int getMinorTickSpacing()`
	متد **setMajorTickSpacing** مقدار محدوده ی تیک بزرگ شی را مشخص میکند.
	متد **getMajorTickSpacing** مقدار محدوده ی تیک بزرگ شی را برمیگرداند.
	متد **setMinorTickSpacing** مقدار محدوده ی تیک کوچک شی را مشخص میکند.
	متد **getMinorTickSpacing** مقدار محدوده ی تیک کوچک شی را

برمیگرداند.	
متد **setPaintTicks** مشخص میکند ایا تیک روی شی نمایش داده شود یا خیر.	`void setPaintTicks(boolean)`
متد **getPaintTicks** مشخص میکند تیک روی شی وجود دارد یا خیر ؟	`boolean getPaintTicks()`
متد **setPaintLabels** مشخص میکند ایا برچسب روی شی نمایش داده شود یا خیر.	`void setPaintLabels(boolean)`
متد **getPaintLabels** مشخص میکند برچسب روی شی وجود دارد یا خیر؟	`boolean getPaintLabels()`

۷-۱۴ کلاس دیالوگ

جاوا دارای چندین **جدول (۷- ۱۶) : متدهای مهم کلاس JSlider** نامه کمک زیادی کند.

این کادرها در حقیقت همان پنجره های عمومی هستند که در بیشتر برنامه های تحت ویندوز مشاهده کرده اید.

به علاوه این کادرها دارای خاصیتهاومتدهای فراوانی هستند که به وسیله ی انها میتوانید این کادرها را با قسمتهای مختلف برنامه ی خود هماهنگ کنید.

۷-۱۴-۱ کادر محاوره ای پیغام

این کادر عموما برای نمایش یک پیغام به کاربر و دریافت جواب به ان پیغام استفاده میشود

شما میتوانید با کلاس JoptionPane پیغام محاوره ای نمایش دهید.

این پیغام میتواند به اشکال مختلف نمایش داده شود.(شکل ۷-۶۳)

شکل ۷-۶۳

شکل (۷- 63) : ایکن های ویندوز و JOptionPane جاوا

شکل ساده پیغام به صورت نمایش Ok

```
Frame frm = new Frame();
JOptionPane.showMessageDialog(frm,"پیغام قابل نمایش","عنوان",ایکن);
```

حالت کلی کلاس JOptionPane

```
1. final JOptionPane optionPane =  new JOptionPane (
2. Component parentComponent,
3. Object message,
4. String title,
5. int optionType,
6. int messageType,
7. Icon icon,
8. Object[] options,
9. Object initialValue)
```

توضیحات مربوط به کد فوق

خط ۲. Component parentComponent

شما به جای متغیر parentComponent میتوانید مقدارnull یا متغیری از نوع JFrame تعریف کرده و یا میتوانید فرمی به برنامه اضافه کرده و متغیری از این فرم تعریف کرده و به جای این متغیر استفاده کنید

خط ۳. Object message

متغیر message پیامی است که در دیالوگ باید نمایش داده شود. معمولا از نوع string میباشد.

نکته ۸ : در جاوا برای برای رفتن به خط بعدی و نمایش پیغام چند خطی از \n استفاده میکینم

خط ۴. String title

متغیر title عنوان دیالوگ را مشخص میکند

خط ۵. int optionType

دگمه هایی که در پایین دیالوگ نمایش داده میشود را مشخص میکند.

```
DEFAULT_OPTION          حالت پیش فرض ( ok ) را نمایش میدهد
, YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION.
```

خط ۶. int messageType

ارگومانی که نوع ایکن دیالوگ را مشخص میکند. نوع ارگومان بدون ایکن میباشد.

توضیحات	پیام
پیام خطا	ERROR_MESSAGE
پیام اطلاع رسانی	INFORMATION_MESSAGE
پیام اخطار	WARNING_MESSAGE
پیام سوالی (سوال)	QUESTION_MESSAGE
پیام ساده	PLAIN_MESSAGE

جدول (۷- ۱۷) : خصوصیات پیام دیالوگ

خط ۷. Icon icon

کاربر میتواند ایکن دیالوگ را به دلخواه تغییر دهد. icon متغیریست از نوع Imageicon که در دیالوگ استفاده میگردد.

```
ImageIcon ico = new ImageIcon("ادرس تصویر");
```

خط ۸. Object[] options

این ارگومان برای امکانات زیادی استفاده میشود. معمولا از نوع رشته ای میباشد. برای مقدار جعبه انتخاب یا تغییر برچسب دگمه ها و... استفاده میگردد

خط ۹. Object initialValue

مقدار پیشفرض انتخابی دیالوگ را مشخص میکند

نمونه هایی از دیالوگ برنامه


```
Frame frm = new Frame();
JOptionPane.showMessageDialog(frm،
"نمایش پیغام با ایکن پیشفرض"،"پیشفرض");
```

شکل (۷- ۶۴) : نمایش پیغام با ایکن پیشفرض


```
Frame frm = new Frame();
JOptionPane.showMessageDialog(frm،
    "نمایش پیغام با ایکن اخطار"،
"اخطار"،
    JOptionPane.WARNING_MESSAGE);
```

شکل (۷- ۶۵) : نمایش پیغام با ایکن اخطار


```
Fram frm= new Frame();
JOptionPane.showMessageDialog(frm،
    "نمایش پیغام با ایکن خطا"،
    "خطا"،
    JOptionPane.ERROR MESSAGE);
```

شکل (۷- ۶۶) : نمایش پیغام با ایکن خطا


```
Fram frm = new Frame();
JOptionPane.showMessageDialog(frm،
    "نمایش پیغام بدون ایکن"،
    "بدون ایکن"،
    JOptionPane.PLAIN_MESSAGE);
```

شکل (۷- ۶۷) : نمایش پیغام بدون ایکن


```
JOptionPane.showMessageDialog(frame،
    "نمایش پیغام با ایکن دلخواه"،
    "ایکن دلخواه"،
    JOptionPane.INFORMATION_MESSAGE،
    icon);
```

شکل (۷- ۸

۷-۱۴-۲ دیالوگهای خاص

در جاوا همچین قابلیتی وجود دارد تا بتوان یک دیالوگ خاص با تغییر برچسب دگمه ها و تغییر ایکن ایجاد کرد.

شکل (۷- ۶۹) : خروجی برنامه دیالوگ خاص

کد مربوطه: (شکل ۷-۶۹ خروجی کد زیر میباشد)

```
1. Object[] options = {"بله - با ذخیره اطلاعات","بله - بدون ذخیره
   اطلاعات","لغو عملیات"};
2. Frame frm = new Frame();
3. Icon ico  = new ImageIcon("ادرس تصویر");
        4. int n = JOptionPane.showOptionDialog(frm," مایلید از برنامه
           خارج شوید ؟ " ,
        5. "خروج از برنامه",
        6. JOptionPane.YES_NO_CANCEL_OPTION,
        7. JOptionPane.INFORMATION_MESSAGE,
        8. ico,
        9. options,
        10.     options[2]);
```

توضیح کد

خط ۱: متغیری از نوع object ایجاد کرده و لیست برچسب دگمه ها را در ان مینویسیم.

خط ۲: متغیری از نوع فرم ایجاد میکند

خط ۳: یک ایکن (تصویر) با ادرس مشخص ایجاد میکند.(توسط کلاس ()Imageicon)

خط ۶: نوع دگمه های دیالوگ را تعیین میکند.

این دیالوگ یکی از حالات سازنده ی کلاس JptionPane میباشد که با تنظیم کردن حالت پیش فرض دیالوگ JOptionPane.INFORMATION_MESSAGE (خط ۷) و ایجاد یک ایکن دلخواه (خط ۸) میتوان ان را ایجاد کرد.

خط ۱۰: دگمه ی انتخابی در هنگام نمایش دیالوگ میباشد.(در حالت پیشفرض کدام دگمه انتخاب شود)

۷-۱۴-۳ دیالوگ ورودی (InputBox)

در جاوا این امکان را به کاربر میدهد تا بتوان اطلاعات را از کاربر دریافت نماید.این کار با متد ShowInputDialog از کلاس jOptionPanel امکان پذیر میباشد.

مثال: نمایش دیالوگ ورودی با انتخاب چندمورد توسط کاربر

شکل (۷- ۷۰) : خروجی برنامه دیالوگ ورودی

شکل (۷- ۷۱) : دیالوگ ورودی

کد مربوطه (شکل ۷-۷۰ خروجی کد زیر میباشد)

```
1.    Icon icon=new ImageIcon("ادرس تصویر");
2.      Frame frm = new Frame();
3.    Object[] possibilities = {"حسین" ,"رضا" ,"علي"};
4.    String ret = (String) JOptionPane.showInputDialog(frm,
5.     "لطفا نام مورد نظر را وارد نمایید ؟\nکاربر گرامي"
6.             "انتخاب نام",
7.             JOptionPane.PLAIN_MESSAGE,
8.            icon,
9.              possibilities,
10.            "رضا");
```

توضیح کد فوق:

خط ۳. متغیر possibilities از نوع اجکت میباشد و لیست مقادیر را در جود جای میدهد.

خط ۱۰. مقدار پیشفرض لیست انتخاب میگردد.

توجه ۹ : برای ایجاد دیالوگ شکل ۷-۷۱،کافیست خط ۹ کد فوق را به null تغییر دهید

۷-۱۵ کلاس تایمر

کلاس تایمر این امکان را به شما خواهد داد تا در یک زمان مشخص رویدادی را فراخوانی و اجرا کنید.

شی timer در جاوا را میتوان از دوکلاس روبرو استفاده کرد

```
1.java.util.Timer;
2.javax.swing.Timer;
```

کلاس ۷-۱۵-۱ Java.Util.Timer

در این کلاس برای استفاده از تایمر باید یک وظیفه ایجاد کرده و به تایمر نسبت داده شود.

برای ایجاد یک وظیفه از کلاس java.util.TimerTask استفاده میگردد.

سازندگان TimerTask

سازنده	توضیحات
`public void schedule( TimerTask task.long time)`	یک وظیفه را در یک ساعت مشخص اجرا میکند ارگومان **time** از نوع **Date** میباشد
`public void schedule( TimerTask task. long delay)`	یک وظیفه را در پس از زمانی مشخص اجرا میکند ارگومان **delay** از نوع **long** میباشد و زمان را مشخص میکند
`public void schedule( TimerTask task.Date time.long delay)`	وظیفه را در یک ساعت مشخص و پس از گذشت **delay** میلی ثانیه از ان ساعت اجرا میکند
`public void schedule( TimerTask task.long delay.long)`	وظیفه را هر **period** میلی ثانیه و با **delay** میلی ثانیه تاخیر در اولین بار اجرا میکند

جدول (۷- ۱۸) : سازندگان مهم کلاس TimerTask

مثال:

در این مثال کلاسی به نام RunMeTask ایجاد کرده که از TimerTask ارث بری میکند.

```
public class RunMeTask extends TimerTask
{
public void run() {
System.out.println("اجرا هر 6 ثانیه و با 3 ثانیه تاخیر در اولین بار اجرا");
                }
        } //end class
import java.util.Timer;
import java.util.TimerTask;
public class App
{
    public static void main( String[] args )
    {
      TimerTask task = new RunMeTask();
      Timer timer = new Timer();
      timer.schedule(task, 3000,60000);
     }  }
```

۷-۱۵-۲ کلاس Javax.Swing.Timer

در این کلاس باید از ActionListener استفاده نمود و عملیاتی که میبایست صورت بگیرد در داخل متد actionPerformed پیاده سازی میگردد.

متدهای مهم شی تایمر javax.swing.Timer

متد	توضیحات
`timer.start();`	باعث فعال شدن شی تایمر میگردد
`timer.stop();`	عملیات شی تایمر متوقف میگردد
`timer.setDelay(int delay);`	این متد باعث تنظیم کردن تاخیر در اجرای تایمر میگردد. تنظیم زمان شی برحسب میلی ثانیه میباشد
`Int timer.getDelay();`	زمان تاخیر تایمر برگردانده میشود
`timer.addActionListener( actionPerformed varibale);`	با استفاده از این متد میتوان یک عمل به تایمر اضافه نمود
`timer.removeActionListener( actionPerformed varibale);`	یک عمل که قبلا به تایمر اضافه شده است از ان حذف خواهد شد
`Bool timer.isRunning();`	تعیین میکند ایا تایمر درحال اجرا میباشد یا خیر.
`ActionListener[] Varible= timer.getActionListeners();`	لیست تمام ActionListener مربوط به تایمر را در متغیری از همان نوع برمیگرداند

جدول (۷- ۱۹) : متدهای مهم کلاس Timer

ایجاد برای شی تایمر به صورت زیر عمل میکنیم

```
   ActionListener variable = new ActionListener() {
     public void actionPerformed(ActionEvent e) {
کد نوشته شده در این قسمت توسط تایمر هر t میلی ثانیه اجرا میگردد//
     } };
```

مثال

در این مثال از 2 ActionListener استفاده شده است که هر ۲۰۰۰ میلی ثانیه پیام مشخصی چاپ میکند

```
 javax.swing.Timer timer = new javax.swing.Timer(2000, new
ActionListener ()    {public void actionPerformed(ActionEvent e)
          System.out.println("Action 1”);}})
```

تعریف ActionListener

```
ActionListener acn=new ActionListener() {
      public void actionPerformed(ActionEvent e) {
          System.out.println("Action 2"); }};

timer.addActionListener(acn);
timer.start();
```

۷–۱۶ کلاس Jlist

این کلاس یکی از زیر کلاسهای swing است که این امکان را به کاربر میدهد تا چندین ایتم را در یک یا چند ستون در لیستی نمایش دهد.

تکه کد زیر یک متغیر از کلاس jlist ایجاد میکند.

```
JList lst = new JList();
```

۷–۱۶–۱ متد های مهم کلاس JList

متد	توضیحات
jList1.setListData(Object[] listData);	لیستی از ایتم ها که نوع **object** میباشد را به لیست اضافه میکند.
jList1.setLayoutOrientation(jList1.HORIZONTAL_WRAP)	ایتم ها را از چپ به راست به صورت افقی در لیست نمایش میدهد.
jList1.setLayoutOrientation(jList1.VERTICAL_WRAP);	ایتم ها را از بالا به پایین به صورت عمودی در لیست نمایش میدهد.
jList1.setLayoutOrientation(jList1.VERTICAL);	ایتم ها را صورت عمودی در لیست نمایش میدهد.

تنها یک ایتم میتواند در هر لحظه انتخاب شود
چندین ایتم همجوار در هر لحظه میتواندانتخاب شود،
(حالت پیشفرض) چندین ایتم به صورت ترکیبی در هر لحظه انتخاب میشود
(برای انتخاب چندین ایتم lrtC نگه داشته میشود)
	`jList1.setSelectionMode( ListSelectionModel.SINGLE_SELECTION)`
	`jList1.setSelectionMode(ListSelectionModel.SINGLE_INTERVAL_SELECTION)`
	`jList1.setSelectionMode( ListSelectionModel.MULTIPLE_INTERVAL_ SELECTION)`

این متد قالبی را به لیست اضافه میکند (قالب متغیری از نوع DefaultListModel میباشد که لیستی از ایتمها را در خود نگه داری میکند)	`jList1.setModel(DefaultListModel varible)`
شماره ایتم انتخابی را برمیگرداند	`Int varible=jList1.getSelectedIndex()`
لیستی از شماره ایتم های انتخابی را برمیگرداند (شماره ها به ترتیب صعودی میباشد)	`Int[] varible=jList1.getSelectedIndices()`
مقدار ایتم انتخابی لیست را برمیگرداند	`Object varible=jList1.getSelectedValue()`
لیست مقادیر انتخابی لیست را برمیگرداند	`Object[] varible=jList1.getSelectedValues()`
اخرین شماره ایتم درج شده در لیست را برمیگرداند	`Int varible=jList1.getLastVisibleIndex()`
کوچکترین شماره ایتم انتخابی در لیست را برمیگردند بزرگترین شماره ایتم انتخابی در لیست را برمیگردند	`Int varible=jList1.getMinSelectionIndex( )` `Int varible=jList1.getMaxSelectionIndex( )`
ارتفاع یا بلندی ایتم های لیست را مشخص میکند عرض یا پهنای ایتم های لیست را مشخص میکند	`jList1.setFixedCellHeight(int varible);` `jList1.setFixedCellWidth(int varible);`
ایا لیست فعال باشد یا خیر	`jList1.setEnabled(false/true)`
توسط این متد میتوان یکی از ایتم های لیست را در حالت انتخاب تنظیم نمود توسط این متد میتوان چندین ایتم از لیست را در حالت انتخاب تنظیم نمود	`jList1.setSelectedIndex(int index);` `jList1.setSelectedIndices(int[] indices);`
ایتمی از لیست که دارای مقدار یکسانی با ورودی تابع باشد را انتخاب میکند.با true بودن ارگومان دوم	`jList1.setSelectedValue( object varible.false/ true);`

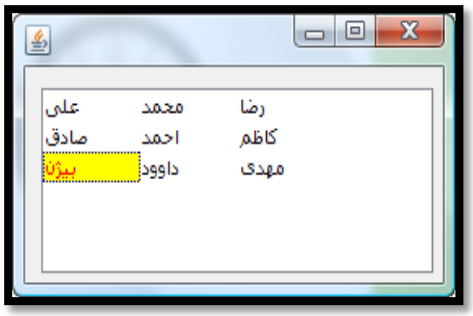
متد، ایتم انتخاب با وجود اسکرول در لیست باز هم نمایش داده میشود.	
این متد باعث تغییر پشت زمینه ی ایتم انتخابی لیست میگردد	`jList1.setSelectionBackground( Color varible)`
این متد باعث تغییر رنگ قلم ایتم انتخابی لیست میگردد	`jList1.setSelectionForeground(Color varible);`
تعداد ردیفهای لیست را تعیین میکند	`jList1.setVisibleRowCount( int VisibleRowCount);`

جدول (۷- ۲۰) : متدهای مهم کلاس JList

مثال ۱

```
Object[]
listData={"علي","محمد","رضا","صادق","احمد","كاظم","بيژن","داوود","مهدي"};
  jList1.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);

  jList1.setLayoutOrientation(jList1.VERTICAL_WRAP);

  jList1.setVisibleRowCount(3);          تعداد نمایش ردیف در لیست
  jList1.setFixedCellWidth(60);          تنظیم کردن عرض سلول در لیست
  Color color=null;
  jList1.setSelectionBackground(color.YELLOW);       پشت زمینه انتخاب
  jList1.setSelectionForeground(color.RED);          رنگ قلم
  jList1.setListData(listData);
```

در انتخاب setLayoutOrientation، ۳ حالت وجود دارد (توضیحات در جدول)



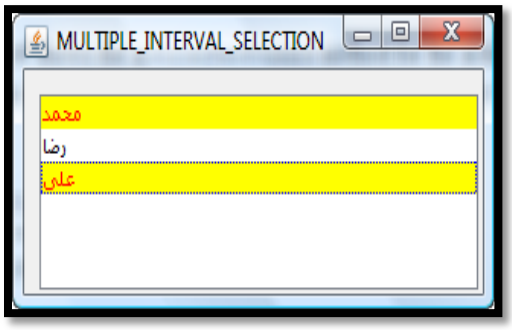
شکل (۷- ۷۲) : صفت Jlist.HORIZONTAL_WRAP

شکل (۷- 73) : صفت VERTICAL_WRAP ,VERTICAL

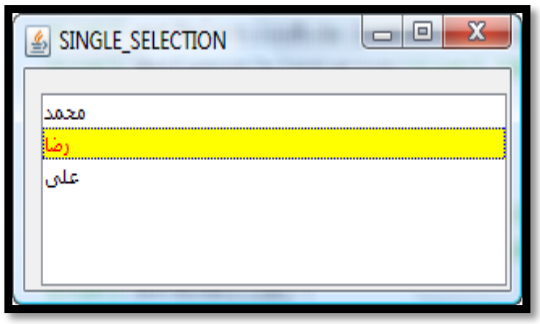
مثال ۲: (تصویر ۷-۷۲ خروجی کد زیر میباشد)

```
DefaultListModel lst=new DefaultListModel();
  lst.addElement("محمد");
  lst.addElement("رضا");
  lst.add(2, "علي");

  jList1.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);

  jList1.setLayoutOrientation(jList1. HORIZONTAL_WRAP);
  jList1.setVisibleRowCount(2);
  jList1.setFixedCellWidth(60);
  Color color=null;
  jList1.setSelectionBackground(color.YELLOW);
  jList1.setSelectionForeground(color.RED);
  jList1.setModel(lst);
```

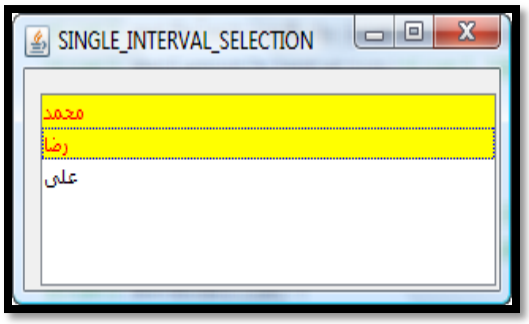
در انتخاب setSelectionMode، ۳ حالت وجود دارد (توضیحات در جدول)



شکل (۷- 74) : انتخاب ListSelectionModel.Multiple_INTERVAL_SELECTION



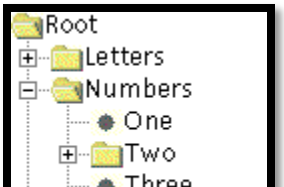
شکل (۷- 75) : انتخاب ListSelectionModel.SINGLE_SELECTION



شکل (۷- 76) : انتخاب ListSelectionModel.SINGLE_INTERVAL_SELECTION

۷-۱۷ کلاس JTree

اگر در برنامه ای قصد نمایش اطلاعاتی را دارید که بخشی از داده ها به عنوان پدر بخش دیگر محسوب می شوند، یعنی در واقع قصد نشان دادن سلسله مراتب بین داده ها را دارید، می توانید از کامپونت JTree استفاده نمایید. البته توجه به این نکته ضروری است که یک شئ JTree واقعاً حاوی اطلاعات نیست بلکه داده‌های موجود را در قالب یک فرم گرافیکی ساده و قابل فهم برای کاربران نمایش می دهد. شکل زیر نشان دهنده یک نمونه از کامپونت JTree یا درخت در جاوا می باشد.



شکل (۷- 77) : کلاس JTree

همانطوریکه در شکل بالا دیده می شود، یک درخت داده ها را به صورت عمودی نمایش می دهد. هر سطر یک درخت، حاوی یک بخش از داده هاست. هریک از این بخش ها « item » را یک گره « node » می نامند. هر درخت دارای یک گره ریشه است که به آن Root گفته می شود.

تمامی node های دیگر درخت از این گره مشتق می شوند. یک درخت هم می تواند دارای فرزندانی باشد و هم نه. گره های دارای فرزند را به ترتیب branch nodes یا شاخه ودیگر گره ها را leaf nodes یا برگ می گویند. یک گره مشخص در درخت همچنین می تواند بوسیله یک مسیر یا TreePath نمایش داده شود.

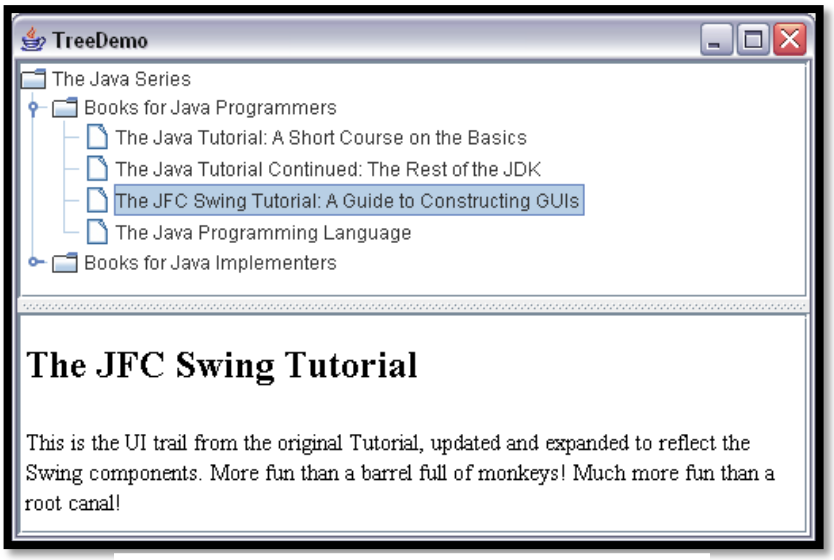
۷-۱۷-۱ چگونه در جاوا یک درخت بسازیم ؟

شکل زیر خروجی برنامه ای را نشان می دهد که در قسمت بالای آن، از یک درخت برای گروه بندی کردن اطلاعات موجود استفاده شده است.

شرح برنامه به این صورت است که توسط چند فایل HTML در موضوعات مختلف جاوا مطالبی تحت عنوان مقالات آموزشی تهیه شده است.

حال برای مطالعه راحتتر این مطالب، در یک درخت، دو شاخه یا **branch** جهت مشخص کردن عنوان گروه های موجود و در هر شاخه تعدادی گره یا برگ یا **node** برای مشخص کردن موضوع محتویات هریک از فایل ها قرار داده شده است.

با کلیک کردن بر روی هریک از گره ها، در قسمت پایین فرم محتویات فایل مورد نظر نمایش داده خواهد شد.(شکل ۷-۷۸)



شکل (۷- 78) :خروجی برنامه JTree

کلاس JTree زیر کلاسی از بسته javax.swing می باشد. این کلاس دارای هفت متد سازنده به شرح زیر می باشد:

JTree() Returns a JTree with a sample model.
JTree(Hashtable<?,?> value) Returns a JTree created from a Hashtable which does not display with root.
JTree(Object[] value) Returns a JTree with each element of the specified array as the child of a new root node which is not displayed.
JTree(TreeModel newModel) Returns an instance of JTree which displays the root node -- the tree is created using the specified data model.
JTree(TreeNode root) Returns a JTree with the specified TreeNode as its root, which displays the root node.
JTree(TreeNode root, boolean asksAllowsChildren) Returns a JTree with the specified TreeNode as its root, which displays the root node and which decides whether a node is a leaf node in the specified manner.
JTree(Vector<?> value) Returns a JTree with each element of the specified Vector as the child of a new root node which is not displayed.

جدول (۷- ۲۱) : سازندگان کلاس JTree

قدم اول ایجاد یک شی از کلاس JTree و اضافه کردن گرههای مورد نظر به آن می باشد.

برای انجام این کار عبارات روبرو را به برنامه اضافه می کنیم.

```
private JTree tree;
...
public TreeDemo() {
    ...
    DefaultMutableTreeNode top =
        new DefaultMutableTreeNode("The Java Series");
    createNodes(top); // متد روبرو توسط کاربر ایجاد میگردد
    tree = new JTree(top);
    ...
```

اگر به کد فوق دقت کنید خواهید دید که برای اضافه کردن فرزندان،از یک متد با نام createNodes که توسط کاربر ایجاد شده، استفاده شده است.

عبارات زیر قسمتی از بدنه این متد است که نشان می دهد، چگونه می توان به یک درخت یک فرزند اضافه نمود.

```
private void createNodes(DefaultMutableTreeNode top) {
    DefaultMutableTreeNode category = null;
    DefaultMutableTreeNode book = null;

                                /*Books for Java Programmers    ایجاد شاخه */
    category = new DefaultMutableTreeNode("Books for Java Programmers");
    top.add(category);

                               /*ایجاد برگه هاي شاخه*/
    book = new DefaultMutableTreeNode(new BookInfo("The Java Tutorial: A Short
Course on the Basics","tutorial.html"));
    category.add(book);
    ...
                                /* Books for Java Implementers   ایجاد شاخه */
 category = new DefaultMutableTreeNode("Books for Java Implementers");
    top.add(category);
                               /*ایجاد برگه هاي شاخه*/
    book = new DefaultMutableTreeNode
(new BookInfo("The Java Virtual Machine Specification ","vm.html"));
    category.add(book);
    ...    }
```

بدنه متد فوق نشان می دهد که ابتدا باید به گره ریشه، اولین گروه یا شاخه را اضافه نمود

«قسمت Books for Java Programmers » سپس فرزندان مربوط به این گروه را اضافه نمود.

«قسمت Books for Java Implementers » و به همین ترتیب گروه دوم و فرزندان گروه دوم و..

کلاس زیر در مثال های کد استفاده شده است که نام و ادرس صفحه کتاب را برمیگرداند.

```
Private class BookInfo{
public String bookName;
public URL bookURL;
 //سازنده ي از كلاس
public BookInfo(String book,String filename)//متغير ارجاعي {
bookName=book;
bookURL = getClass().getResource(filename);
if(bookURL==null) {System.err.println("Couldn't file file :"+filename);}}
public String toString(){return bookName;}}
```

البته می توان برای ایجاد گره ها از یک حلقه استفاده نمود:

```
    // نام برچسبهاي هر گره
   String NameNodes[] =
{"First Node","Second Node","Third Node","Fourth Node","Fifth Node"};
    DefaultMutableTreeNode Node[]= new
DefaultMutableTreeNode[NameNodes.length];
    …
    DefaultMutableTreeNode top = new DefaultMutableTreeNode("Top Node");
    for (int i=0;i<5;i++){
    Node[i] = new DefaultMutableTreeNode(NameNodes[i]);
    top.add(Node[i]);    }
     tree = new JTree(top);
```

نکته بعد آن است که معمولا درخت ها دارای فرزندهای متعددی می باشند به همین خاطر به فضای بیشتری برای نمایش کل درخت نیاز است. از طرفی چون فرم برنامه دارای فضای محدودی می باشد، نیاز است تا از خاصیت Scroll برای درخت بهره ببریم. برای انجام این کار، عبارت زیر را به برنامه اضافه می کنیم.

```
    ...
    JScrollPane treeView = new JScrollPane(tree);
    ...
```

در مرحله بعد باید برنامه را بگونه ای تنظیم کنیم که زمانی که کاربر روی گره های درخت کلیک کرده و یکی از فرزندان موجود را انتخاب کند، برنامه بتواند به آن پاسخ مناسبی بدهد.

برای این کار کافیست که از یک tree selection listener استفاده کرده و آن را به درخت پیوند دهیم ، کد های زیر نحوه انجام اینکار را نشان می دهند.

```
tree.getSelectionModel().setSelectionMode(
TreeSelectionModel.SINGLE_TREE_SELECTION );
     //Listen for when the selection changes.
tree.addTreeSelectionListener(this);
    ...
```

```
    public void valueChanged(TreeSelectionEvent e) {

        //اخرين مسير عنصر انتخابي را برميگرداند
//اين متد هنگاميكه مدل اجازه انتخاب تنها يک گره را داشته باشد كمک خواهد كرد

DefaultMutableTreeNode node =
(DefaultMutableTreeNode)tree.getLastSelectedPathComponent();
      if (node == null)
     //هيچ گره اي انتخاب نشده است
         return;
       Object nodeInfo = node.getUserObject();
     if (node.isLeaf())// اگر گره انتخابي برگ ميباشد {
         BookInfo book = (BookInfo)nodeInfo;
         displayURL(book.bookURL); // نمايش صفحه اينترنت
     } else {
         displayURL(helpURL);
     }
}
```

نکات مهم در تکه کد فوق عبارتند از:

«۱» TreeSelectionModel.SINGLE_TREE_SELECTION در این حالت فقط انتخاب یک گره در هر زمان ممکن است.

«۲» addTreeSelectionListener به کمک این عبارت برنامه به گونه ای تنظیم می شود که به رویدادهای مربوط به درخت « مانند رویداد حاصل از انتخاب یک گره از درخت » گوش فرا دهد.

«۳» getLastSelectedPathComponent برنامه با فراخوانی متد فوق متوجه می شود که کدام گره از درخت انتخاب شده است.

«۴» getUserObject استفاده از این متد سبب بدست آوردن اطلاعات موجود در گره مورد نظر می شود.

توجه ۱۰ : کد اجرایی برنامه در محتوای بسته و در پوشه ی sample Jtree\Jtree وجود دارد

۷-۱۷- ۲ چگونه می توان به گره های یک درخت کنترل Tooltip اضافه نمود؟

در این مبحث میخواهیم متدهایی و خصوصیاتی را معرفی کنیم که یک درخت ایجادکرده و با رفتن موس روی هر گره توضیحات مربوط به ان ظاهر شود.

شکل (۷- 79) : خروجی برنامه Tree Tooltip

کد این برنامه به شرح زیر می باشد.

```
/*کتابخانه هاي استفاده شده در کد*/
import java.awt.BorderLayout;
import java.awt.Component;
import java.util.Dictionary;
import java.util.Properties;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTree;
import javax.swing.ToolTipManager;
import javax.swing.tree.DefaultMutableTreeNode;
import javax.swing.tree.DefaultTreeCellRenderer;
import javax.swing.tree.TreeCellRenderer;
```

کد اجرایی: (شکل ۷-۷۹ خروجی کد زیر میباشد)

```
public class TreeTips {
   public static void main(String args[]) {
      JFrame frame = new JFrame("Tree Tips");
      Properties props = System.getProperties();
      JTree tree = new JTree(props);
      ToolTipManager.sharedInstance().registerComponent(tree);
      TreeCellRenderer renderer = new ToolTipTreeCellRenderer(props);
      tree.setCellRenderer(renderer);
      JScrollPane scrollPane = new JScrollPane(tree);
      frame.getContentPane().add(scrollPane, BorderLayout.CENTER);
      frame.setSize(300, 150);
      frame.setVisible(true);
   }
}

/*_____________________________________________________________________*/
class ToolTipTreeCellRenderer implements TreeCellRenderer {
```

```
        DefaultTreeCellRenderer renderer = new DefaultTreeCellRenderer();
        Dictionary tipTable;

public ToolTipTreeCellRenderer(Dictionary tipTable) {
              this.tipTable = tipTable;
        }
public Component getTreeCellRendererComponent(JTree tree, Object value,
boolean selected, boolean expanded, boolean leaf, int row,boolean hasFocus) {
renderer.getTreeCellRendererComponent(tree, value, selected, expanded,leaf,
row, hasFocus);
         if (value != null) {
                   Object tipKey;
         if (value instanceof DefaultMutableTreeNode) {
                    tipKey = ((DefaultMutableTreeNode) value).getUserObject();
           } else {
tipKey = tree.convertValueToText(value, selected, expanded,leaf, row,
hasFocus);
           }
           Object tip = tipTable.get(tipKey);
           if (tip != null) {
                    renderer.setToolTipText(tip.toString());
           } else {
                    renderer.setToolTipText(null);
           }
        }
        return renderer;
    }
}
```

برنامه فوق با وجود اینکه از لحاظ خطوط کد کوچک می باشد ولی حاوی نکات مهمی می باشد که باید آنها را درک نمایید. این نکات عبارتند از:

۱- همانطور که می بینید برنامه فوق از دو کلاس تشکیل شده است. کلاس اصلی برنامه با نام TreeTips و کلاس ToolTipTreeCellRenderer.

۲- داده هایی که در این برنامه برای تشکیل درخت و گره های آن مورد استفاده قرار گرفته است، ویژگی ها یا مشخصات محیط اجرای برنامه جاوا می باشند. برای بدست آوردن این اطلاعات و تحویل آن به متد سازنده کلاس JTree، از متد getProperties کلاس System موجود در بسته java.lang استفاده شده است. از جمله این مشخصات می توان به موارد زیر اشاره نمود:

Key	Meaning(توضیحات)
"file.separator"	File separator (e.g., "/")
"java.class.path"	Java classpath

```
"java.class.version"     Java class version number
"java.home"              Java installation directory
"java.vendor"            Java vendor-specific string
"java.vendor.url"        Java vendor URL
"java.version"           Java version number
"line.separator"         Line separator
"os.arch"                Operating system architecture
"os.name"                Operating system name
"path.separator"         Path separator (e.g., ":")
"user.dir"               User's current working directory
"user.home"              User home directory
"user.name"              User account name
```

همانطور که در لیست بالا مشاهده می کنید، این مشخصات به صورت Key/Value می باشد. یعنی به ازای هر کلید یک مقدار باز گردانده می شود.

اگر در متد فوق کلید خاصی را به عنوان آرگومان ورودی ارسال کنید، فقط مقدار معادل آن باز گردانده می شود ولی اگر متد فوق را بدون آرگومان استفاده نمایید، کلیه مشخصات پیش فرض را به خروجی باز می گرداند. به عبارت زیر دقت کنید.

```
System.getProperty("path.separator");
```

۳- نکته سوم در این کد، استفاده از یکی از متدهای کلاس ToolTipManager با نام registerComponent می باشد.
وظیفه کلاس فوق مدیریت رفتارهای کامپوننت Tooltip مورد استفاده در سایر کامپوننت های جاوا می باشد. این کلاس بر نحوه نمایش متن راهنمای Tooltip زمانی که اشاره گر ماوس روی item مورد نظر می رود و پنهان کردن آن، زمانی که ماوس از روی محدوده Item مورد نظر خارج می شود و همچنین مدت زمان نمایش متن tooltip و... نظارت دارد. این کلاس برای انجام وظایف مطرح شده از کلاس MouseAdapter و واسط MouseMotionListener استفاده می کند.

وظیفه متد registerComponent در این کلاس آن است که سایر کنترل های دیگر جاوا مانند JTree را که قصد اضافه کردن Tooltip به خود را دارند، جهت مدیریت Tooltip شان، به این کلاس معرفی می کند.

۴- در برنامه فوق از واسط TreeCellRenderer استفاده شده است. هرگاه بخواهیم در ظاهر یک درخت تغییراتی ایجاد کنیم، مثلا رنگ یا نوع یا قلم فونت، آیکون مورد استفاده و... را تغییر دهیم از این واسط یا واسط پیش فرض، یعنی DefaultTreeCellRenderer استفاده می کنیم. در این برنامه قصد داریم مقدار متد setToolTipText را برای هر گره تنظیم کنیم. به همین خاطر نیاز است تا از واسط های فوق استفاده نماییم.

به عنوان نمونه در کد زیر نحوه تغییر آیکون گره های مختلف یک درخت نمایش داده شده است.

```
DefaultTreeCellRenderer renderer = new DefaultTreeCellRenderer();
Icon customIcon = null;
...
renderer.setLeafIcon(customIcon );
renderer.setClosedIcon(customIcon );


renderer.setOpenIcon(customIcon );
setCellRenderer(renderer);
```

توجه ۳ : کد اجرایی برنامه در محتوای بسته و در پوشه ی sample Jtree\TreeTips وجود دارد

۷-۱۸ کلاس Jtable

JTable یکی از زیر کلاس های swing است که به کاربر اجازه می دهد داده های مرتبط به یکدیگر را بصورت مرتب در جدول نشان دهد.(شکل ۷-۸۰)

این کلاس در حقیقت محلی برای نگهداری از داده ها نیست، بلکه صرفا جهت نمایش منظم داده مورد استفاده قرار می گیرد. همچنین این کلاس در صورت تمایل کاربر، قابلیت ویرایش هر یک از سلول های جدول را به کاربر می دهد.از مهمترین کاربردهای این کلاس، قابلیت آن در نشان دادن و ویرایش کردن داده های جداولی است که توسط پایگاه داده ی مرتبط با برنامه، ساخته شده اند.

شکل (۷- 80) : اجزای JTable

۷-۱۸-۱ چگونگی ساختن یک Jtable

اولین گام در ساختن یک جدول، ایجاد عناوین هر یک از ستونهای آن می‌باشد. به عنوان مثال عناوین جدول موجود در برنامه را بصورت زیر ایجاد می شوند.

سپس اطلاعاتی که قرار است در جدول مشاهده کنیم را در آرایه‌ای ۲ بعدی از اشیا می آوریم. با این کار محتویات هریک از سلول‌های جدول راست می‌کنیم.

```
Object[][] data = {
    {"Mary", "Campione",
     "Click for combo box", new Integer(5), new Boolean(false)},
    {"Alison", "Huml",
     "Click for combo box", new Integer(3), new Boolean(true)},
    {"Kathy", "Walrath",
     "Click for combo box", new Integer(2), new Boolean(false)},
    {"Sharon", "Zakhour",
     "Click for combo box", new Integer(20), new Boolean(true)},
    {"Philip", "Milne",
     "Click for combo box", new Integer(10), new Boolean(false)}
 };
```

پس از انجام تعریفات بالا، این دو آرایه را به صورت زیر در متد سازنده جدول قرار می دهیم.

```
JTable table = new JTable(data,columnNames);
```

JTable دو نوع متد سازنده دارد که بطور مستقیم داده ها را مورد پذیرش و استفاده قرار می دهند.

```
1- JTable(Object[][] rowData, Object[] columnNames)
3- JTable(Vector rowData, Vector columnNames)
```

مزیت استفاده از متد های فوق آن است که استفاده از آنها سبب پیاده سازی ساده تر و راحتتر جداول می شود.

به عنوان مثال:

۱- تمام سلول های یک جدول که با متدهای فوق ایجاد می گردند، قابل ویرایش می باشند.

۲- یک جدول در این حالت با تمام انواع داده ای به یک صورت

(به صورت نوع رشته String) رفتار می کند.

۳- دو متد فوق نیاز دارند که تمام داده های ورودی توسط یه آرایه یا یک Vector به جدول داده شوند.

توجه ۴ : اگر شما نمی خواهید از دو متد فوق با محدودیت های خاص آنها استفاده کنید، باید مدل جدول مورد نظر خود را پیاده سازی نمایید . در ادامه به این مدل خواهیم پرداخت.

۷-۱۸-۲ اضافه کردن خاصیت Scroll به Table

برای اضافه کردن خاصیت Scroll به جدول از تابع سازنده کلاس JScrollPane استفاده می کنیم و سپس آن را به container برنامه اضافه می کنیم.

```
JScrollPane scrollPane = new JScrollPane(table);
table.setFillsViewportHeight(true);
add(scrollPane);
```

خط دوم در تکه برنامه فوق جهت تنظیم کردن مقدار خاصیت fillsViewportHeight در جدول به کار می رود. زمانی که مقدار این ویژگی true باشد، جدول از تمام ارتفاع کانتینر Scroll استفاده خواهد کرد. (حتی زمانی که تعداد سطرهای جدول کمتر از ارتفاع کانتینر باشد.)

استفاده از Scroll pane سبب می شود تا Header جدول بصورت ثابت قرار بگیرد، یعنی زمانی که عمل اسکرول کردن سطرهای جدول را انجام می دهیم اولین سط یا همان عنوان ستون ها ثابت با قی می ماند.

اگر از Scroll pane استفاده نکنیم برای تنظیم جدول به صورت قبل از تکه کد زیر استفاده می نماییم:

```
container.setLayout(new BorderLayout());
container.add(table.getTableHeader(), BorderLayout.PAGE_START);
container.add(table, BorderLayout.CENTER);
```

۷-۱۸-۳ تنظیم کردن عرض ستون های جدول

بطور پیش فرض، تمام ستون های جدول دارای سایز یکسان هستند و ستون های جدول تمام پهنای جدول را پر می کنند. هنگامیکه کاربر سایز فرم را تغییر می دهد (و در نتیجه‌ی آن اندازه جدول نیز عریض تر یا باریکتر می شود) عرض ستون های آن نیز بطور خودکار بر اساس اندازه جدید جدول، تنظیم می شود.

توسط متد setPreferredWidth می توان عرض هریک از ستون های جدول را بطور دلخواه تنظیم نمود. بطور مثال اگر تکه برنامه زیر را به برنامه قبل اضافه نماییم، عرض سومین ستون آن بیشتر از سایر ستون های جدول خواهد شد.

```
TableColumn column = null;
   for (int i = 0; i < 5; i++) {
        column = table.getColumnModel().getColumn(i);
        if (i == 2) {
            column.setPreferredWidth(100); //third column is bigger
        }
        else {
            column.setPreferredWidth(50);
        }
  }
```

همانطور که در کدهای بالا نشان داده شده است ،با استفاده از یک شی column از کلاس TableColumn و توسط متد getColumnModel که تمام اطلاعات مربوط به ستون های جدول رابر می گرداند، می توان سایز ستون های جدول را تغییر داد.

۷-۱۸-۴ تغییر رنگ سرستون جدول

با استفاده از کلاس JTableHeader می توان رنگ Header (سرستون) جدول را تغییرداد.(شکل ۷-۸۱)

```
JTableHeader header = table.getTableHeader();
header.setBackground(Color.ORANGE);
```

همچنین می توان بجای استفاده از دو دستور بالا ، از دستو زیر بصورت یکجا در برنامه استفاده کرد.

```
table.getTableHeader().setBackground(Color.ORANGE);
```

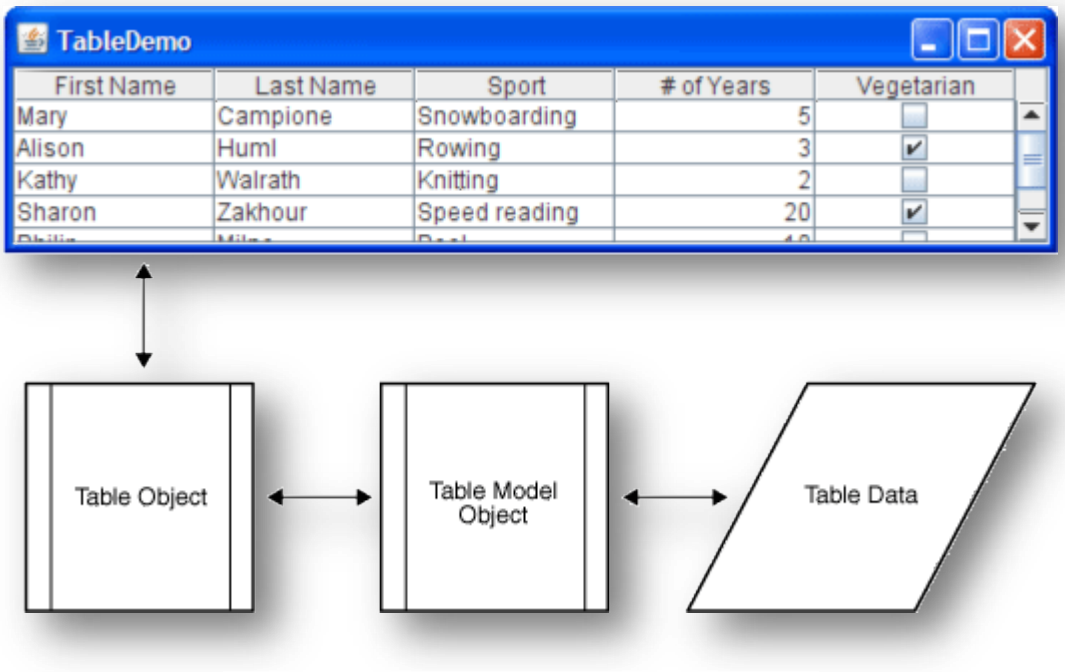
شکل (۷- ۸۱) : سرستون جدول

۷-۱۸-۵ ساختن مدلی از جدول

در جاوا هر جدول، از یک شی table model برای مدیریت داده های خود استفاده می کند.

شی table model باید از واسط TableModel استفاده کند. اگر کاربر برای جدول خود یک شی table model ایجاد نکرده باشد، JTable به صورت اتوماتیک یک نمونه از DefaultTableModel ایجاد می نماید.

این روابط در شکل ۷-۸۲ نشان داده شده است.



شکل (۷- ۸۲) : ساختار TableModel

برای استفاده از شی Table Model کافیست ان را تعریف نموده و ان را به جدول نسبت دهیم.

به تکه کد زیر دقت نمایید.

```
A. Object columns[]={“code Member”,”Chracter Member”,”Tel”,”Address”};
B. int current_row=0;
C. DefaultTableModel model= new DefaultTableModel();
D. model.setColumnIdentifiers(columns);
E. String str_value[]= {100,”AliReza”,”00985714463369”,”iran-sabzevar”};
F. model.insertRow(current_row,obj);
G. table.setModel(model);
```

در خط A متغیر columns ارایه ای رشته ای میباشد که سر تیتر جدول را مشخص میکند و در خط D این متغیر به مدل نسبت داده شده است.

در خط E متغیر str_value ارایه ای از مقادیری یک سطر جدول را در خود نگه داری میکند. در خط F میتوان مقداری از نوع ارایه رابه یک سطر مدل اضافه نمود.نکته حائض اهمیت اینجا میباشد که باید در تابع insertRow شماره سطر (از صفر شروع میشود) و مقدار درجی در جدول را وارد نمود.و در نهایت در خط اخر این مدل ایجاد شده به جدول اضافه میشود.

۷–۱۸–۶ چگونگی ارتباط با پایگاه داده و نشان دادن Query های درخواستی در Table

قطعه کد زیر نحوه ی ارتباط یک برنامه با پایگاه داده طراحی شده با Access را نشان می دهد.

هنگام استفاده از کلاس DefaultTableModel برای قرار دادن سرستون ها (Header) و محتویات جدول، می توانیم از عبارت

```
final DefaultTableModel model = new DefaultTableModel(null,columnName);
```

استفاده کنیم

```
try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
          link = DriverManager.getConnection("jdbc:odbc:Identify","","");
          statement = link.createStatement();
String Query="select field_Long,field_String,field_Double,
                                   Field_Int from MyTb";
           results = statement.executeQuery(Query);
        }
        catch(ClassNotFoundException cnfEx) {
                        System.out.println("* Unable to load driver! *");
                              System.exit(1);
        }        catch(SQLException sqlEx) {
System.out.println("* SQL error! *");
                              System.exit(1);
        }
```

با توجه به تعریف **نوع** هریک از صفات جدول پایگاه داده (**ستون ها**)، متد مربوط به آن از پایگاه فراخوانی می شود. یعنی مثلاً ستون آخر در پایگاه داده بصورت Integer تعریف شده است، باید از متد **getInt** استفاده کرد.

نکته مهم اینجاست که **در جدول پایگاه داده ستون ها از اندیس۱ شروع می شوند** و آرگمان هریک از این متد ها با توجه به این موضوع مقدار می گیرند.

```
    try {
              while (results.next()) {
                    model.insertRow(table.getRowCount(),new Object[]
                              {
                                 results.getLong(1),
                                 results.getString("field_String"),
                                 results.getDouble("field_Double"),
                                 results.getInt(4),
                               });
                         }}
               catch(SQLException sqlEx)
                    {     System.out.println("* SQL error! *");
                          System.exit(1); }
```

با توجه به کد های بالا در هنگام پر کردن جدول از یک حلقه استفاده می کنیم. این حلقه توسط متد next() بوسیله ی شیء تعریف شده(results) از واسط ResultSet ، با قرار دادن دستور results.next() درشرط حلقه، در پایگاه داده رکورد به رکورد (سطر به سطر) جلو می رود و در صورت وجود،نتایج را به جدول منتقل می کند.

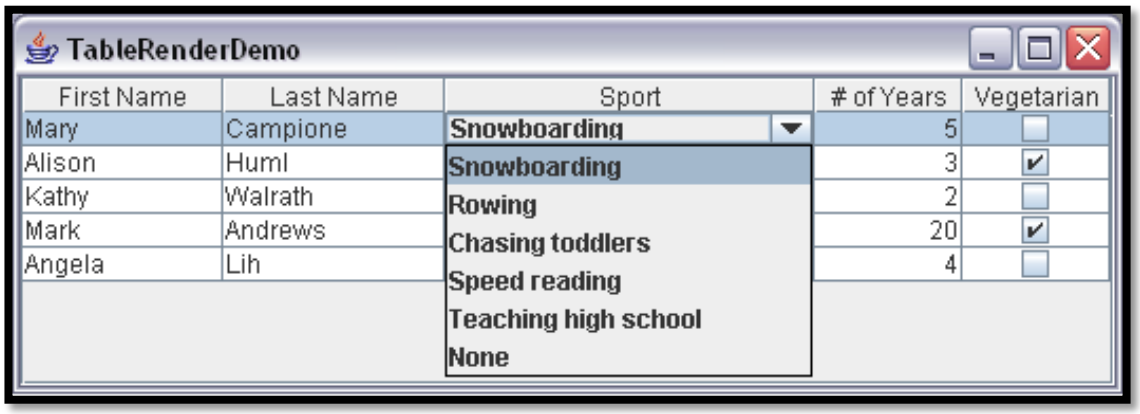
۷–۱۸–۷ قرار دادن یک comboBox در یکی از ستون های جدول به عنوان یک Editor

فرض کنید می خواهیم در سومین ستون جدول یعنی ستون sportColumn برای راحتی عمل ویرایش

از comboBox استفاده نماییم. برای انجام چنین کاری به صورت زیر عمل می کنیم.

```
TableColumn sportColumn = table.getColumnModel().getColumn(2);
...
JComboBox comboBox = new JComboBox();
comboBox.addItem("Snowboarding");
comboBox.addItem("Rowing");
comboBox.addItem("Chasing toddlers");
comboBox.addItem("Speed reading");
comboBox.addItem("Teaching high school");
comboBox.addItem("None");
sportColumn.setCellEditor(new DefaultCellEditor(comboBox));
```

با اجرای قطعه کد فوق خروجی شکل ۷-۸۳ حاصل میگردد.



شکل (۷- 83) : خروجی برنامه ComboBox در جدول

۷-۱۸-۸ چگونه می توان سطرهای یک جدول در جاوا را بر اساس نیاز حذف یا اضافه نمود؟

برای استفاده از امکان حذف و اضافه در سطرهای یک جدول، از کلاس DefaultTableModel استفاده می کنیم.

یعنی عملیات حذف و اضافه روی شی این کلاس انجام گرفته و این شی به شی کلاس JTable اضافه می شود.

```
final DefaultTableModel model = new DefaultTableModel(data,columnName);
final JTable table = new JTable(model);
```

کلاس DefaultTableModel زیر کلاسی از کلاس JTable است که می توانیم توسط آن بسیاری از عملیات مورد نیاز را انجام دهیم. از جمله متد هایی که توسط این کلاس پشتیبانی می شود ولی نمی توان از آنها به صورت مستقیم توسط کلاس JTable استفاده کرد، می توان به موارد زیر اشاره نمود:

```
1. Vector getDataVector()
2. void setDataVector(Object[][] newData, Object[] columnIDs)
3. void setDataVector(Vector newData, Vector columnIDs)
4. void addColumn(Object columnID)
5. void addColumn(Object columnID, Object[] columnData)
6. void addRow(Object[] rowData)
7. void insertRow(int row, Object[] rowData)
8. void removeRow(int index)
9. void setColumnIdentifiers(Object[] columnIDs)
```

توجه ۵ : می توان بجای استفاده از آرایه ای از اشیاء در آرگمان متد های شماره ۵ و ۶ و ۷ و ۹ ، آرایه ای ازنوع Vector قرار داد.

برنامه زیر برنامه ایست که دارای قابلیت حذف و اضافه کردن سطر به جدول است.

//عملیات درج

```
InsertRowbutton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
               //درج در اخر جدول
               model.insertRow(table.getRowCount(),new Object[]
               {
                textField1.getText(),
                textField2.getText(),
                textField3.getText(),
                textField4.getText()
               }
        );
```

قطعه کد فوق عملیات درج در جدول را به صورت یک عملیات به دگمه InsertRowbutton اضافه میکند

//عملیات حذف اخرین رکورد

```
RemoveRowbutton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
               //حذف اخرین رکورد
               if(model.getRowCount()!=0)
                  model.removeRow(model.getRowCount()-1);
            }
    });
```

قطعه کد فوق عملیات حذف اخرین رکورد از جدول را به صورت یک عملیات به دگمه RemoveRowbutton اضافه میکند

//حذف تمام رکودهای جدول

```
RemoveAllRowsbutton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
               //حذف تمام رکورد هاي جدول
               int numrows = model.getRowCount();
               for(int i = numrows - 1; i >=0; i--)
               {
                model.removeRow(i);
               }
            }
    });
```

قطعه کد فوق عملیات حذف تمام رکوردها از جدول را به صورت یک عملیات به دگمه

RemoveAllRowsbutton اضافه میکند

۷-۱۸-۹ اضافه کردن ستون جدید به جدول

به کد زیر دقت کنید :

```
InsertColumnbutton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
               //اضافه کردن ستون جدید
                model.addColumn("Sport",new Object[]
                {
                "فوتبال",
                "گلف",
                "پینگ پنگ"
                }
                );
            }    });
```

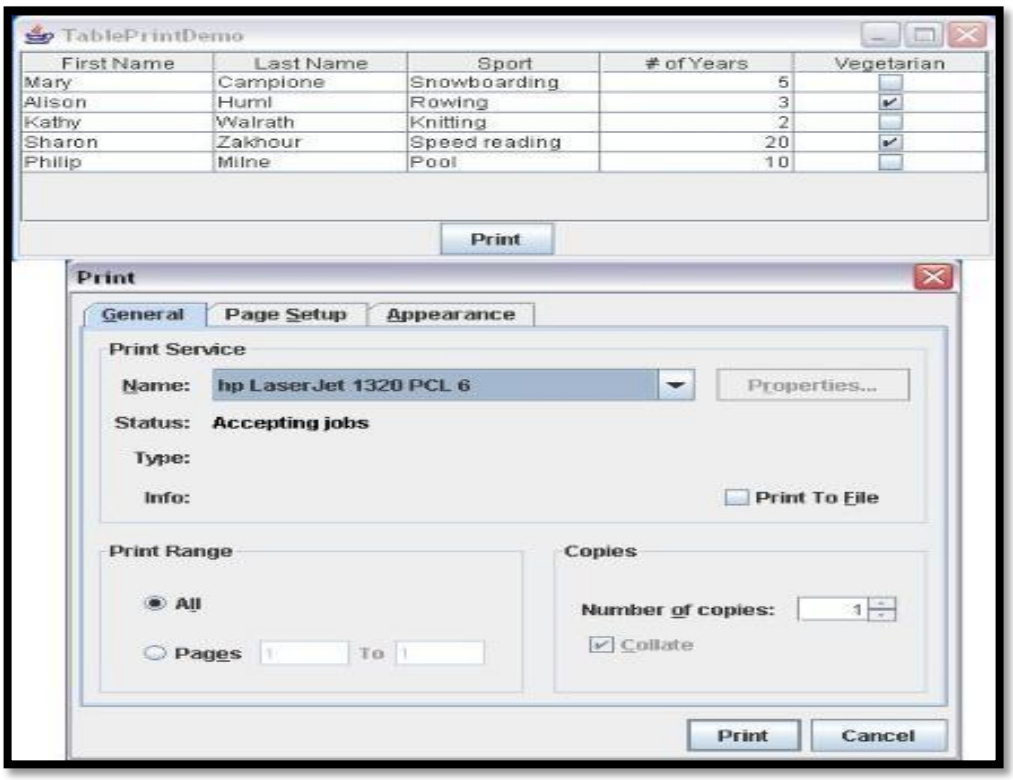
با اجرای کد فوق یک ستون به جدول، با نام Sport و مقادیر مربوط به نام ورزش های مورد علاقه افراد اضافه خواهد شد. پس از اضافه شدن ستون جدید، اگر کاربر بخواهد یک سطر به سطرهای موجود اضافه کند، چون textField ی برای این ستون پیش بینی نشده، مقدار آن خالی می ماند.

۷-۱۸-۱۰ چگونه می توان محتویات یک جدول در جاوا را چاپ نمود؟

برنامه نویسان و مخصوصا طراحان واسط های کاربری، همواره به دنبال یافتن راه حل های مختلفی جهت بهبود کارایی واسط های کاربری برنامه های خود می‌باشند. یکی از این موارد آن است که کاربر قادر باشد تا محتوای جداول موجود در برنامه را به دستگاه پرینتر ارسال نموده و نسخه ای از آن را چاپ نماید. برای انجام چنین کاری می توان ازمتد **JTable.print** بدون هیچ آرگومانی استفاده نمود.

تکه کد زیر نحوه استفاده از این متد را نمایش داده است.(شکل ۷-۸۴ خروجی تکه کد زیر میباشد)

```
try {
    if (! table.print()) {
        System.err.println("User cancelled printing");
    }
} catch (java.awt.print.PrinterException e) {
    System.err.format("چاپ نمیتواند صورت بگیرد", e.getMessage());
}
```



شکل (۷- ۸۴) : خروجی برنامه چاپ محتوای جدول

حال اگر قصد اضافه کردن یک عبارت جهت تعیین عنوان صفحه در زمان چاپ یا page header را داشته باشید، بصورت زیر عمل می نماییم.

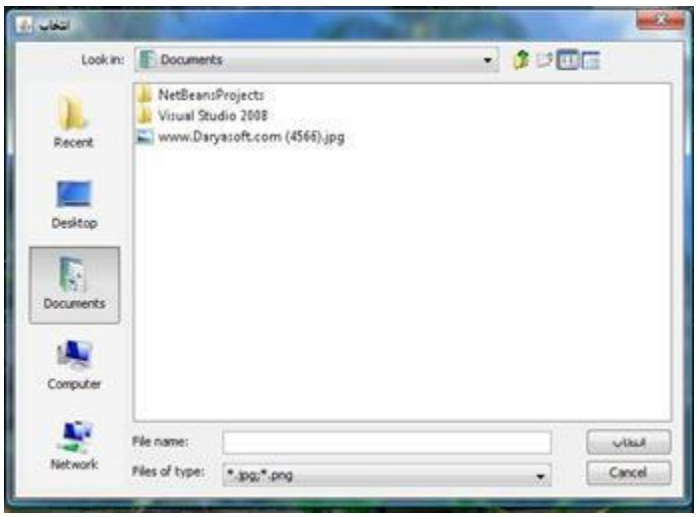
```
MessageFormat header = new MessageFormat("Page {0,number,integer}");
try {
    table.print(JTable.PrintMode.FIT_WIDTH, header, null);
} catch (java.awt.print.PrinterException e) {
    System.err.format("عملیات چاپ نمیتواند صورت بگیرد", e.getMessage());
}
```

۷-۱۹ کلاس JFileChooser

JFileChooser ها این اجازه را به کاربران میدهند تا بتوانند فایلی را ویرایش،لود و یا ذخیره کنند.

برای ایجاد یک شی JFileChooser به صورت زیر عمل میکنیم.

```
JFileChooser fileChooser = new JFileChooser();

//--------------------------- OpenDialog
fileChooser.showOpenDialog(this);
//--------------------------- SaveDialog
fileChooser.showSaveDialog(this);
```



شکل (۷- ۸۵) :خروجی برنامه JFileChooser

روش فوق نحوه ی ایجاد یک OpenDialog و SaveDialog است.

از جمله سازنده ها و متدهای مهم این کلاس میتوان به موارد زیر اشاره نمود.

سازنده	توضیحات

تنظیم مسیر جاری در نمایش دیالوگ	`fileChooser = new FileChooser(currentDirectory)`
مسیری به صورت متغیر رشته ای دریافت و مسیر جاری در نمایش را بر ان اساس تنظیم مینماید	`fileChooser = new JFileChooser( string currentDirectoryPath)`

۷–۱۹–۱ سازندگان مهم کلاس

۷–۱۹–۲ متدهای مهم

جدول (۷– ۲۲) : سازندگان کلاس JFileChooser

توضیحات	متدها
تمام فیلترهایی که در دیالوگ وجود دارد را حذف مینماید مثلا (*.jpg)	`fileChooser.removeChoosableFileFilter()`
توسط این متد میتوان فیلترهایی به دیالوگ اعمال نمود	`fileChooser. (FileFilter);`
با استفاده از این متد میتوان حالت انتخاب فایلها را تغییر نمود	`fileChooser.setFileSelectionMode(int);`
دیالوگی با عنوان و فریمی مشخص نمایش میدهد	`fileChooser.showDialog(Fram، tilte);`
پنجره OpenDialog را نمایش میدهد	`fileChooser.showOpenDialog(component);`
پنجره SaveDialog را نمایش میدهد	`fileChooser.showSaveDialog(component);`
میسر فایل انتخاب شده را برمیگرداند	`fileChooser.getSelectedFile()`

جدول (۷– ۲۳) : متدهای کلاس JFileChooser

۷–۱۹–۳ چگونه یک دیالوگ را فیلتر کنیم ؟

برای فیلترکردن دیالوگ،بایستی از متد addChoosableFileFilter و به صورت زیر عمل کنیم.

در این مثال سعی شده است که با ایجاد یک کلاس که از کلاس File Fileter ارث بری میکند، پنجره OpenDialog را برای گرفتن تصاویری را پسوندهای مشخص،فیلتر کنیم

```
public class clsFilterImg extends javax.swing.filechooser.FileFilter {

 public boolean accept(File file)
       { String filename = file.getName();
         if (file.isDirectory()) { return true; }
         return filename.endsWith(".jpg") || filename.endsWith(".png");
       }
       public String getDescription() { return "*.jpg;*.png"; } }
```

متد accept() برای فیلتر کردن فایل ها استفاده میگردد و متد getDescription() متنی که در جلو Files Of Type در دیالوگ وجود دارد را،تعیین میکند.

به کد زیر دقت کنید.

این قطعه کد با استفاده از کلاس فوق پنجره OpenDialog را نمایش میدهد.(شکل ۷-۸۵ خروجی تکه زیر میباشد)

```
Void show(){
JFileChooser fileChooser = new JFileChooser();
حذف تمام فیلترهای دیالوگ (برای حذف (ALL File
   fileChooser.removeChoosableFileFilter(fileChooser.getFileFilter());
   fileChooser.addChoosableFileFilter(new clsFilterImg());
فایل ها و پوشه ها نمایش داده شود
   fileChooser.setFileSelectionMode(JFileChooser.FILES_AND_DIRECTORIES);
int result = fileChooser.showDialog(new Frame(),"Selected Image");
اگر دگمه انصراف گردد از تابع خارج شود
if(result == JFileChooser.CANCEL_OPTION) return;

 else if (result == JFileChooser.APPROVE_OPTION){
                عملیات
 }
  }
```

فصل هشتم

کار با MYSQL در نت بینز

۸–۱ مقدمه

مای اس کیو ال (my Sql) یک سامانه ی مدیریت دادگان (DBMS) است که به دلیل سادگی نصب و مدیریت و باز متن (Open Source) بودن از محبوبیت زیادی برخودار است.SQL مخفف Structured Query Languale میباشد که از محبوبیت زبان کامپیوتری برای ایجاد ; تغییر و بازیابی و عملیات بر روی داده ها در مدل رابطه ای میباشد.

این زبان به سمت مدل شی گرا_رابطه ای نیز پیشرفت کرده است.

۸–۲ نصب mySql (5.0.67)

۱. برای نصب mysql ابتدا باید ان را از سایت زیر دانلود کرد.

http://dev.mysql.com

نکته ۱ : نسخه ویندوز ضروری x86 میباشد (۳۲ بیت)

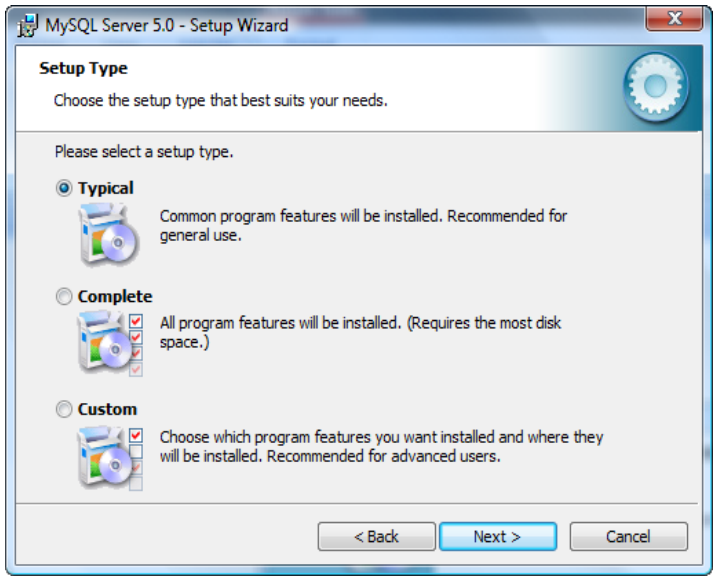
برنامه ای برای اجرا و مدیریت پایگاه داده SQL از طریق ایجاد یک سرور بر روی سیستم (Local Host)

۲. روی Setup.exe دوبار کلیک کرده تا برنامه شروع به نصب گردد.

۳.در پنجره خوش امد گویی برای نصب روی next کلیک کنید.(شکل ۸–۱)

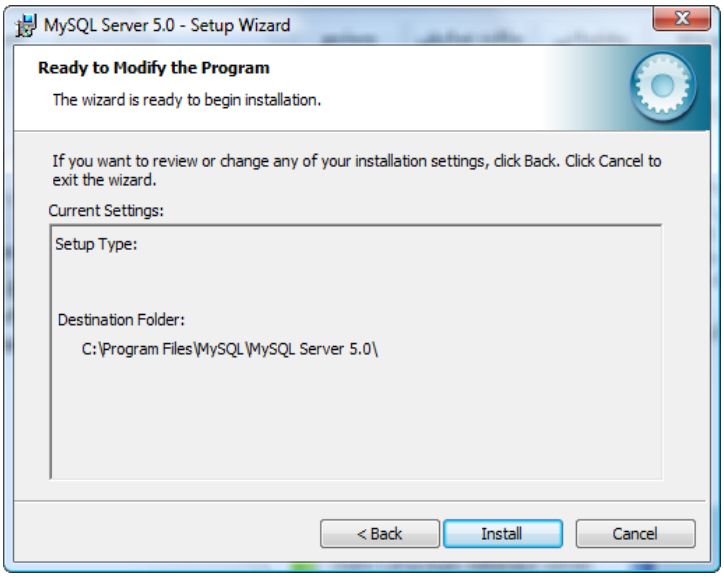
شکل (8- 1) : مرحله سوم نصب Mysql

۴.در نوع نصب گزینه ی typical را انتخاب کنید و روی دگمه next کلیک کنید.(شکل ۸-۲)



شکل (8- 2) : مرحله چهارم نصب Mysql

۵.در این مرحله پنجره اماده نصب نرم افزار ظاهر میشود که مسیر نصب نرم افزار نیز نمایش داده شده است.(شکل ۸-۳)



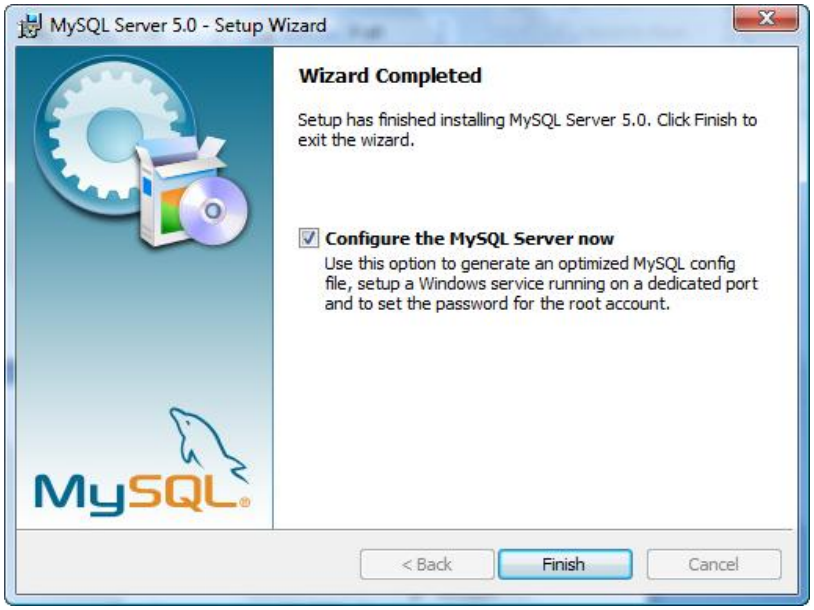
شکل (8- ۳) : مرحله پنجم نصب Mysql

برای نصب روی Install کلیک کنید

۶.همکنون موتور دیتابیس اس کیوال نصب گردیده است.

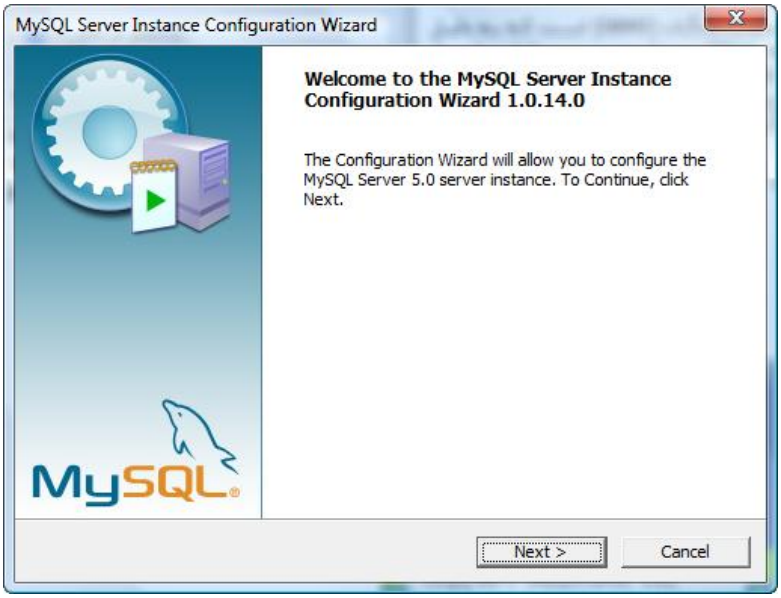
گزینه Configure the MySQL Server now

را انتخاب کرده و finish را زده تا پیکربندی مربوط به MySql صورت گیرد.(شکل ۸-۴)



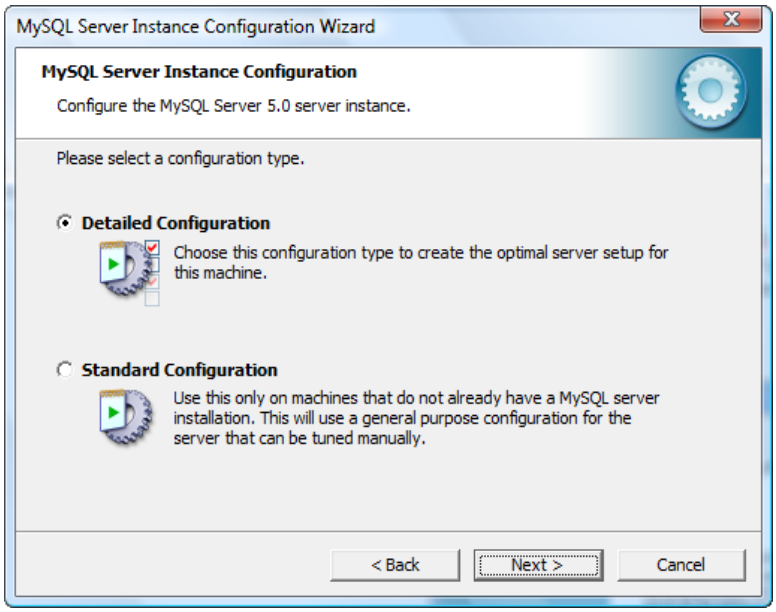
شکل (8- ۴) : مرحله ششم نصب Mysql

۷.در پنجره ظاهر شده ی MySQL Server Instance Confguration Wizard روی دگمه next کلیک کنید.(شکل ۸-۵)



شکل (8- ۵) : مرحله هفتم نصب Mysql

۸.در پنجره ظاهر شده نوع پیکربندی Sql ،گزینه Detailed Confguration را انتخاب کنید تا تمام جزئیات تنظیم نمایش داده شود.(شکل ۸-۶)



شکل (8- ۶) : مرحله هشتم نصب Mysql

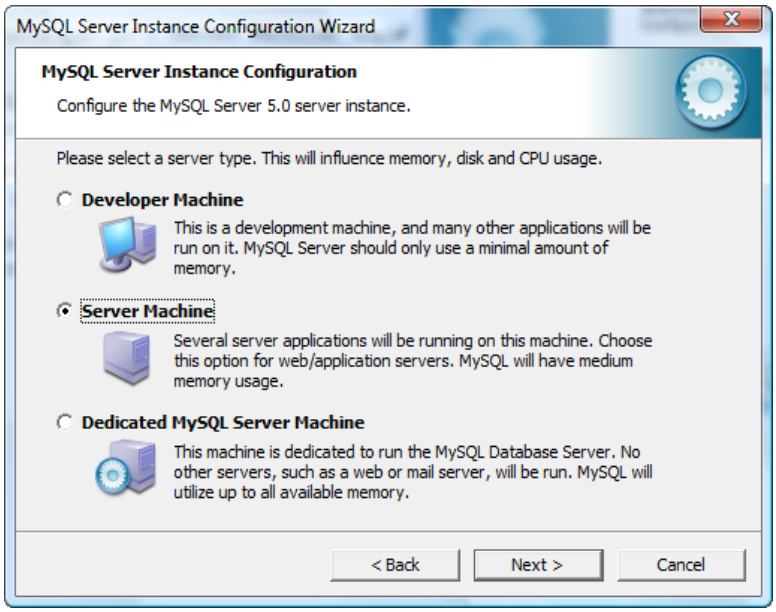
۹.در این پنجره نوع ماشینی که MySql در روی ان نصب میگردد را مشخص میکند.(شکل ۸-۷)

گزینه Developer Machine : این ماشین همانند یک Pc خانگی که اکثر برنامه کاربردی روی ان نصب شده اند عمل میکند و در این ماشین موتور اس کیو ال حداقل حاظفه را اشغال میکند.

گزینه Server Machine: در این ماشین چندین برنامه ی سرور در حال اجرا میباشد. گزینه ای برای انتخاب سرور های کاربردی (Application) و وب (Web) میباشد.در ای ماشین موتور مای اس کیو ال متوسط حافظه را اشغال میکند.

گزینه Dedicated MySql Server Machin: این ماشین مختص اجرای سرور دیتابیس MySql میباشد.هیچ سرور دیگری همچون Web یا Mail Server نباید اجرا باشد.

در این ماشین موتور مای اس کیوال همه حافظه را در دسترس دارد.گزینه ی Server Machine را انتخاب کرده و Next را کلیک کنید.



شکل (8- ۷) : مرحله نهم نصب Mysql

۱۰. در این پنجره نوع دیتابیس مشخص میگردد.(شکل ۸-۸)

گزینه MultiFunctional Database:

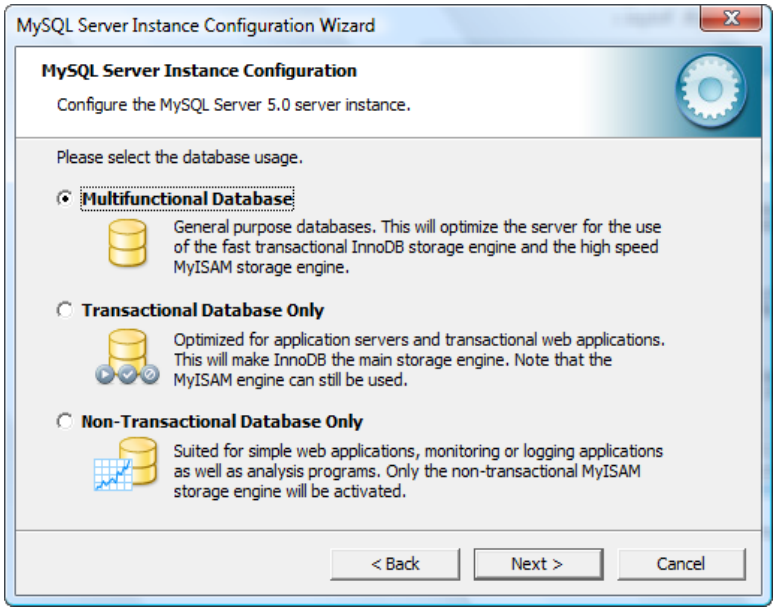
دیتابیس هایی با عملکرد و اهداف عمومی میباشد.این گزینه سرور را برای استفاده از موتور ذخیره سازی InnoDB و myISAM بهینه خواهد ساخت.

گزینه Transactional Database Only:

این گزینه برای سرورهای Application و Web بهینه شده است.این یک موتور ذخیره سازی اصلی InnoDB ایجاد خواهد کرد. توجه کنید که هنوز هم موتور myISAM میتواند استفاده شود.

گزینهNon-Transactional Database Only:این گزینه برای برنامه های Web وMonitoring و یا برنامه های واقعه نگاری (logging) همچون برنامه های تجزیه و تحلیل مناسب است.تنها موتور ذخیره سازی myISAM فعال خواهد بود.

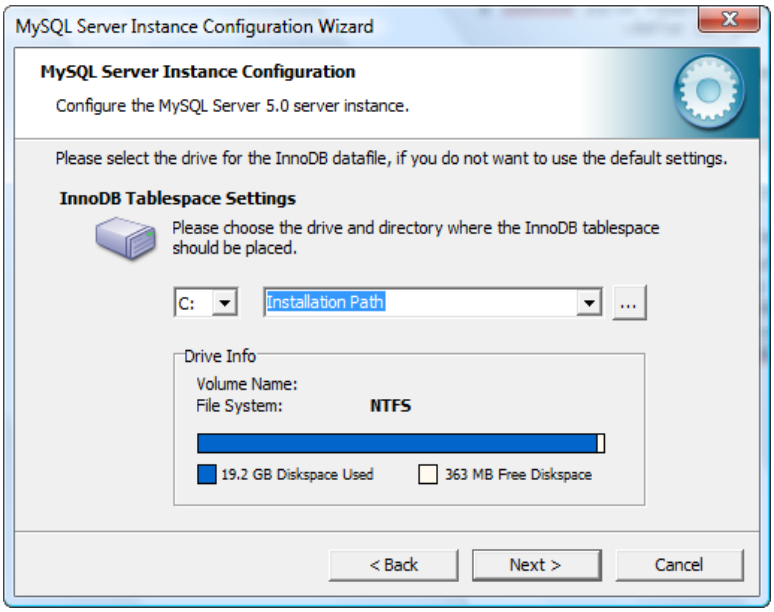
گزینه ی MultiFunction Database را انتخاب و با انتخاب next به مرحله بعد رفته.



شکل (8- ۸) : مرحله دهم نصب Mysql

۱۱.در این مرحله اگر نمیخواهید از تنظیمات پیشفرض استفاده کنید میتوانید مکان ذخیره سازی جدول های (InnoDb Tablespace) مای اس کیوال را تغییر دهید.

با تغییرات لازم گزینه Next را کلیک کنید.(شکل ۸-۹)



شکل (8- ۹) : مرحله یازدهم نصب Mysql

۱۲. در پنجره ظاهر شده تعداد تقریبی اتصالات (Connections) همزمان به سرور تعیین میشود.(شکل ۸-۱۰)

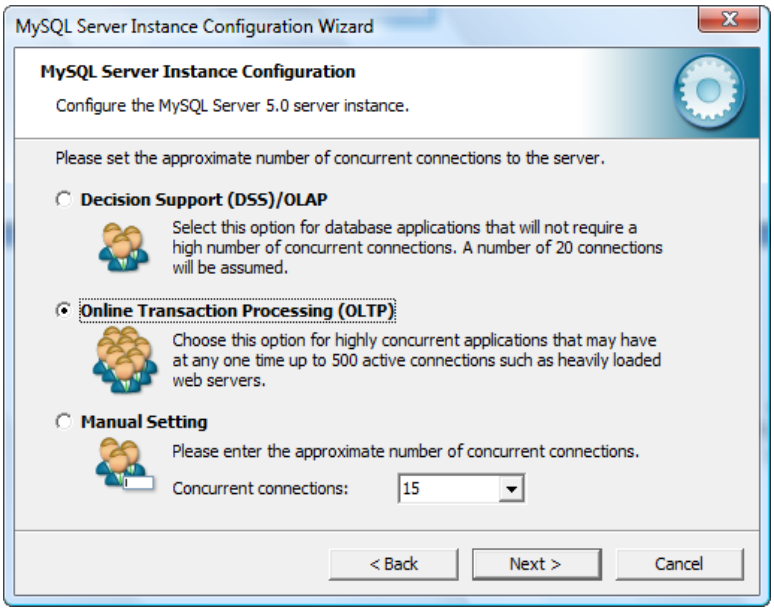
گزینه ی Decision Support(DDS)/OLAP:

انتخاب این گزینه برای دیتابیس برنامه کاربردهایی ست که به تعداد بالایی اتصال همزمان احیتاج نباشد.تعداد ۲۰ اتصال پیش بینی شود.

گزینه Online Transaction Processing (OLTP):انتخاب این گزینه برای برنامه های کاربردی است که دارای اتصال همزمان بالایی (بیش از ۵۰۰ اتصال) در هر لحظه باشد.مثلا Web Servers

گزینه Manual Setting: تعداد تقریبی اتصالات به صورت دستی وارد میشود.

گزینه Online Transaction Processing (OLTP) را انتخاب نموده و Next را بزنید

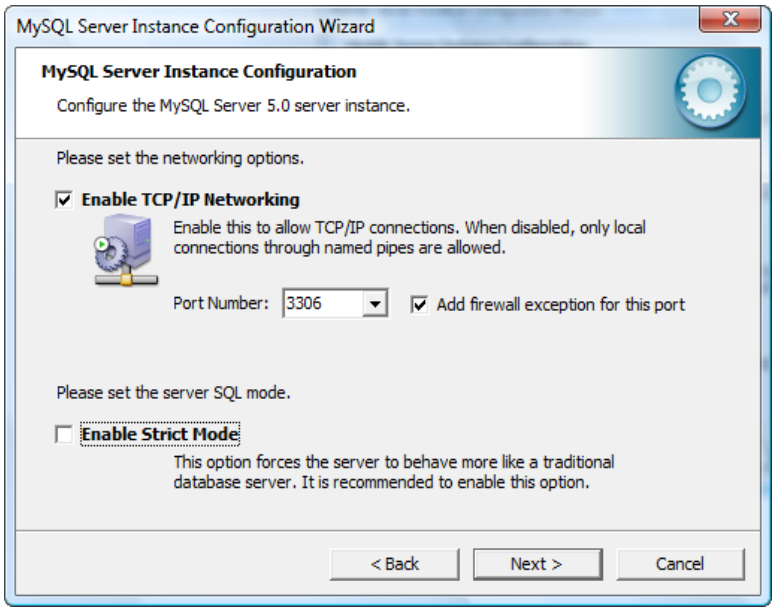


شکل (8- 10) : مرحله دوازدهم نصب Mysql

۱۳. در این صفحه تنظیمات مربوط به شبکه میباشد.

- با فعال سازی Tcp/Ip مربوط به شبکه این امکان وجود دارد تا کاربران دیگر به بانک دسترسی داشته باشند در غیر اینصورت بانک به صورت Local قابل دستیابی میباشد.
- گزینه Add firewall Exception for this port باعث میگردد فیروال اینپورت را به عنوان پورت های مجاز شبکه شناسایی و در برابر ان گارد نیگیرد.
- با فعال سازی گزینه Enable Strict Mode سرور بیشتر دارای رفتار دیتابیس سرورهای قدیمی را بخود میگیرد.فعال سازی این گزینه یک پیشنهاد میباشد.

تنظیمات شبکه را تعیین کرده و Next را کلیک کنید.



شکل (8- 11) : مرحله سیزدهم نصب Mysql

۱۴. در این صفحه تنظیمات مربوط به کراکتر های بانک میباشد.این قسمت یکی از قسمتهای مهم نصب میباشد که با انتخاب گزینه دوم Mysql کراکترهای UTF8 را پشتیبانی میکند.(شکل ۸-۱۲)

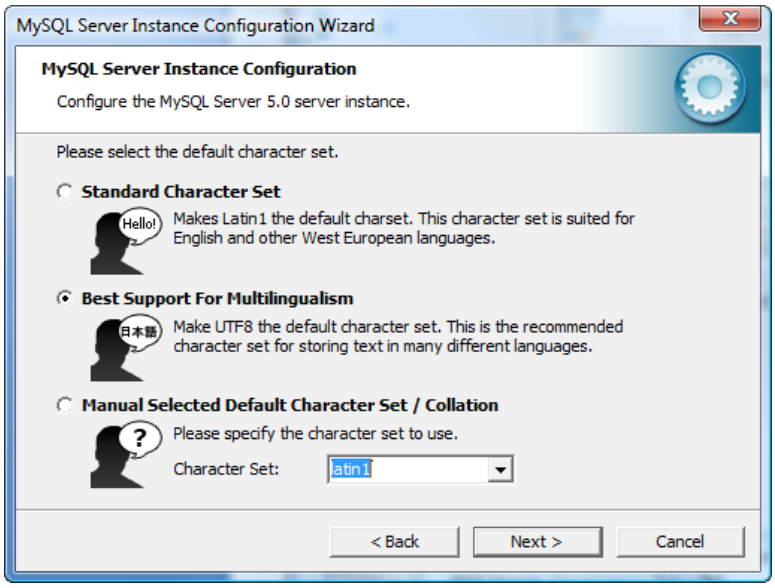
گزینه ۱ Standard Character Set:

این گزینه مجموعه ای از کراکترهای استاندارد را در برمیگیرد.به طور پیشفرض کراکتر Latin 1 میباشد.این کراکتر برای زبانهای انگلیسی و غربی مناسب میباشد.

گزینه ۲ Best Support For Multilingualism: این گزینه بهترین انتخاب برای زبانهای خاص میباشد. کراکتر پیشفرض ان UTF8 میباشد.این کراکتر برای ذخیره سازی متن در زبانهای مختلف پیشنهاد میشود.

گزینه ۳ Manual Select Default Character Set/Collation:

در این گزینه کاربر کراکترهای پیشفرض مای اس کیو ال را مشخص میکند.گزینه Best Support For Multilingualism را انتخاب کرده و بر روی next کلیک کنید.

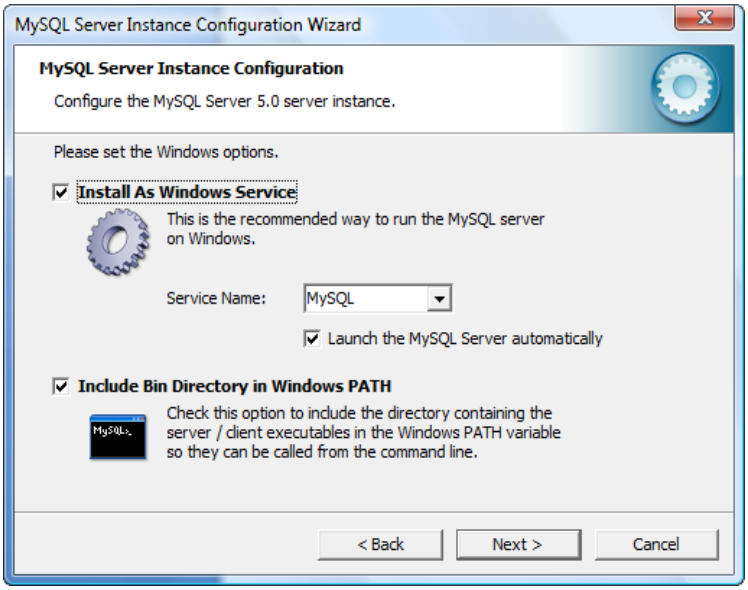


شکل (8- 12) : مرحله چهاردهم نصب Mysql

۱۵.در صفحه ظاهر شده تنظیمات مربوط به ویندوز میباشد.(شکل ۸-۱۳)

- با انتخاب گزینه Install As Windows Service مای اس کیو ال یک سرویس با نام دلخواه که میتوان در قسمت Service Name وارد کرد ایجاد کرده و روشی میباشد بتوان موتور اس کیوال را اجرا نمود.با انتخاب Launch The MySQL Server Automatically سرویس Mysql به طور خودکار راه اندازی میشود.

- اگر گزینه Include Bin Directory in Windows Path انتخاب شود محتویات پوشه Bin به صورت متغیرهای سیستم در ویندوز ایجاد شده و میتوان ان را از طریق Command line اجرا نمود.تنظیمات مربوط به ویندوز را انجام داده و بر روی next کلیک کنید.

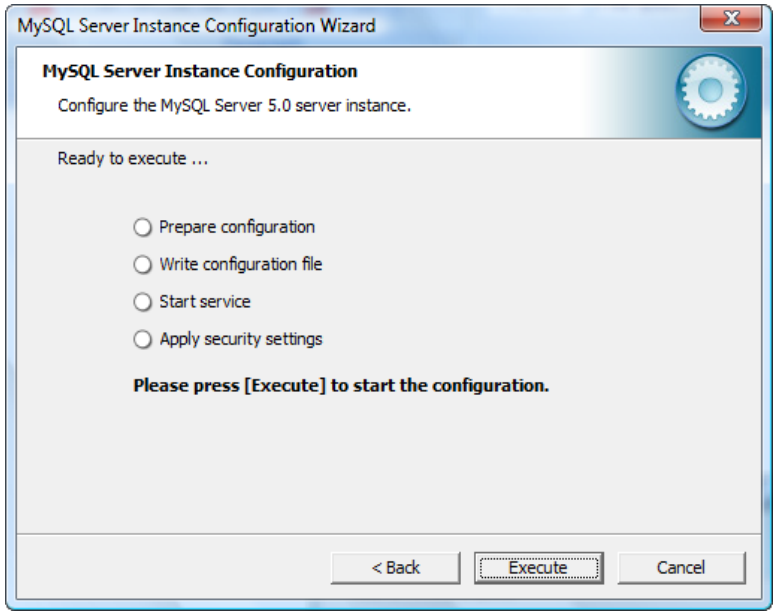


شکل (8- 13) : مرحله پانزدهم نصب Mysql

۱۶.در این مرحله گزینه های مربوط به امنیت Mysql تنظیم میشود.با انتخاب گزینه Modify Security Settings میتوان تنظیمات مربوط به امنیت همچون تغییر پسورد root و فعال سازی دستیابی یوزر root برای دستیابی به ماشینهای دور.با انتخاب گزینه Ctreate An Anonymouse Account یک یوزر بینام روی سرور ایجاد میگردد.معمولا این گزینه برای سیستم های ناامن مفید میباشد.تنظیمات مربوط به این صفحه را تنظیم و با انتخاب Next به صفحه بعدی بروید.(شکل ۸-۱۴)

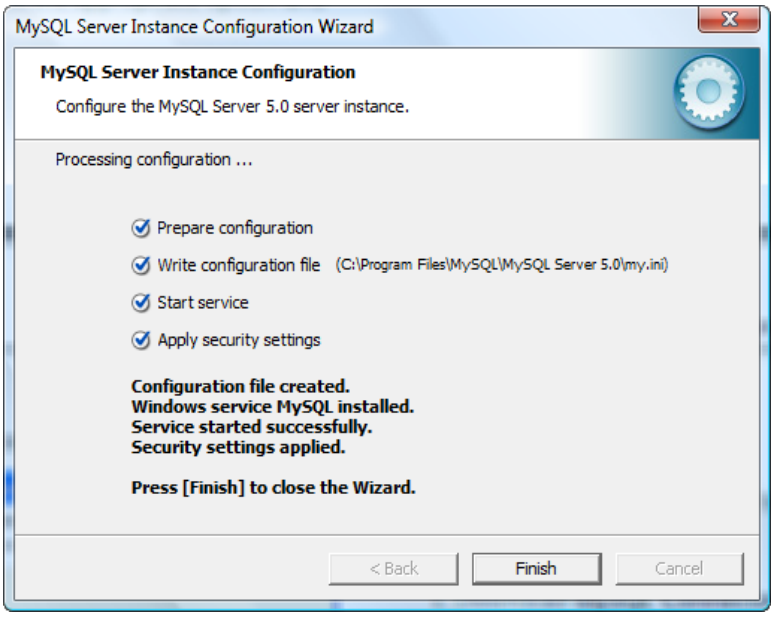
شکل (8- ۱۴) : مرحله شانزدهم نصب Mysql

۱۷.روی گزینه Excute کلیک کرده تا پیکربندی Mysql صورت گیرد.(شکل ۸-۱۵)



شکل (8- ۱۵) : مرحله هفدهم نصب Mysql

۱۸.در این مرحله پیکربندی موتور مای اس کیوال به پایان رسیده است و با انتخاب Finish پنجره بسته خواهد شد.(شکل ۸-۱۶)



شکل (8- ۱۶) : مرحله هجدهم نصب Mysql

۸–۲–۱ کار با MySql

برای کار با MySql میتوان از MySQL Command Line Client (مربوط به خود Mysql) یا نرم افزار هایی برای مدیریت Mysql استفاده نمود.

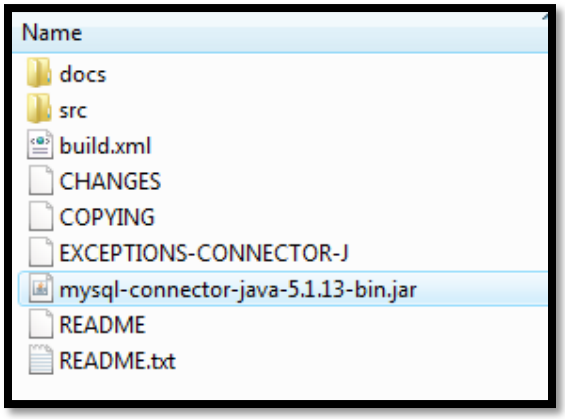
توجه ۱: نرم افزار مدیریت اس کیو ال که میتوان استفاده نمود

PREMIUMSOFT NAVICAT FOR MYSQL ENTERPRISE EDITION 8.1.15 میباشد.

۸–۳ نحوه اضافه نمودن کلاس Mysql به پروژه

قدم اول (نصب کتابخانه): ابتدا بایستی Connector/J (فایل ارتباطی بانک با پروژه) را با نام mysql-connector-java-5.1.13.zip را از سایت http://dev.mysql.com/downloads/connector/j/5.1.html دانلود نمود.

نام فایل دریافتی، بسته به نسخه ای که ان را دانلود مینمایید متفاوت میباشد.بعد از دانلود فایل ان را در محلی از حالت فشرده سازی خارج نمایید.(شکل ۸–۱۷)



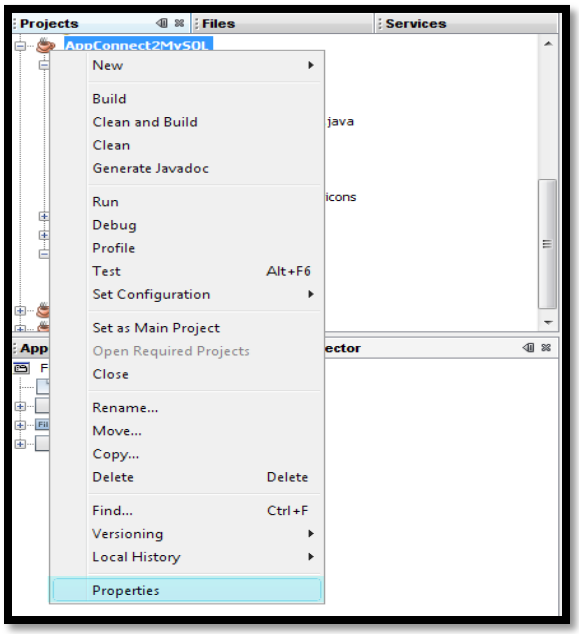
شکل (8– ۱۷) : قدم اول در اضافه نمودن کلاس mysql به پروژه

توجه ۲: فایل mysql-connector-java-5.1.13-bin.jar در پوشه، کتابخانه ای است که برای ارتبط لازم میباشد.تنها این فایل را در پوشه کتابخانه جاوا کپی کنید.

مثلا C:\Program Files\Java\jre1.6.0_04\lib

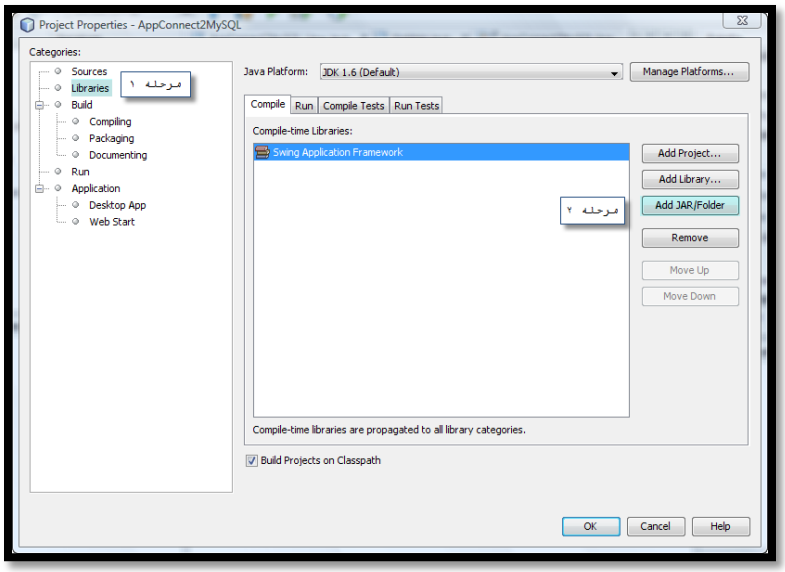
قدم دوم (اضافه کردن درایور JDBC به پروژه نت بین):

مرحله اول: ابتدا یک پروژه جدید ایجاد نموده،و دربرگه projects روی پروژه جدید راست کلیک کرده و گزینه ی Properties را انتخاب نمایید. (شکل ۸-۱۸)



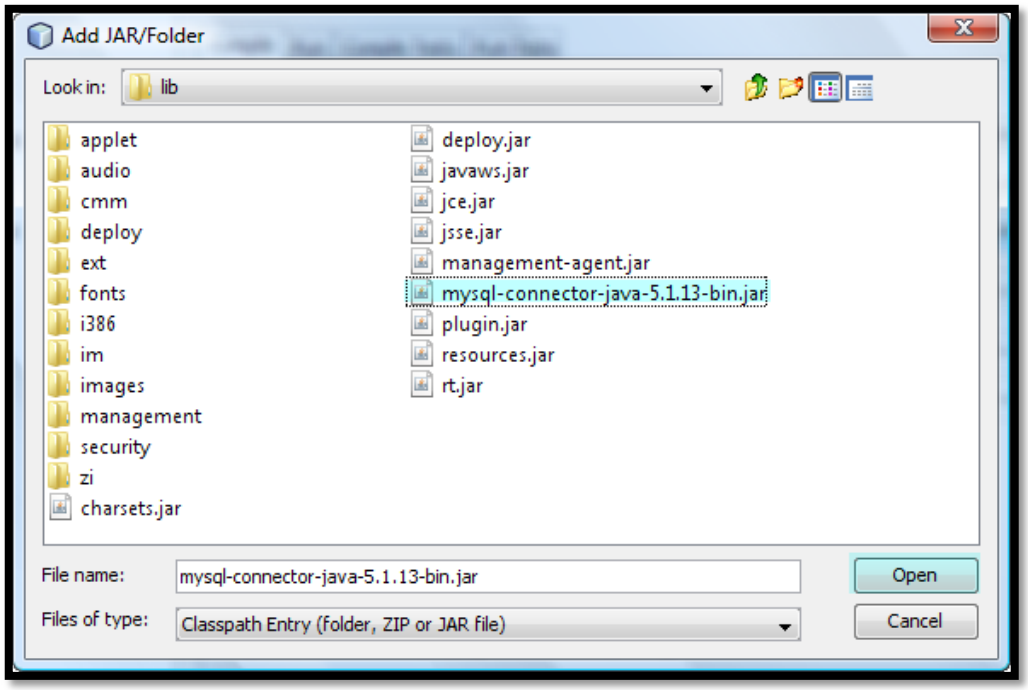
شکل (8- 18) : قدم دوم در اضافه نمودن کلاس mysql به پروژه

مرحله دوم: با ظاهر شده پنجره Project Properties،از سمت چپ رده Libraries را انتخاب نموده و از سمت راست در برگه Compile روی دگمه Add Jar/Folder کلیک کنید. (شکل ۸-19)



شکل (8- 19) : قدم سوم در اضافه نمودن کلاس mysql به پروژه

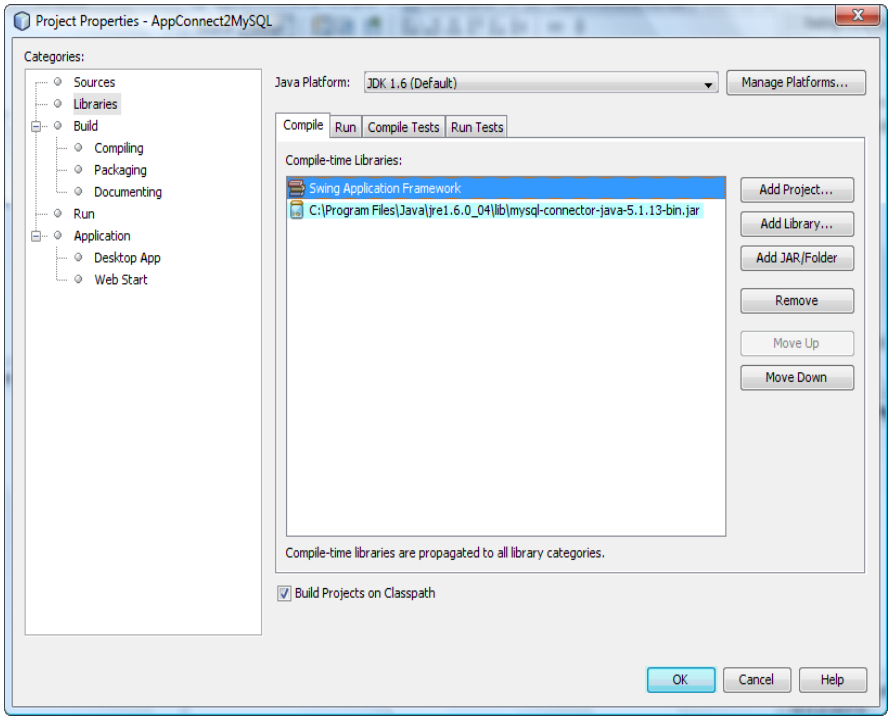
مرحله سوم: با پنجره Browser ظاهر شده فایل کتابخانه را انتخاب،و روی دگمه Open کلیک کنید.(شکل ۸-20)



شکل (8- 20) : قدم چهارم در اضافه نمودن کلاس mysql به پروژه

مرحله چهارم: در این مرحله همان طور که مشاهده میکنید این فایل به کتابخانه پروژه اضافه و از ادرس C:\Program Files\Java\jre1.6.0_04\lib فراخوانی میشود.(شکل ۸-21)

اگر فایل را بدون حذف لینک ان در پروژه حذف کنید پروژه با خطا مواجهه میشود.



شکل (8– 21) : قدم پنجم در اضافه نمودن کلاس mysql به پروژه

توجه ۳ : فایل **mysql-connector-java-5.1.13-bin.jar** را در پوشه پروژه نگه داری کنید.

۸–۴ استفاده از دستورات Sql در پروژه

به طور معمول میتوان دستوراتی که در پروژه استفاده میشوند را به دو گروه تقسیم نمود.

۱. دستوراتی که هیچ نتیجه برنیمگردانند

به عنوان مثال میتوان به موارد زیر اشاره نمود

```
Insert into mytable  (field1,field2,…) values (value1,value2)
Update myTable set field1=value1,field2=value2,... where ...
Delete myTable where ...
Exec myProcedure
...
```

۲. دستوراتی که یک چند سطر (چندتایی) را به عنوان خروجی برمیگردانند

به عنوان مثال میتوان به عبارات زیر اشاره نمود.

```
Select field1,field2,... from myTable where ...
Select myFunction(parametr1,parametr2,...)
...
```

برای استفاده از از این دستورات ،ابتدا بایستی یک ارتباط (Connection) بین پروژه و mySql وجود داشته باشد.

برای ایجاد ارتباط از کلاس Connection که یکی از زیر کلاسهای java.sql میباشد استفاده نمود.

به کد زیر توجه کنید.

```
Class.forName("com.mysql.jdbc.Driver");
    String connectionUrl = "jdbc:mysql://localhost:3306/db_lib?" +
                                "user=root&password=100";
     Connection cnn = DriverManager.getConnection(connectionUrl);
```

در خط اول، Class.forName("com.mysql.jdbc.Driver") باعث لودشدن درایورهای mysql میگردد.

رشته اتصال به بانک را اینطور مشخص میگردد

```
jdbc:mysql://localhost:شماره پورت/نام دیتابیس?user=نام کاربری&password=پسورد
```

دراینجا شماره پورت،همان شماره پورت TCP/IP میباشد که در زمان نصب MYSQL تنظیم میشود.(رجوع شود به نصب mysql)

ونیز نام کاربری (root)و پسورد،مقادیری هستند که در زمان نصب mysql تنظیم میشوند.

بعد از ایجاد یک ارتباط به mysql نوبت به کلاسی خواهد رسید تا بتواند این دستورات را اجرا نماید.برای اینکار، از کلاس Statement که یکی از زیر کلاسهای java.sql میباشد استفاده میکنیم.

به مثال زیر توجه کنید.

```
Class.forName("com.mysql.jdbc.Driver");
    String connectionUrl = "jdbc:mysql://localhost:3306/db_lib?" +
                                "user=root&password=100";
     Connection cnn = DriverManager.getConnection(connectionUrl);
  Statement stmt = cnn.createStatement();
...
```

دستوری که با رنگ زرد مشخص شده است یک متغیر از نوع کلاس Statement ایجاد کرده و ان را برای اجرای دستورات sql اماده میسازد.

این کلاس دارای دو متد مهم به شرح زیر میباشد

```
stmt.executeUpdate(Query);
      از متد براي استفاده از گروه اول دستورات mysql استفاده ميگردد
                  (دستوراتي كه هيچگونه نتيجه را برنميگردانند)

ResultSet result = stmt.executeQuery(Query);
      از متد براي استفاده از گروه دوم دستورات mysql استفاده ميگردد
                  (دستوراتي كه هميشه نتيجه اي را برميگردانند)
```

در متد دوم برای انکه نتیجه ی دستورات (مجموعه از رکوردها) را در جایی قرارداده شود از کلاس ResultSet

استفاده میگردد.

این کلاس دارای متدهایی مهمی می باشد و لیستی از ان در زیر اورده شده است.

متد	توضیحات
`result.next()`	**اگر نتیجه وجود داشته باشد به رکود بعدی پرش خواهد کرد**
`result.first()`	**اشاره گر result به اولین رکورد پرش میکند**
`result.last()`	**به اخرین رکورد پرش میکند**
`result.close()`	**اشاره result بسته و به هیچ رکودی اشاره نمیکند**

`result.getByte(condition)`
`result.getInt(condition)`
…
	مقداری فیلد مشخص شده در ورودی تابع را به صورت Byte یا Int و... دریافت میکند.
	(شماره یا نام فیلد condition)

جدول (8- 1) : متدهای مهم کلاس ResultSet

به مثال زیر دقت کنید

```
Void ExecuteNonQuery(){
Class.forName("com.mysql.jdbc.Driver");
    String connectionUrl = "jdbc:mysql://localhost:3306/db_lib?" +
                                    "user=root&password=100";
     Connection cnn = DriverManager.getConnection(connectionUrl);
String Query="insert into mytb values(1,'ali','alirezae')";

  Statement stmt = cnn.createStatement();
stmt.executeUpdate(Query);
cnn.close();
 }
```

تابع ExecuteNonQuery با فرض اینکه جدول **mytb** دارای سه فیلد شماره دانشجویی،نام و نام خانوادگی

می باشد،رکوردی را در جدول ثبت میکند.

حال برای نمایش اطلاعات جدول،بایستی به صورت زیر عمل کنیم.

```
Void ExecuteQuery(){
Class.forName("com.mysql.jdbc.Driver");
    String connectionUrl = "jdbc:mysql://localhost:3306/db_lib?" +
                                    "user=root&password=100";
     Connection cnn = DriverManager.getConnection(connectionUrl);
String Query="Select name,family from mytb Where id=1";

 Statement stmt = cnn.createStatement();
ResultSet rs=stmt.executeQuery(Query);
  while(rs.next()){
System.out.println("name = "+rs.getString("name")+" , family =
"+rs.getString("family"));
       }
cnn.close();
 rs.close();
 }
```

خروجی برنامه

```
name = ali    ,  family  = alirezae
```

۸–۵ ذخیره کردن تصویر در MySql

برای ذخیره کردن تصویر در باید بایستی نوع فیلد جدول،Blob باشد.

فرض کنید جدول mytb دارای دو فلید با مشخصات زیر باشد.

نوع	فیلد
Int - اندازه تصویر	Size
Blob	Img

جدول (8– 2) : فیلدهای جدول تصویر

کد ذخیره اطلاعات در جدول به صورت زیر میباشد.

```
Class.forName("com.mysql.jdbc.Driver");
String connectionUrl = "jdbc:mysql://localhost:3306/db_lib?" +
"user=root&password=100";
Connection cnn = DriverManager.getConnection(connectionUrl);
ایجاد پارامتر برای ورود اطلاعات
```

```
PreparedStatement pr = cnn.prepareStatement("insert into mytb (size,img)
values (?,?)");

FileInputStream in = new FileInputStream("ادرس تصویر");
int len = in.available();

pr.setString(1, len.toString());
pr.setBinaryStream(2,in,len);

pr.executeUpdate();
```

۸–۶ لود کردن تصویر از MySql

برای لودکردن تصویر از دیتابیس با فرض انکه جدول mytb دارای شمای قبلی میباشد

کد ان را به صورت زیر مینویسیم.

```
Class.forName("com.mysql.jdbc.Driver");
String connectionUrl = "jdbc:mysql://localhost:3306/db_lib?" +
"user=root&password=100";
Connection cnn = DriverManager.getConnection(connectionUrl);
Statement stmt = cnn.createStatement();
ResultSet rs = stmt.executeQuery("select size,img from mytb");

byte [] b;
InputStream in = null;
int size;
while(rs.next()){
   size = rs.getInt("size");
   b = new byte[size];
   in = rst.getBinaryStream("img");
   }
in.close();
in.read(b);
Image img = Toolkit.getDefaultToolkit().createImage(b);
 ImageIcon imgIcn = new ImageIcon(img);
ایجاد یک برچسب برای نمایش تصویر
  JLabel ImgShow = new JLabel(imgIcn);
```

فصل نهم

ابزار گزارش گیری JasperReports

۹–۱ مقدمه ای بر JasperReports

شکل (۹– ۱) : Teodor Danciu

در سال ۲۰۰۱ فردی با نام Teodor Danciu (شکل ۹-۱)روی پروژه ای که نیازمند ابزارهای گزارش گیری بود، شروع به فعالیت کرد و نتیجه این تحقیقات سبب معرفی ابزاری با نام JasperReports گردید.

در آن زمان بروز مشکل بالا بودن هزینه توسعه پروژه نرم افزاری، سبب متوقف شدن آن گردید.

با این وجود او کار روی JasperReports را در اوقات فراغتش آغاز کرد.

سرانجام او پروژه ی خود را در سپتامبر ۲۰۰۱ در سایت http://sourceforge.net ثبت کرد.

مدتی بعد، (با وجود اینکه او هنوز هیچ کدی از محصول خود را منتشر نکرده بود) email هایی را از کاربران علاقه مند به JasperReports دریافت نمود.

نسخه ی ۰.۱.۵ JasperReports در نوامبر ۲۰۰۱ منتشر شد. از آن پس JasperReports محبوب شد و هنوز هم یکی از محبوب ترین ابزارهای گزارشگیری جاوا میباشد.

سپس در April سال ۲۰۰۵ شرکت JasperSoft (توسعه دهنده ابزارهای مانند iReport Visual Designer) تصمیم به پشتیبانی همه جانبه از این ابزار گرفت و با بستن یک قرارداد همکاری با Teodor Danciu از آن تاریخ به بعد توسعه JasperReports را از طرف شرکت فوق صورت می پذیرد.

شرکت مذکور در ادامه حدود ۸ میلیون دلار روی JasperReports سرمایه گذاری نمود. با توجه به آمارهای ارائه شده توسط شرکت JasperSoft در هر ماه حدود ۲۰۰۰۰ بار ابزار JasperReports دانلود می شود.

۹–۱–۱ JasperReports چیست؟

JasperReports یک ابزار قوی گزارش گیری می باشد که توسط زبان جاوا پیاده سازی شده است و در واقع متشکل از تعدادی کتابخانه جاوا و بصورت کد باز یا open source می باشند.

هدف از تولید این ابزار آن است که توسعه دهندگان جاوا قادر به اضافه کردن قابلیت های گزارش گیری در برنامه های خود گردند.

به دلیل‌اینکه JasperReports یک ابزار مستقل نیست، نمی‌تواند به تنهایی نصب شود، ولی با معرفی مسیر قرارگیری کتابخانه JasperReports در CLASSPATH می توان از آن در برنامه های خود استفاده نمود.

در آغاز کار JasperReports برای اضافه کردن قابلیت های گزارش گیری به برنامه های مبتنی بر وب (ایجاد شده با تکنولوژی API Servlet) استفاده می شود، ولی هیچگونه وابستگی به Servlet API یا هر کتابخانه EE Java ندارد و به هیچ وجه به برنامه‌های وب محدود نمی‌شود.

JasperReports چیزی نیست جز یک کتابخانه جاوا که دارای یک رابط برنامه نویسی کاربر برای آسان کردن توانایی تولید گزارشها از هر نوع در Java Application ها می باشد.

JasperReports نیاز به JDK ۱.۳ یا نسخه ی جدیدتر آن دارد تا بتواند برنامه هایی که شامل کلاس های JasperReports هستند را بطور موفقیت آمیز کامپایل کند و همچنین نیاز به Java Runtime Environment 1.3 یا نسخه ی جدید تر آن دارد تا بتواند این برنامه ها را به طور موفقیت آمیز اجرا نماید.

نسخه های مختلف JasperReports

(2009-07-03)	**JasperReports 3.5.2**
(2009-05-04)	JasperReports 3.5.1
(2009-03-26)	JasperReports 3.5.0
(2009-02-10)	JasperReports 3.1.4
(2009-01-12)	JasperReports 3.1.3
(2008-11-04)	JasperReports 3.1.2
(2008-09-17)	JasperReports 3.1.0
(2008-08-08)	JasperReports 3.0.1
(2008-05-19)	JasperReports 3.0.0
(2008-03-12)	JasperReports 2.0.5

(2008-01-11)	JasperReports 2.0.4
(2007-12-11)	JasperReports 2.0.3

جدول (۹- ۱) : نسخه های مختلف JasperReports

شما می توانید نسخه های جدیدتر این محصول را از سایت sourceforge.net دانلود نماید.

۹-۲ ویژگی های JasperReports

علاوه بر داده های متنی، JasperReports قادر است تا گزارش های حرفه ای که شامل تصاویر، نمودارها و گراف ها باشند را تولید نماید. بعضی از ویژگی های اصلی این محصول عبارتند از:

- محیط صفحه آرایی در این محصول دارای انعطاف پذیری زیادی می باشد.
- قادر است داده ها را به صورت متنی یا گرافیکی نمایش دهد.
- می تواند داده ها را از چندین منبع داده بپذیرد. مثلاً می توان یک متن ثابت را تنظیم کرد تا در گزارش نشان داده شود.

پس در این صورت از یک empty datasource استفاده شده است. یا می توان از یک منبع داده مثل database استفاده کرد و اطلاعات ذخیره شده در پایگاه داده را به صورت یک گزارش نشان داد و...

- قادر است تا گزارش ها را به شکل های مختلف نمایش دهد. مثلا می تواند از تکنیک Watermarks استفاده نماید و یا قادر به تولید زیر گزارش (subReports) می باشد.

گزارش های تولیدی توسط JasperReports را می توان در قالب های مختلفی ارائه نمود. از جمله:

- **PDF** (Portable Document Format)
- **XLS** (Excel)
- **RTF** (Rich Text Format)
- **HTML** (HyperText Markup Language)
- **XML** (Extensible Markup Language)

- **CSV** (Comma-Separated Values)
- **plain text** (متن ساده)

در ادامه تصویری از خروجی یک گزارش (شکل ۹-۲) که توسط JasperReports ایجاد شده وبه صورت یک فایل PDF می باشد را مشاهد می نمایید.

شکل (۹- 2) : خروجی گزارش JasperReports

۹-۲-۱ انعطاف پذیری صفحه بندی گزارش

همانطور که گفته شد، JasperReports دارای یک محیط صفحه آرایی انعطاف پذیر می باشد.

ابزار فوق به ما اجازه می دهد تا داده های مختلف خود را در بخش ها جداگانه محیط قرار دهیم.

بخش های مختلف محیط گزارش گیری JasperReports عبارتند از:

«۱» عنوان گزارش که فقط یکبار و در ابتدای هر گزارش قرار می گیرد.

«۲» بخش سر صفحه (Header Page) که در بالای هر صفحه از گزارش قرار می گیرد.

«۳» بخش جزئیات (Sections Detail) که شامل داده های مورد نظر ما جهت نمایش بوده و بین دو بخش ۲ و ۳ قرار می گیرد.

«۴» بخش پایین صفحه (Footer Page) که در پایین هر صفحه از گزارش قرار می گیرد.

«۵» بخش خلاصه وضعیت (Sections Summery) که در انتهای هر گزارش قرار می گیرد.

توجه ۱ : JasperReports بجز پنج بخش فوق، شامل بخش های دیگری نیز می باشد که در قسمتهای بعدی به آنها خواهیم پرداخت.

ویژگی دیگر وجود امکان گروه بندی داده های بر اساس یک فیلد یا مجموعه ای از فیلدها می باشد. این ویژگی یکی از پر کاربردترین ویژگیهای JasperReports میباشد که به کمک آن می توان گزارش های متنوعی را ایجاد نمود.

در حین گروه بندی داده ها وجود امکان subTotal سبب می شود تا در پایان هر گروه و در صورت نیاز بتوان یک جمع بندی از دادهای گروه فوق بدست آورد. مثلا می خواهیم طی گزارشی بفهمیم فروشنده های یک فروشگاه (به تفکیک هر فروشنده) چه کالاهایی را فروخته و مجموع درآمد حاصل از این فروش چقدر است. ایجاد چنین گزارشی به کمک دو ویژگی فوق امکان پذیر است.

روش های مختلف برای چاپ داده ها (Multiple Ways to Present Data)

داده‌های مورد نیاز یک گزارش از طرق مختلفی قابل دسترس و ارسال به گزارش می‌باشند. به عنوان نمونه می‌توان داده‌های مورد نیاز را از یک بانک اطلاعات یا از یک فایل متنی دریافت نمود.

راه دیگر ارسال داده های بصورت دستی و به کمک عبارات متنی می باشد. اما روش دیگری نیز وجود دارد و آن بدست آوردن داده های مورد نیاز از طریق انجام محاسبات بر روی فیلدهای عددی یا ترکیب مقادیر فیلدهای متنی و... می باشد.

روش های مختلف برای دریافت اطلاعات از منبع(Multiple Ways to Supply Data)

توسعه دهندگان نرم افزار، برای ارسال داده های مورد نیاز به گزارش می توانند از پارامترهای موجود در محیط ایجاد گزارش استفاده نمایند. این پارامتر ها در حقیقت نمونه‌هایی از کلاس‌های جاوا می‌باشند که بر اساس نوع

داده مورد استفاده قرار می‌گیرند. یکی از انواع ویژه کلاس ها برای ارسال داده ها به گزارش، datasource ها می باشند. از جمله datasource های معروف می توان به موارد زیر اشاره نمود:

```
1- XML files
2- Plain Old Java Objects (POJOs)
3- Any class implementing the java.util.Map interface
4- And any class implementing the javax.swing.TableModel interface
5- JDBC datasources
6- custom datasources
```

custom datasources: اگر قرار باشد از یک منبع داده ای که توسط JasperReports پشتیبانی نمی شود استفاده کنیم، باید یک datasources ایجاد نموده و از آن جهت انتقال داده ها به JasperReports استفاده نماییم.

۹–۲–۲ Watermarks (پشت زمینه گزارش)

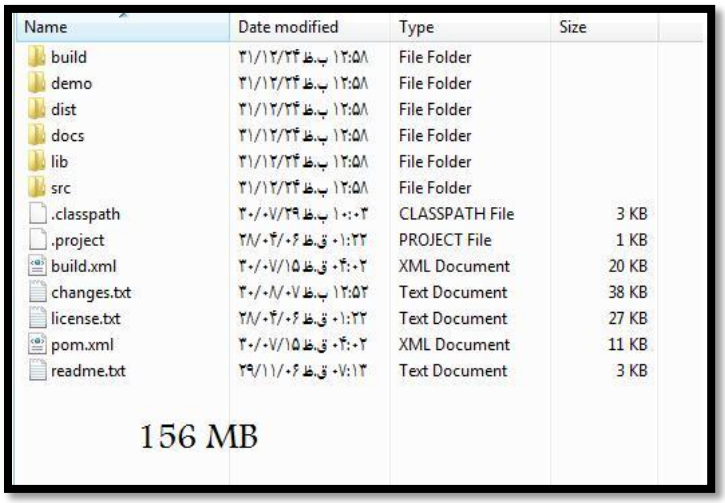
یکی از ویژگی های JasperReports استفاده از امکان Watermarks می باشد. به کمک این ویژگی در گزارش می توان به عنوان نمونه از یک تصویر در زمینه کار سود جست. مزایای انجام چنین کاری عبارتند از: اول آنکه ممکن است شرکتی تمایل به قرار دادن آرم یا لوگوی شرکت خود در زمینه گزارشات خود باشد. دومین کار آن است اگر تصویری در زمینه گزارش قرار بگیرد، به راحتی نمی توان از روی گزارش فوق جعل نمود و یک گزارش جعلی ایجاد کرد.

۹–۳ چگونه ابزار JasperReports را در NetBeans نصب کنیم؟

عمل نصب و اضافه کردن امکانات گزارش گیری ابزار JasperReports به برنامه های کاربردی جاوا، بسیار ساده و راحت می باشد.

شما می تواند قابلیت های این ابزار را به برنامه های خود چه در سمت سرور و چه در سمت کلاینت به سادگی اضافه نمایید. برای انجام این کار، کافیست مراحل زیر را به ترتیب انجام دهید.

ابتدا به آدرس **http://www.sourceforge.net/projects/jasperreports/files** مراجعه کرده و روی لینک مربوط به جدیدترین نسخه این ابزار کلیک نمایید. در این مقاله فرض بر آن است که ابزار jasperreports-3.5.3 جدیدترین نسخه می باشد. از بین لینک های موجود روی لینک jasperreports-3.5.3-project.zip کلیک کنید. بعد از دانلود فایل آن را در یک پوشه به انتخاب خودتان Extract نمایید. اکنون پوشه jasperreports-3.5.3 که شامل فایل ها و دایرکتوریهای زیر است را مشاهده می نمایید.(شکل ۹-۳)



شکل (۹- ۳) : فایل و دایرکتوریهای JasperReports

محتویات پوشه های موجود در این مسیر عبارتند از:

build: این دایرکتوری شامل فایل های کامپایل شده کلاس JasperReports است.

demo: این دایرکتوری شامل مثال های متنوعی در باره ویژگی های مختلف کارکرد JasperReports می باشد.

Dist: این دایرکتوری حاوی یک فایل JAR می باشد که این فایل شامل کتابخانه JasperReports است. این فایل JAR باید به CLASSPATH اضافه شود تا بتوان از مزایای JasperReports استفاده کرد.

Docs: این دایرکتوری شامل یک کپی محلی از وب سایت JasperReports است.

lib : این دایرکتوری شامل همه کتابخانه های مورد نیاز برای ایجاد JasperReports و استفاده از آن در برنامه های جاوا می باشد.

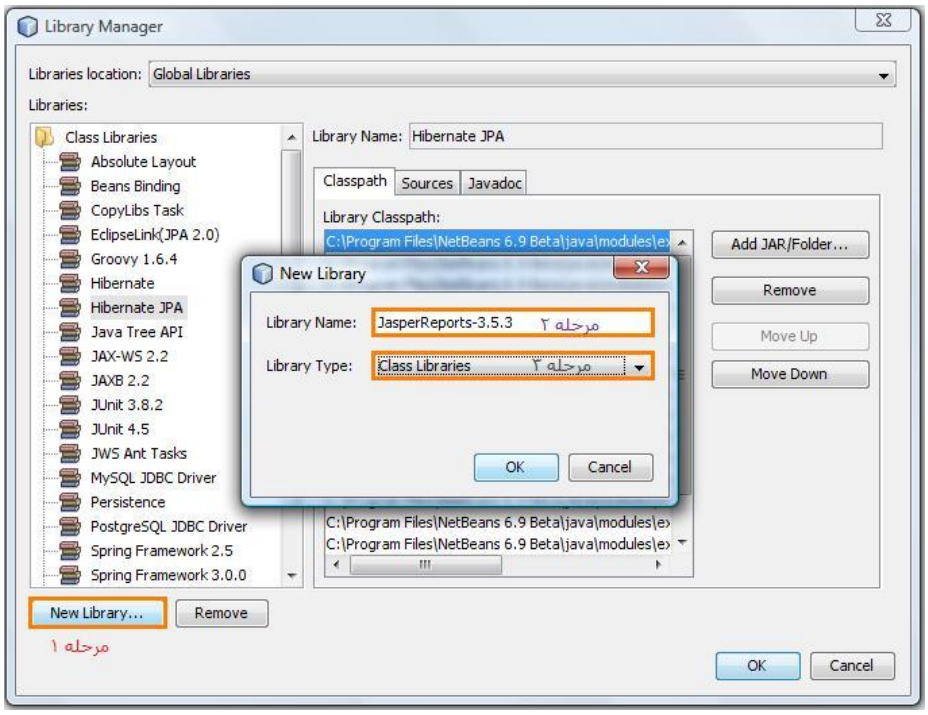
Src : این دایرکتوری حاوی JasperReports Source Code است.

پس از انجام مراحل بالا، حال نوبت به تنظیمات محیط NetBeans می رسد. برای انجام این تنظیمات مراحل زیر را انجام دهید:

۱- محیط NetBeans را باز کنید.

۲- پنجره Library Manager را باز کنید (Tools\libraries)

۳- New Library را کلیک کرده و در آن مقدار JasperReports-3.5.3 را در فیلد Library Name تایپ کنید. در قسمت Library Type عبارت Class Libraries را انتخاب کنید.(شکل ۹-۴)



شکل (۹- ۴) : پنجره Library Manager

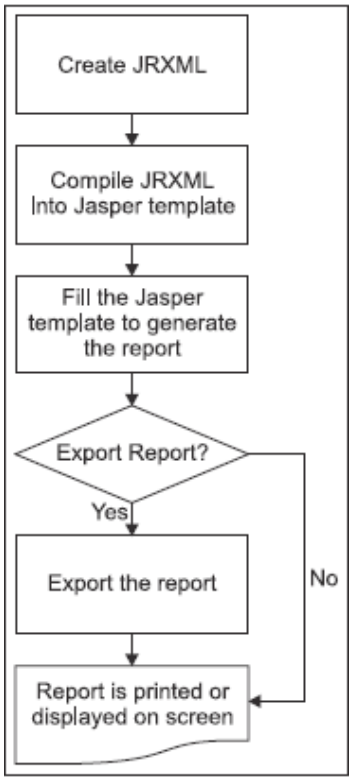
۴- در تب Classpath فایل jasperreports-3.5.3.jar را از دایرکتوری dist و تمام فایل jar را از دایرکتوری lib، اضافه کنید.

۵- در تب Sources ، دایرکتوری src را اضافه کنید.

با انجام مراحل فوق محیط NetBeans آماده تولید گزارش به کمک ابزار JasperReports می باشد.

۹-۴ چگونه با استفاده از JasperReports یک گزارش در جاوا ایجاد نماییم؟

فلوچارت شکل ۹-۵ روند کار تولید گزارش توسط JasperReports را نشان می دهد.



شکل (۹- ۵) : فلوچارت روند کار تولید گزارش توسط JasperReports

همانطور که در فلوچارت فوق مشخص است اولین مرحله، تولید الگوی گزارش می باشد که این الگو به صورت یک فایل XML ذخیره می شود. اگر چه الگوهای گزارشی JasperReports، بصورت فایل های XML هستند، ولی این فایلها با پسوند jrxml ذخیره می شوند.

در ادامه نمونه ای از این الگوها را مشاهده می‌نمایید:

```
<?xml version="1.0" encoding="UTF-8"?>
 <!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
  "http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
   <jasperReport name="simple_template">
          <title>
                  <band height="50">
                  </band>
         </title>
  <pageHeader>
       <band height="50">
       </band>
  </pageHeader>
  <columnHeader>
       <band height="30">
       </band>
  </columnHeader>


  <detail>
```

```
        <band height="100">
        </band>
  </detail>
  <columnFooter>
        <band height="30">
        </band>
  </columnFooter>
        <pageFooter>
                <band height="50">
                </band>
        </pageFooter>
  <lastPageFooter>
        <band height="50">
        </band>
  </lastPageFooter>
  <summary>
        <band height="50">
        </band>
  </summary>
</jasperReport>
```

الگوی فوق نشان دهنده تگ های اصلی این نوع فایل می باشد.

اگر این الگو را کامپایل کنیم و نتیجه کار را اجرا نماییم، یک گزارش خالی «صفحه سفید» به ما نشان خواهد داد. اگر به الگوی فوق دقت کنید، خواهید دید که تگ <band>، تگی است که زیاد مورد استفاده قرار گرفته است.

این تگ در واقع محلی است که داده ها و فرمت نمایش آنها، در آن قرار می گیرند. در الگوی بالا تمام تگ های <band> موجود خالی می باشند. تولید الگوها از دو طریق ممکن می باشد. اول بصورت دستی و دوم به کمک نرم افزارهای موجود مانند iReport.

در مرحله بعد، فایلJRXML یا همان الگوی ایجاد شده برای تولید گزارش به کمک متد های مناسب موجود در **library JasperReports class** مانند JasperCompileManager.compileReport کامپایل می شود.

نتیجه حاصل از عمل کامپایل الگو، به صورت یک فایل Jasper شناخته می شود و با پسوند jasper ذخیره می شود.

اکنون نوبت به انجام مرحله با نام report the filling می رسد.

در این مرحله فایل Jasper ایجاد شده آماده تولید گزارش مورد نظر بوده و فقط کافیست که با داده های مناسب پر شود. به عبارت دیگر Query مورد نظر را اجرا کرده و داده های حاصل از اجرای آن را به فایلJasper منتقل نماییم. در این حالت خواهید دید که به نوعی در حال پر کردن الگوی گزارش تولید شده با داده ها و در نتیجه ایجاد گزارش نهایی می باشیم.یک فایل JRXML یا الگو، تنها باید یکبار کامپایل شود اما فایل Jasper تولید شده می تواند چندین بار برای تولید و نمایش گزارش ها filled شود.

« منظور از این جمله آن است که پس از تولید و کامپایل الگوی گزارش، با ورود داده های متفاوت که حاصل از Query های متفاوت است، می توان عمل گزارش گیری را انجام داد.» گزارش های filled شده با عناون فایل های JasperPrint شناخته شده و با پسوند jrprint ذخیره می شوند.

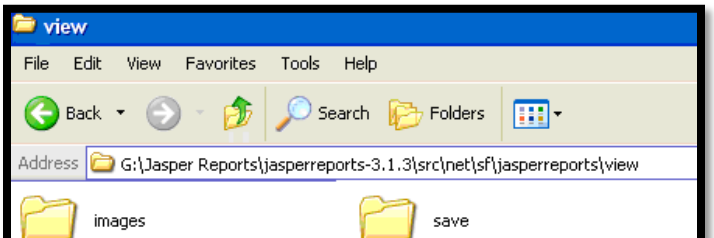
فایل های JasperPrint تنها توسط ابزارهای نمایش گزارش JasperReports قابل مشاهده می‌باشند. این فایل ها می توانند به فرمت های دیگر نیز تبدیل شوند، مانند PDF.

بنابراین می توانند با ابزارهایی مثل Viewers PDF و Processors Word نیز نمایش داده شوند.

۹-۴-۱ JasperReports Viewer چیست؟

JasperReports شامل یک کلاس کمکی به نام net.sf.jasperreports.view.JasperViewer میباشد که از آن برای دیدن گزارشات استفاده میشود. در حقیقت از این ابزار برای مشاهده پیش نمایش گزارش استفاده می‌شود.

این کلاس در مسیر jasperreports-3.1.3\src\net\sf\jasperreports\view قرار دارد.(شکل ۹-۶)



شکل (۹- ۶) : مسیر فایل JasperViewer

در این مسیر فایلی به نام JasperViewer.java وجود دارد.

در این فایل چگونگی ایجاد پنجره گزارش یا JasperViewer و تنظیمات مربوط به آن نوشته شده است.

۹-۴-۲ ایجاد اولین گزارش

قبل از شروع کار، ابتدا از نصب JasperReports بر روی سیستم و راه اندازی آن در محیط Netbeans مطمئن شوید. حال مراحل زیر را به ترتیب انجام دهید.

۱- وارد محیط NetBeans شده و File » New Project را انتخاب نمایید. از قسمت java ، Categories و از قسمت Java Application ،Projects را انتخاب کرده و گزینه Next را کلیک نمایید.

۲- در پنجره باز شده، اسم پروژه را JasperReportsDemoApp تایپ کرده و دکمه Finish را کلیک نمایید.

۳- در پنجره سمت چپ محیط NetBeans که لیست پروژه ها در آن قرار دارد، پروژه JasperReportsDemoApp را باز کرده و روی Libraries کلیک راست کرده و گزینه Add Library را انتخاب نمایید. سپس کتابخانه JasperReports-3.5.3 را به آن اضافه نمایید.

۴- اکنون یک برنامه جاوا با نام JasperReportsDemoApp ایجاد کرده و کد زیر را در آن قرار دهید. «می توانید فایل اجرایی ان را که در بسته وجود دارد اجرا کنید»

```
import java.util.HashMap;
import java.util.Map;
import net.sf.jasperreports.engine.JREmptyDataSource;
```

```
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperExportManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.engine.JasperReport;
import net.sf.jasperreports.view.JasperViewer;

public class JasperReportsDemoApp {
   public static void main(String []args){
      String reportSource = "./report/templates/HelloReportWorld.jrxml";
      String reportDest = "./report/results/HelloReportWorld.jasper";
      Map<String, Object> params = new HashMap<String, Object>();
      try
         {
  JasperCompileManager.compileReportToFile(reportSource , reportDest);
  JasperPrint jasperPrint =
JasperFillManager.fillReport(reportDest, params, new JREmptyDataSource());
          JasperViewer.viewReport(jasperPrint);
         }
      catch (JRException ex){
               ex.printStackTrace();
         }
   }
}
```

نکات مهم در کد فوق عبارتند از:

۱. برای خوانایی بهتر برنامه و مدیریت بهتر پروژه، در پوشه اصلی پروژه، یک پوشه با نام Reports و در آن دو پوشه با نامه های templates و results ایجاد نمایید. در پوشه templates الگوهای مورد استفاده در برنامه ودر پوشه results نتایج اجرای برنامه را قرار می دهیم.

در برنامه نیز با کمک دو خط پررنگ زرد برنامه مسیر این پوشه ها را اعلام می نماییم.

۲. در مرحله بعد الگوی تعیین شده توسط کامپایلر JasperReports کامپایل شده و بصورت یک فایل با پسوند jasper در پوشه results قرار می گیرد.

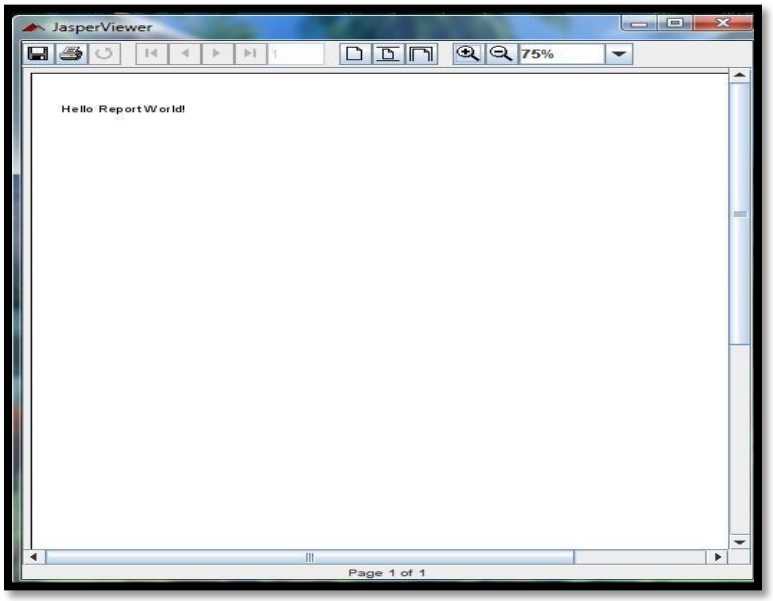
۳. سومین مرحله پر کردن الگوی کامپایل شده با داده ها می باشد. این عمل توسط متد fillReport انجام می شود.

۴. آخرین مرحله، نمایش نتیجه کار توسط متد viewReport یا همان پیش نمایش گزارش می باشد.

۵. این مرحله آخرین عملی است که باید انجام دهید. در این مرحله باید الگوی گزارش را ایجاد نمایید. برای این کار فایلی با نام HelloReportWorld.jrxml در پوشه templates ایجاد نمایید و عبارات زیر را در آن قرار دهید.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">

<jasperReport name="HelloReportWorld">
   <detail>
         <band height="200">
               <staticText>   از تگ براي متن در گزارش استفاده ميگردد
                     <reportElement x="0" y="0" width="500" height="20"/>
                     <text><![CDATA[Hello Report World!]]></text>
               </staticText>
          </band>
   </detail>
</jasperReport>
```

۶- حال برنامه خود را اجرا کرده و نتیجه کار را با تصویر ۹-۷ مقایسه نمایید.



شکل (۹- ۷) : خروجی برنامه HelloReportWorld.jrxml

۹-۴-۳ رسم خط در گزارش

برای رسم خط در گزارش کد زیر را در یکی از قسمتهای گزارش قرار میدهیم.

```
<line>
<reportElement positionType="Float" x="0" y="12" width="270" height="1"
```

```
            forecolor="#808080"/>
         <graphicElement>
<pen lineWidth="0.5"/>
         </graphicElement>
</line>
```

۹-۴-۴ قرار دادن تصویر در گزارش

برای قراردادن تصویر در گزارش ابتدا بایستی تصویر مورد نظر را در مقصد فایل گزارش کنار فایل jasper* کپی کرده و سپس تکه کد زیر را در یکی از قسمتهای گزارش قرار دهید(شکل ۹-۸)

```
<image scaleImage="Clip">

            <reportElement x="0" y="5" width="165" height="40"   تنظیم محل
قرارگیری تصویر />

            <graphicElement/>
<imageExpression class="java.lang.String"><![CDATA[" نام
تصویر.*"]]></imageExpression>

<hyperlinkTooltipExpression>" Tooltip   متن"</hyperlinkTooltipExpression>
</image>
```

شکل (۹- ۸) : خروجی برنامه قراردادن تصویر در گزارش

۹-۵ پارامتر در JasperReports

با استفاده از پارامترها میتوان مقداری را از برنامه به گزارش ارسال نمود و از ان استفاده نمود.

برای استفاده از پارامتر باید ابتدا ان را تعریف نمود.

به تکه کد زیر توجه کنید

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
 "http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
      <parameter name="prTiltle" class="java.lang.String"/>
...
```

در تکه کد بالا متنی که با رنگ زرد مشخص شده خط تعریف پارامتر در گزارش میباشد.

در این تگ پارامتری با نام prTitle و با نوع String تعریف شده است.

یک پارامتر میتواند دارای نوع های متفاوتی به شرح زیر باشد.

java.lang.String
java.lang.Boolean
java.lang.Byte
java.util.Date
java.sql.Timestamp
java.sql.Time
java.lang.Double
java.lang.Float
java.lang.Integer
java.lang.Long
java.lang.Short
java.math.BigDecimal
java.lang.Number
java.util.Collection
java.util.List
java.lang.Object
java.io.InputStream
net.sf.jasperreports.engine.JREmptyDataSource

جدول (۹- 2) : نوع های مختلف گزارش

پس از تعریف پارامتر برای استفاده کردن از ان و قرار دادن ان در یکی از قسمتهای گزارش مثلا Title به صورت زیر عمل میکنیم.

```
<title>
   <band height="79" splitType="Stretch">

      <textField>
            <reportElement x="223" y="31" width="100" height="20"/>
```

```
            <textElement/>
<textFieldExpression class="java.lang.String"><![CDATA[$P{prTiltle}]]>
                                </textFieldExpression>
      </textField>

  ...
   </band>
</title>
```

همان طور که مشاهده میکنید پارامتر تعریف شده بین تگ <textField> قرار دارد.

تگ <reportElement> مربوط به تنظیمات پارامتر همچون مکان قرارگیری ان در گزارش و طول و عرض ان میباشد. در تگ <textFieldExpression> نوع پارمتر و نام پارمتر قرار میگیرد.

به کلاس زیر توجه کنید.این کلاس نحوه استفاده پارامتر در برنامه را شرح میدهد.

```
import java.util.HashMap;
import java.util.Map;
import java.util.logging.Level;
import java.util.logging.Logger;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperCompileManager;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
import net.sf.jasperreports.view.JasperViewer;

public class clsReport{
public static void main(String []args) throws JRException{

    String reportSource = "./report/templates/report.jrxml";
    String reportDest = "./report/result/report.jasper";

  Map<String, Object> params = new HashMap<String, Object>();

params.put("prTiltle", "Hello Report World");

 JasperCompileManager.compileReportToFile(reportSource,reportDest);
    JasperPrint prnt = null;
        try {
            prnt = JasperFillManager.fillReport(reportDest, params);
        } catch (JRException ex) {
            Logger.getLogger(clsReport.class.getName()).log(Level.SEVERE,
null, ex);
        }
       JasperViewer.viewReport(prnt);
    }
 }
```

شرح کد کلاس:

- متغیر های reportSource و reportDest محل مبدا و مقصد گزارش میباشند.

براحتی کار در پوشه پروژه و در کنار پوشه src پوشه ای به نام report ایجاد کرده و در داخل ان دو پوشه با نامهای rerult و templates ایجاد میکنیم.

فایل گزارش طراحی شده را در پوشه templates کپی کرده و با اجرای برنامه خواهید دید JasperViewer یک فایل در پوشه مقصد ایجاد خواهد کرد.

- متغیر ی از کلاس Map که در بسته java.unil میباشد برای جای دادن نام و مقدار پارامتر مورد استفاده قرار میگیرد.

توجه داشته باشید که ورودی دوم متغیر params از نوع object میباشد و بسته به اینکه پارامتر گزارش از چه نوعی میباشد ورودی تغییر خواهد کرد.

```
params.put(نام پارامتر, مقدار);
```

- متد compileReportToFile که از کلاس JasperCompileManager میباشد برای کامپایل کردن فایل مبدا و ایجاد فایل **jasper**. در مقصد میباشد.

اگر فایل مبدا از نظر syntax خطایی داشته باشد عمل گزارش گیری صورت نخواهد گرفت.

- متد fillReport از کلاس JasperFillManager عمل قرار دادن دادها به اصطلاح لود انها از فایل مقصد را انجام میدهد.
- خروجی این تابع از نوع JasperPrint میباشد و دارای چندین سازنده میباشد که در ادامه به انها اشاره خواهد شد. یکی از سازنده های این تابع ادرس فایل مقصد و پارامترهای ارسالی به گزارش را دریافت میکند.
- متد viewReport از کلاس JasperViewer برای نمایش گزارش ایجاد شده مورد استفاده قرار میگیرد. این متد نیز دارای سازنده های زیادی میباشد.

در این کلاس این متد یک متغیر از نوع JasperPrint که قبلا گزارش لود شده در ان میباشد را دریافت و ان را نمایش میدهد.

۹-۶ گزارش دیتابیس

یکی از کارهای مهمی که میتوان با JasperReports کرد ایجاد گزارش از بانک اطلاعاتی میباشد.

در این مبحث سعی شده است که بتوان روشهای مختلف گزارش گیری از بانک را انجام دهیم.

۹-۶-۱ گزارش گیری ایستا

به کد زیر دقت کنید

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport name="dbReport">
<queryString>
                    <![CDATA[select cdBook,chrBook from tbbook]]>
</queryString>
<field name="cdBook" class="java.math.BigDecimal"/>
<field name="chrBook" class="java.lang.String"/>
...
<detail>
              <band height="125" splitType="Stretch">
                   <textField>
              <reportElement x="371" y="30" width="100" height="20"/>
                   <textElement textAlignment="Center"
                   verticalAlignment="Middle">
                         <font fontName="Tahoma" size="9" isBold="true"/>
              </textElement>
<textFieldExpression Class="java.math.BigDecimal"><![CDATA[$F{cdBook}]]>
                              </textFieldExpression>
                   </textField>
                   <textField>
                         <reportElement x="216" y="50" width="100"
                         height="20"/>
                         <textElement/>
<textFieldExpression class="java.lang.String"><![CDATA[$F{chrBook}]]>
                              </textFieldExpression>
                   </textField>
              </band>
</detail>
...
```

همانطور که در کد فوق مشاهده میکنید برای گزارش از بانک اطلاعاتی ابتدا باید کوئری اطلاعاتی که میخواهید گزارش گرفته شود مشخص کنید.

```
<queryString>
                    <![CDATA[select   cdBook,chrBook   from tbbook]]>
</queryString>
```

در کوئری فیلد هایی که باید نمایش داده شود اورده میشود.

کوئری میتواند ترکیبی از چند جدول باشد و یا دارای شرط خاص داشته باشد.

بعد از تعریف کوئری مربوط به بانک نوبت به تعریف فیلد های نمایش داده شده در گزارش میرسد.

```
<field name="cdBook"  class="java.math.BigDecimal"/>
<field name="chrBook" class="java.lang.String"/>
```

همانطور که مشاهده میکند برای تعریف فیلد های کوئری از تگ <field> استفاده میگردد و در ان نام فیلد و نوع ان مشخص میگردد.

معمولا برای نمایش فیلدهای جدول ان را در قسمت Detail قرارمیدهیم و ان را به صورت زیر تعریف مینماییم.

```
<detail>
              <band height="125" splitType="Stretch">

                  <textField>
              <reportElement x="371" y="30" width="100" height="20"/>
                <textElement textAlignment="Center" verticalAlignment="Middle">
                        <font fontName="Tahoma" size="9" isBold="true"/>
                 </textElement>
<textFieldExpression
Class="java.math.BigDecimal"><![CDATA[$F{cdBook}]]></textFieldExpression>
                  </textField>


                  <textField>
                        <reportElement x="216" y="50" width="100"
                        height="20"/>
                        <textElement/>
<textFieldExpression
class="java.lang.String"><![CDATA[$F{chrBook}]]></textFieldExpression>
                  </textField>

              </band>
</detail>
```

تگ <textFiled> محلی میباشد که میتوان بین انها تنظیمات مربوط به فیلد را انجام داد.

تگ <reportElement> مربوط به اندازه و محل قرار گیری فیلد گزارش میباشد.

تگ <textElement> مربوط به تنظیمات متن گزارش میباشد.مثلااندازه و نوع فونت و یا محل قرارگیری متن که

همانطور که مشاهده میکنید کلاس فوق همانند کلاس قبل تر میباشد با این تفاوت که در این کلاس پارامتری وجود ندارد و Connection ی ایجاد شده است که در پارامتر متد fillreport به کار رفته است.

توجه ۲: برای ارسال پارامتر به گزارش کافی است در ورودی متد `fillReport` به جای `new HashMap()` متغیری از ان را تعریف نموده و پرامترها را به ان اضافه کنید

۹-۶-۱-۲ تغییر کوئری گزارش از طریق ارسال پارامتر

اکثر اوقات شما مجبور خواهید بود پارامتری را به گزارش ارسال کنید تا بر اساس ان کوئری و یا همان لیست گزارش تغییر کند.

برای اینکار کافیست در گزارش یک پارامتر تعریف نموده و به صورت زیر تغییری در QueryString دهید.

```
<parameter name="param" class="java.lang.Integer"/>

<queryString>
<![CDATA[select  cdBook,chrBook  from tbbook where cdBook=$P{param}]]>
</queryString>
```

توجه داشته باشید که نوع پارامتر باید همنوع فیلد ی که در کوئری مقایسه میگردد باشد.

برای استفاده از این روش در برنامه کافیست در متد fillReport متغیر پرامتر خود را به گزارش پاس دهیم.

۹-۶-4 گزارش گیری پویا

در این روش گزارش گیری همانند قبل عمل میشود با این تفاوت که در این گزارش گیری تگ **<QueryString>** وجود ندارد و تنها بایستی فیلدهای گزارش تعریف گردد.

به کد زیر توجه کنید.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport name="dbReport">
<field name="cdBook" class="java.math.BigDecimal"/>
<field name="chrBook" class="java.lang.String"/>
...
<detail>
              <band height="125" splitType="Stretch">
                   <textField>
              <reportElement x="371" y="30" width="100" height="20"/>
                   <textElement textAlignment="Center"
                   verticalAlignment="Middle">
                         <font fontName="Tahoma" size="9" isBold="true"/>
              </textElement>
<textFieldExpression Class="java.math.BigDecimal">
                        <![CDATA[$F{cdBook}]]></textFieldExpression>
                   </textField>
                   <textField>
                         <reportElement x="216" y="50" width="100"
                         height="20"/>
                         <textElement/>
<textFieldExpression class="java.lang.String">
                        <![CDATA[$F{chrBook}]]></textFieldExpression>
                   </textField>
              </band>
</detail>
...
```

در برنامه برای استفاده از گزارش به صورت زیر عمل میکنیم.

```
public class clsReport{
    private static Statement statement;
public static void main(String []args) throws JRException, SQLException,
IOException{
        try {
            String reportSource = "./report/templates/report1.jrxml";
           String reportDest = "./report/result/report1.jasper";

            JasperCompileManager.compileReportToFile(reportSource,
reportDest);

           Class.forName("com.mysql.jdbc.Driver");
   String connectionUrl = "jdbc:mysql://localhost:3306/db_lib?" +
                                    "user=root&password=123";
  Connection cnn = DriverManager.getConnection(connectionUrl);

            statement = cnn.createStatement();
            Statement stmp = cnn.createStatement();
            ResultSet rs = stmp.executeQuery("select cdBook,chrBook from
tbBook");

  JRResultSetDataSource resultSetDataSource = new JRResultSetDataSource(rs);

            JasperPrint prnt = null;
```

```
  prnt = JasperFillManager.fillReport(reportDest,new HashMap(),
resultSetDataSource);
            stmp.close();
            cnn.close();
            JasperViewer.viewReport(prnt);
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(clsReport.class.getName()).log(Level.SEVERE,
null, ex);
        }
}}
```

همانطور که مشاهده میکنید در این روش کوئری در برنامه انتخاب میگردد و نتیجه ان به متغیر rs نسبت داده میشود.

برای پاس دادن این نتیجه به گزارش باید یک DataSource ایجاد شده و نتیجه را در ان قرار بدهیم.

```
JRResultSetDataSource resultSetDataSource = new JRResultSetDataSource(rs);
```

بعد از ایجاد متغیری از نوع دیتاسورس ان را با استفاده از یکی از سازنده های متد fillReport به گزارش پاس میدهیم.

۹-۶-۲-۱ اعمال روی فیلدها

بسیاری از اوقات شما مجبور هستید تا جمع یک فیلد خاص یا میانگین ان را در قسمتی از گزارش مثلا pageFooter نمایش دهید.

JasperReports این امکان را کاربر میدهد تا با تعریف یک متغیر در گزارش از این امکانات استفاده نماید.

در این مثال میخواهیم تعداد کتابهای موجود در کتابخانه را در قسمت pageFooter نمایش دهیم.

ابتدا بایستی فیلد تعداد کتاب (cnBook) را تعریف و متغیری که جمع تعداد را در خود نگه داری میکند را مشخص نماییم.و در اخر نیز ان را در قسمت pageFooter قرار دهیم.

به تکه کد زیر توجه نمایید.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
 "http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport name="Report">
...
<field name="cnBook" class="java.lang.Integer"/>
```

```
<variable name="cnSum" class="java.lang.Integer" calculation="Sum">
      <variableExpression><![CDATA[$F{cnBook}]]></variableExpression>
</variable>
...
<columnFooter>
        <band height="30">
        </band>
</columnFooter>
        <pageFooter>
                 <band height="54">
              <textField>
                        <reportElement x="50" y="0" width="55" height="20"/>
                        <textElement/>
   <textFieldExpression class="java.lang.Integer">
                             <![CDATA[$V{cnSum}]]></textFieldExpression>
            </textField>
            </band>
        </pageFooter>
...
```

همانطور که در کد فوق مشاهده میکنید فیلدی با نام cnBook در گزارش تعریف شده است.

برای تعریف متغیر از تگ <varible> استفاده میگردد که باید در ان نوع متغیر و محاسباتی که باید صورت بگیرد مشخص شود.

محاسباتی که توسط گزارش صورت میگیرد به شرح زیر میباشد.

محاسبات	توضیح
calculation="Sum"	عمل جمع فیلد را انجام میدهد
calculation="Count"	تعداد فیلد در گزارش را محاسبه میکند
calculation="DistinctCount"	تعداد فیلد با مقدار غیر تکراری را نشان میدهد
calculation="Average"	میانگین مقدار فیلد را نشان میدهد
calculation="Lowest"	کوچکترین مقدار فیلد را مشخص میکند
calculation="Highest"	بزرگترین (بیشترین) مقدار فیلد را نشان میدهد
calculation="StandardDeviation"	انحراف معیار فیلد را محاسبه میکند
calculation="Variance"	واریانس فیلد را محاسبه میکند
calculation="First"	اولین فیلد گزارش را نشان میدهد

جدول (۹- ۳) : محاسبات در گزارش

برای نمایش متغیر همانند نمایش فیلدها عمل میکنیم با این تفاوت که در متغیرها نامشان در میان کلمه کلیدی $v{} قرار میگیرد.

تگ فیلد در گزارش

```
<textFieldExpression Class="java.lang.Integer">
                                <![CDATA[$F{نام فیلد}]]></textFieldExpression>
```

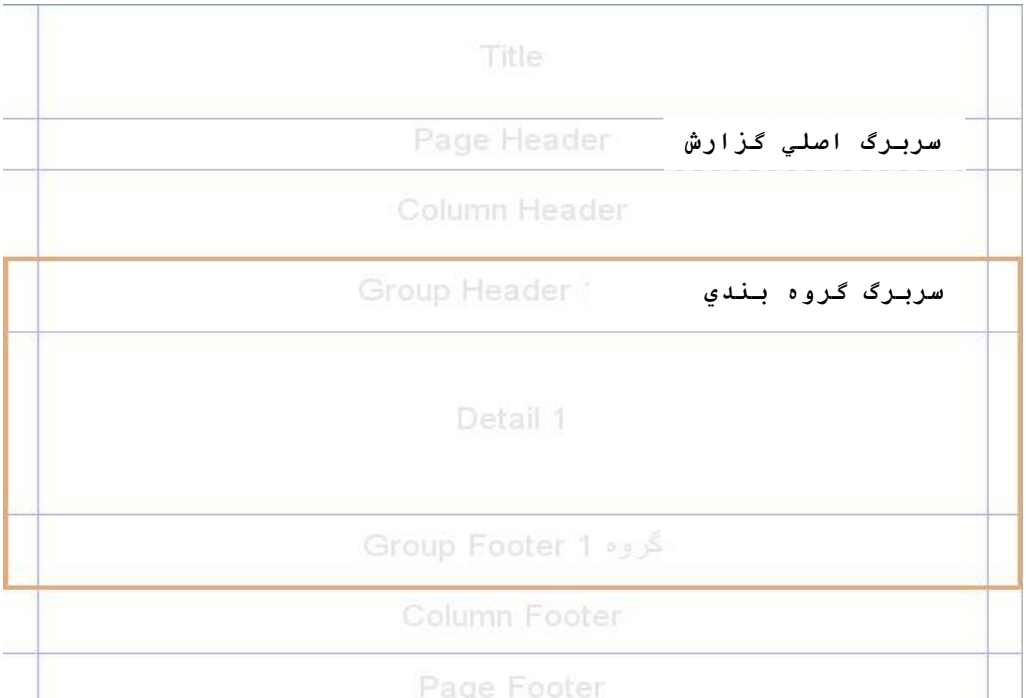
تگ متغیر در گزارش

```
<textFieldExpression class="java.lang.Integer">
                                <![CDATA[$V{ نام
متغیر}]]></textFieldExpression>
```

۹-۷ گروه بندی در گزارش

در JasperReports چنین قابلیتی وجود دارد تا شما بتوانید داده های خود را براساس فلیدی گروه بندی کنید.

با گروه بندی گزارش شما یک groupHeader و groupFooter خواهید داشت و گزارش به قسمت های زیر تقسیم خواهد شد.(شکل ۹-۹)



پـايـين صفحه گروه بـنـدي

پـايـين صفحه گـزارش

شکل (۹- ۹) : قسمتهای گزارش

قسمتی که با کادر مشخص شده است در گزارش براساس گروه خاص تکرار خواهد شده ولی قسمتهای دیگر ثابت میباشند.

مثال: برای نمونه میخواهیم لیست تمام کتب را بر اساس موضوعشان نشان دهیم و هر نام موضوع را در بالای لیست کتب مربوط به ان بیاوریم.برای این کار کافی است همانند قبل عمل کرده وفیلد های گزارش را تعریف نماییم و فقط گزارش را بر اساس موضوع کتاب گروه بندی کنیم.به کد زیر دقت کنید.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "//JasperReports//DTD Report Design//EN"
 "http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport name="ReportGroup">
<field name="cdbook" class="java.math.BigDecimal"/>
<field name="chrbook" class="java.lang.String"/>
<field name="cnBook" class="java.lang.Integer"/>
<field name="subject" class="java.lang.String"/>

      <group name="grpSubject">
<groupExpression><![CDATA[$F{subject}]]></groupExpression>
      <groupHeader>     سربـرگ گـروه بـنـدي
            <band height="50">
     <textField>    قـرار دادن فـيـلـد مـوضوع در سربـرگ
      <reportElement x="32" y="14" width="100" height="20"/>
            <textElement/>
<textFieldExpression class="java.lang.String">
                        <![CDATA[$F{chrBook}]]></textFieldExpression>
      </textField>
            </band>
      </groupHeader>

      <groupFooter>     پـايـين صفحه گـروه بـنـدي
                  <band height="50"/>
      </groupFooter>
      </group>

      <title>
            <band height="20"></band>
```

```
</title>
<pageHeader>
            <band height="20"></band>
</pageHeader>
<detail>
قراردادن فيلدهاي گزارش
</detail>
<columnFooter>
        <band height="30">
        </band>
</columnFooter>
        <pageFooter>
                <band height="54"></band>
        </pageFooter>
</jasperReport>
```

همانطور مشاهده میکنید تعریف گروه با تگ group name شروع میشود و در ادامه نام گروه ذکر میگردد.

برای ذکر کردن فیلدی که باید گروه بندی شود از تگ groupExpression به صورت زیر استفاده میگردد.

```
<groupExpression><![CDATA[$F{نام فيلد}]]></groupExpression>
```

و در ادامه برای تعیین سر برگ و پایین صفحه گروه از دو تگ <groupHeader> و </groupFooter> استفاده میگردد.

بقیه روال به صورت قبل انجام میگیرد.

منابع

FOURTH EDITION، Teach Yourself Programming with Java in 24 Hours

NetBeans IDE 6.0 Java Quick Start Tutorial

JasperReports for Java Developers – David R.Heffelfinger – Chapter 4

کتاب چگونه با جاوا برنامه بنویسیم، ویرایش ششم، اثر دایتل و دایتل

کتاب تفکر در جاوا، ویرایش سوم، اثر بروس اکل

سایت ها

www.download.oracle.com