جاوا به زبان فوق العاده ساده

ترجمه و تالیف : هادی خداپناه

بسم تعالی

زبان برنامه نویسی جاوا در دنیا روز به روز با اقبال بیشتری مواجه می شود و عده ی زیادی به برنامه نویسی با این زبان روی می آورد. دلیل این هم روشن است: امن بودن و در عین حال قدرت زبان جاوا باعث گردیده، تعداد برنامه نویس هایی که با جاوا کار می کنند از تعداد برنامه نویس هایی که با سایر زبان های برنامه نویسی کار می کنند به مراتب بیشتر باشد. در کنار این موضوع شاهد رواج گسترده ی گوشی های هوشمند بین تمام مردم دنیا هستیم، که اکثرا از سیستم عامل اندروید استفاده میکنند و حال آنکه ما برای ایجاد و نوشتن اپلیکیشن ها و برنامه های جانبی روی اندروید بایستی از جاوا استفاده کنیم. باید توجه داشت که بخش رابط کابری اندروید با زبان جاوا نوشته شده است و برنامه های اندرویدی میتوانند برای ارتباط با لایه های زیرین سیستم عامل اندروید از کتابخانه های جاوایی اندروید بهره ببرند .همچنین برای کار با پایگاه داده هایی همچون اوراکل، بدون دانستن زبان جاوا امری غیر ممکن است. زیرا که برای کاربا پایگاه داده ی اوراکل می بایستی حتما ، با جاوا آشنایی داشته باشیم. ما با استفاده از جاوا قدر خواهیم بود تا برنامه های کاربردی، برای لینوکس ، اندروید و ویندوز بنویسیم.

هدف بنده از نگارش این کتاب ارتقای سطح علمی جاوا در ایران بوده و معیار اصلی من هنگام تالیف کتاب مورد توجه قرار دادن خوانندگان مبتدی و کاربرانی که اصلا برنامه نویسی نکرده اند بوده است. در عین حال از کابران حرفه ای نیز غافل نشده ام، و این کتاب شامل نکات ارزنده ای برای متخصصین حرفه ای کامپیوتر است.

رویه آموزشی کتاب هم از مطالب مقدماتی شروع شده و تا سطح پیشرفته پیش میرود. سعی من در تالیف این کتاب به صورت خود آموز بوده به صورتی که شما دیگر نیازی به کلاس آموزشی یا استاد ندارید.

و اما وجه تسمیه این کتاب با سایر کتاب های فارسی جاوا موجود در ایران ، در این است که بنده تمام مطالب فصل های کتاب را از آخرین ویرایش های کتاب های معتبر انگلیسی اخذ کرده و به جرات می توانم بگویم حتی از یک منبع فارسی (که معمولا پر از اشتباه اند) استفاده نکرده ام.

پیش فرض من در هنگام نوشتن کتاب این بوده است، که خوانندگان این کتاب در سطح مقدماتی و پایین قرار دارند و قدم به قدم توسط مطالعه ی این کتاب به سطح خوبی در جاوا خواهند رسید.

. خواننده گان عزیز میتوانند نظرات و پیشنهادات خود را از طریق زیر به مولف کتاب ، هادی خداپناه ارسال کنند:

۱- آی دی تلگرام : https://telegram.me/hadi_khodapanah_computer

۲- ایمیل *hadi_khodapanah@hotmail.com*

۳- برای اطلاع و دانلود از آخرین تالیفات دیگر و نیز آخرین ویرایش این کتاب، میتوانید به کانال زیر رجوع کنید:

https://telegram.me/khodapanah_hadi_IT

هادی خداپناه

۱۳۹۵/۶/۱۳

فهرست مطالب

فصل ٤

فصل ۵

فصل ۶

فصل ۷

فصل 1

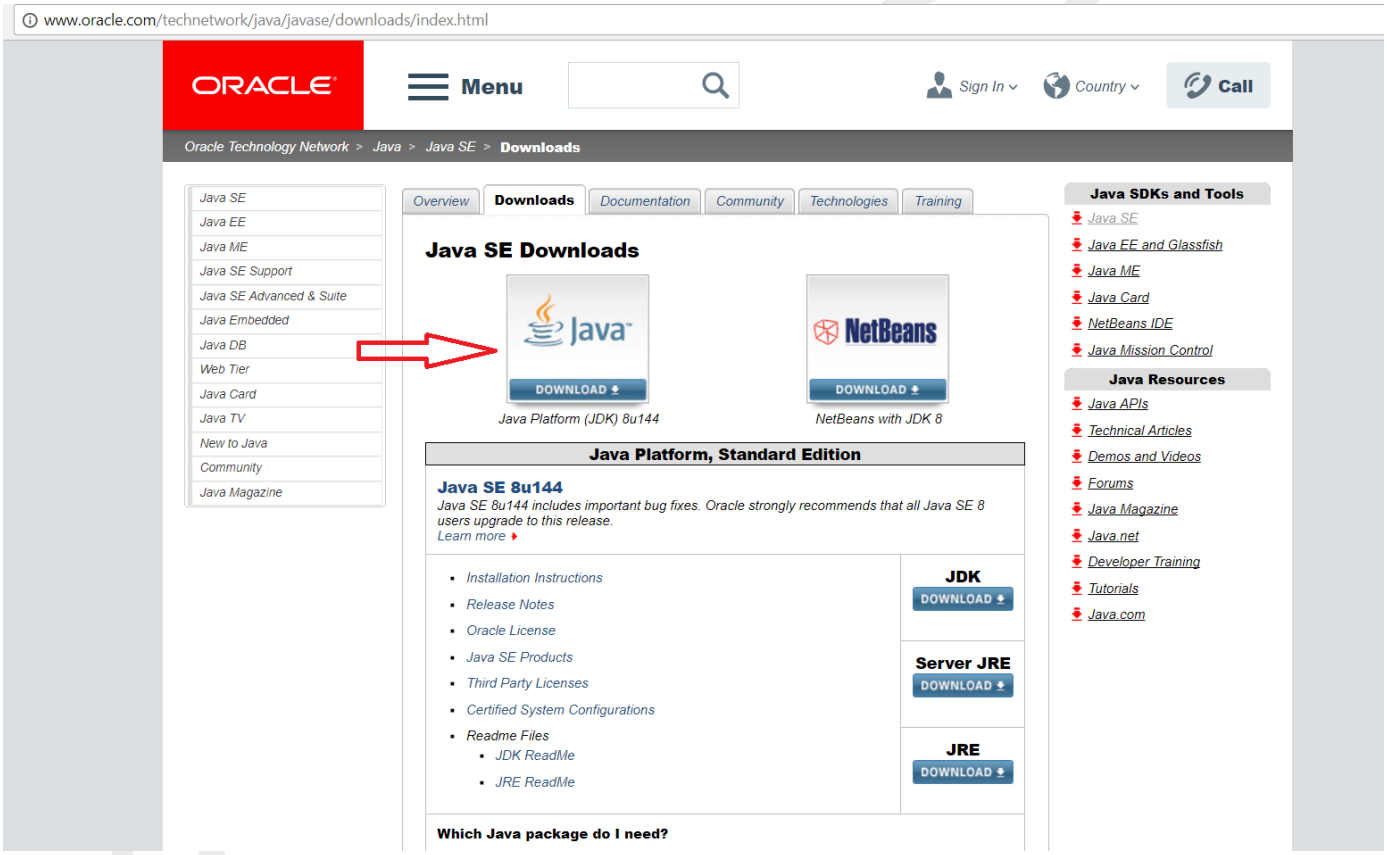
نصب JAVA

در ابتدا فایل ***JDK (Java SE Development Kit)*** و نیز فایل ***JRE: (Java Runtime Environment)*** رو از سایت رسمی آن به آدرس

http://www.oracle.com/technetwork/java/javase/downloads/index.html

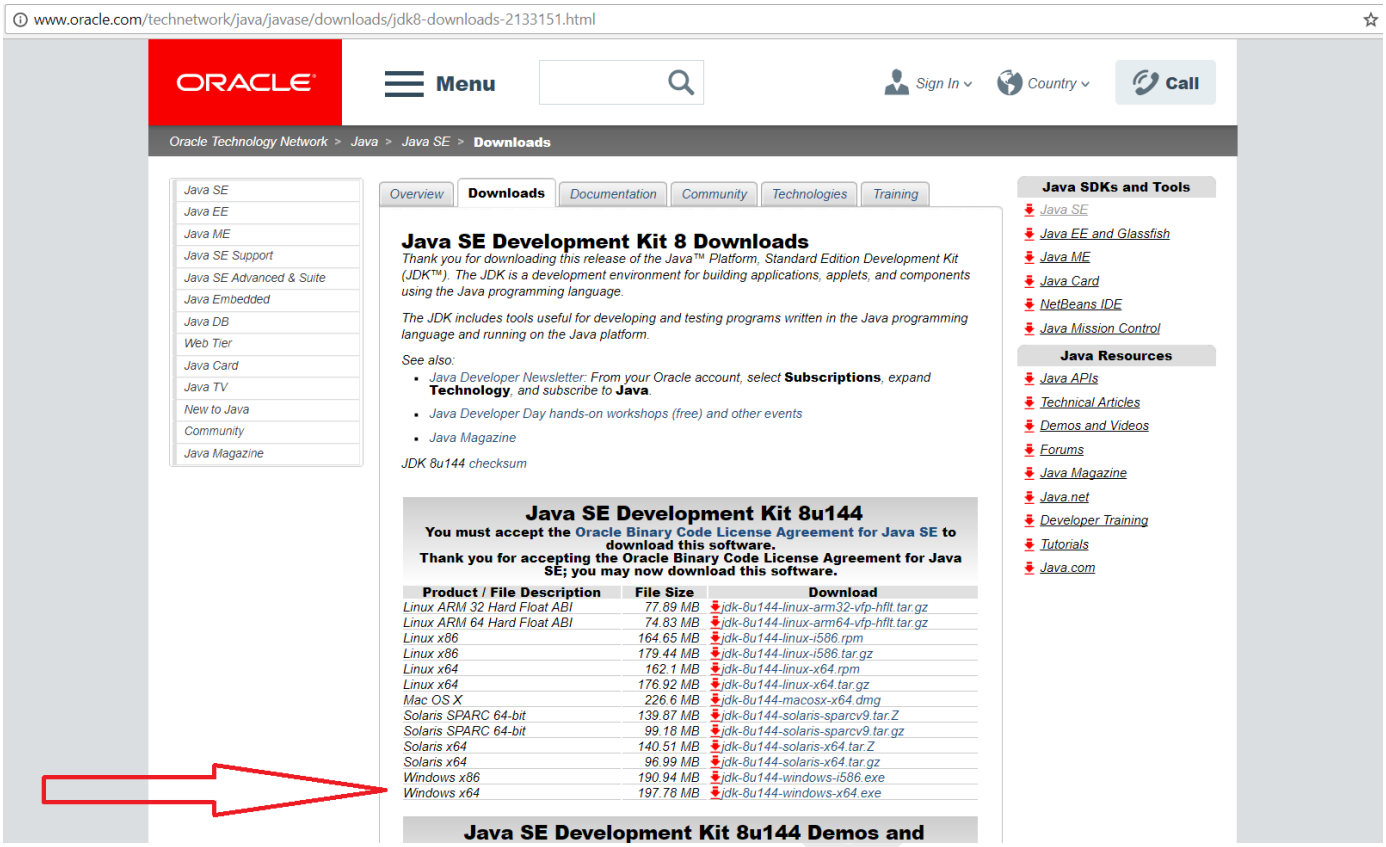
دانلود کنید.و هر دو را نصب کنید. .

توجه کنید که در زمانی که این مطلب رو می نویسم، گویا ایران تحریم شده است و باید از **VPN** (همون فیلتر شکن خودمون)استفاده کنیم از سایت بالا این دو فایل رو دانلود کنید. با اینکه از سایت های ایرانی آنها را تهیه نمایید.

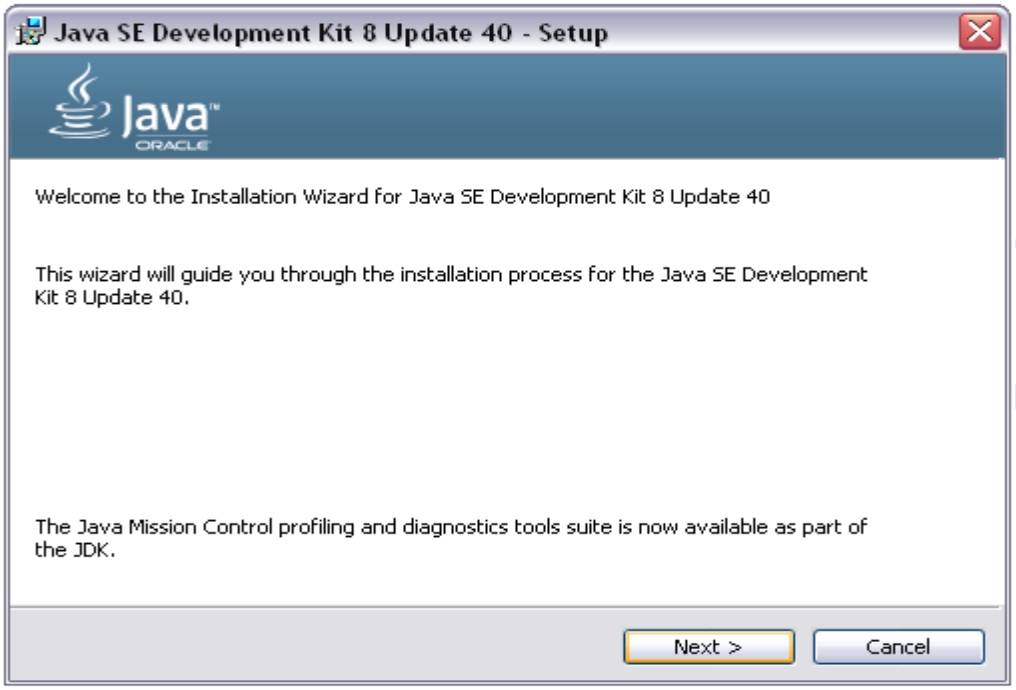


بعد کلیک روی دکمه ی مشخص شده در تصویر زیر رو به خواهید گشت.

سپس گزینه ی Accept License Agreement را تیک دار کنید. و از داخل کتدر قرمز رنگ JDK متناسب با سیستم عامل خود را انتخاب کنید.

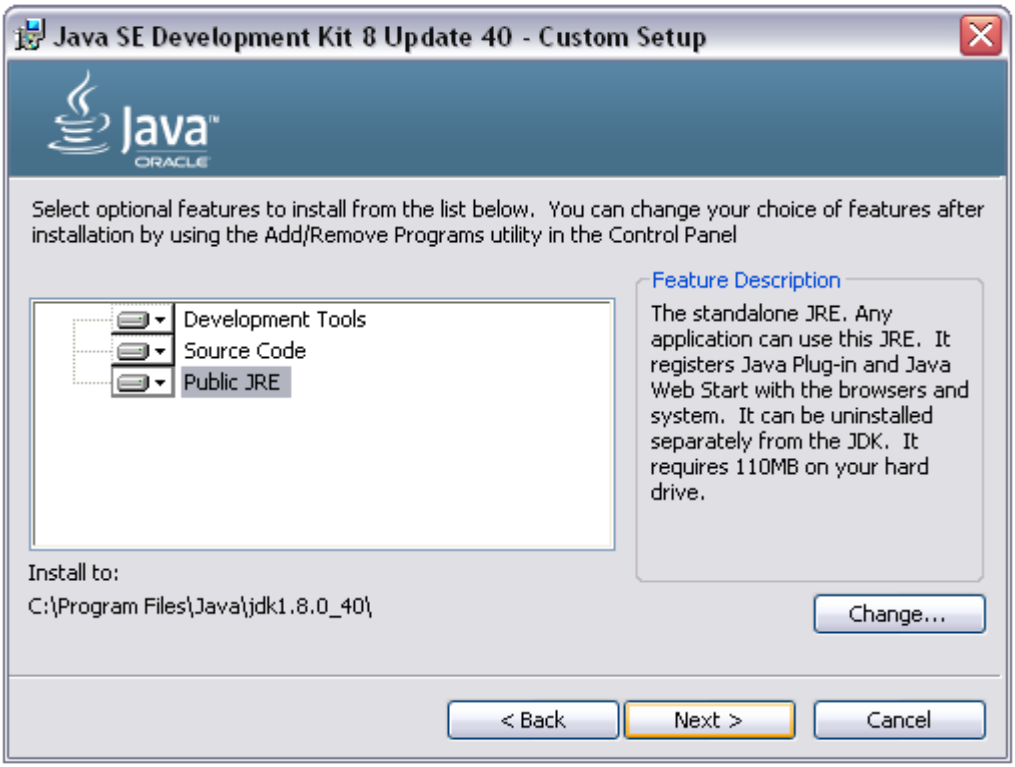

www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
ORACLE
Menu
Sign In
Country
Call
Oracle Technology Network > Java > Java SE > Downloads
Java SE
Java EE
Java ME
Java SE Support
Java SE Advanced & Suite
Java Embedded
Java DB
Web Tier
Java Card
Java TV
New to Java
Community
Java Magazine
Overview
Downloads
Documentation
Community
Technologies
Training
Java SE Development Kit 8 Downloads
Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.
The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.
See also:
Java Developer Newsletter: From your Oracle account, select Subscriptions, expand Technology, and subscribe to Java.
Java Developer Day hands-on workshops (free) and other events
Java Magazine
JDK 8u144 checksum
Java SE Development Kit 8u144
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.
Product / File Description
File Size
Download
Linux ARM 32 Hard Float ABI 77.89 MB jdk-8u144-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI 74.83 MB jdk-8u144-linux-arm64-vfp-hflt.tar.gz
Linux x86 164.65 MB jdk-8u144-linux-i586.rpm
Linux x86 179.44 MB jdk-8u144-linux-i586.tar.gz
Linux x64 162.1 MB jdk-8u144-linux-x64.rpm
Linux x64 176.92 MB jdk-8u144-linux-x64.tar.gz
Mac OS X 226.6 MB jdk-8u144-macosx-x64.dmg
Solaris SPARC 64-bit 139.87 MB jdk-8u144-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit 99.18 MB jdk-8u144-solaris-sparcv9.tar.gz
Solaris x64 140.51 MB jdk-8u144-solaris-x64.tar.Z
Solaris x64 96.99 MB jdk-8u144-solaris-x64.tar.gz
Windows x86 190.94 MB jdk-8u144-windows-i586.exe
Windows x64 197.78 MB jdk-8u144-windows-x64.exe
Java SE Development Kit 8u144 Demos and
Java SDKs and Tools
Java SE
Java EE and Glassfish
Java ME
Java Card
NetBeans IDE
Java Mission Control
Java Resources
Java APIs
Technical Articles
Demos and Videos
Forums
Java Magazine
Java.net
Developer Training
Tutorials
Java.com

بعد از دانلود فایل JDK **نوبت به نصب آن میرسد**

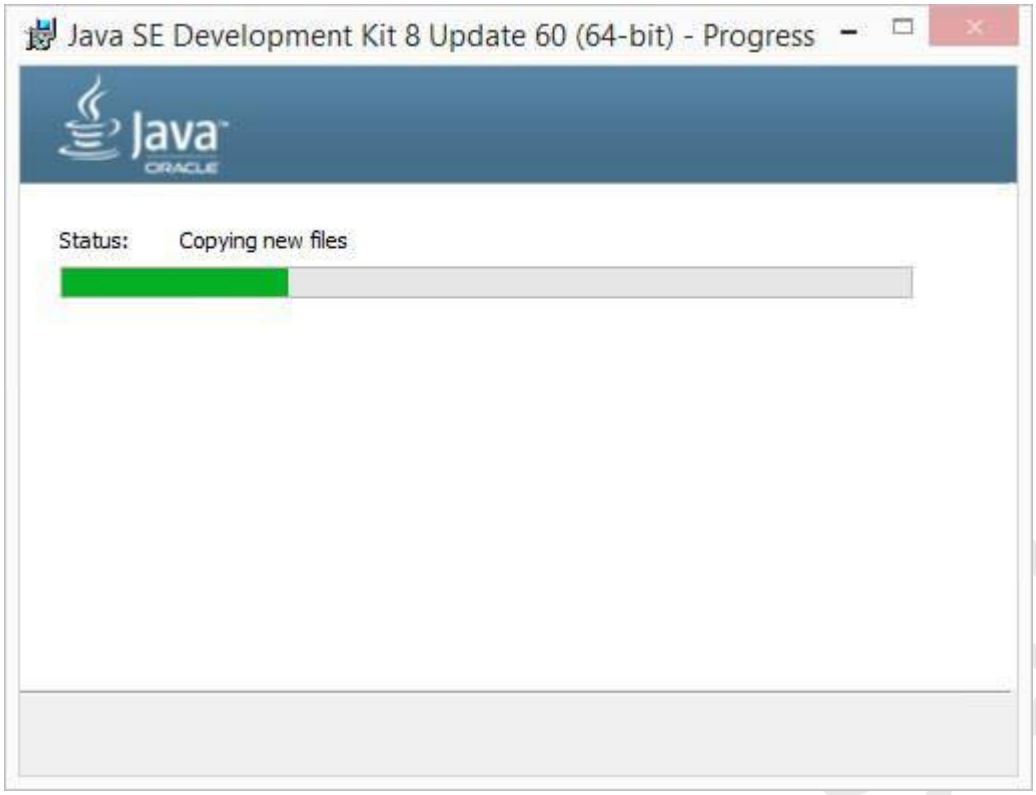


فایل اجرایی جاوا (jdk) را اجرا کنید.

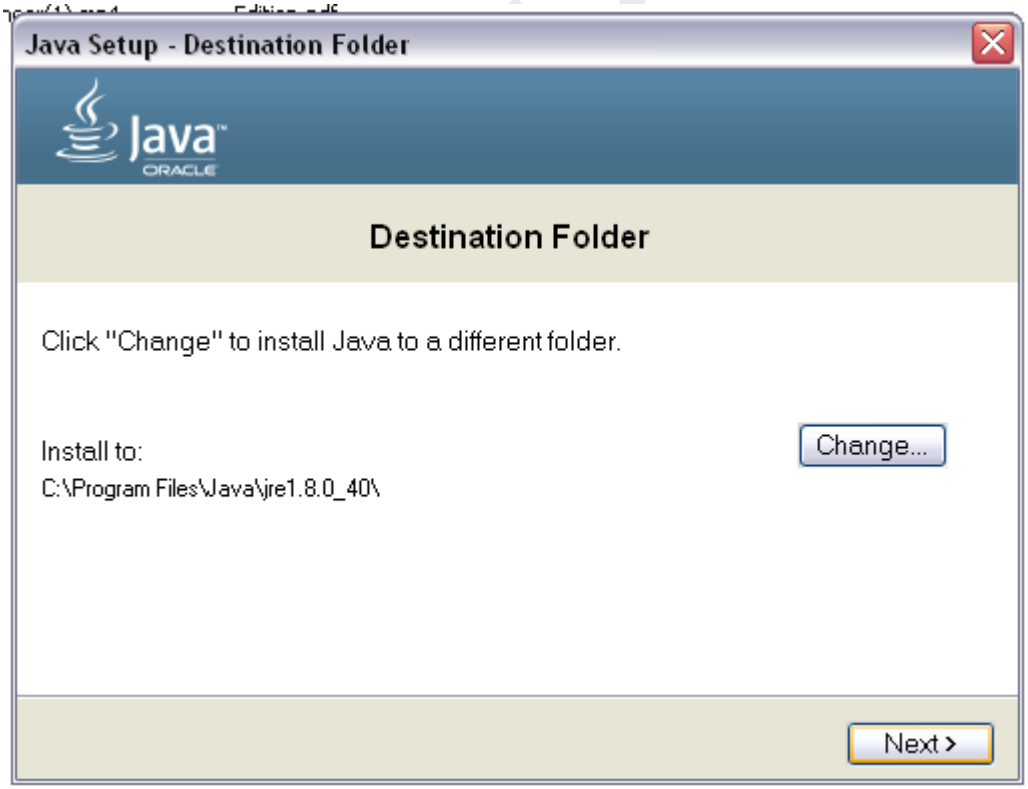
در کادر اول زیر روی Next کلیک کنید.



در کادر دوم نیز بعد از مشخص کردن مسیر نصب جاوا روی Nextکلیک کنید

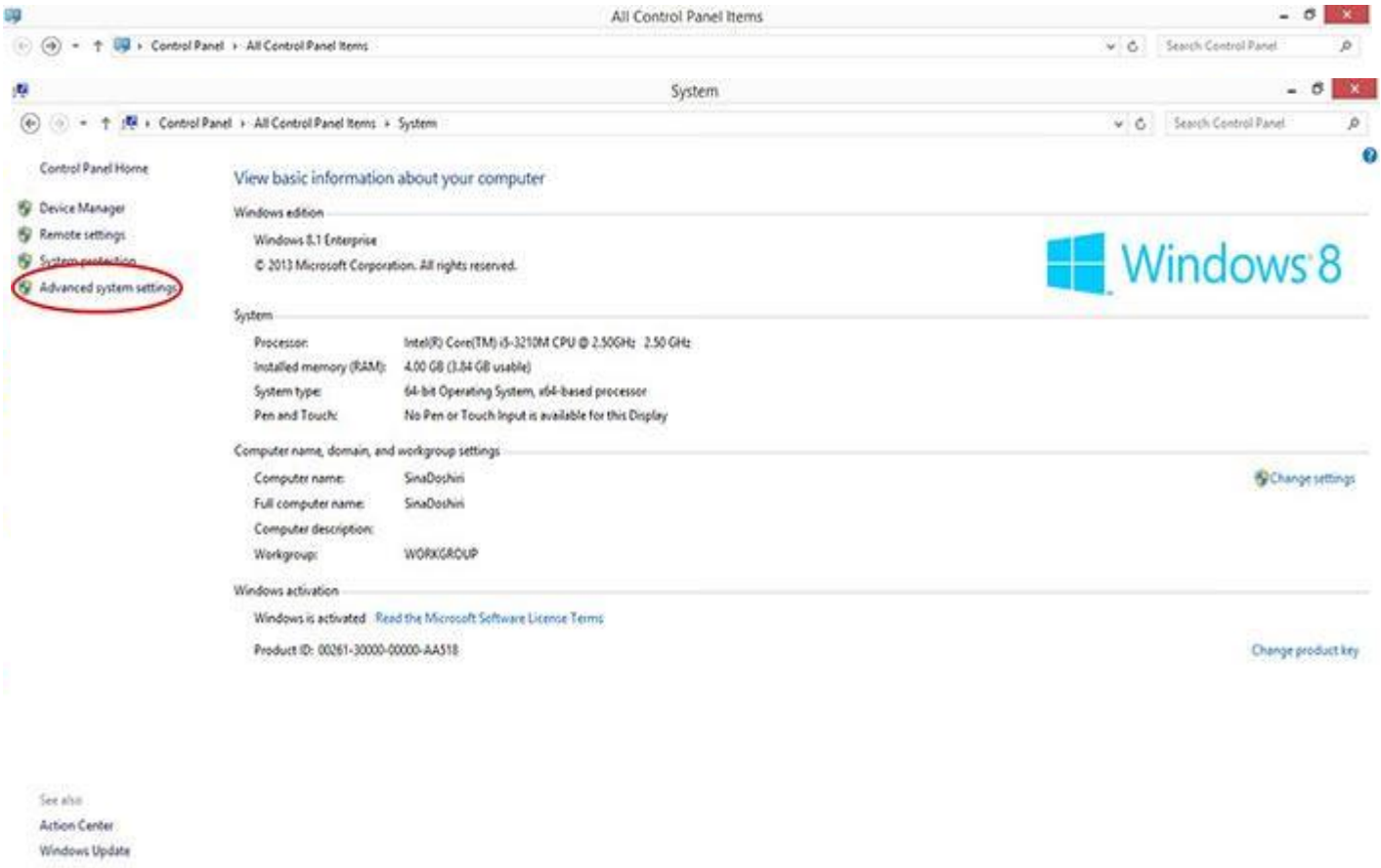


در اینجا بعد از مشخص کردن پوشه ی مقصد برای نصب جاوا روی Nextکلیک کنید.

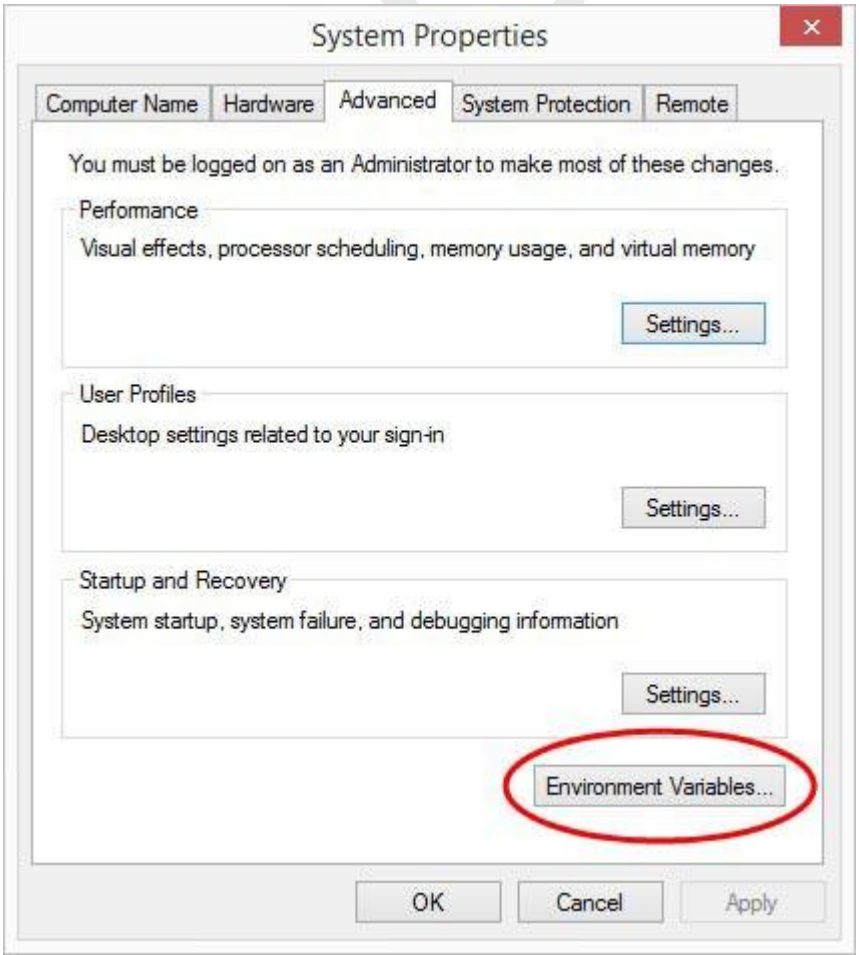


در کادر پایین اظهار میکند که جاوا با موفقیت نصب شد...روی Closeکلیک کنید...

حال که ما جاوا را نصب کردیم.بایستی جاوا رو به سیستم عامل خود معرفی کنیم. برای اینکار وارد کنترل پنل شوید و روی System کلیک کنید

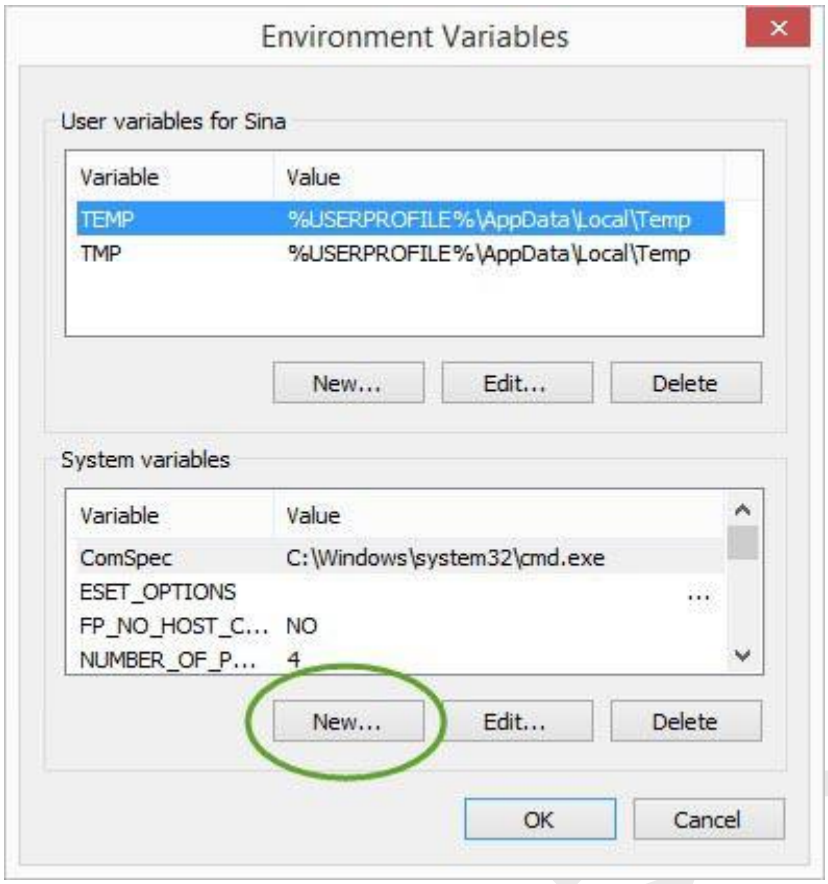


آنگه پنجره ای ظاهر خواهد گشت که روی Advanced system settings کلیک کنید

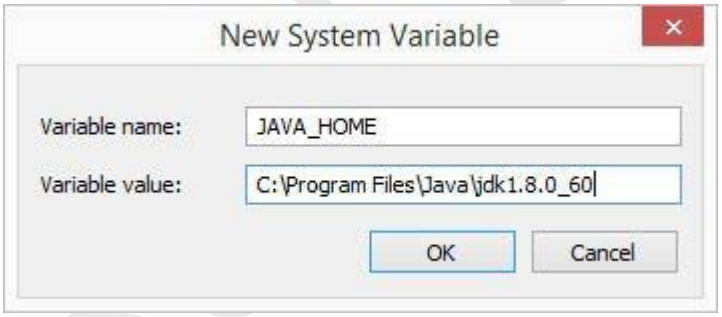


حال پنجره ای ظاهر میشود که باید روی Environment Variables کلیک کنید

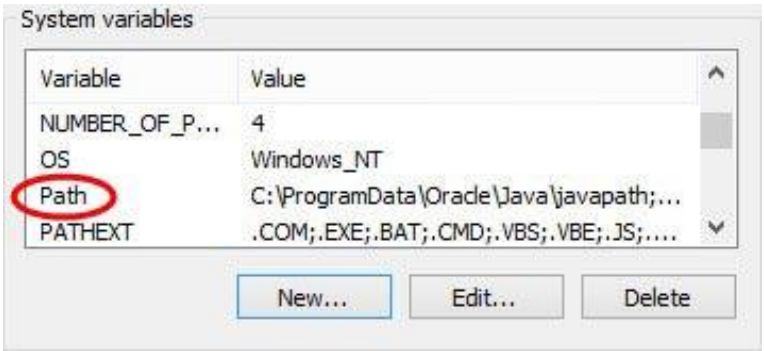
دوباره پنجره ی جدیدی ظاهر خواهد گشت که در قسمت system variables روی دکمه ی New کلیک کنید



با این کادر یک متغیر جدید ایجاد میشود که بایستی مقدار دهی کنید

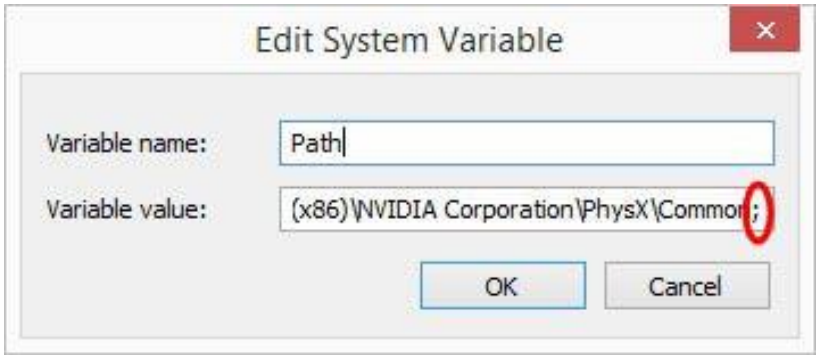


پنجره ظاهر شده را مانند رو به رو پر کنید(با این کار متغییر را مقداردهی می کنید). در قسمت Value مسیری که قبلا مشخص کرده اید را بنویسید



بعد از این مرحله باید مسیر دایرکتوری bin را در متغیر سیستمی PATH قرار دهیم. برای این کار

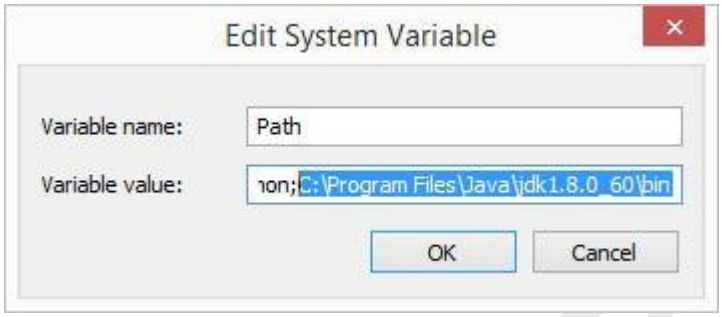
دوباره در قسمت system variables به دنبال متغیر path بگردید.



Path را انتخاب کرده و روی Edit کلیک کنید.

در قسمت value به آخر خط رفته و یک سِمی کالِن قرار دهید .

بعد از قرار دادن سمی کالن، نشانگر ماوس خود را بعد از سمی کالن قرار دهید و بعد مسیر دایرکتوری bin را در این قسمت paste کنید.



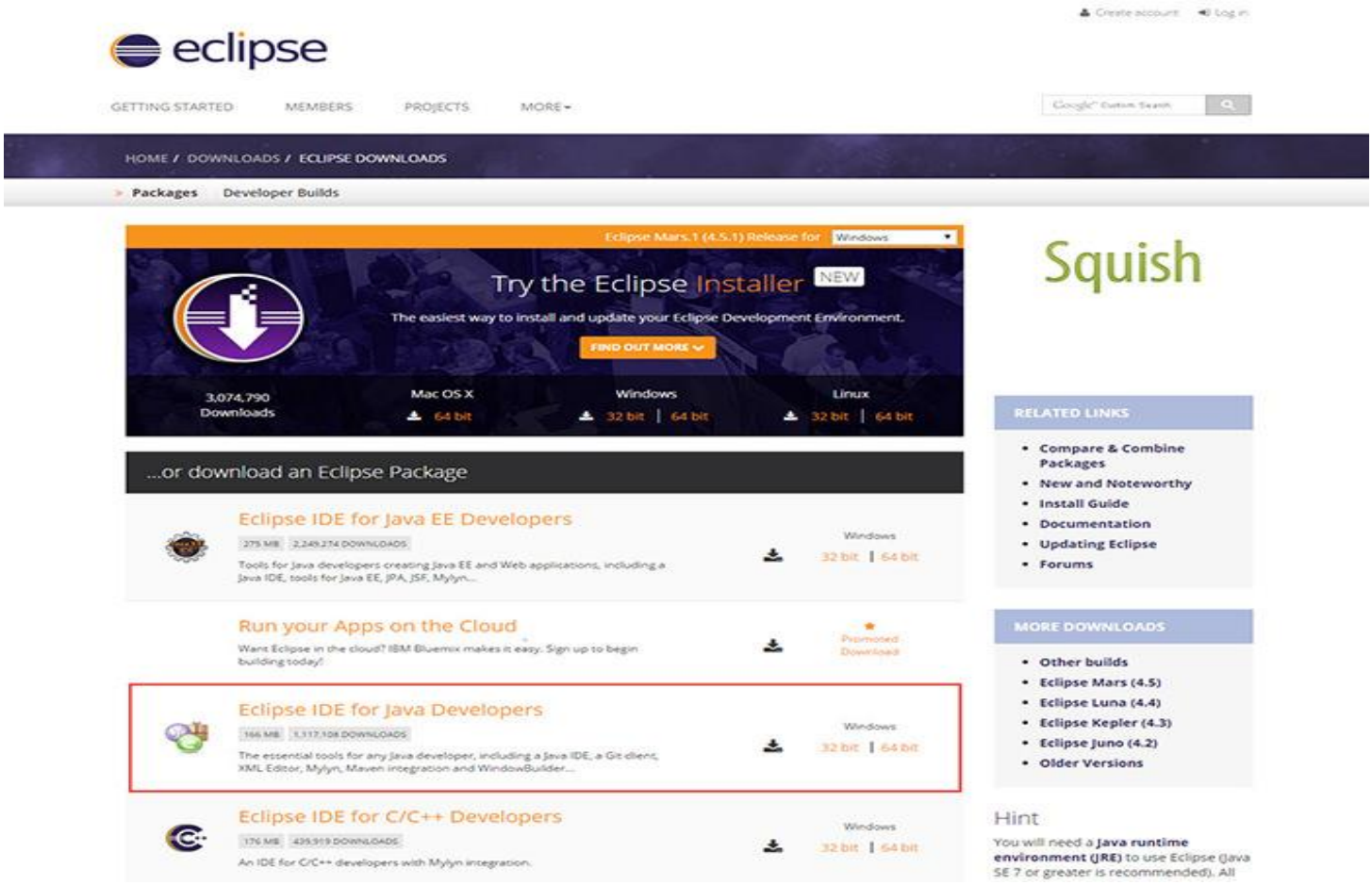
در نهایت رو OK کلیک کنید. و کامپیوتر خود را رستارت کنید.

نصب eclipse

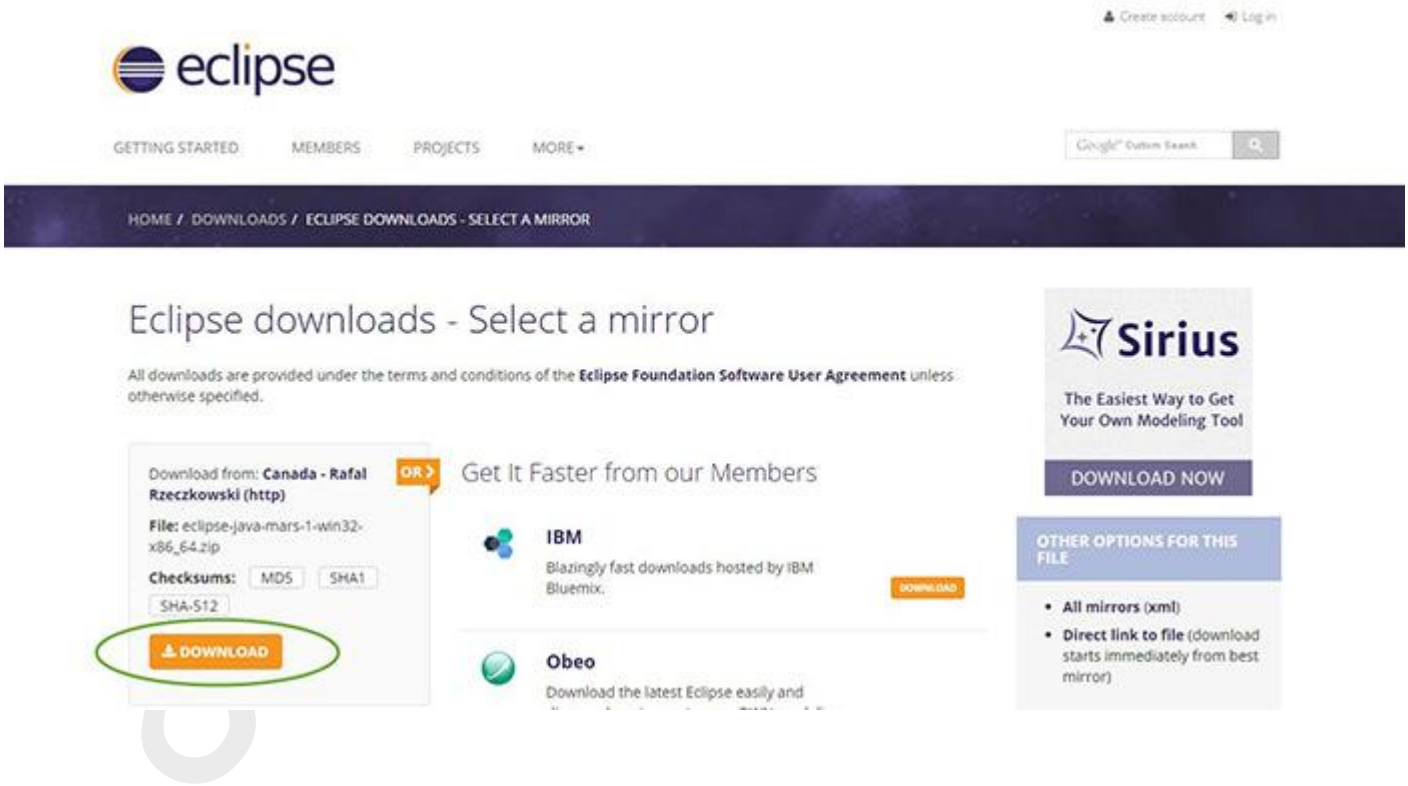
در آخر نوبت به نصب برنامه ی eclipse میرسه که میتونید اون رو از سایت زیر دانلود و نصب کنید .آموزش دانلود از سایت را در زیر قرار دادم.

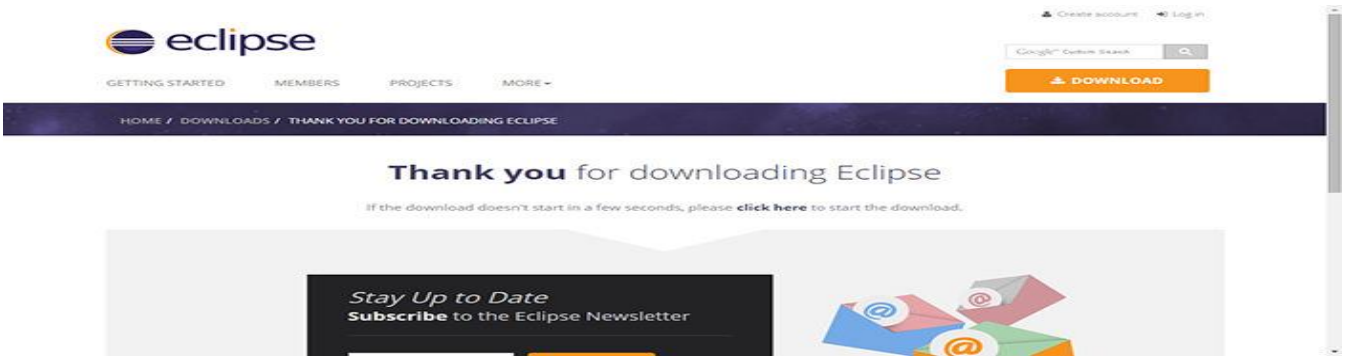
https://eclipse.org/downloads/


eclipse
GETTING STARTED
MEMBERS
PROJECTS
MORE
DOWNLOAD
Eclipse Is...
An amazing open source community of Tools, Projects and Collaborative Working Groups. Discover what we have to offer and join us.
DISCOVER
IDE & Tools
Community of Projects
Collaborative Working Groups


eclipse
GETTING STARTED
MEMBERS
PROJECTS
MORE
HOME / DOWNLOADS / ECLIPSE DOWNLOADS
Packages
Developer Builds
Eclipse Mars.1 (4.5.1) Release for Windows
Try the Eclipse Installer
NEW
The easiest way to install and update your Eclipse Development Environment.
FIND OUT MORE
3,074,790 Downloads
Mac OS X 64 bit
Windows 32 bit | 64 bit
Linux 32 bit | 64 bit
...or download an Eclipse Package
Eclipse IDE for Java EE Developers
Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...
Windows 32 bit | 64 bit
Run your Apps on the Cloud
Want Eclipse in the cloud? IBM Bluemix makes it easy. Sign up to begin building today!
Promoted Download
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven integration and WindowBuilder...
Windows 32 bit | 64 bit
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers with Mylyn integration.
Windows 32 bit | 64 bit
Squish
RELATED LINKS
Compare & Combine Packages
New and Noteworthy
Install Guide
Documentation
Updating Eclipse
Forums
MORE DOWNLOADS
Other builds
Eclipse Mars (4.5)
Eclipse Luna (4.4)
Eclipse Kepler (4.3)
Eclipse Juno (4.2)
Older Versions
Hint
You will need a Java runtime environment (JRE) to use Eclipse (Java SE 7 or greater is recommended). All


eclipse
GETTING STARTED
MEMBERS
PROJECTS
MORE
DOWNLOAD
HOME / DOWNLOADS / PACKAGES / ECLIPSE IDE FOR JAVA DEVELOPERS
RELEASES
Neon Packages
Mars Packages
Luna Packages
Kepler Packages
Juno Packages
Indigo Packages
Helios Packages
Galileo Packages
Ganymede Packages
Europa Packages
All Releases
Eclipse IDE for Java Developers
Package Description
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven integration and WindowBuilder
This package includes:
Eclipse Git Team Provider
Eclipse Java Development Tools
Maven Integration for Eclipse
Mylyn Task List
Code Recommenders Tools for Java Developers
WindowBuilder Core
Eclipse XML Editors and Tools
Download Links
Windows 32-bit
Windows 64-bit
Mac OS X (Cocoa) 64-bit
Linux 32-bit
Linux 64-bit
Downloaded 1,117,108 Times
Checksums...
Bugzilla


eclipse
GETTING STARTED
MEMBERS
PROJECTS
MORE
HOME / DOWNLOADS / ECLIPSE DOWNLOADS - SELECT A MIRROR
Eclipse downloads - Select a mirror
All downloads are provided under the terms and conditions of the Eclipse Foundation Software User Agreement unless otherwise specified.
Download from: Canada - Rafal Rzeczkowski (http)
File: eclipse-java-mars-1-win32-x86_64.zip
Checksums: MD5 SHA1 SHA-512
DOWNLOAD
OR
Get It Faster from our Members
IBM
Blazingly fast downloads hosted by IBM Bluemix.
Obeo
Download the latest Eclipse easily and
Sirius
The Easiest Way to Get Your Own Modeling Tool
DOWNLOAD NOW
OTHER OPTIONS FOR THIS FILE
All mirrors (xml)
Direct link to file (download starts immediately from best mirror)

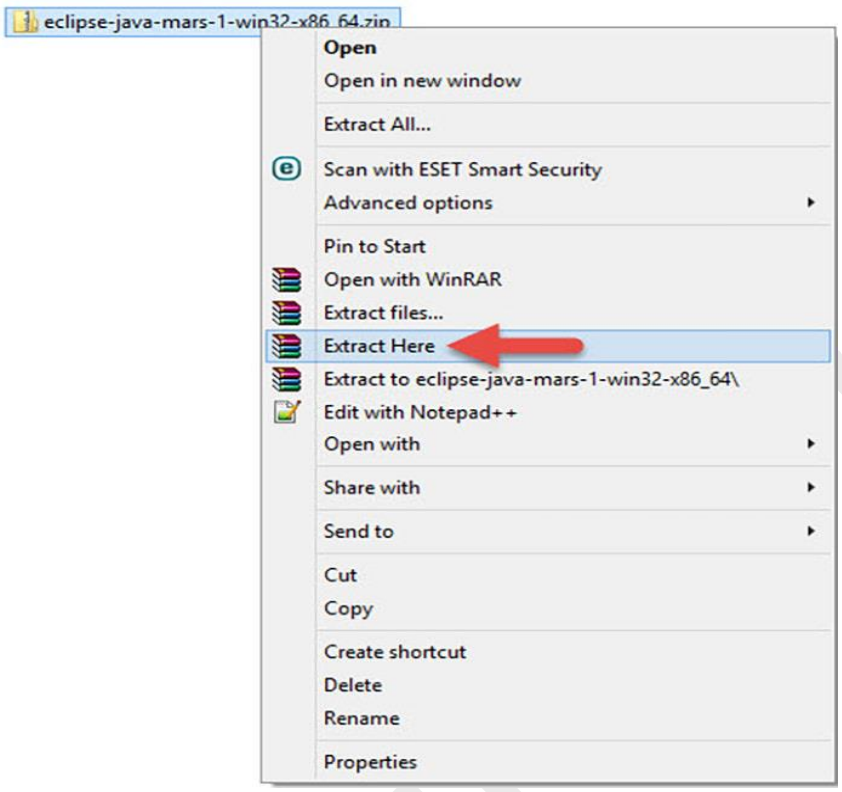

eclipse
GETTING STARTED
MEMBERS
PROJECTS
MORE
DOWNLOAD
HOME / DOWNLOADS / THANK YOU FOR DOWNLOADING ECLIPSE
Thank you for downloading Eclipse
If the download doesn't start in a few seconds, please click here to start the download.
Stay Up to Date
Subscribe to the Eclipse Newsletter

در تصویر شماره ی سه در بالا ، باید با توجه به سیستم عامل خود یک گزینه رو انتخاب کنید.

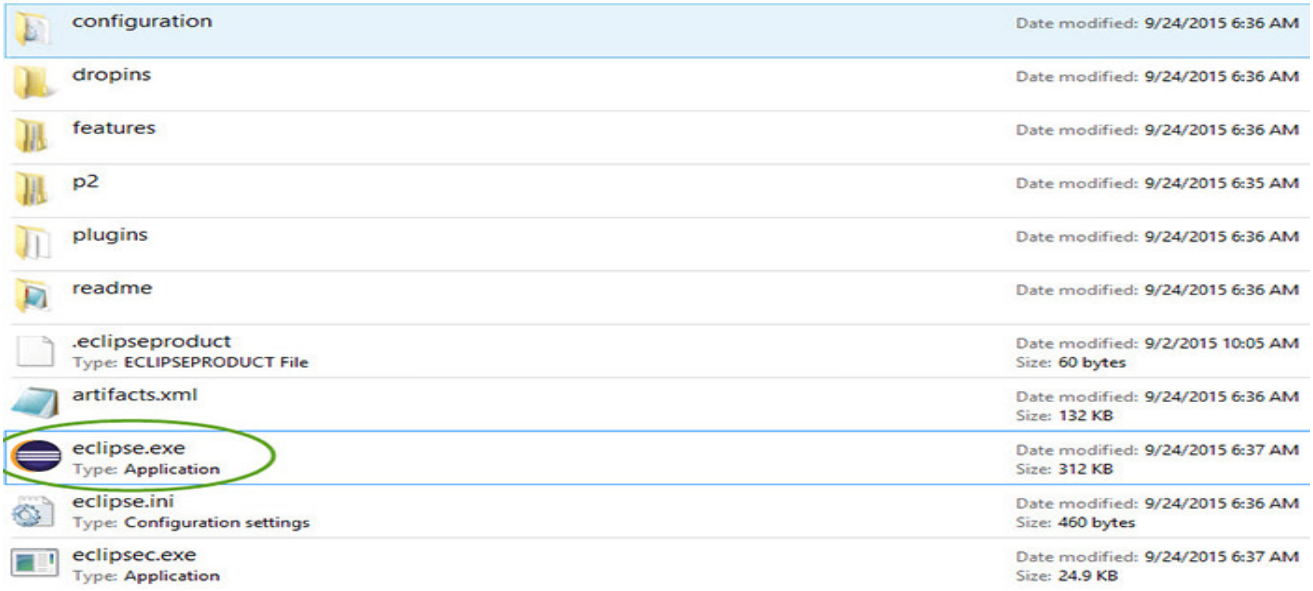
بعد از دانلود برنامه نوبت به نصب و اجرای آن میرسد.

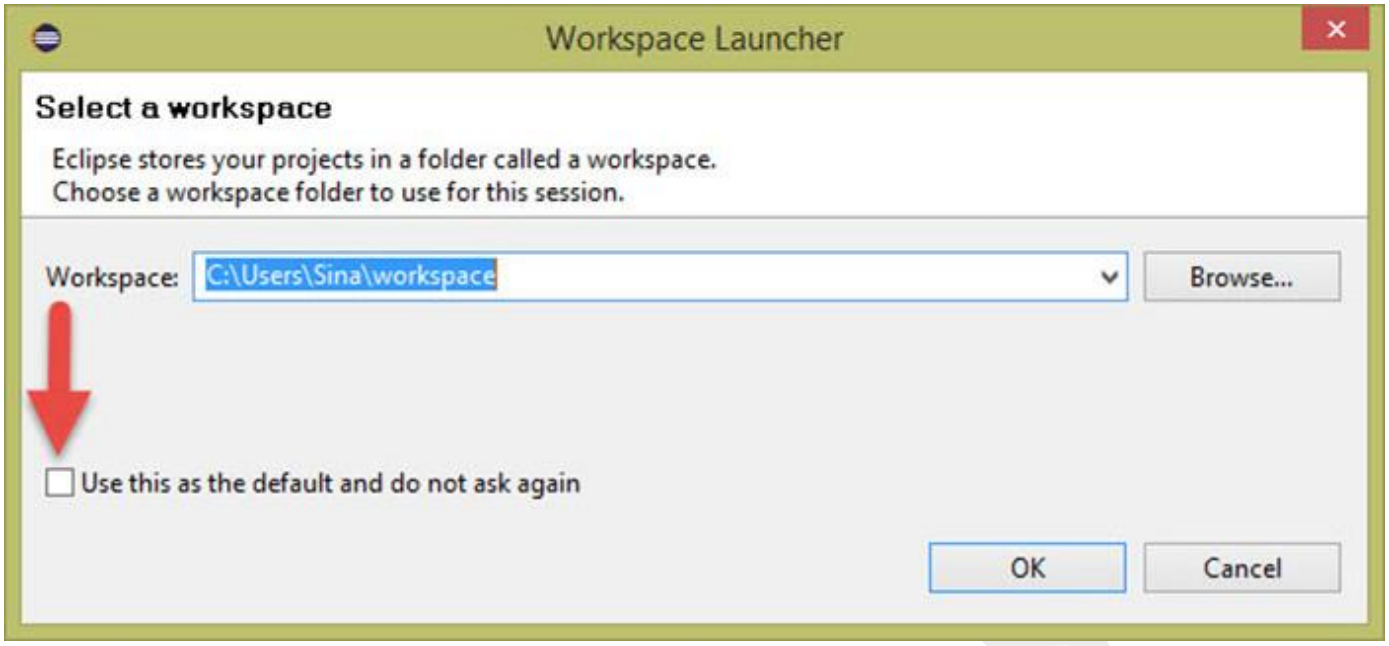
ما با استفاده از این برنامه ی ایکلیپس برنامه های نوشته شده ی جاوا را اجرا و تست خواهیم کرد

ابتدا فایل فشرده رو از حالت فشرده خارج کنید.



فایل eclipse رو اجرا کنید.

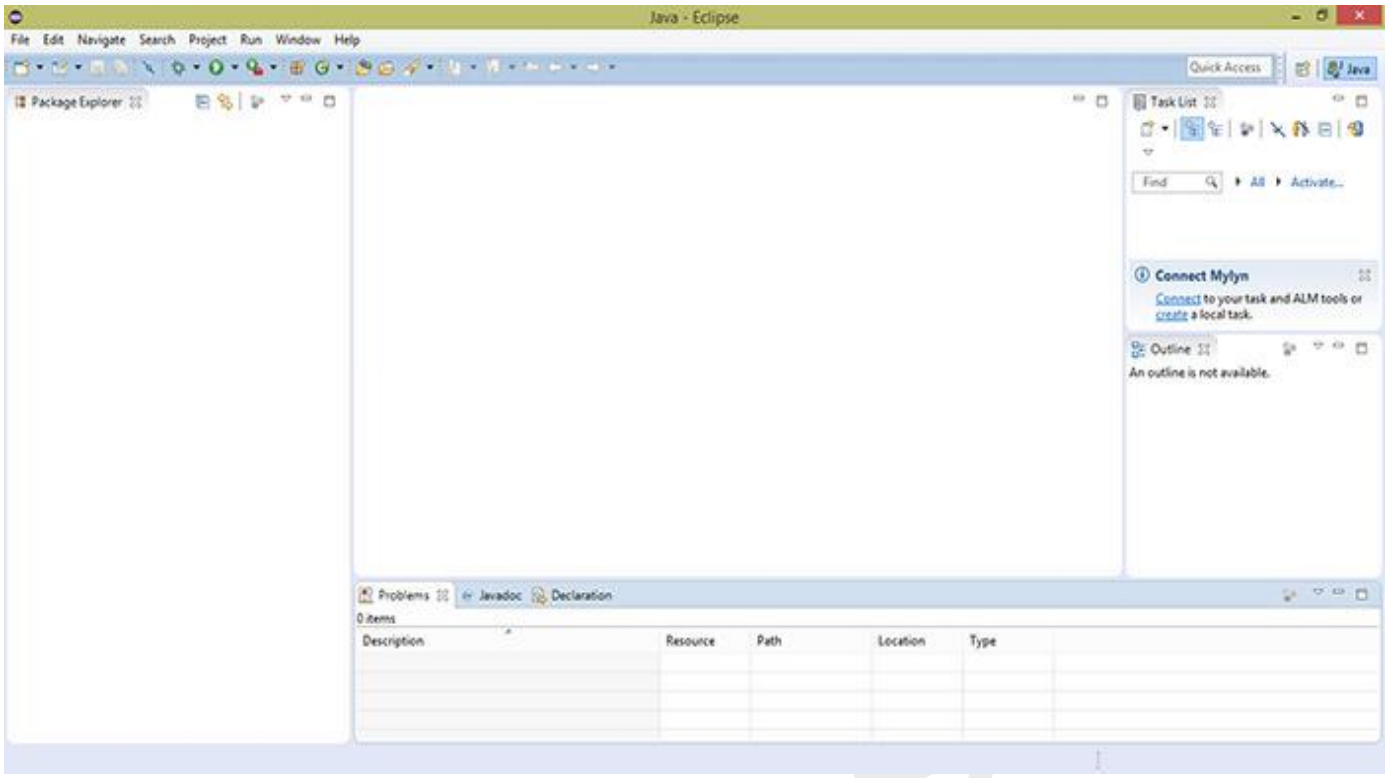




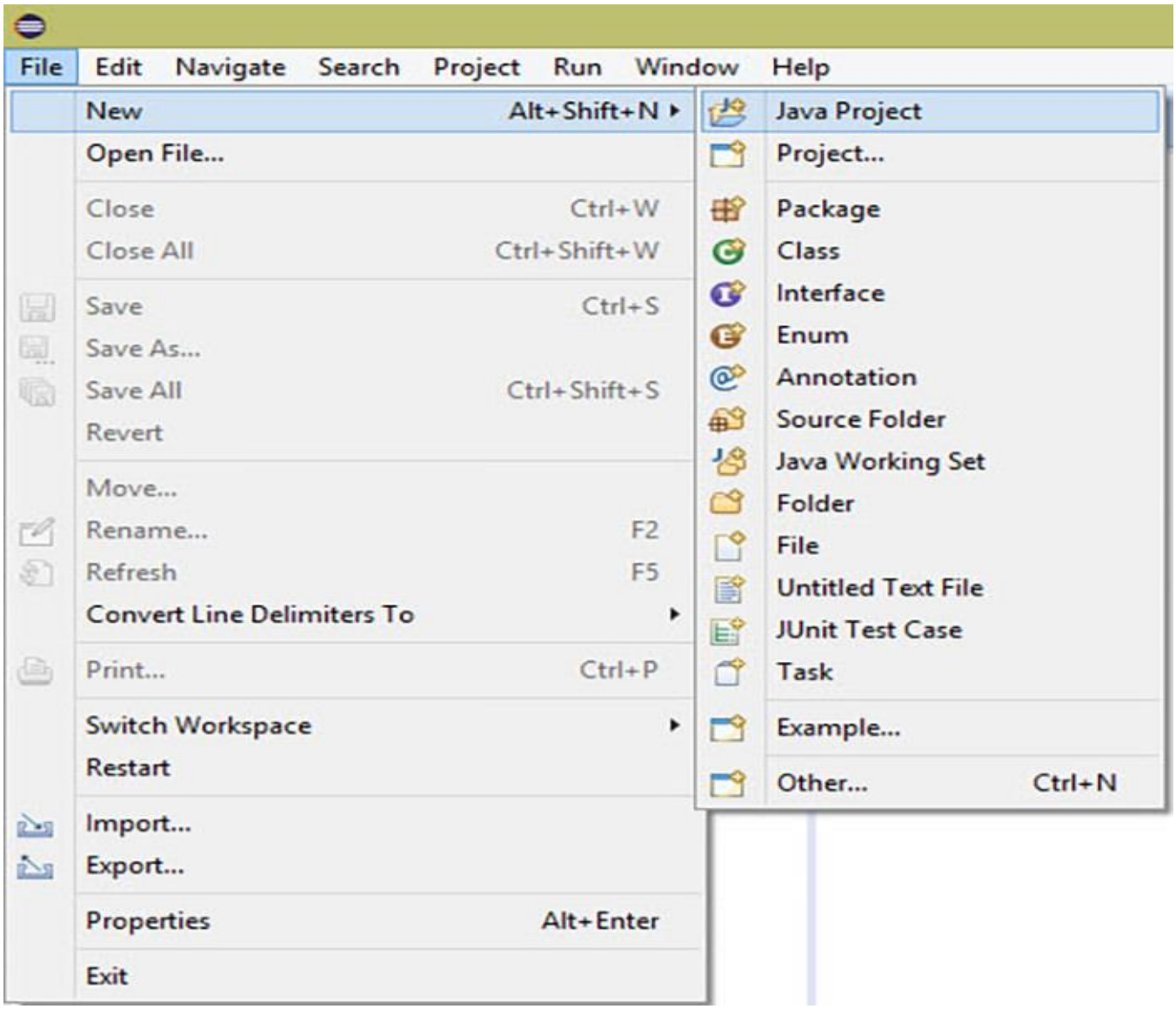
در کادر بالا مسیری که پروژه های جاوا ذخیره خواهند شد رو مشخص کنید.

همچنین اگر گزینه ی پایینی را تیک بزنید آنگاه دیگر این کادر نمایش داده نخواهد شد و مسیری که برای ذخیره ی برنامه ها مشخص کرده اید به عنوان پیش فرض تعبین میگردد.

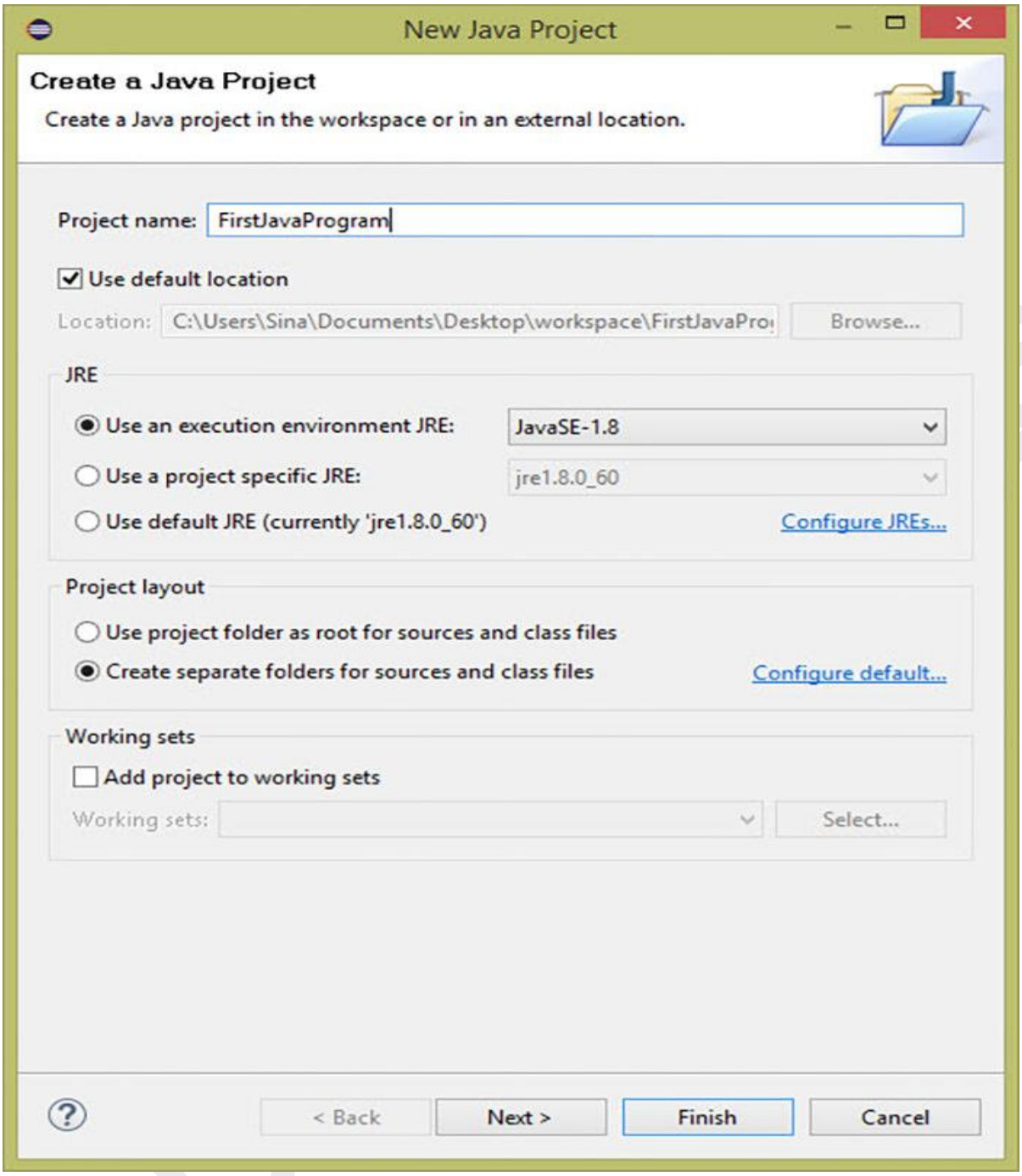
زمانی که فایل eclipse رو اجرا کردید صفحه خوشامد گویی مانند بالا ظاهر خواهد گشت. رو دکمه ی Worckbench که با فلش سبز مشخص شده است کلیک کنید.



حال نوبت آن است که اولین پروژه جاوای خودمان را ایجاد کنیم. برای این من مانند تصویر زیر عمل کنید.

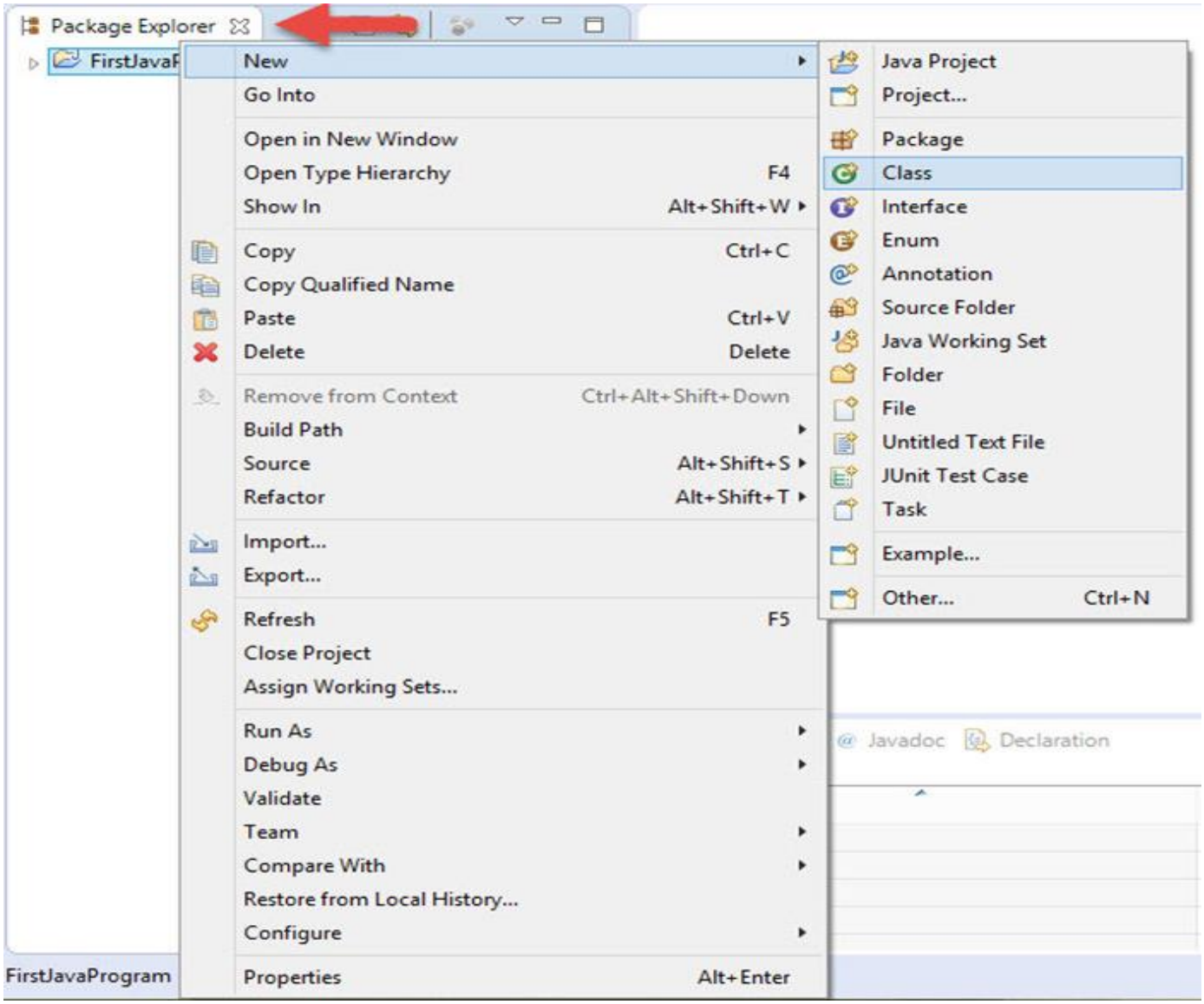


بعد از انتخاب Java Project پنجره ی زیر نمایش داده خواهد شد . در کادر بالا نام پروژه ی خود را بنویسید که من FirstJavaProgrm را برای نام برنامه نوشتم



سپس روی Finish کلیک کنید.

حال باید یک کلاس برای پروژه ی خود ایجاد کنیم. برای اینکار روی پروژه ی خود یعنی FirstJavaProgrm راست کلیک کنید و مانند تصویر زیر عمل کنید



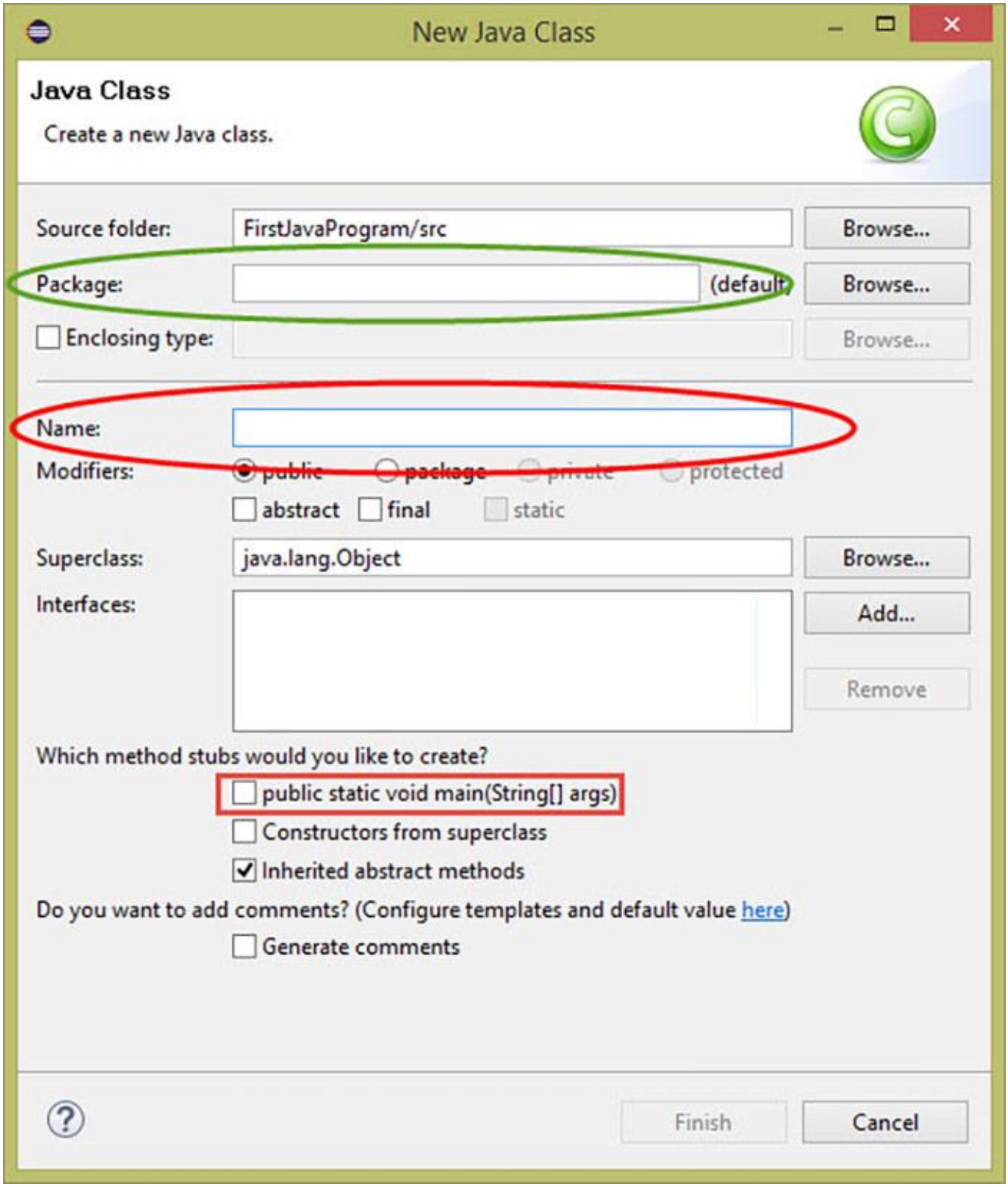
توجه : قسمتی که با فلش قرمز مشخص شده را Package Explorer نامیده میشود . تمام پروژه هایی که ما ایجاد می کنیم در این قسمت قرار خواهد گرفت.

بعد از کلیک روی class با پنجره ی زیر روبه رو خواهیم شد.

در بیضی سبز رنگ Package پکیج مشخص میشود. که بعدا در این مورد توضیح خواهم داد.

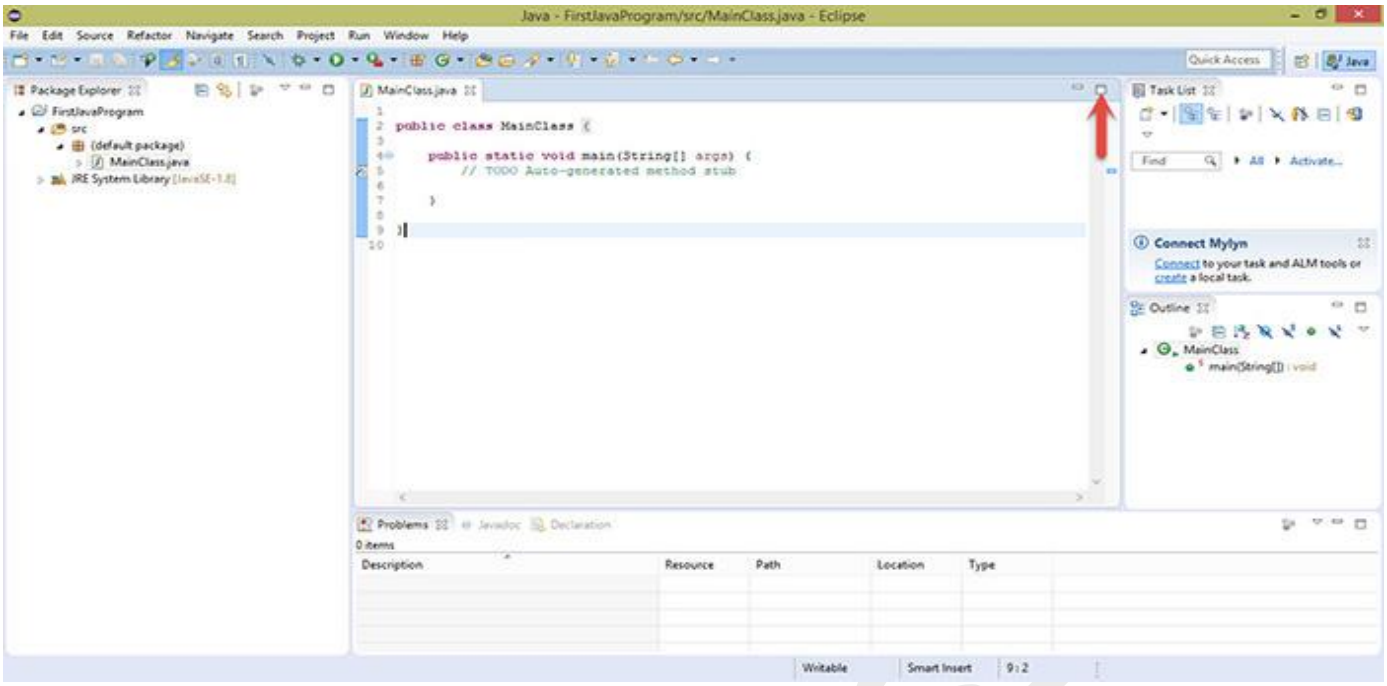
در قسمت Name که ب بیضی قرمز مشخص شده نام کلاس خود را می نویسیم.

در قسمت آخر که با مستطیل قرمز مشخص شده است اگر تیک دار باشد Eclipse به ظور خودکار متد main را در کلاس های ما پیاده سازی میکند.



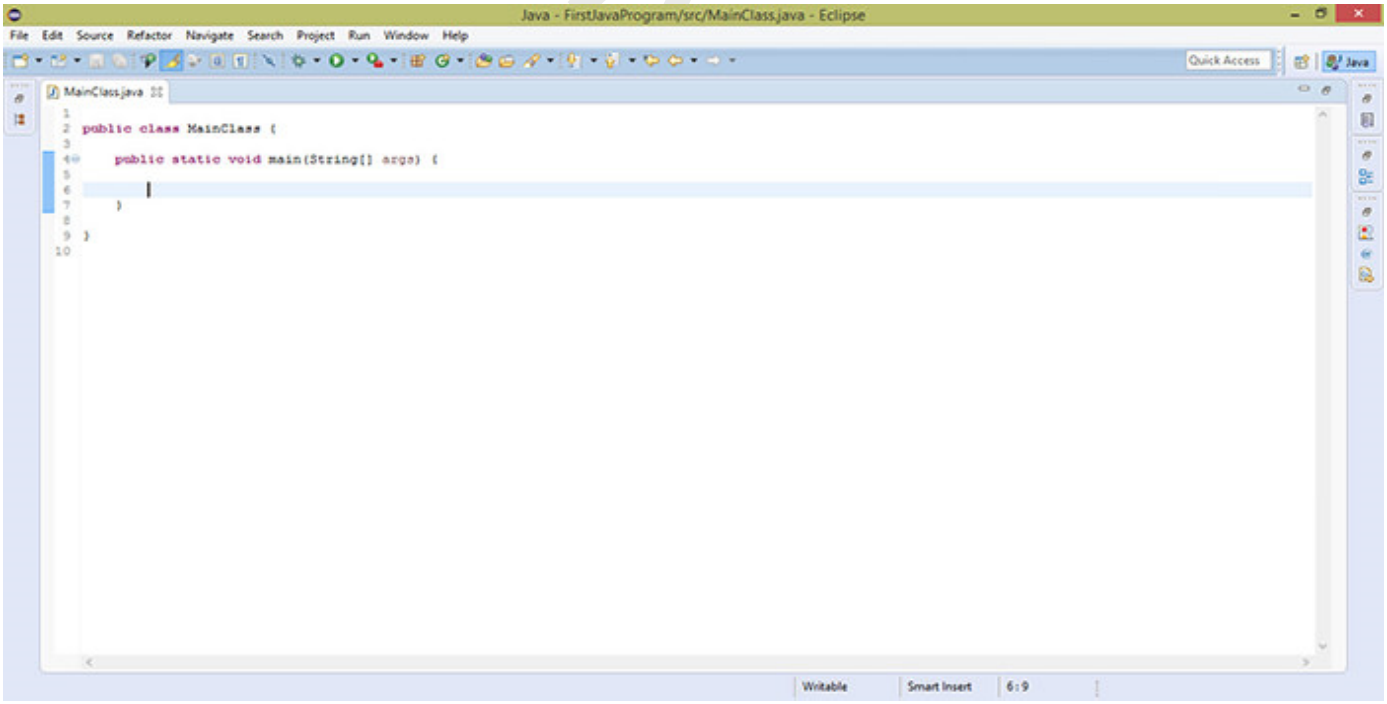
روی Finish کلیک کنید.

بعد از ساخته شدن کلاس با تصویر زیر مواجه خواهید شد

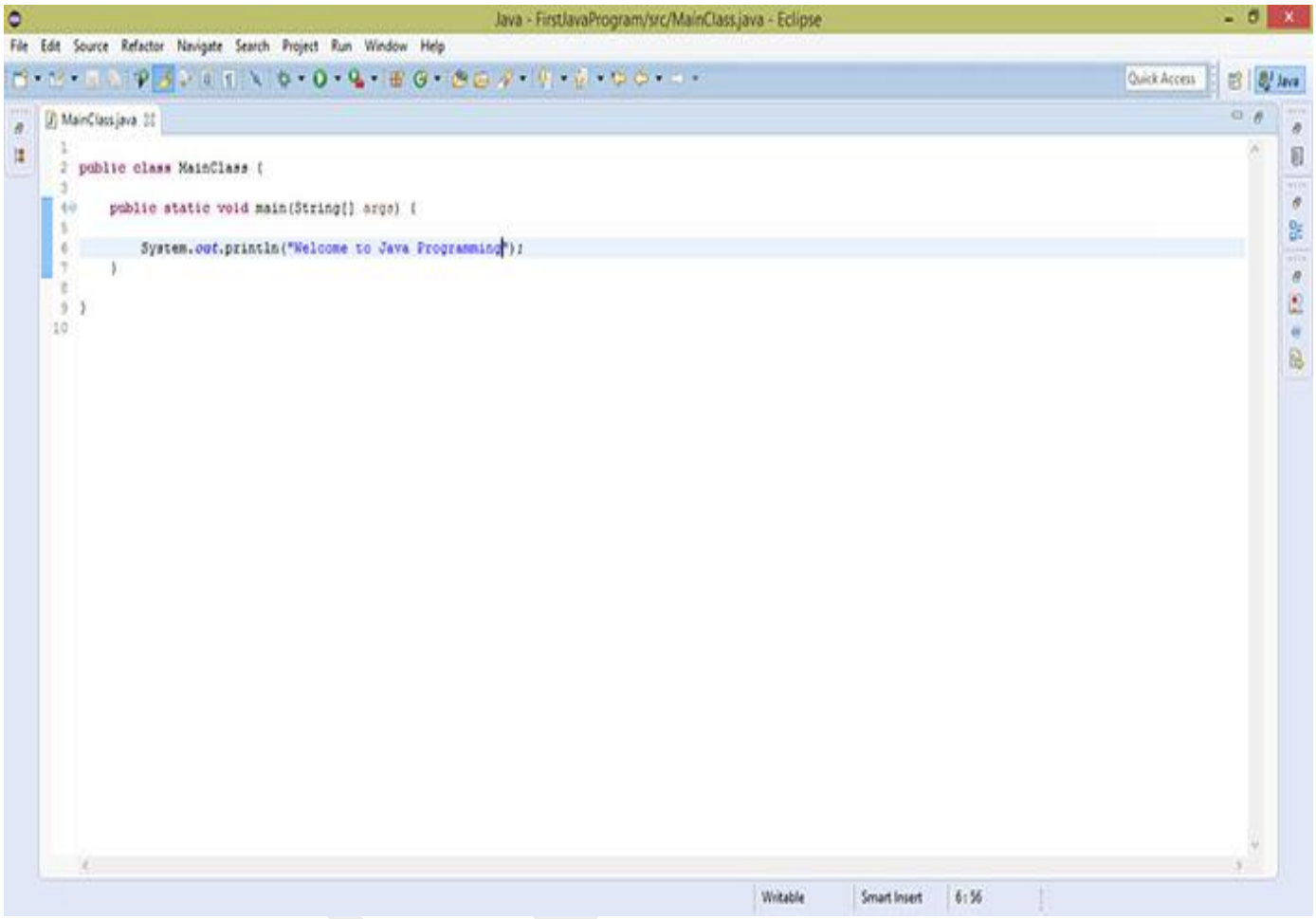


روی دکمه ی Maximize که با فلش قرمز مشخص شده کلیک کنید تا فضای بیشتری برای کد نویسی داشته باشیم.

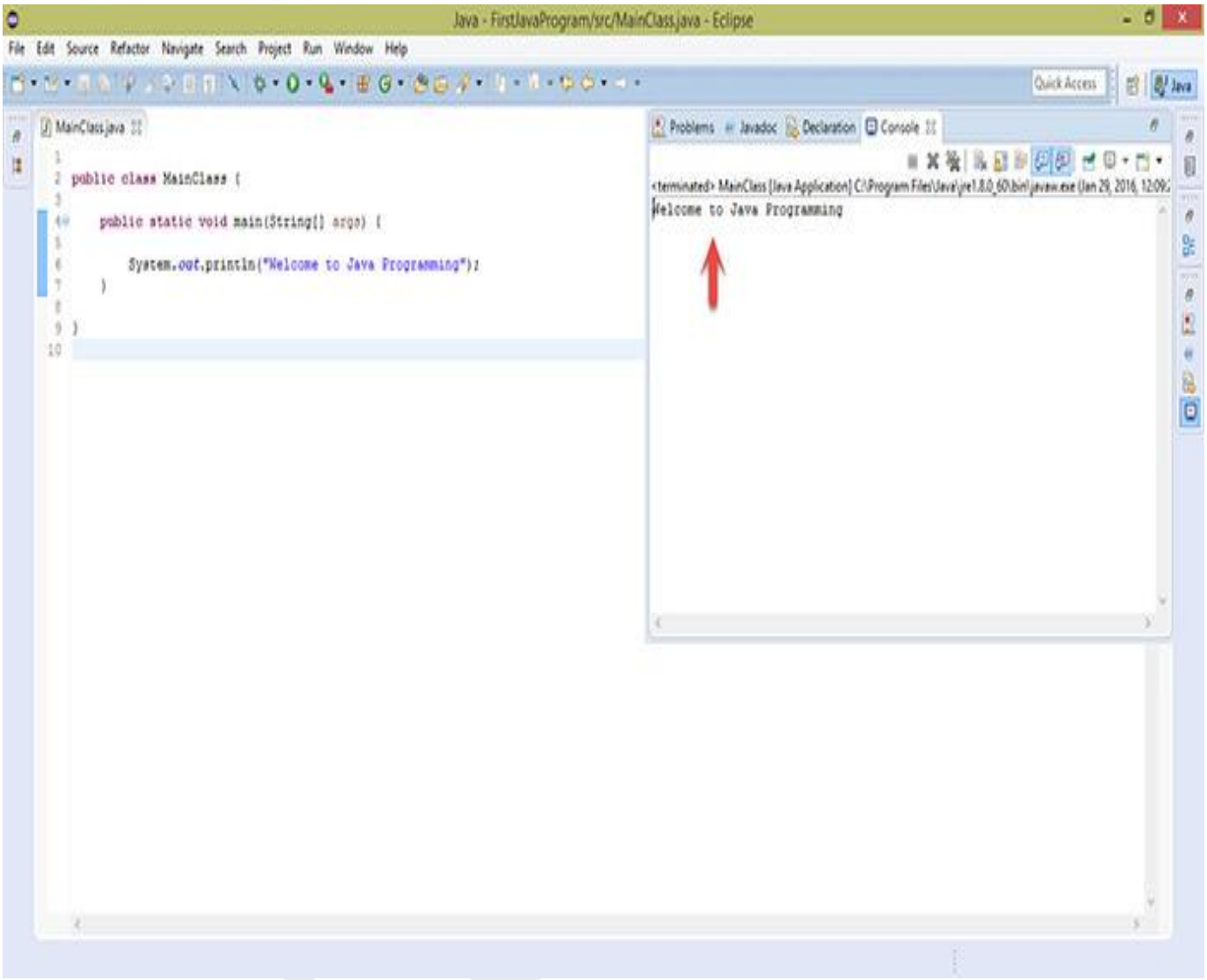
حال یک برنامه ی نمونه ایجاد میکنیم تا مطلب کاملا جا بیافتد. البته نگران نباشید این برنامه فقط برای آشنایی شما با چگونگی ایجاد و اجرای یک برنامه ی جاوا در ایکلیپس است. و اگر کد های نوشته شده رو نمی فهمید اصلا مهم نیست!!!!! چون در فصلهای بعدی همه ی کدها را ریز به ریز توضیح خواهم داد.



به عنوان مثال میخواهیم برنامه ی ما عبارت Welcome to Java Programming رو چاپ کند بنابرین باید از دستور System.out.println() در جاوا استفاده کنیم.



در نهایت کلیدهای ترکیبی CTRL + S رو بزنید تا برنامه ذخیره شود. در دست آخر کلید های CTRL + F۱۱ رو نگه دارید تا برنامه اجرا گردد. حاصل اجرای برنامه رو در تصویر پایین می بینیم.


Java - FirstJavaProgram/src/MainClass.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
MainClass.java
public class MainClass {
public static void main(String[] args) {
System.out.println("Welcome to Java Programming");
}
}
Problems Javadoc Declaration Console
<terminated> MainClass [Java Application] C:\Program Files\Java\jre1.8.0_60\bin\javaw.exe (Jan 29, 2016, 12:09:2
Welcome to Java Programming

فصل ۲

متغیر:

به طور خلاصه میتوان متغیررا اینگونه تعریف نمود :

"متغیر، اسمی برای مکانی از حافظه ی کامپیوتر است، که داده ها در آنجا ذخیره میشوند."

انواع داده ها در جاوا :

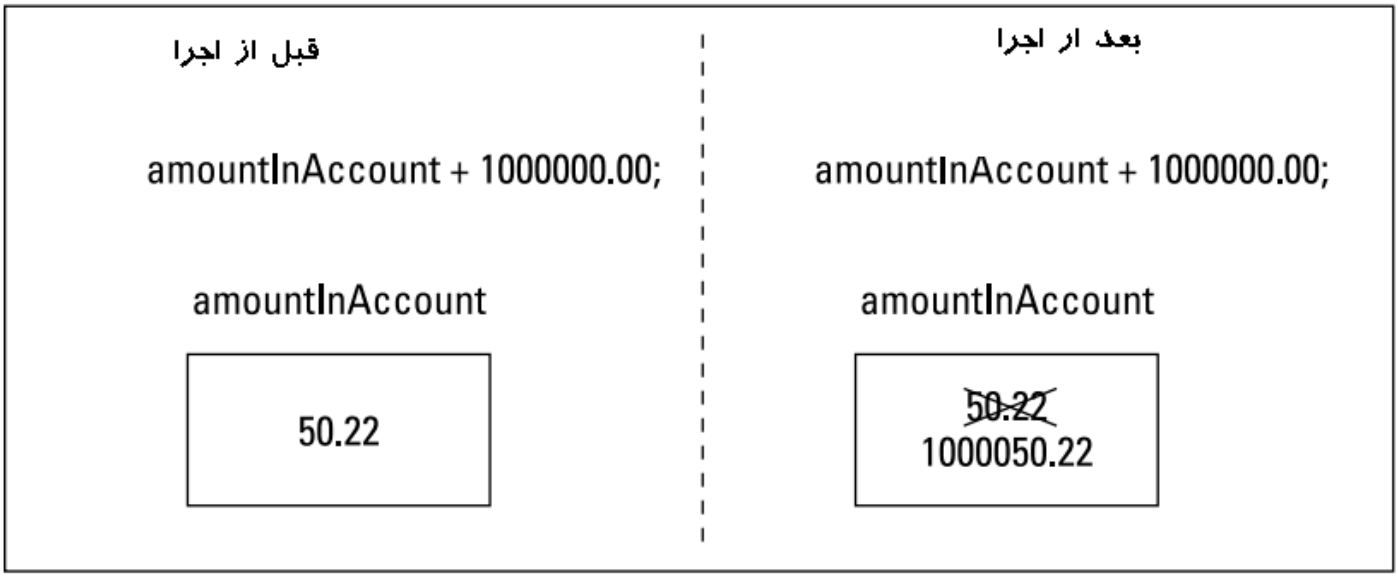
به طور دقیق، جاوا دارای هشت نوع داده ی اصلی می باشد، که جدول ۱-٤ لیست کامل انواع اصلی در جاوا را نمایش میدهد

انواع داده های اصلی جاوا — جدول ۱-٤

نام نوع داده	لیترال نمونه	محدوده ی مقادیر
انواع عددی صحیح		
byte	(byte)42	–128 to 127
short	(short)42	–32768 to 32767
int	42	–2147483648 to 2147483647
long	42L	–9223372036854775808 to 9223372036854775807
انواع عددی دیسیمال		
float	42.0F	-3.4×10^{38} to 3.4×10^{38}
double	42.0	-1.8×10^{308} to 1.8×10^{308}
انواع کاراکتری		
char	'A'	Thousands of characters, glyphs, and symbols
انواع منطقی		
boolean	true	true, false

تخصیص مقدار به متغیر های دارای مقدار اولیه :

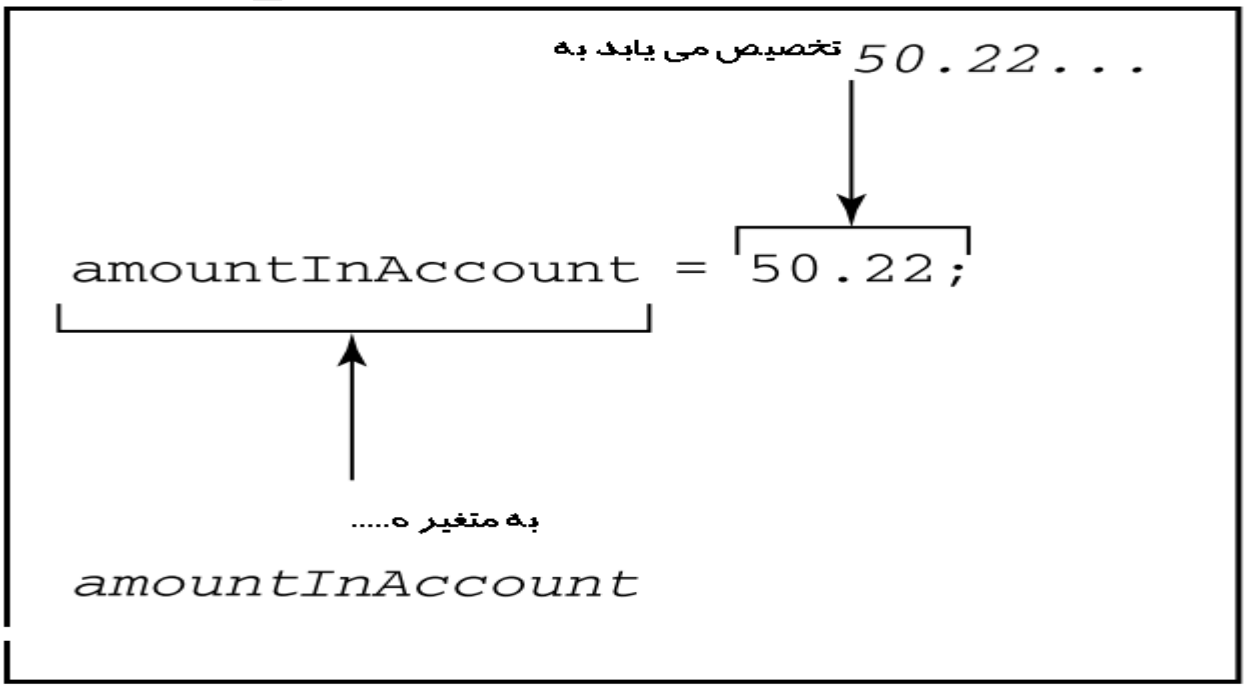
برای اینکه کاملا درک نمایید که چه اتفاقی برای متغیری که دارای مقدار اولیه میباشد هنگام تخصیص مقدار جدید می افتد به شکل ۱-٤ دقت نمایید:



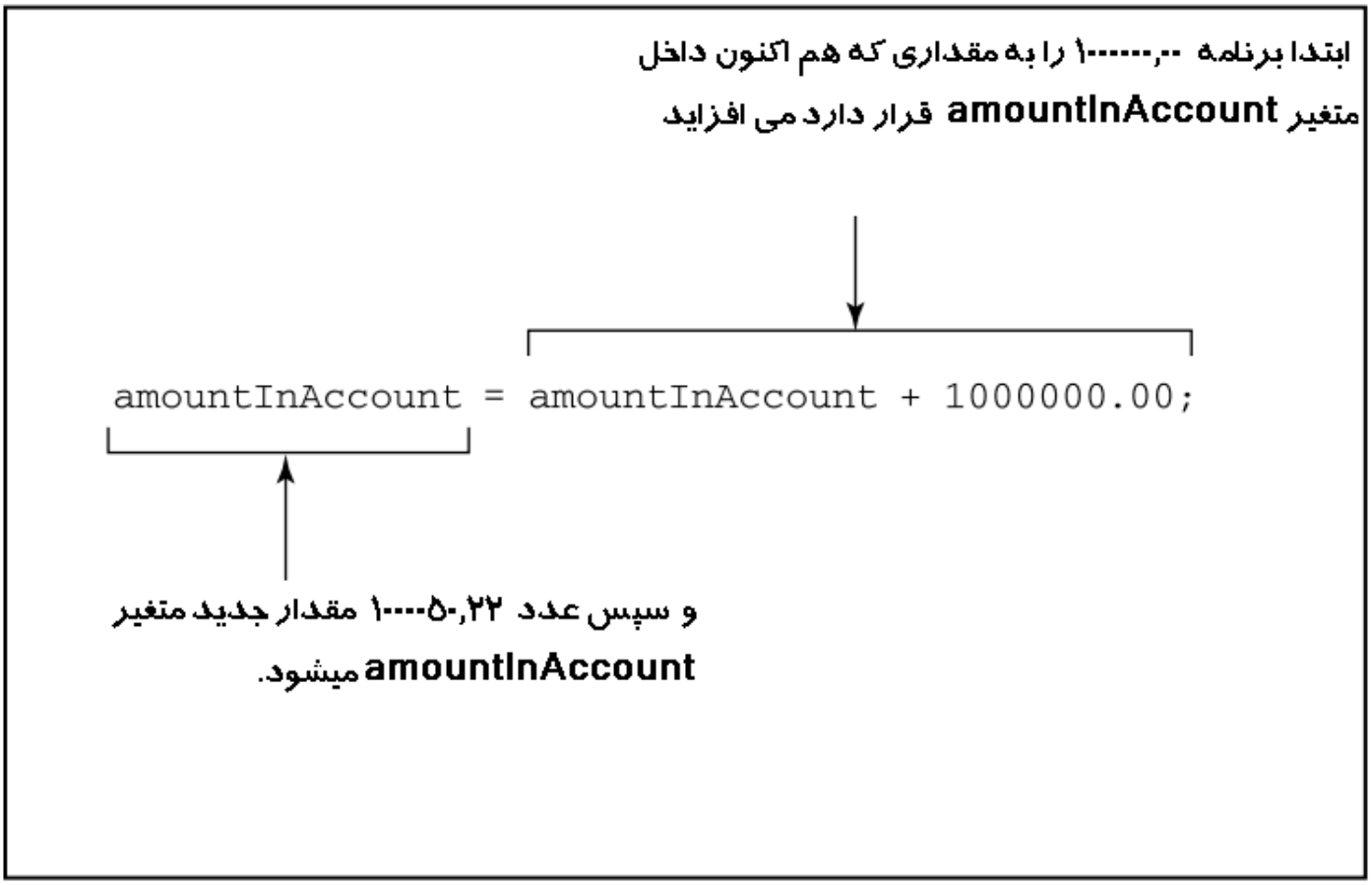
توجه : متغیر تغییر میکند ولی یک لیترال تغییر ناپذیر است.

عبارات تخصیص (Assignment Statements)

در تخصیص، شما یک مقدار را به چیزی که در بیشتر مواقع متغیر است اختصاص میدهید در موقع تخصیص شما باید از عملگر = استفاده نمایید. بشکل ۲-نشان دهنده ی همین موضوع است :



شکل ۳-٤ چگونگی افزایش مقدار متغیر amountInAccount را توضیح میدهد، که مقدار قبلی متغیر را به اندازه ی ۱۰۰۰۰۰۰ واحد افزایش می دهد.



(با توجه به شکل ۲-٤ مقدار اولیه متغیر amountInAccount مقدار ۵۰,۲۲ بوده است) .

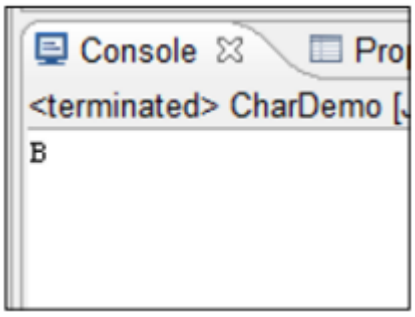
نوع داده ی *char* :

نوع داده ای که در جاوا برای ذخیره سازی کاراکتر ها استفاده میشود ***char*** نام دارد.

مثال ٤- ٤ یک برنامه ساده که از نوع داده ی ***char*** استفاده میکند را نشان میدهد:

```
public class CharDemo {
public static void main(String args[]) {
charmyLittleChar = 'b';
charmyBigChar = Character.toUpperCase(myLittleChar);
System.out.println(myBigChar);
 }
}
```

خروجی حاصل از اجرای مثال ٤-٤ :



در خط ۳ متغیر myLittleCharتعریف شده و در ضمن با مقدار *b*مقدار دهی شده است، دقت کنید که *b*میان دو تا تک کوتیشن قرار گرفته است.در جاوا همیشه لیترال های *char* میان تک کوتیشن قرار میگیرند.

در خط ٤ متغییر *myBigChar* تعریف میگردد، و در مقدار دهی به آن ، برنامه یک متد API را که *Character.toUpperCase* نام دارد فراخوانی میکند.متد *Character.toUpperCase* همانطور که از نامش میتوان حدس زد. حرف را میگیرد و معادل *UpperCase* (حروف-بزرگ) آن را باز میگرداند.که در اینجا متغییر myLittleChar*را که حاوی لیترال b میباشد را گرفته ، و معادل حرف بزرگ*(UpperCase)آن را که *B*میباشد باز میگرداند. این Bبازگزدانده شده در *myBigChar*ذخیره میگردد.

توجه:

```
char mychar='Hadi';   //Error
```

اگر سعی کنید کدی مانند بالا بنویسید ، با خطا روبرو خواهید شد شما نمیتوانید، در یک زمان بیش از یک حرف در در متغیری از نوع *char* ذخیره کنید. اگر میخواهید کلمه یا جمله ای را در جاوا ذخیره کنید به چیزی به نام ***String*** نیاز خواهید داشت. (در مورد *string* در فصل های بعدی توضیح داده شده است).

نوع boolean :

یک متغیر نوع Boolean فقط دو مقدار را ذخیره میکند : true یا false .

مثال ٥-٤ طریقه ی استفاده از متغیر بولین را روشن میسازد.:

```
public class ElevatorFitter2 {
public static void main(String args[]) {
System.out.println("True or False?");
System.out.println("You can fit all ten of the");
System.out.println("Brickenchickerdectuplets");
System.out.println("on the elevator:");
System.out.println();
intweightOfAPerson = 150;
intelevatorWeightLimit = 1400;
intnumberOfPeople =
elevatorWeightLimit / weightOfAPerson;


booleanallTenOkay = numberOfPeople>= 10;
System.out.println(allTenOkay);
   }
}
```

```
True or False?
You can fit all ten of the
Brickenchicker dectuplets
on the elevator:

false
```

خروجی حاصل از اجرای مثال ٥-٤ :

در مثال بالا در خط یازدهم، متغیر allTenOkay از نوع boolen می باشد.برای یافتن ارزش (مقدار) allTenOkay برنامه‌چک میکند که آیا numberOfPeople بزرگتر تا مساوری ده است یا نه ؟

(سمبل =< نماد بزرگتر یا مساوی در جاوا میباشد).

numberOfPeople >= 10 یک *عبارت* است. ارزش این عبارت به مقدار ذخیره شده در numberOfPeople بستگی دارد. اما شما با نگاه کردن به کد برنامه میتوانید بفهمید مه مقدار numberOfPeople بزرگتر مساوی ده نیست. بنابراین ارزش عبارت numberOfPeople >= 10 برابر با false خواهد بود.

نکته : هر قسمت از برنامه های جاوا که ارزش (value) داشته باشد یک عبارت است.

در خط هفتم ، از System.out.println() استفاده شده ، درحالی که هیچ چیزی درون پارانتز های آن وجود ندارد، این باعث میشود جاوا یک خط خالی ایجاد کند.

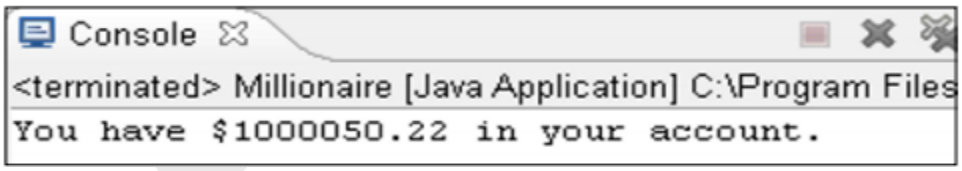
نوع داده های اعشاری در جاوا :

جاوا داری دو نوع داده اعشاری اصلی می باشد: double و float . تفاوت اساسی این دونوع این در این است که double دارای دقتی مضاعف نسبت به float می باشد و میتوانید اعداد اعشاری بزرگتری را در خود جا دهد. برای مشاهده ی محدوده ی مقادیر به جدول ۱-٤ رجوع کنید. برای روشن شدن مطلب به مثال ۲-٤

```
public class Millionaire {
public static void main(String args[]) {
doubleamountInAccount;
amountInAccount = 50.22;
amountInAccount = amountInAccount + 1000000.00;
System.out.print("You have $");
System.out.print(amountInAccount);
System.out.println("in your account.");
   }
}
```

در زیر توجه کنید:

خروجی مثال ***4-2 :***



در خط سوم ، متغیر amountInAccount از نوع double تعریف شده است. در خط پنجم مقدار قبلی amountInAccount با ***1000000*** جمع شده و حاصل داخل amountInAccount قرار گرفته است.

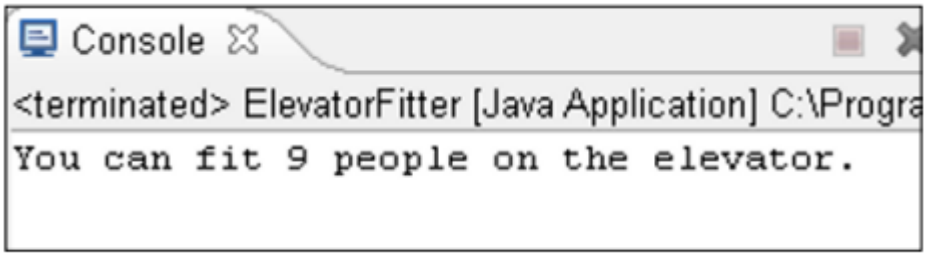
توجه کنید که F موجود در انتهای لیترال های ، برای مشخص کردن اینکه اینها از نـوع float هسـتند بـه کـار رفته اند، و این مقادیر از نوع هگزادیسیمال نمی باشند.

نوع داده ای صحیح در جاوا :

جاوا دارای چهار نوع داده ای نوع صحیح اصلی را پشتیبانی میکند. محدوده مقادیر این انواع داده ای را میتوانید در جدول ۱-٤ مشاهده نمایید. توجه داشته باشید که در مواردی که مقدار متغیر در طول برنامه به طور مکرر تغییر میکند و یا از مقدار دقیق مقادیر این متغییر ها در طول اطلاع ندارید ، بهتر است از نوع داده ای را استفاده کنید که به اندازه ی کافی بزرگ باشد تا احتمال سریز (overflow) را به حداقل برسانید. جهت فهم بهتر به مثال ۳-٤ دقت نمایید

```
public class ElevatorFitter{
public static void main(String args){ ([]
intweightOfAPerson;
intelevatorWeightLimit;
intnumberOfPeople;
weightOfAPerson = 150;
elevatorWeightLimit = 1400;
numberOfPeople =
elevatorWeightLimit / weightOfAPerson;
System.out.print("You can fit");
System.out.print(numberOfPeople);
System.out.println("people on the elevator.");
  }
}
```

خروجی حاصل از اجرای برنامه ی بالا:



در خط سوم متغییر weightOfAPersonاز نوع intتعریف گردیده. سپس متغیر های elevatorWeightLimitو numberOfPeopleاز نوع intدر خطوط بعدی تعربف شده است.

در خط هشتم ابتدا داخل پارانتز حساب میشود، یعنی اول عبارت elevatorWeightLimit/PersonWeight محاسبه شده و سپس نتیجه ی نهایی در متغیر numberOfPeople ذخیره میگردد.

انوع مرجع(Reference Types)

با ترکیب چیزی های ساده شما عناصر پیچیده تری را به دست می آورید. به طور مشابه چند نوع اصلی در

```
importjavax.swing.JFrame;
public class ShowAFrame {
public static void main(String args[]) {
JFramemyFrame = new JFrame();
      String myTitle = "Blank Frame";
myFrame.setTitle(myTitle);
myFrame.setSize(300, 200);
myFrame.setDefaultCloseOperation
         (JFrame.EXIT_ON_CLOSE);
myFrame.setVisible(true);
   }
}
```

جاوا را میتوان با یکدیگر ترکیب کرد و انواع مرجع را به دست آورد. برای روشن سازی بهتر مطلب به مثال ٤-٦ را که در زیر آورده به دقت توجه کنید:

خروجی حاصل از برامه ٦-٤ :

در مثال ٦-٤ دو نوع مرجع تعریف گردیده است که هر دوی آنها در API جاوا تعریف شده اند.(اولی را که شما در بیشتر اوقات استفاده خواهید کرد) String نام دارد. نوع دیگر (که شما از آن در ساخت GUI ها استفاده خواهید کرد) JFrame نام دارد.

String گروهی از کاراکتر هاست، آن شبیه آن است که چندین مقدار char را در یک سطر داشته باشد. متغیر myTitleاز نوع String اعلان گردیده و "Blank Frame" به آن تخصیص داده شده است. توجه داشته باشید که کلاس String در API جاوا اعلان شده است.

در برنامه های جاوا ، علامت دابل کوتیشن در ابتدا و انتهای لیترال های String باید وجود داشته باشد.

JFrame در جاوا بسیار شبیه به windowمیباشد.(با این تفاوت که شما JFrameرا بجای window فراخوانی میکنید).

سعی نکنید تمام خطوط مثال ٦-٤ تفسیر کنید. تنها چیزی که باید از این مثال یادبگیرید، اعلان دو متغییر مرجع ، یعنی myFrameاز نوع JFrame ونیز myTitleاز نوع String میباشد.(در فصول بعدی در مورد دیگر خطوط این مثال توضیح داده خواهد شد).

نکته ی مهم دیگر آن هست که String و JFrame نام کلاس میباشند و در حالت *کلی* ***هر کلاس در جاوا یک نوع مرجع میباشد***.(در مورد کلاس های جاوا دو فصل های بعدی توضیح داده خواهد شد).

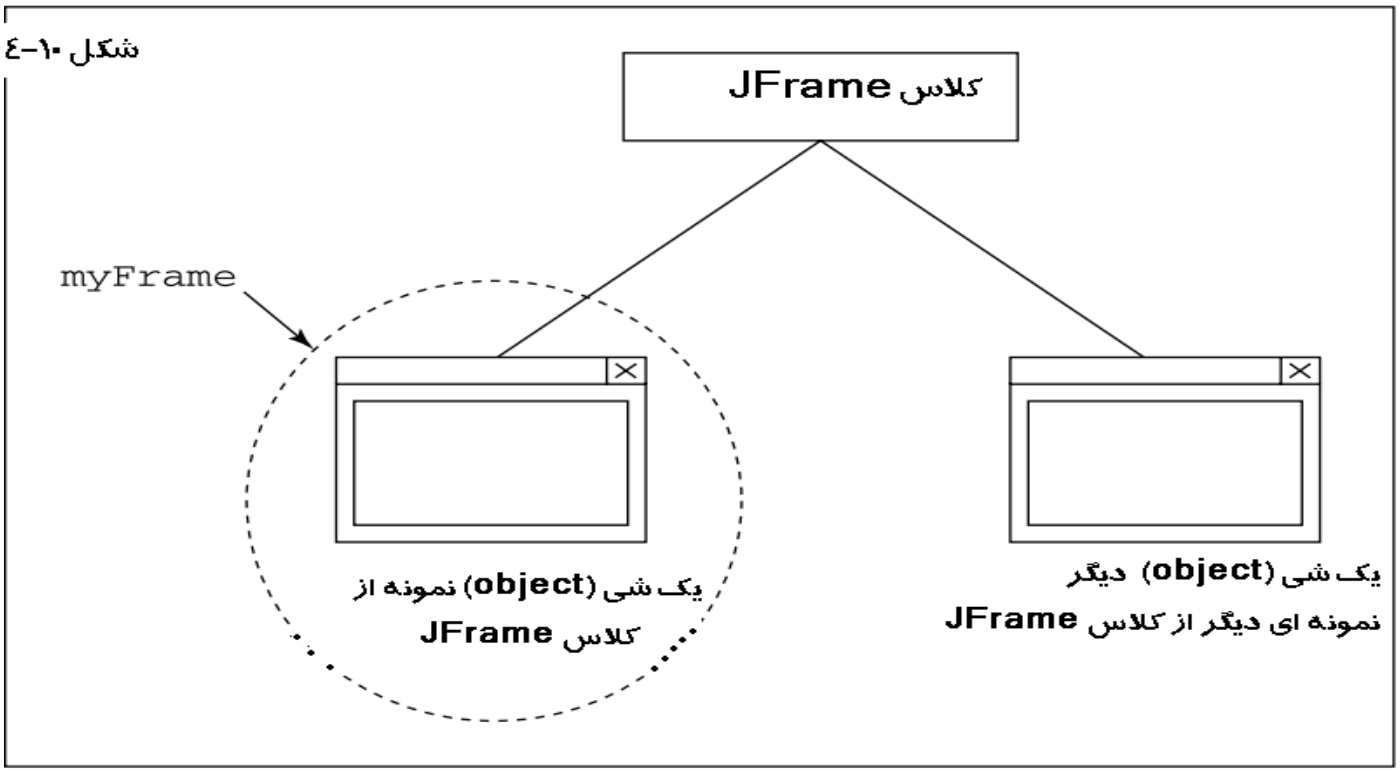
شمامیتوانیدمتغییریازنوعdouble را به صورت زیراعلان کنید:

```
doubleamountInAccount;

doubleamountInAccount = 50.22;
```

```
JFramemyFrame;

JFramemyFrame = new JFrame();
```

شما همچنین میتوانید myframe را از نوع JFrame اختصاص دهید :

در شکل ۱۰-٤ متغیر myframe به مثال و نمونه ای از کلاس JFrame ارجاع میکند. (در فصول بعدی در مورد کلاس و شی به طور کامل توضیح داده خواهد شد).



در خط اول مثال ۶-٤ شما اعلان میکنید کف در این برنامه از javax.swing.JFrame در طول برنامه استفاده خواهید کرد. اگر خط یک را از برنامه ۶-٤ حذف کنید مجبور هستید هر جا که از JFrame استفاده میکنید، نام کامل javax.swing.JFrame را تکرار کنید. به درک بهتر برنامه زیر را با حذف خط اول از مثال ۶-٤ باید

```
public class ShowAFrame {
public static void main(String args[]) {
javax.swing.JFramemyFrame =
newjavax.swing.JFrame();
      String myTitle = "Blank Frame";
myFrame.setTitle(myTitle);
myFrame.setSize(3200, 200);
myFrame.setDefaultCloseOperation
         (javax.swing.JFrame.EXIT_ON_CLOSE);
myFrame.setVisible(true);
   }
}
```

برنامه را به شکل زیر تغییر دهید.

ایجاد مقادیر جدید با اعمال عملگرها (Creating New Values by Applying Operators) :

عملیات روی داده ها: مقدار یک متغیر ممکن است نتیجه ی یک عبارت محاسباتی و یا سایر عملیات روی داده ها باشد. برای انجام عملیات مختلف بر روی داده ها از **عملگر** ها استفاده میشود.

ابتدا برای توضیح کارکرد عملگرها محاسباتی در جاوا و اینکه از آن ها چگونه در برنامه هایمان استفاده کنیم به جدول زیر توجه نمایید

نوع عملگر	عملگر	نام عملگر	مثال	نتیجه
محاسباتی	*	ضرب	٥*٤	٢٠
	/	تقسیم	١٠/٤	٢
	+	جمع	٢+٥	٧
	-	منها	٩-٥	٤
	٪	باقیمانده	٥٪٢٣	٣

عملگر های حسابی در جدول شرح داده شده اند نکته ای که نباید فراموش کرد این است حاصل تقسیم عدد صحیح بر عدد صحیح دیگری عدد صحیح خواهد بود . البته ابن نکته از جدول بالا هم قابل مشاهده واست.حال برای روشن شدن هر چه بهتر عملگرهای حسابی با مثال ٧-٤ که در زیر آورده شده توجه نمایید:

```
import static java.lang.System.out;
public class MakeChange {
public static void main(String args[]) {
int total = 248;
int quarters = total / 25;
intwhatsLeft = total % 25;
int dimes = whatsLeft / 10;
whatsLeft = whatsLeft % 10;
int nickels = whatsLeft / 5;
whatsLeft = whatsLeft % 5;
int cents = whatsLeft;
out.println("From" + total + "cents you get");
out.println(quarters + "quarters");
out.println(dimes + "dimes");
out.println(nickels + "nickels");
out.println(cents + "cents");
   }
}
```

خروجی مثال بالا :

```
From 248 cents you get
9 quarters
2 dimes
0 nickels
3 cents
```

به خط اول برنامه ی بالا توجه نمایید: ; import static java.lang.System.out آن را با خط اول مثال ۶-٤ import javax.swing.JFrame; مقایسه نمایید.

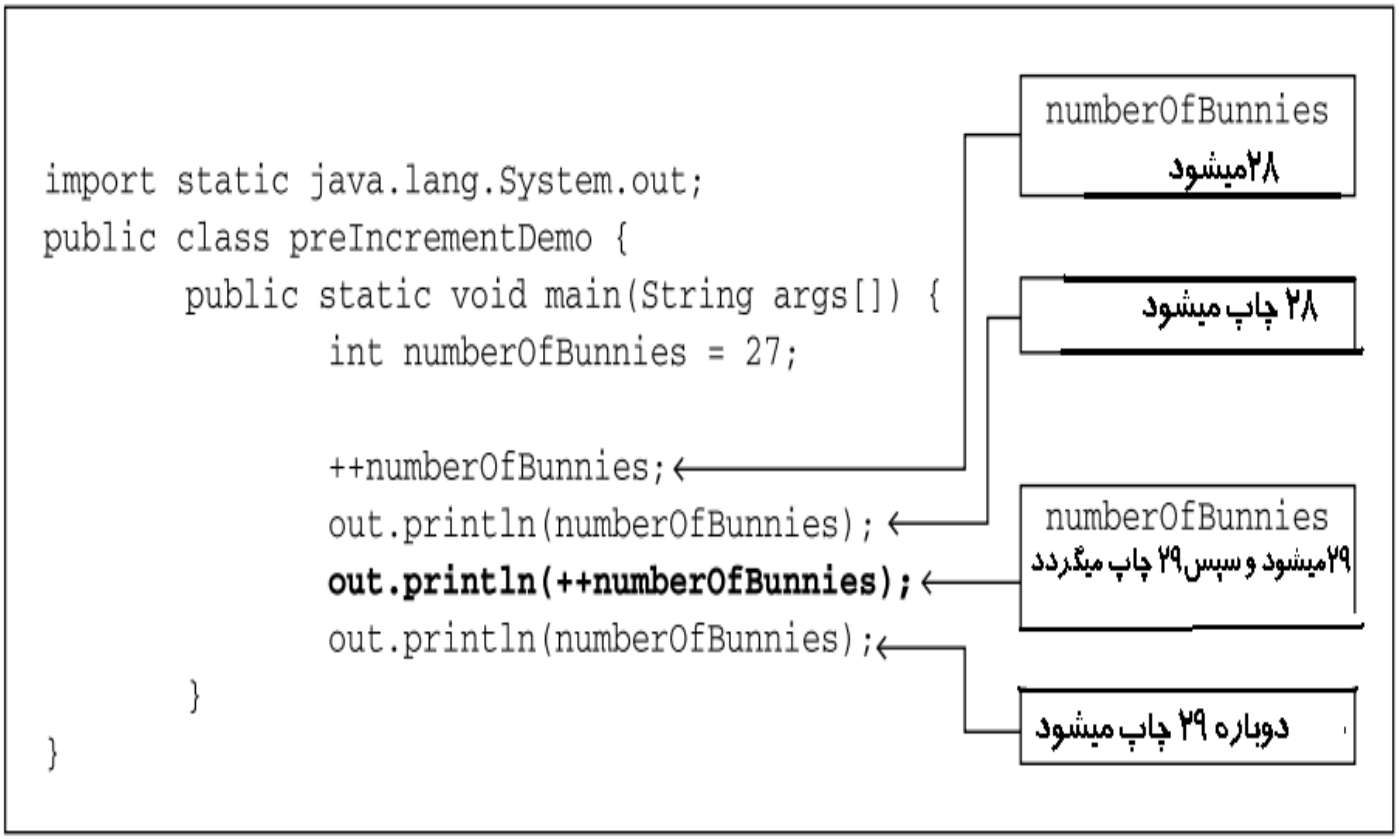
با اضافه نمودن ;import static java.lang.System.out به مثال ۷-٤ کد برنامه کمی برای نوشتن آسانتر شده و حتی خواندن آن را نیز سهل میشود. بدین صورت که شما بجای System.out.println فقط از عبارت کوتاهتر out.println استفاده کرده اید.

باید توجه نمود شما میتوانید خط اول برنامه import static java.lang.System.out را حذف نموده و به جای عبارت out.println عبارت System.out.println را جایگزین آن نمایید.

ولی اما در مثال ۷-٤ کاری که برنامه انجام میدهد: خرد کردن پول ۲٤۸ سنتی به سکه های رایج در ایالات متحده یعنی : ۱سنتی ، ۵ سنتی (nickel) ، ۱۰ سنتی(dime) ، ۲۵ سنتی(quarter) میباشد. کلاس MakeChange کمترین تعداد سکه های که با آن ها میتوان یک ۲٤۸ سنتی را خرد کرد ارائه میدهد.

عملگر های افزایش (increment) و کاهش (decrement) :

جاوا عملگر های بسیار مرتبی دارد که کار را برای ما ساده میکند. عملگر افزایشی یک واحد افزایش و عملگر کاهشی یک واحد کم میکند.عملگر افزایشی را با دو تا بعلاوه (++) و عملگر کاهش را با دوتا منها (--) نشان خواهیم داد . برای آنکه بفهمید آن ها چگونه کار میکنند به چند تا مثال نیاز دارید.

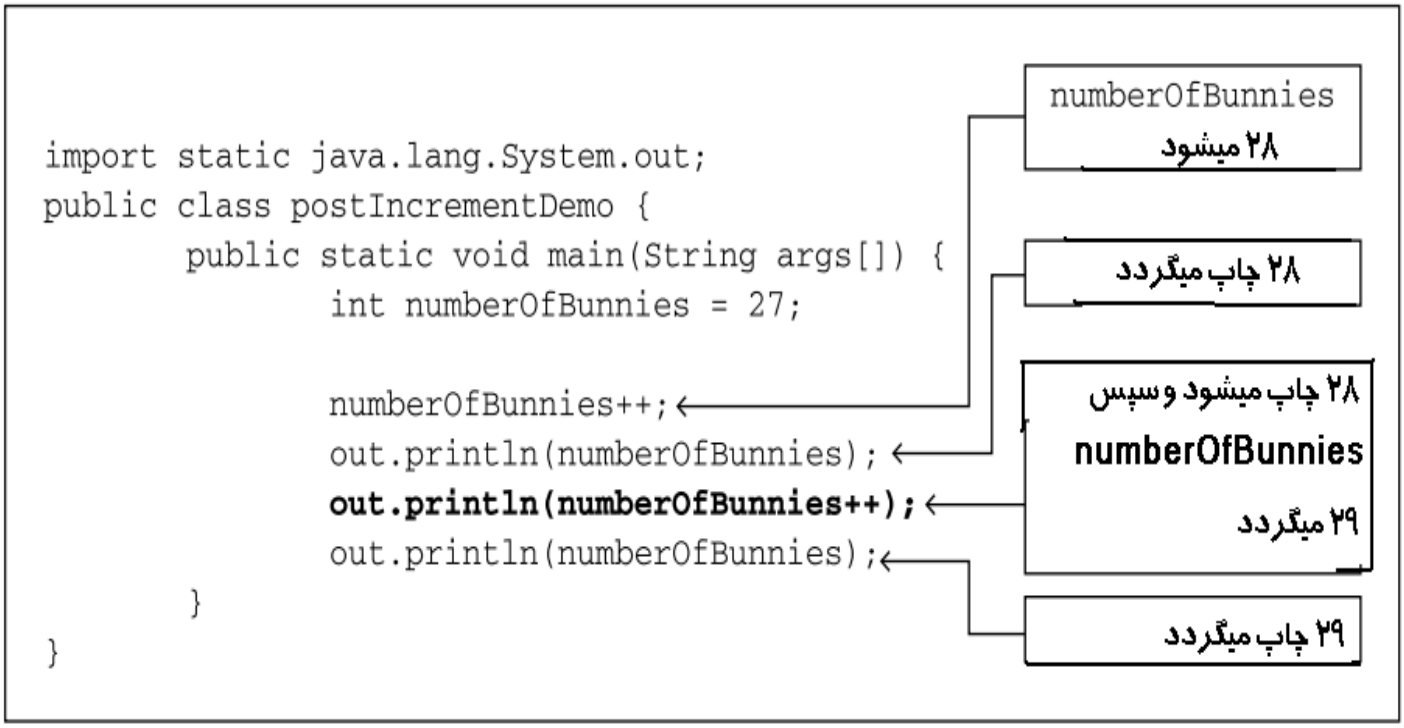


کارکرد این عملگر ها بستگی دارد که شما آن ها را کجای متغیر به کار برده باشید..اگر آن را قبل از یک متغیر به کار ببرید آن ***preincrement*** نامیده میشود. به مثال ۱۲-٤ توجه نمایید:

شما ++ را قبل از متغیر قرار میدهید و کامپیوتر قبل از اینکه از متغیر مورد نظر در یک عبارت دیگر استفاده کند مقدار آن را یک واحد افزایش میدهد.

برای مثال در خط هفتم برنامه ابتدا numberOfBunniesیک واحد افزایش یافته و سپس چاپ میگردد.

ولی اگر عملگر افزایش یعد از متغیر قرار گیرد (postincrement)، نمایده میشود و ابتدا متغیر با همان مقدار دز یک عبارت شرکت کرده و سپس افزایش صورت میگیرد. برای درک بهتر به مثال ١٤-٤ توجه نمایید:



نکته ای که ممکن است بی دقتی به آن موجب فاجعه شود نادیده گرفتن تفاوت مابین preincrement و postincrement میباشد و این به برنامه ی شما بستگی دارد که از کدام یک استفاده کنید.

نکته ی ۱ : در predecrement(--numberOfBunnies) کامپیوتر ایتدا یک واحد از متغیر کم کرده و بعد آن را در بقیه ی عبارت استفاده میکند.

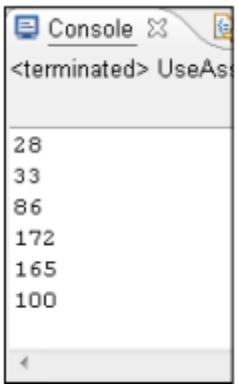
نکته ۲ : در postdecrement (numberOfBunnies--) کامپیوتر ابتدا متغیر را با مقدار قبلی در عبارت حساب کرده و سپس از مقدار آن یک واحد میکاهد.

عملگر های تخصیص (Assignment Operators) :

جاوا عملگر های تخصیص گوناگونی دارد که شما میتوانید عملیات های جمع و تفریق و ضرب تقسیم و .... را انجام دهید.برای مثال در مثال پایین توجه کنید چگونه 5=+ به مقدار متغیر numberOfBunnies پنج واحد می افزاید. و یا چگونه ۲=* مقدار numberOfBunnies را دوبرابر میکند.....برای آموزش عملکرد این عملگر ها به مثال ۸-٤ توجه نمایید :

```
public class UseAssignmentOperators {

public static void main(String args[]) {

intnumberOfBunnies = 27;

intnumberExtra = 53;

numberOfBunnies += 1;

System.out.println(numberOfBunnies);

numberOfBunnies += 5;

System.out.println(numberOfBunnies);

numberOfBunnies += numberExtra;

System.out.println(numberOfBunnies);

numberOfBunnies *= 2;

System.out.println(numberOfBunnies);

System.out.println(numberOfBunnies -= 7);

System.out.println(numberOfBunnies = 100);

   }

}
```

خروجی :



دو خط آخر مثال ۸-٤ ویژگی های خاص عملگر هاس جاوا میباشند، شما میتوانید عملیات تخصیص در به عنوان بخشی از یک عبارت در جاوا انجام دهید.در خط سیزدم عملگر ۷ واحد از numberOfBunnies میکاهد و مقدار

numberOfBunnies از ۱۷۲ به ۱۶۵ کاهش پیدا میکند. و بعد با این مقدار جدید System.out.println فراخوانی میشود .و در خط چهاردهم نشان میدهد که تخصیص مقدار به یک متغیر میتواند خود بخشی از یک عبارت بزرگتر باشد.

فصل ۳

Making Decisions (Java if Statements)

زمانی که شما در حال نوشتن برنامه های کامپیوتری هستید دائما در معرض انتخاب مسیر برنامه قرار دارید- (آیا کاربر پسوردش را به درستی واردکرده است؟ اگر بله، پس اجازه بده کاربر وارد سیستم شده و با آن کار کند. اگر خیر، پس به کاربر اجازه ی ورود نده.) . پس شما در برنامه های جاوا به راهی برای شاخه شاخه کردن برنامه و انتخاب بین شاخه ها احتیاج دارید. خوشبختانه جاوا برای این منظور راه حلی دارد که عبارات if نامیده میشوند.

ابتدا یک مثال زده و سپس از روی مثال زیر عبارات ifو چند مبحث دیگر را توضیح خواهم داد.

مثال ۱-۵ ::

```
import static java.lang.System.out;
importjava.util.Scanner;
importjava.util.Random;
public class GuessingGame {
public static void main(String args[]) {
      Scanner keyboard = new Scanner(System.in);
out.print("Enter an int from 1 to 10: ");
intinputNumber = keyboard.nextInt();
intrandomNumber = new Random().nextInt(10) + 1;
if (inputNumber == randomNumber) {
out.println("**********");
out.println("*You win.*");
out.println("**********");
      } else {
out.println("You lose.");
out.print("The random number was ");
out.println(randomNumber + ".");
      }
out.println("Thank you for playing.");
keyboard.close();
   }
}
```

خروجی مثال ۱-۵ :

```
Enter an int from 1 to 10: 2
**********
*You win.*
**********
Thank you for playing.


Enter an int from 1 to 10: 4
You lose.
The random number was 10.
Thank you for playing.
```

برنامه ۱-۵ یک بازی ساده است. یک عدد حدسی را از کاربر گرفته و سپس یک عدد تصادفی مابین ۱ تا ۱۰ تولید میکند..اگر عدد وارد شده توسط کاربر با عدد تصادفی تولید شده توسط برنامه یکسان باشد، کاربر برنده شده و در غیر این صورت بازنده است و برنامه به کاربر میگوید که عدد تصادفی چه بوده است.

کنترل کلید های فشار داده شده از کیبورد:

به این تکه کد که برشی از مثال ۱-۵ است دقت نمایید :

```
importjava.util.Scanner;

    Scanner keyboard = new Scanner(System.in);

intinputNumber = keyboard.nextInt();
```

این شبه کد هر عددی که کاربر از طریق کیبورد وارد کند را گرفته و در خط سوم آن را در متغیر inputNumber قرار میدهد. زمانی که شما میخواهید از کیبورد داده ای را دریافت کنید دو خط اول را فقط یکبار در برنامه قرار میدهید. بعدا در برنامه تان، هر زمان که نیاز باشد کاربر داده ی int ای را تایپ

کند باید کدی همانند خط سوم از شبه کد بالا که شامل فراخوانی nextInt باشد را بنویسید . تا داده ای

```
importjava.util.Scanner;
    Scanner readingThingie = new Scanner(System.in);
intvalueTypedIn = readingThingie.nextInt();
```

وارد شده از کیبورد گرفته شده و در متغیر مورد نظر ذخیره شود.

تمام کلمات موجود در شبه کد سه خطی بالا بجز دوتای آن ها جزئی از جاوا بوده و در جاوا موجود میباشند. آن دو نامی که من از خودم انتخاب کردم inputNumber و keyboard میباشند. پس برای روشن تر ساختن موضوع شبه کد بالا را فقط در از جنبه ی نام هایی که خودم انتخاب کردم تغییر میدهم :

من نمیخواهم تمام مفاهیم و جزئیات مثال ۱-۵ را توضیح بدهم ولی ذکر چند نکته ضروری ست :

نکته ۱ : وقتی import java.util.Scanner را به کار بردید از کلمه ی کلیدی static استفاده نمیکنید.ولی وقتی import java.lang.System.out را به کار بردید از کلمه ی کلیدی static استفاده کردید. این امر به این خاطر است که Scanner نام کلاس است. در حالی که System.out نام هیچ کلاسی در جاوا نیست.

نکته ی **۲**: معمولا (روی کامپیوتر های رومیزی و لب تاب ها) کلمه ی System.in برای استفاده کیبورد به کار میرود.

نکته ی۳: زمانی که شما انتظار دارید کاربر مقدار عددی صحیحی را وارد نماید از ()nextIn استفاده کنید. اگر انتظار دارید مقداری اعشاری وارد نماید از ()nextDouble استفاده نمایید. اگر انتظار دارید یکی از مقادیر درست یا غلط را وارد نماید از ()nextBoolean استفاده نمایید. اگر انتظار دارید که کاربر کلماتی نظیر Hadi, Java, Computer و... را وارد نماید از ()next استفاده نمایید.

نکته ی٤: شما میتوانید چندین مقدار را از طریق کیبورد یکی پی از دیگری دریافت نمایید .برای این کار از keyboard.nextInt() چندین بار استفاده نمایید.

نکته ی٥: زمانی که شما از Scanner جاوا استفاده میکنید، شما باید بعد از آخرین فراخوانی nextInt ، متد close را فراخوانی نمایید.مانند خط بیست و سوم از مثال ۱-۵.

ایجاد کردن مقادیر تصادفی

رسیدن به مقادیر واقعی تصادفی به طور شگفت آوری مشکل است.یک کامپیوتر در تولید مقادیر تصادفی ممکن است به نظر رسد که واقعا دارد مقادیر تصادفی تولید میکند.و لی در انتها تنها چیزی که به آن گفته شده را انجام میدهد و این یک فشن و روش کاملا قطعی ست نه تصادفی.

پس در مثال ۱-۵ زمانی که کامپیوتر کد زیر را اجرا میکند:

```
importjava.util.Random;

intrandomNumber = new Random().nextInt(10) + 1;
```

به نظر مقادیر وتصادفی مابین ۱ تا ۱۰ تولید میکند ولی این مقادیر دراصل، واقعا تصادفی نیستند. کامپیوتر فقط از دستور العمل های داد شده به آن پیروی میکتد.

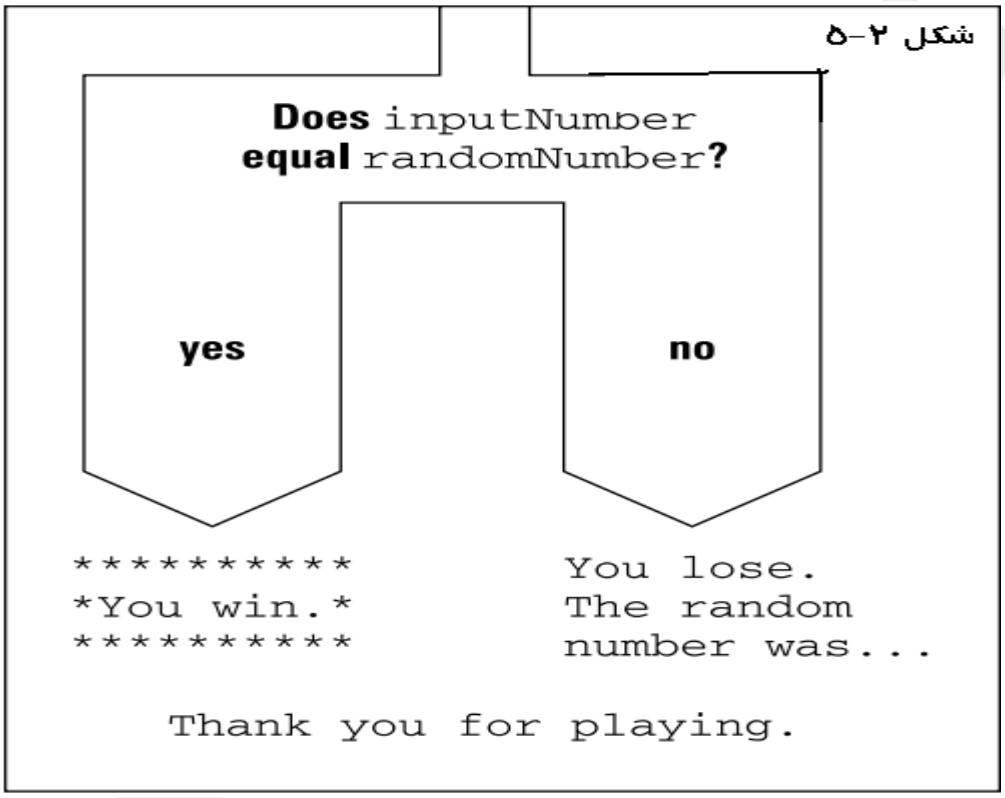
نگران نباشید که new Random().nextIn چه معنی میدهد. هدف از مثال ۱-۵ یادگیری یک سری مباحث مورد نظر این فصل میباشد. اگر معنی برخی کد ها را در مثال هایی که می آوریم جای نگرانی نیست. چون برخی از این کدها و مفاهیم و فصول بعدی توضیح داده خواهد شد.

دستورات انتخابی :

همان طور که در مثال ۱-۵ مشاهده میکنید عبارت if یک انشعاب در برنامه را نشان میدهد. کامپیوتر فقط یکی از دو شاخه را انتخاب خواهد کرد شاخه You win یا شاخه ی You lose را. کامپیوتر یکی از دو شاخه را با تست کردن درست یا غلط بودن شرط ، انتخاب خواهد کرد. شرطی که تست خواهد شد :

```
inputNumber == randomNumber
```

شکل ۲-۵ دستور if را به صورت گرافیکی نشان میدهد :



شکل ۲-۵

همانطور که از شکل پیداست. یک عبارت if شبیه یک انتشعاب و شاخه شاخه شدن در یک مسیر و راه میباشد.

آیا inputNumber با randomNumber برابر است ؟ اگر شرط درست باشد ، برنامه بلوک مابین شرط و کلمه ی else را اجرا خواهد کرد. اما اگر شرط نادرست باشد کامپیوتر بلوک بعد از کلمه ی else را اجرا

خواهد نمود. به هر حال کامپیوتر آخرین فراخوانی println را اجرا خواهد کرد و Thank you for playing را نمایش خواهد داد.

علامت ==(The double equal sign):

علامت مساوی = برای تخصیص یک لیترال به یک متغیر یا تخصیص مقدار یک متغیر به یک متغیر دیگر و..... به کار میرود. در حال که علامت دو تا مساوی == برای مقایسه ی دو مقدار به کار برده میشود تا بفهمیم که آیا آنها دارای مقداری یکسان اند یا خیر ؟

کاربرد if بدون else

:به مثال ۱-۵ دوباره نگاهی بیاندازید، شاید نخواهید به کاربر بگویید که او باخته است. در این صورا باید مثال ۵-۱ را به صورت زیر ویرایش کنید مثال ۲-۵ :

```
import static java.lang.System.in;
import static java.lang.System.out;
importjava.util.Scanner;
importjava.util.Random;
public class DontTellThemTheyLost {
public static void main(String args[]) {
      Scanner keyboard = new Scanner(in);
out.print("Enter an int from 1 to 10: ");
intinputNumber = keyboard.nextInt();
intrandomNumber = new Random().nextInt(10) + 1;
if (inputNumber == randomNumber) {
out.println("*You win.*");
      }
out.println("That was a very good guess :-)");
out.print("The random number was ");
out.println(randomNumber + ".");
out.println("Thank you for playing.");
keyboard.close();
   }
}
```

خروجی مثال ۲-۵ :

```
Enter an int from 1 to 10: 4
*You win.*
That was a very good guess :-)
The random number was 4.
Thank you for playing.


Enter an int from 1 to 10: 4
That was a very good guess :-)
The random number was 6.
Thank you for playing.
```

مثال ۲-۵ بخش else ندارد اگر inputNumber با randomNumber برابر باشد برنامه You win را چاپ خواهد نمود در غیر این صورت برنامه You win را چاپ نخواهد کرد.

ایجاد شرط با عملگر های مقایسه ای و منطقی :

جاوا عملگر ها و ابزار های بسیار فراوانی برای نیاز های متنوع شما در ایجاد شرط داراست. در این بخش قصد داریم به این موضوع بپردازیم.

جدول ۱-۵ به شما نشان میدهد که چگونه ی میتوانید از عملگر های مقایسه ای برای مقایسه یک چیز با دیگری استفاده کنید.

عملگر های مقایسه ای — جدول ۱-۵

نماد عملگر	معنا	مثال
==	برابر است با	`numberOfCows == 5`
!=	برابر نیست با	`buttonClicked != panicButton`
<	کوچک تر است از	`numberOfCows < 5`
>	بزرگتر است از	`myInitial > 'B'`
<=	کوچکتر مساوی است	`numberOfCows <= 5`
>=	بزرگتر مساوی است	`myInitial >= 'B'`

مقایسه ی اشیاء (Comparing objects)

زمانی که شما کار کردن با اشیاءِ را شروع کردید، شما خواهید دانست که میتوانید از == و =! برای مقاسه ی اشیاءِ با یکدیگر استفاده کنید. برای مثال یک دکمه که شما روی نمایشگر کامپیوتر می بینید یک شیءِ است. شما میتوانید با نوشتن یک برنامه جویا شوید که آیا یک چپ کلیک روی دکمه خاص روی نمایشگر کامپیوتر اتفاق است یا نه ؟شما میتوانید این کار را با استفاده از عملگر های کیفیت جاوا (Java's equality operator) انجام دهید.

```
if (e.getSource() == bCopy) {
clipboard.setText(which.getText());
```

اما برای مقایسه دو String باید توجه داشته باشید که، هنگام این مقایسه استفاده از == بدین معنی است که " آیا این String دقیقا در محلی از حافظه که String دیگر قرار دارد، ذخیره شده است". معمولا شما چنین خواسته اس ندارید. . حال برای پرسیدن اینکه " آیا این String دقیقا همان کاراکترهایی را دارد که String دیگر داراءِ میباشد" یک متد به اسم *equals* وجود دارد که در API (Application Programming Interface) تعریف گردیده است. *equals* دو String را مقایسه می مند که ببیند که آیا آن ها کاراکتر های یکسانی دارند یا خیر؟ به مثال ۳-۵ توجه نمایید .

در مثال ۳-۵ keyboard.next() فراخوانی شده و هر چیزی که کاربر روی کیبورد کامپیوتر تایپ کند را گردآوری میکند. سپس کلمه ی تایپ شده را در متغیری به نام password ذخیره میکند.و در نهایت با استفاده از if تفاوت == و متد equals نشان داده می شود.

در فراخوانی متد equals مهم نیست که کدام String داخل پارانتز و کدام یک قبل از نقطه باشد ، برای مثال میتوانستیم بخط بیست و دمم را به شکل زیر نیز بنویسیم:

```
if ("swordfish".equals(password))
```

که در این صورت معنی و کارکرد برنامه ۳-۵ هیچ نغییری نمیکرد.

```
import static java.lang.System.*;
importjava.util.Scanner;
public class CheckPassword {
public static void main(String args[]) {
out.print("What's the password?");
      Scanner keyboard = new Scanner(in);
      String password = keyboard.next();
out.println("You typed >>" + password + "<<");
out.println();
if (password == "swordfish") {
out.println("The word you typed is stored");
out.println("in the same place as the real");
out.println("password. You must be a");
out.println("hacker.");
      } else {
out.println("The word you typed is not");
out.println("stored in the same place as");
out.println("the real password, but that's");
out.println("no big deal.");
      }
out.println();
if (password.equals("swordfish")) {
out.println("The word you typed has the");
out.println("same characters as the real");
out.println("password. You can use our");
out.println("precious system.");
      } else {
out.println("The word you typed doesn't");
out.println("have the same characters as");
out.println("the real password. You can't");
out.println("use our precious system.");
      }
keyboard.close();
   }
}
```

خروجی مثال ۳-۵ :

```
What's the password? swordfish
You typed >>swordfish<<

The word you typed is not
stored in the same place as
the real password, but that's
no big deal.

The word you typed has the
same characters as the real
password. You can use our
precious system.
```

در اولین خط برنامه ۳-۵ یک راه تنبلی را نشان میدهد شما با نوشتن import static java.lang.System.* یک راه سهل و آسان را انتخاب کرده اید ، این یک خط تقریبا معادل ۳۰ عدد import جداگانه ی دیگر مانند System.in و System.nanoTime , System.err, System.out و..... میباشد. کابرد این نوع به کاربردن *(ستاره) در برنامه های بزرگ است که باعث کوتاه تر شدن کد برنامه خواهد شد.

عملگر های منطقی در جاوا

جاوا تعدای عملگر برای کار ئ تست مقادیر منطقی دارد که در جدول زیر نشان داده شده اند :

جدول ۲-۵ — عملگر های منطقی

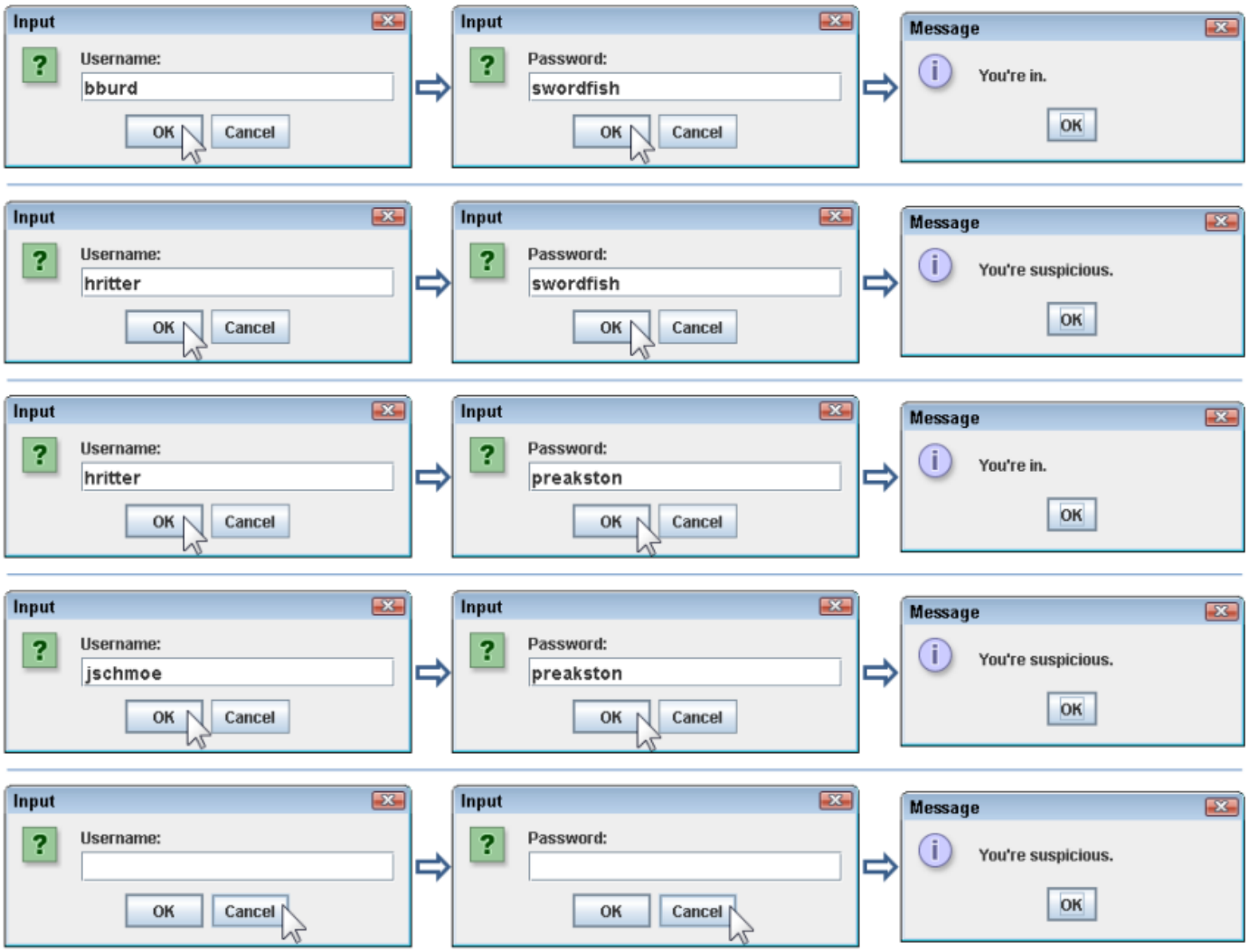
نماد عملگر	معنا	مثال
&&	و	`5 < x && x < 10`
||	یا	`x < 5 || 10 < x`
!	نقیض(نه منطقی)	`!password.equals("swordfish")`

شما میتوانید از این عملگر ها در شرط ها به طور مهرانه ای استفاده کنید. به مثال ٤-۵ دقت نمایید:

حاصل اجرای چندین بار مثال ٤-٥ در زیر نشان داده شده است ، زمانی که یوزرنیم bburd و پسورد swordfish می باشد و یا زمانی که یوزرنیم hritter و پسورد preakston می باشد. کاربر پیام مناسب را که نشان می دهد یوزرنیم و پسورد صحییح هستند را دریافت میکند، در غیر این صورت پیامی مبنی در نامعتبر بودن یوزرنیم و پسورد دریافت میکند.

```
importjavax.swing.JOptionPane;
public class Authenticator {
public static void main(String args[]) {
      String username =
JOptionPane.showInputDialog("Username:");
      String password =
JOptionPane.showInputDialog("Password:");
if (
username != null && password != null &&
        (
          (username.equals("bburd") &&
password.equals("swordfish")) ||
          (username.equals("hritter") &&
password.equals("preakston"))
        )
       )
     {
JOptionPane.showMessageDialog
          (null, "You're in.");
     } else {
JOptionPane.showMessageDialog
          (null, "You're suspicious.");
     }
  }
}
```

خروجی حاصل از اجرای مثال ***5-4*** :



مثال ٤-٥ یک روش تازه را برای گرفتن ورودی از کاربر نشان میدهد. روس صفحه نمایش یک کادر پیام (دیالوگ) مبنی در درخواست از کاربر برای وارد کردن ورودی، نشان داده می شود.

```
String password =    JOptionPane.showInputDialog("Password:");
```

درمثال ٤-٥ دوباز از JOptionPane.showInputDialog استفاده شده است. یک بار برای گرفتن یوزرنیم و دیگر بار برای گرفتن پسورد. برای مثال JOptionPane.showInputDialog("Username") یک کادر محاوره ای (دیالوگ) شامل فیلدی برای وارد کردن متن و نیز دکمه های OK و Cancel به نمایش در می

آورد. زمانی که کاربر بروی OK کلیک می نماید. کامپیوتر هر آنچه که در فیلد متن وارد شده را داخل یک متغیر قرار می دهد.

حال به قسمت دوم ، خط یازدهم برنامه دقت نمایید:

```
JOptionPane.showMessageDialog   (null, "You're in.");
```

این خط نیز یک کادر محاوره ای(دیالوگ) نمایش می دهد با این تفاوت که هیچ فیلد متنی برای وترد کردن اطلاعات وجود ندارد. نام JOptionPaneدر API جاوا درون چیزی که javax.swing نامیده میشود، تعریف شده است. دلیل وجود خط اول نیز همین موضوع است.

در مثال بالا JOptionPane.showInputDialog به زیبایی کار میکند زیرا کاربر یوزرنیم و پسورد را به صورت رشته ای از کاراکتر ها وارد میکند. اگر شما میخواهد کاربر مقدار int را وارد کند باید چیزی شبیه

```
intnumberOfCows = Integer.parseInt(JOptionPane. showInputDialog("How many cows?"))
```

تایپ کنید. ویا برای دریافت کردن مقدار double از کار بر باید چیزی شبیه به زیر را تایپ کنید.

```
doublefractionOfHolsteins = Double. parseDouble(JOptionPane.showInputDialog("Holsteins:"))
```

توجه داشته باشید که username != null به این معنی ست که یوزرنیم تهی نیست، یااینکه یوزنیم دارای یک مقدار است.

توجه : دو عبارات username != null و نیز password != null اختیاری نیستند. اگر شما آن ها را از برنامه حذف کنید و سپس برنامه را اجرا کنید و در کادر ظلاهر شده روی Cancel کلیک کنید برنامه مقابل چشمانتان خطای NullPointerException ر امیگیرد. برای تمرین این خطوط را حذف کرده و دوباره رنامه را اجرا کنید تا عملا با این نوع خطلا آشنا شوید.

کاربرد کلمه ی کلیدی null در خطوط دهم و یازدهم کاملا متفاوت است، فراخوانی showMessageDialog به جاوا میگوید که یک کادر محاوره ای (دیالوگ) جدید ایجاد کند. کلمه null بر این موضوع اشاره میکند که کادر محاوره ای جدید بر روی هیچ یک از کادر های محاوره ای قبلی سبز نشود.

ساخت if های تودرتو

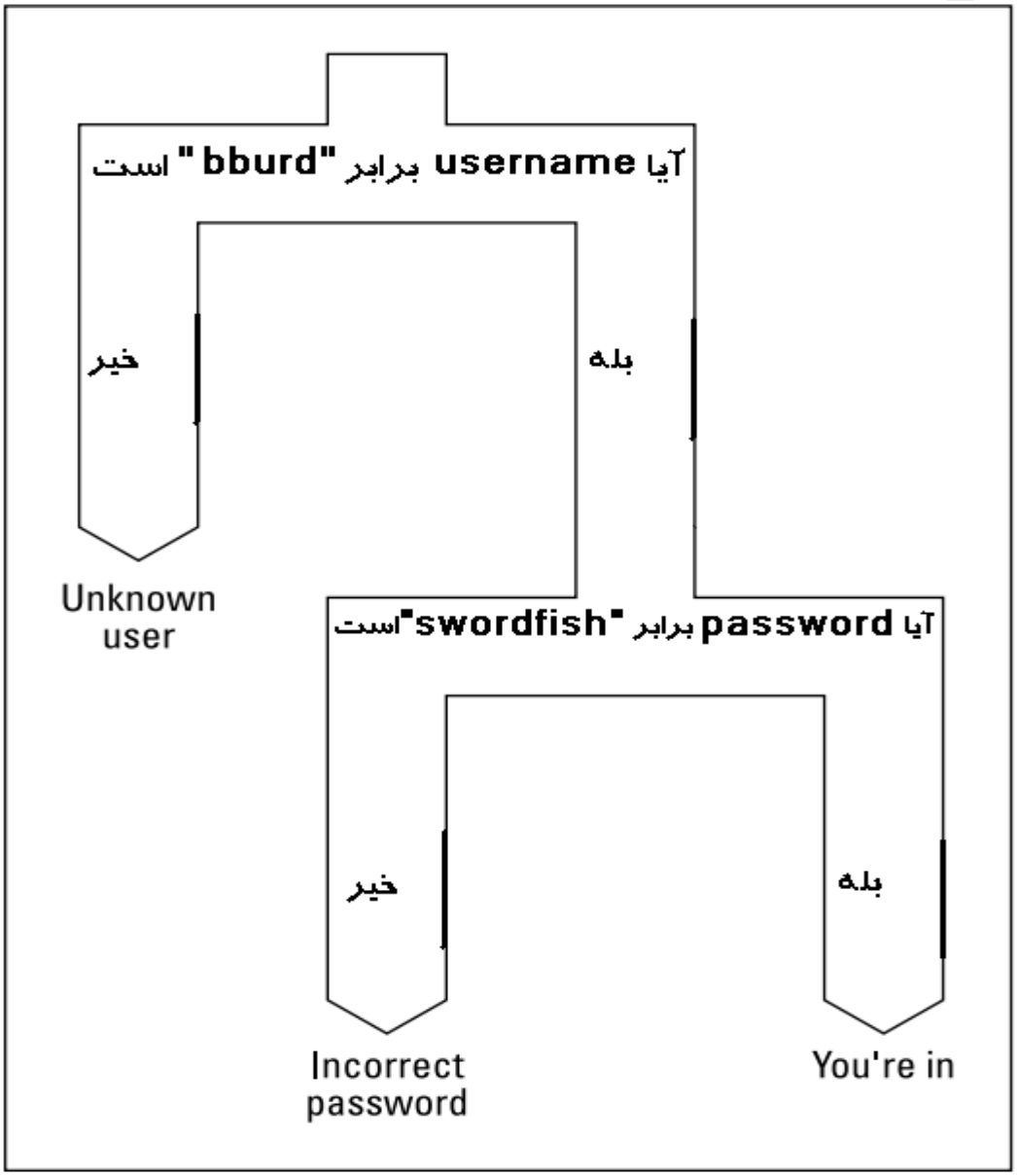
در جاوا می توان از یک if داخل if دیگر استفاده کرد. به مثال ۵-۵ دقت کنید :

```
import static java.lang.System.out;
importjava.util.Scanner;
public class Authenticator2 {
public static void main(String args[]) {
    Scanner keyboard = new Scanner(System.in);
out.print("Username: ");
    String username = keyboard.next();
if (username.equals("bburd")) {
out.print("Password: ");
        String password = keyboard.next();
if (password.equals("swordfish")) {
out.println("You're in.");
        } else {
out.println("Incorrect password");
        }
    } else {
out.println("Unknown user");
    }
keyboard.close();
  }
}
```

خروجی حاصل از چندین بار اجرای مثال ۵-۵ :

```
Username: bburd
Password: swordfish
You're in.


Username: bburd
Password: catfish
Incorrect password


Username: jschmoe
Unknown user
```

شما برای وارد شدن باید هر دوی یوزرنیم و پسورد را درست وارد کنید.(به عبارتی دیگر هر دو شرط if باید درست باشد). شرط اول درستی یوزرنیم را بررسی میکند. شرط دوم درستی پسورد را بررسی میکند. اگر شما اولین شرط(یوز نیم) را درست وارد کنید. شما به سوی if دیگری که شرط دوم(پسورد) را بررسی میکند پیش روی میکنید. در اولین if شرط(یوزنیم) را نادرست وارد کنید، هرگز شرط if دوم (پسورد) از شما پرسیده نخواهد شد.



برای درک بهتر موضوع به نمودار گرافیکی ۵-۷ دقت کنید :

البته این مثال فقط برای یادگیری if های تو در تو آورده شده و شاید در کاربرد آن در عمل باید کمی تغییرات روی آن انجام دهیم: ۱- پسورد وارد شده توسط کاربر به صورت ستاره دار نشان داده نمیشود. این امنیت را پایین می آورد ۲- هیچ گونه عمل کدگذاری(encrypting) روی پسورد انجام نمیشود. ۳- کبه کابران غیر قانونی که سعی در وارد شدن به سیستم دارند نشان داده میشود که کدام یک از username یا پسورد را اشتباه وارد میکنند.....هدف از گفتن این مسائل فقط آشنا کردن شما بود و قصد پیاده کردن این مسائل را در این فصل نداریم.

Switchدر جاوا

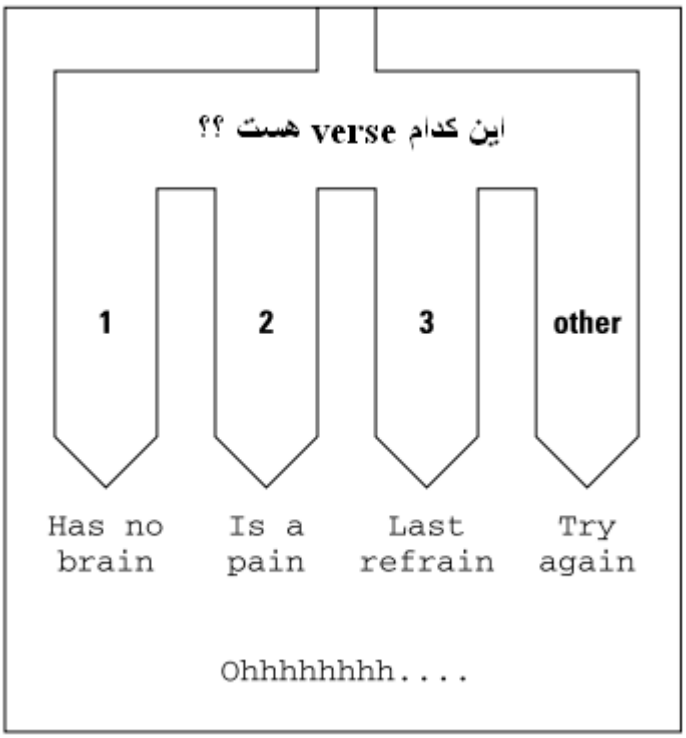
وقتی که قصد داریم از تعداد زیادی گزینه فقط یک مورد آنها را انتخاب کنیم دستور switch کاربرد دارد :ابتدا به مثال ۶-۵ دقت نمایید :

```
import static java.lang.System.out;
importjava.util.Scanner;
public class JustSwitchIt {
public static void main(String args[]) {
    Scanner keyboard = new Scanner(System.in);
out.print("Which verse? ");
int verse = keyboard.nextInt();
switch (verse) {
case 1:
out.println("That's because he has no brain.");
break;
case 2:
out.println("That's because he is a pain.");
break;
case 3:
out.println("'Cause this is the last refrain.");
break;
default:
out.println("No such verse. Please try again.");
break;
    }
out.println("Ohhhhhhhh . . . .");
keyboard.close();
  }
 }
```

```
Which verse? 2
That's because he is a pain.
Ohhhhhhhh. . . .


Which verse? 6
No such verse. Please try again.
Ohhhhhhhh. . . .
```

. ابتدا کاربر یک عدد مانند عدد ۲ را وارد میکند. سپس برنامه به ابتدای عبارت switch یعنی خط هشتم میرسد. کامپیوتر مقدار متغیر verse را برسی میکند. بعد از آنکه کامپیوتر تشخیص داد که مقدار متغیر verse عدد ۲ است. هر یک از case های عبارت switch را برسی میکند. عدد ۲ با اولین case مطابقت ندارد . پس کامپیوتر به case دوم حرکت میکند.. مقدار متغیر verse با مقدار case دوم مطابقت دارد پس کامپیوتر دستوری که بلافاصله بعد از case دوم میآید را اجرا میکند.

حال case دوم دارای 2 دستور است که اولی That's because he is a pain.. را نمایش میدهد. و دومی دستور break است.زمانی که کامپیوتر به دستور break میرسد ، برنامه از switch خارج شده و به اولین دستور بعد switch که در اینجا (out.println("**********") میباشد میرسد. شکل ۵-۹ :



حال اگر دستور break را حذف کنیم چه اتفاقی می افتد؟؟ در این صورت دستورات تمام case های بعدی نیز اجرا خواهند شد.

مثلا اگر break را از case دوم حذف کنیم بعد از نمایش That's because he is a pain.برنامه از switch خارج نشده و Cause this is the last refrain نیز نمایش داده خواهد شد. خوب با توجه با این نکته break موجود در default اجباری نیست . اما فراموش کردن break در انتهای هر case یک خطلای رایج در برنامه نویسی است . برای که شما با عواقب ناخوشایند این فراموشی آشنا شوید به مثال ۷-۵ دقت کنید.

```
import static java.lang.System.out;
importjava.util.Scanner;
public class FallingForYou {
public static void main(String args[]) {
      Scanner keyboard = new Scanner(System.in);
out.print("Which verse? ");
int verse = keyboard.nextInt();
switch (verse) {
case 3:
out.print("Last refrain, ");
out.println("last refrain,");
case 2:
out.print("He's a pain, ");
out.println("he's a pain,");
case 1:
out.print("Has no brain, ");
out.println("has no brain,");
      }
out.println("In the rain, in the rain.");
out.println("Ohhhhhhhh...");
out.println();
keyboard.close();
   }
}
```

خروجی حاصل :

```
Which verse? 1
Has no brain, has no brain,
In the rain, in the rain.
Ohhhhhhhh...


Which verse? 2
He's a pain, he's a pain,
Has no brain, has no brain,
In the rain, in the rain.
Ohhhhhhhh...


Which verse? 3
Last refrain, last refrain,
He's a pain, he's a pain,
Has no brain, has no brain,
In the rain, in the rain.
Ohhhhhhhh...


Which verse? 6
In the rain, in the rain.
Ohhhhhhhh...
```

همانطور که مشاهده میکنید. ما ۳ را وارد کردیم ، که عدد ۳ با اولین case مطابقت دارد و دستورات اولین case اجرا میشود , ولی چون break را فراموش کردیم، دستورات case های بعدی هم به ترتیب اجرا شده اند.

Switch جدید و بهبود یافته

در مثال های ۶-۵ و ۷-۵ ، متغیر verse (از نوع int) دستور switch را به یکی از case ها هدایت میکند.. یک متغیر int داخل داستور switch در همه ورژن های جاوا کار میکند- در ورژن های قدیمی جاوا ، فقط انواع char و تعدار کمی از انواع دیگر داخل دستور switch کار میکرند- اما اگر شما از جاوا۷ با نسخه های جدید تر استفاده میکنید، میتوانید از مقادیر String داخل switch برای انتخاب و اجرای یکی از case ها استفاده کنید. اگر شما از جاوا برای ایجاد اپلیکیشن های Android استفاده میکنید توجه کنید که در جاوا ۶ و کلیه ورژن های قدیمی جاوا ، شما نمی توانید برای پرش میان case ها داخل دستور switch ، از مقادیر string استفاده نمایید. به مثال ۸-۵ دقت کنید

```
import static java.lang.System.out;
importjava.util.Scanner;
public class SwitchIt7 {
public static void main(String args[]) {
    Scanner keyboard = new Scanner(System.in);
out.print("Which verse (one, two or three)? ");
    String verse = keyboard.next();
switch (verse) {
case "one":
out.println("That's because he has no brain.");
break;
case "two":
out.println("That's because he is a pain.");
break;
case "three":
out.println("'Cause this is the last refrain.");
break;
default:
out.println("No such verse. Please try again.");
break;
    }
out.println("Ohhhhhhhh. . . .");
keyboard.close();
  }
}
```

خروجی :

```
Which verse (one, two or three)? two
That's because he is a pain.
Ohhhhhhhh. . . .
```

فصل ٤

تکرار دستور العمل ها بارها و بارها (دستور while در جاوا)

در اینجا یک بازی حدس زدن برای شما مثال زده شده است. کامپیوتر اعداد تصادفی از یک تا ده تولید میکند. و سپس کامپیوتر از شما میخواهد که اعداد را حدس بزنید اگر حدس شما اشتباه بود بازی ادامه می یابد، و چنانچه حدس شما درست باشد بازی خاتمه می یابد. سپس با استفاده از ایم مثال به توضیح حلقه ی while خواهیم پرداخت. مثال ۱-۶ :

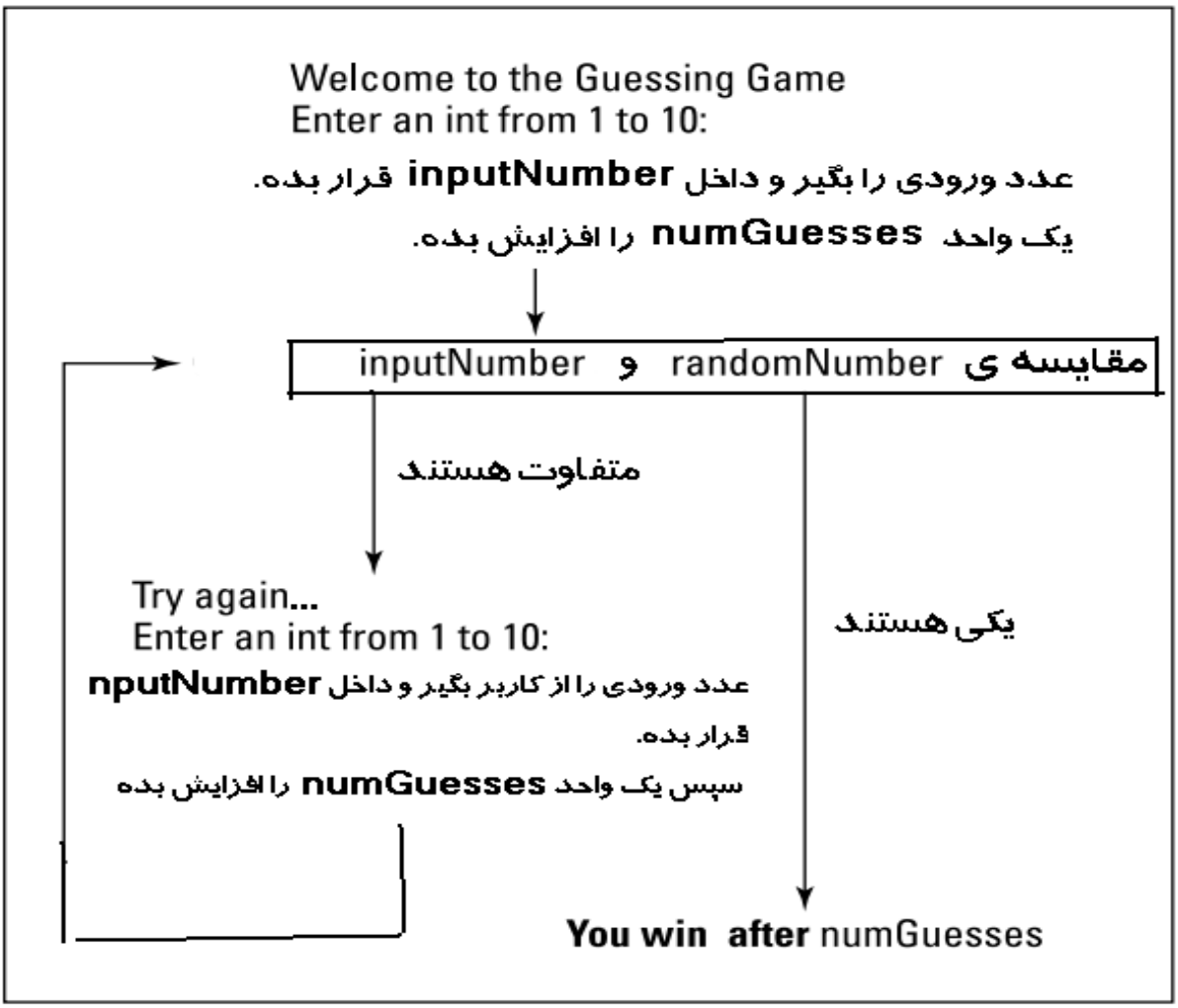
```
import static java.lang.System.out;
importjava.util.Scanner;
importjava.util.Random;
public class GuessAgain {
public static void main(String args[]) {
      Scanner keyboard = new Scanner(System.in);
intnumGuesses = 0;
intrandomNumber = new Random().nextInt(10) + 1;
out.println("      ************        ");
out.println("Welcome to the Guessing Game");
out.println("      ************        ");
out.println();
out.print("Enter an int from 1 to 10: ");
intinputNumber = keyboard.nextInt();
numGuesses++;
while (inputNumber != randomNumber) {
out.println();
out.println("Try again...");
out.print("Enter an int from 1 to 10: ");
inputNumber = keyboard.nextInt();
numGuesses++;
      }
out.print("You win after ");
out.println(numGuesses + " guesses.");
keyboard.close();
   }
}
```

خروجی مثال ۱-۶:

```
        ************
Welcome to the Guessing Game
        ************

Enter an int from 1 to 10: 2

Try again...
Enter an int from 1 to 10: 5

Try again...
Enter an int from 1 to 10: 8

Try again...
Enter an int from 1 to 10: 3
You win after 4 guesses.
```

در این خروجی کامپیوتر یک عدد تصادفی تولید کرده و از کاربر خواسته و عدد تولید شده را حدس بزند، کاربر در اولین حدس خود اشتباه میکند. ولی در تلاش دوم عدد تصادفی را به درستس حدس میزند. و پیام .... You Win را دریافت میکند، توجه کنید که به ازای هر حدس متغیر numGuesses یک واحد افزایش می یابد.

به خط شانزدهم دقت کنید کاری که این دستور انجام میدهد این است که تا زمانی که متقغیر randomNumber (عدد تصادفی تولید شده توسط کامپیوتر) و متغیر inputNumber (عدد وارد شده توسط کاربر) نامساوی باشند. دستورات خطوط ۱۷ الی ۲۱ را تکرار نماید. ۲-۶ کارکرد این مثال را روشن میکند:



اگر یکبار دیگر کد مثال بالا را مرور کنید. خواهید دانست که تنها کاری که حلقه ی while انجام میدهد این است که تا زمانی که شرط اش درست باشد،، دستورات موجود در اولین بلوک بعدی اش را تکرار میکند..

تکرار تا تعدادی معین(حلقه ی for در جاوا)

ابتدا به مثال ۲-۶ توجه نمایید، سپس حلقه for از روی این مثال توضیح داده خواهد شد.

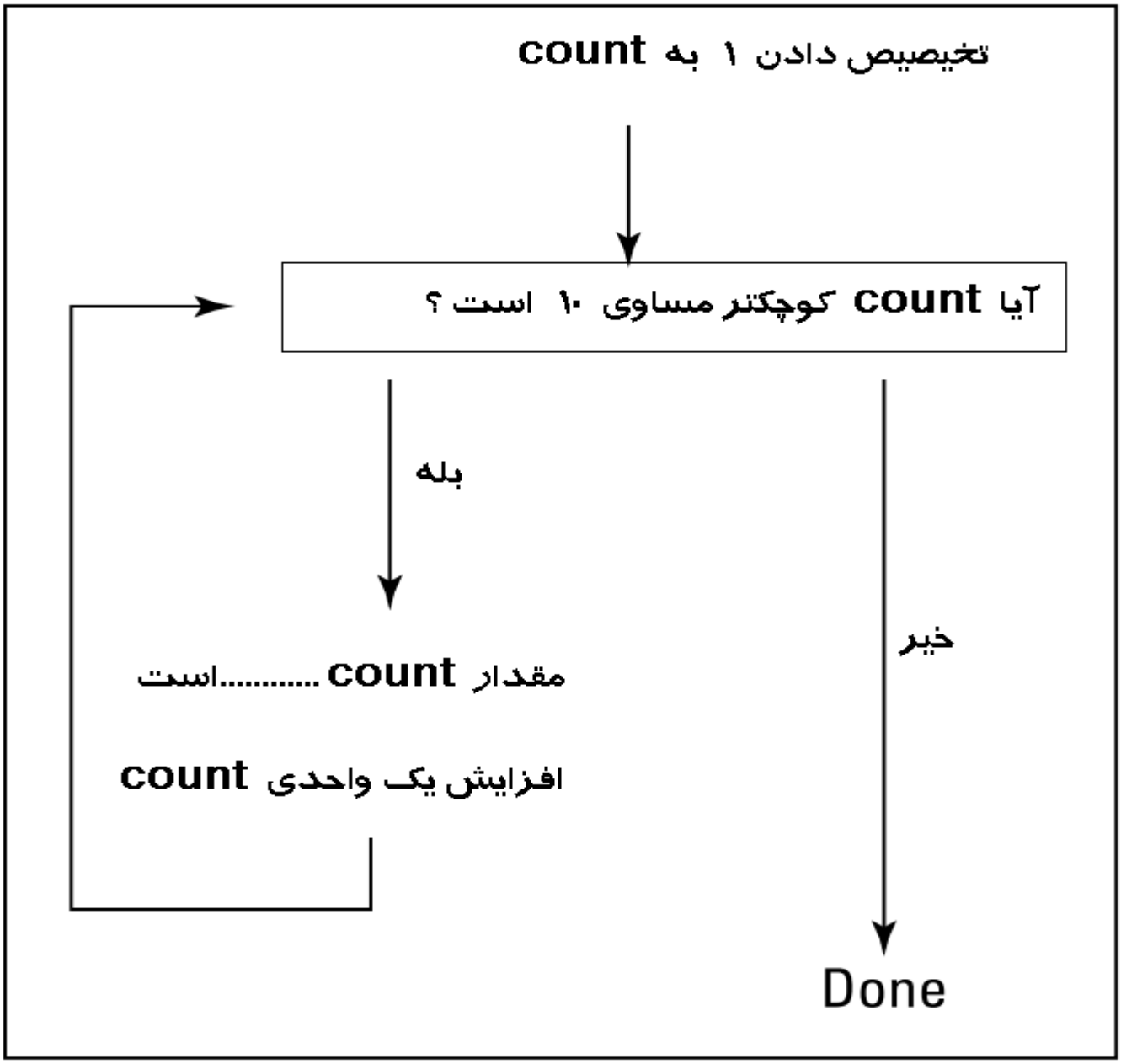
```
import static java.lang.System.out;
public class Yawn {
public static void main(String args[]) {
for (int count = 1; count <= 10; count++) {
out.print("The value of count is ");
out.print(count);
out.println(".");
      }
out.println("Done!");
   }
}
```

```
The value of count is 1.
The value of count is 2.
The value of count is 3.
The value of count is 4.
The value of count is 5.
The value of count is 6.
The value of count is 7.
The value of count is 8.
The value of count is 9.
The value of count is 10.
Done!
```

خروجی :

زمانی که برنامه را اجرا می کنید، حلقه ی for با تخصیص دادن یک به متغیر count شروع میشود. سپس مقدار count برسی میشود تا اطمینان حاصل شود که کوچکتر مساوی ده میباشد. اگر درست بود برنامه پیش میرود ، و دستورات مابین دو آکولاد را اجرا میکند، و در آخر یک واحد به count می افزاید.

شکل ۲-۶ کارکرد حلقه for در این مثال را توضیح میدهد :



اجرای حلقه ی for ادامه می یابد تا اینکه مقدار count به ۱۱ می رسد. در این هنگام شرط حلقه (کوچکتر مساوی بودن مقدار count از ۱۰) نادرست می شود. پس اجرای حلقه به پایان میرسد. و برنامه از اولین دستوری که بلافاصله بعد از for آمده است ، ادامه پیدا می کند.

حالت کلی دستور for به صورت زیر است:

for (initialization ; expression ; update)

initialization(مقدار دهی اولیه) : زمانی که برنامه برای اولین بار به عبارت for میرسد فقط یکبار اجرا میشود.

Expression(شرط) : چندین بار ارزیابی میشود (قبل از هر تکرار).

Update(به روز رسانی متغیر حلقه) : چندین بار اعمال میشود(بعد از هر تکرار).

```
import static java.lang.System.out;
public class AlsAllWet {
public static void main(String args[]) {
for (int verse = 1; verse <= 3; verse++) {
out.print("Al's all wet. ");
out.println("Oh, why is Al all wet? Oh,");
out.print("Al's all wet 'cause ");
out.println("he's standing in the rain.");
out.println("Why is Al out in the rain?");
switch (verse) {
case 1:
out.println("That's because he has no brain.");
break;
case 2:
out.println("That's because he is a pain.");
break;
case 3:
out.println("'Cause this is the last refrain.");
break;
      }
switch (verse) {
case 3:
out.println("Last refrain, last refrain,");
case 2:
out.println("He's a pain, he's a pain,");
case 1:
out.println("Has no brain, has no brain,");
      }
out.println("In the rain, in the rain.");
out.println("Ohhhhhhhh...");
out.println();
    }
out.print("Al's all wet. ");
out.println("Oh, why is Al all wet? Oh,");
out.print("Al's all wet 'cause ");
out.println("he's standing in the rain.");
  }
}
```

مثال بالا مروری هم به مطالب فصل قبلی میکند. دو عبارت switch داخل حلقه ی for به کار رفته اند. یکی از این switch ها از دستور break استفاده میکند و دیگری نه.

متغیر verse به عنوان متغیر حلقه ی for عمل میکند ابتدا مقدار ۱ و سپس ۲ و درنهایت مقدار ۳ را به خود میگیرد که در هر یک از این حالات case های مناسب درعبارات switch اجرا میشوند. زمانی که شرط حلقه نادرست میشود (مقدار count بزرگتر از سه میشود) برنامه بلافاصله از حلقه خارج شده و چهار دستور انتهای برنامه را اجرا میکند.

اجرا تا زمانی که آنچه میخواهید را به دست آورید (حلقه ی do در جاوا)

اولین سوالی که ممکن از خود بپرسید این است که این حلقه چه تفاوتی با حلقه ی while دارد. جواب این است که حلقه ی while حداقل یکبار اجرا خواهد شد. چون شرط حلقه در انتهای حلقه برسی میشود).

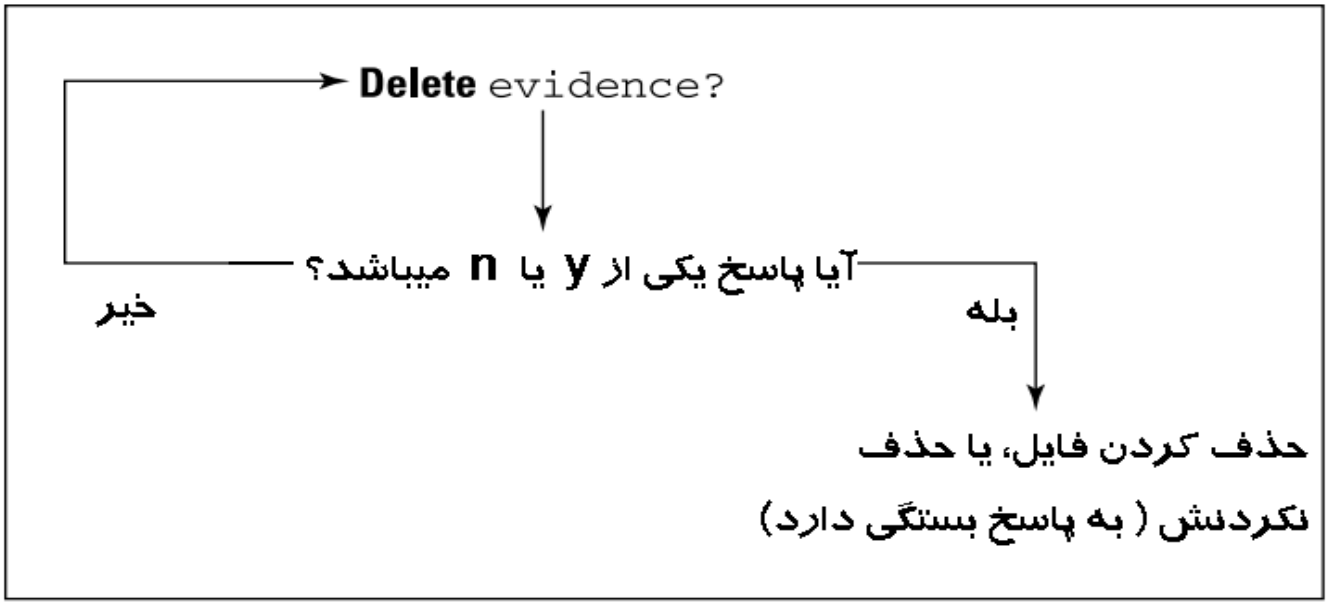
بعد مثال ٤-۶ توضیحات بیشتری داده خواهد شد.

```
importjava.io.File;
import static java.lang.System.out;
importjava.util.Scanner;
public class DeleteEvidence {
public static void main(String args[]) {
    File evidence = new File("cookedBooks.txt");
    Scanner keyboard = new Scanner(System.in);
char reply;
do {
out.print("Delete evidence? (y/n) ");
reply =
keyboard.findWithinHorizon(".",0).charAt(0);
    } while (reply != 'y' && reply != 'n');
if (reply == 'y') {
out.println("Okay, here goes...");
evidence.delete();
out.println("The evidence has been deleted.");
    } else {
out.println("Sorry, buddy. Just asking.");
    }
keyboard.close();
  }
}
```

خروجی حاصل از دوبار اجرای مثال بالا :

```
Delete evidence? (y/n) n
Sorry, buddy. Just asking.


Delete evidence? (y/n) u
Delete evidence? (y/n) Y
Delete evidence? (y/n) L
Delete evidence? (y/n) 8
Delete evidence? (y/n) .
Delete evidence? (y/n) y
Okay, here goes...
The evidence has been deleted.
```

قبل از حذف فایل برنامه از کاربر می پرسد که فایل را خذف کند یا نه ؟؟ اگر کاربر y یا nجواب دهد برنامه همانطور که کاربر انتظار دارد پیش میرود. اما اگر کاربر از هرگونه کارکتر دیگر(مانند اعداد ، حروف بزرگ، سمبل ها.....) استفاده کند. برنامه از کاربر دوباره سوال خواهد کرد. حال سوالی که مطرح میشود این است که استفاده از حلقه ی do در اینجا چه لزومی دارد؟؟؟در پاسخ باید گفت که در اینجا برنامه برای اینکه از کاربر سوالی بپرسد نیازی به برسی شرط حلقه ندارد(البته قبل از کاربر پاسخی بدهد، چیزی برای برسی وجود ندارد) و بعد اینکه کاربر به سوال مربوطه جواب داد شرط حلقه برسی خواهد شد.



توجه: در مورد کاردن با فایل ها در فصول توضیح داده خواهد شد. هدف این مثال فقط آشنایی با حلقه ی do میباشد نه آموزش کار با فایل ها.

شما میتوانید کیفیت برنامه را حتی بهتر هم کنید چون کاربر ممکن است به بزرگ یا کوچک بودن حروف بی توجه باشد شما میتوانید شرط حلقه را به صورت زیر گسترش دهید :

```
do {
out.print("Delete evidence? (y/n) ");
reply = keyboard.findWithinHorizon(".", 0).charAt(0);
} while (reply != 'y' && reply != 'Y' &&
reply != 'n' && reply!='N');

if (reply == 'y' || reply == 'Y') {
```

ذخیره ی یک کارکتر تنها

در مثال ۳-۵ کاربر یک کلمه ای را در کیبورد تایپ می کرد و متد keyboard.next کلمه تایپ شده را گرفته و داخل متغیر String به نام password ذخیره میکرد. در آن مثال همه چیز به خوبی به پیش میرفت چون String می تواند تعداد زیادی کارکتر را در خود جاء دهد . اما در مثال بالا شما انتظار ندارید که کاربر چندین کاراکتر وارد کند، شما انتظار دارید که کاربر فقط یک حرف (y یا n) وارد نماید، به همین دلیل متغیری از نوع String به کار نمی برید. در عوض متغیر از نوع char به کار می برید، متغیر که فقط یک سمبل را در یک زمان می تواند نگه دارد. API جاوا چیزی به نام nextChar ندارد. پس برای خواندن چیزی که با متغیر نوع char هم خوانی داشته باشد باید چیزی شبیه تکه کد زیر را به کار ببرید:

```
keyboard.findWithinHorizon(".", 0).charAt(0)
```

شما میتوانید دقیقا از کد بالا هر جا که نیاز دازید تنها یک کاراکتر تنها را بخوانید ، استفاده کنید.

کارکردن با فایل ها در جاوا

جزئیات این موضوع در فصول بعدی ارائه خواهد شد، اما برای آشنایی با مطلب در اینجا توضیحاتی ارائه میشود. شما میتوانید در API جاوا یک کلاس به نام java.io.File پیدا کنید. عبارت :

```
File evidence = new File("cookedBooks.txt");
```

درون حافظه ی کامپیوتر یک شی جدید ایجاد می کند. این شی از کلاس java.io.File شکل گرفته است.

این شی هر چیزی را که برنامه نیا دازد در مورد فایل cookedBooks.txt بداند توصیف میکند. با توجه ب این نکته متغیر evidence به فابل روی دیسک cookedBooks.txt رجوع میکند.

شی evidence یک مثال از کلاس java.io.File است که دارای متد delete می باشد. زمانی که شما evidence.delete را فراخوانی میکنید، کامپیوتر فایل مربوطه را پاک میکند. البته شما نمی توانید چیزی را ککه وجود ندارد را پاک کنید، برای مثال در دستور

```
File evidence = new File("cookedBooks.txt");
```

جاوا برسی نمی کند که آیا فایلی به نام cookedBooks.txt وجود دارد یا نه ؟ برای اینکه جاوا را مجبور کنید که این برسی را انجام دهد چنین راه دارید، ساده ترین راه فراخوانی متد exists میباشد. زمانی که شما evidence.exists() را فراخوانی می کنید. متد به فولدری که جاوا اتظار دارد cookedBooks.txt را پیدا کند ، نگاه میکند فراخوانی evidence.exists() مقدار true را برمی گرداند اگر cookedBooks.txt را پیدا کند و در غیر این صورت مقدار false بر میگرداند. در اینجا نسخه ی دیگر از مثال ٤-٦ را باقبلیت چک کردن وجود فایل آورده می شود :

```
importjava.io.File;
import static java.lang.System.out;
importjava.util.Scanner;
public class DeleteEvidence {
public static void main(String args[]) {
   File evidence = new File("cookedBooks.txt");
if (evidence.exists()) {
    Scanner keyboard = new Scanner(System.in);
char reply;
do {
out.print("Delete evidence? (y/n) ");
reply =
keyboard.findWithinHorizon(".", 0).charAt(0);
    } while (reply != 'y' && reply != 'n');
if (reply == 'y') {
out.println("Okay, here goes...");
evidence.delete();
out.println("The evidence has been deleted.");
    } else {
out.println("Sorry, buddy. Just asking.");
    }
keyboard.close();
   }
 }
}
```

محدوده ی متغیر ها

مجموعه از عبارات که درون دو آکولاد {} قرار گرفته اند، یک بلوک را ایجاد میکنند. اگر شما متغیری را داخل یک بلوک تعریف کنید نمی توانید خارج از آن بلوک از آن متغیر استفاده کنید. برای نمونه اگر در مثال برای نمونه اگر مثال ٤-٦ را به صورت زیر تغییر می دادید خطا دریافت می کردید:

```
do {
out.print("Delete evidence? (y/n) ");
char reply =
keyboard.findWithinHorizon(".", 0).charAt(0);
} while (reply != 'y' && reply != 'n');
if (reply == 'y')
```

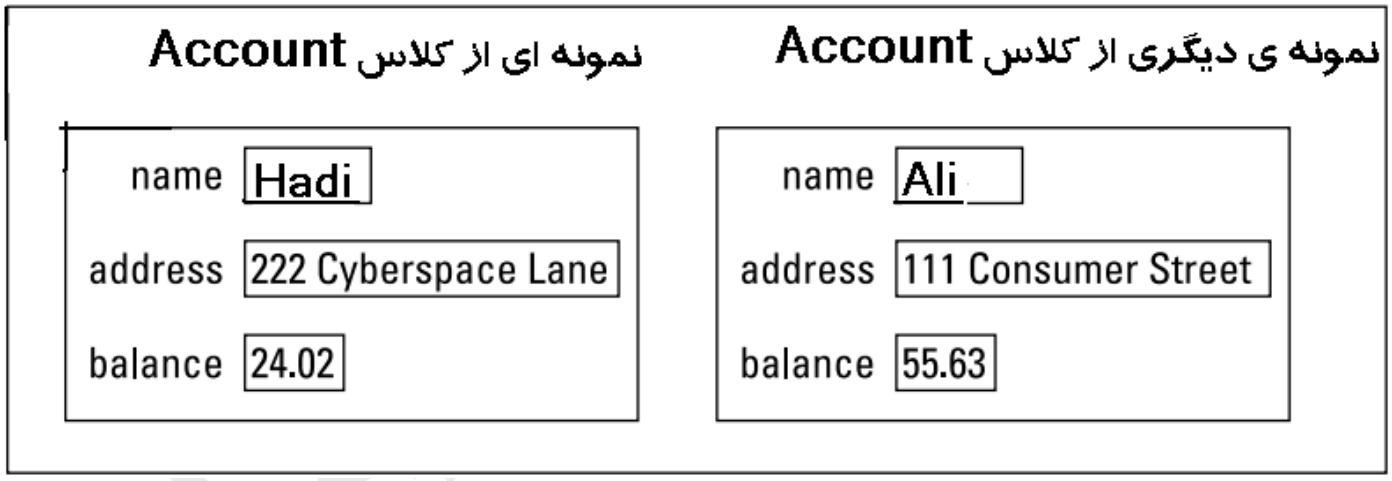
زیرا متغیر reply داخل بلوک مربوط به حلقه ی do اعلان گردیده و فقط در همانجا قابل استفاده است، ولی در بالا خارج از بلوک مربوطه نیز ازreply سه مرتبه استفاده شده که این باعث ایجاد خطا خواهد گردید.

فصل ۵

تعریف کلاس

چه چیزی دو حساب بانکی را در یک بانک مشترک متمایز میکند؟؟ اگر از یک بانک دار این سوال را بپرسید، پاسخ طولانی خواهد شنید ، ولی منظور اصلی ما از تفاوت حساب های بانکی در برنامه ی بانکداری و چگونگی برنامه نویسی برای ایجاد این تفاوت است. برای مثال احتمالا در حساب های بانکی یک متغیر به نام balance وجود دارد، که برای حساب من دارای مقدار ۲٤,۰۲ و برای حساب شما دارای مقدار ۵۵,۶۳ می باشد. حال سوال این است: چگونه در برنامه ی بانکداری مابین متغیر balance حساب من، و متغیر balance حساب شما تفاوت قائل شده و آنها را از هم تشخیص دهیم.

تفاوت ایجاد دو شی (object) جداگانه است. متغیر balance اولی را داخل شی اول ، و متغیر balance دوم را داخل شی دوم قرار دهید. شما همچنین میتوانید متغیر های name و addressدرون هریک قرار دهید.، پی شما دو شی خواهید داشت، و هر شی یک حساب بانکی را نشان میدهد. حال در اینجا هر شی مثالی ونمونه ای از کلاس Accountاست.



ولی با این راحل مشکل به طور کامل حل نشده است.در واقع تا به اینجا شما دو شی در برنامه ی کامپیوتری خود دارید. حال شما میتوانید دو متغیر برای رجوع به این دو شی داشته باشید، فرض کنید اولین متغیر را به نام myAccount و متغیر دوم را با نام yourAccount ایجاد کنید.که اولین متغیر به شی اول (حساب بانکی من) که و دومین متغیر به شی دوم (جساب بانکی شما) اشاره میکند.برای مثال :

```
1  myAccount.balance  //  به balance در حساب بانکی من رجوع میکند
2  myAccount.name     //به name در حساب بانکی من رجوع میکند
3  myAccount.balance = 24.02; // balance حساب من را برابر ۲۴,۰۲ قرار میدهد
4  out.println(yourAccount.name);  // name حساب شما را چاپ میکند
```

در بالا دو شی که مثالی از یک class به نام Account بودند آورده شده بود. ولی بر استفاده از این موضوع شما باید کلاس Account را از قبل تعریف کنید برای روشن شدن موضوع به تعریف کلاس Account در جاوا دقت نمایید:

```
public class Account {
   String name;
   String address;
double balance;
}
```

در واقع این تعریف به برنامه میگوید که هر مثال و نمونه ای از کلاس Account (یعنی اشیا) سه متغیر خواهند داشت- name و address و balance - در برنامه نویسی جاوا متغیر هایی که داخل کلاس (نه داخل) method تعریف شده اند ، فیلد (field) نامیده میشوند.

تعریف متغیر ها و تعریف اشیاء

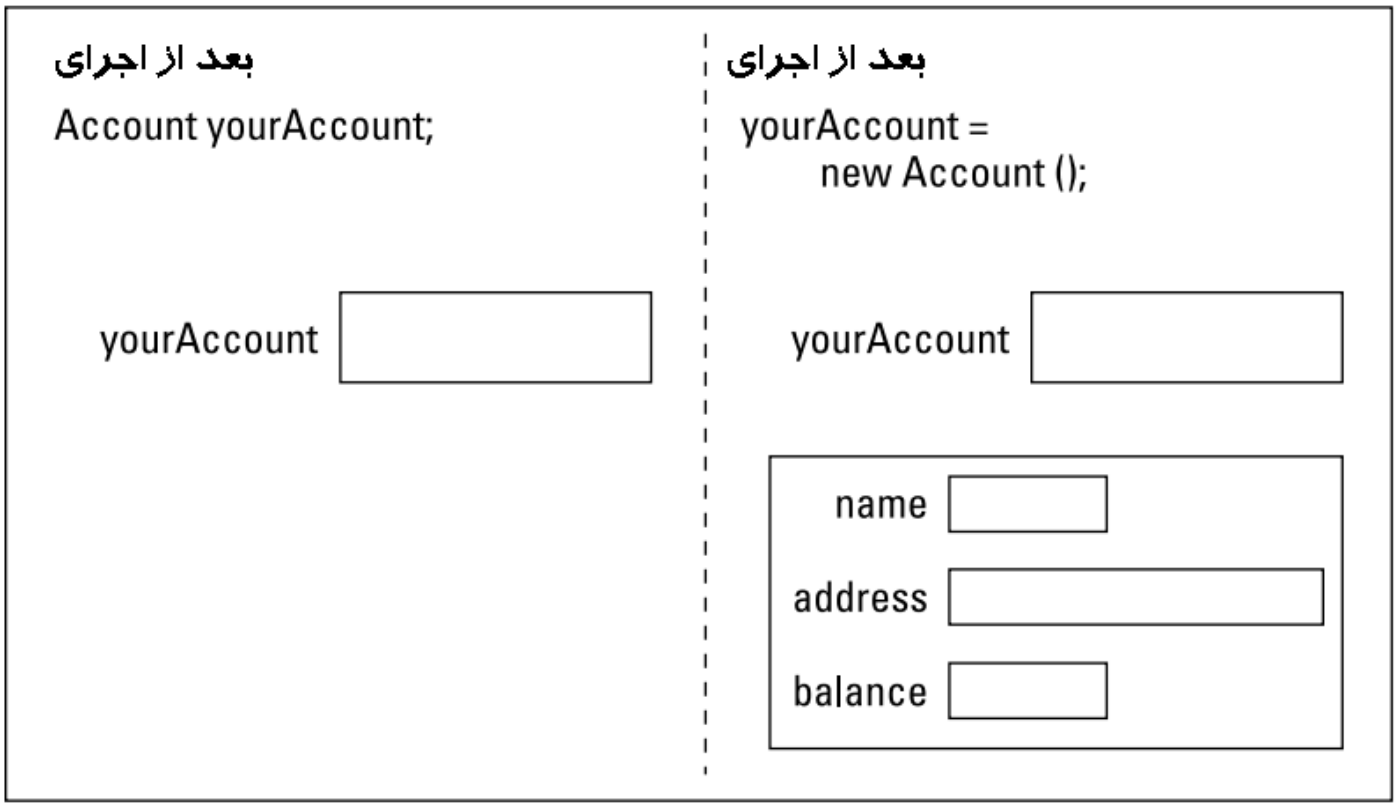
ابتدا به مثال ۲-۷ دقت کنید، سپس از روی این مثال به توضیح مطالب خواهیم پرداخت.

```
import static java.lang.System.out;
public class UseAccount {
public static void main(String args[]) {
      Account myAccount;
      Account yourAccount;
myAccount = new Account();
yourAccount = new Account();
   myAccount.name = "HadiKhodapanah";
myAccount.address = "222 Cyberspace Lane";
myAccount.balance = 24.02;
      yourAccount.name = "Ali Alizadeh";
yourAccount.address = "111 Consumer Street";
yourAccount.balance = 55.63;
out.print(myAccount.name);
out.print(" (");
out.print(myAccount.address);
out.print(") has $");
out.print(myAccount.balance);
out.println();
out.print(yourAccount.name);
out.print(" (");
out.print(yourAccount.address);
out.print(") has $");
out.print(yourAccount.balance);
   }
}
```

در اینجا دو کلاس Account و UseAccount وجود دارد که یک برنامه ی بالا را شکل میدهد. مثال مثال ۲-۷ کلاس UseAccount را تعریف میکند و کلاس UseAccount یک متد اصلی (main method) دارد. متد main متغیر های مخصوص خود را داراست (یعنی متغیر های yourAccount و myAccount).

دو خط اول داخل متد main (خطوط چهرم و پنجم) کمی گمراه کننده هستند به معنی این خطوط دقت کنید : خط Account yourAccount بدین معنی است که : اگر و زمانی که من متغیر yourAccount را ایجاد میکنم، این متغیر به چیزی رجوع میکند که آن چیز مثال و نمونه ای از کلاس Account خواهد بود.

از لحاظ فنی زمانی که کامپیوتر کد new Account() را اجرا میکند، یک شی به وسیله ی فراخوانی کلاس سازنده ی Account ساخته میشود..



زمانی که کامپیوتر کد yourAccount = new Accoun () را اجرا می کند. کامپیوتر یک شی جدید ساخته و متغیر yourAccount به آن شی ارجاع میکند. برای روشن شدن موضوع ، به شکل بالا رجوع کنید.

حال به خروجی حاصل از کمپایل و اجرای برنامه های ۱-۷ و ۲-۷ دقت کنید:

```
Hadi Khodapanah (222 Cyberspace Lane) has $24.02
Ali Alizadeh    (111 Consumer Street) has $55.63
```

کلاس های عمومی (Public classes)

کلاس Account به صورت public است، یک کلاس public برای همه ی کاربران به وسیله ی کلاس های دیگر در دسترس است، برای نمونه کلاس UseAccount نیز به صورت public هست. وقتی که کلاسی شامل متد main است، برنامه نویسان جاوا تمایل دارند که کلاس را به صورت public تعریف کنند، بدون اینکه زیاد در مورد کسی که از کلاس استفاده خواهد کرد فکر کنند !!!!

زمانی که شما کلاسی را به صورت public اعلان میکنید، شما باید کلاس را در فایلی که هم نام کلاس است، معرفی و ذخیره کنید. برای مثال زمانی که شما public class MyImportantCode را اعلان میکنید، شما باید کدهای کلاس را در فایلی به نام MyImportantCode.java قرار دهید. اگر کد شما دارای دو کلاس public باشد شما باید دست کم دو فایل java. را در نظر بگیرید، به عبارتی دیگر شما نمی توانید دو کلاس public را در یک فایل java. ذخیره کنید.

تعریف متد داخل کلاس

محتویای اطلاعات موجود در دو حساب بانک را در نظر بگیرید :

جدول ۱-۷ بدون برنامه نویسی شیء گرا

Name	Address	Balance
Hadi Khodapanah	222 Cyberspace Lane	24.02
Ali Alizadeh	111 Consumer Street	55.63

جدول بالا نشان میدهد که هر حساب سه چیز دارد – balance , address , name - این شیوه ی بود که قبل از اینکه برنامه نویسی شیء گرا وارد عرصه شود مورد استفاده قرار میگرفت. اما برنامه نویسی شیء گرا تغییر بزرگی را در تفکر برنامه نویسی ایجاد کرد. با برنامه نویسی شیء گرا هر حساب (account) می تواندیک name و یک address و یک balance داشته. و به طریقی نمایش داده شوند. به طوری که در برنامه نویسی شیء گرا هر شی یک سری عملگر یا functionality توکار دارد. هر حساب(account) میداند که چگونه خود را نمایش دهد. یک string میتواند به شما بگوید که آیا کاراکتر های یکسانی با یک string دیگر دارد. به عنوان مثال یک PrintStream مانند System.out میداند که چگونه println کند.

در برنامه نویسی شیءِ گرا هر شی متد های خودش را داراست. متد ها زیر برنامه های کوچکی هستند کهشما میتوانید آن ها را برای انجام کارهای مورد نظرتان فراخوانی کنید.

حال یک جدول تسهیل شده را تصور کنید، در جدول جدید هر شیءِ یک سری functionality توکار (built in) دارد.. هر account میداند که چگونه خود را روی مانیتور ، نمایش دهد. هر سطر از جدول یک کپی از متد display را برای خود دارد.

جدول ۲-۷	روش شیء گرایی		
Name	***Address***	***Balance***	***Display***
Hadi Khodapanah	222 Cyberspace Lane	24.02	`out.print....`
Ali Alizadeh	111 Consumer Street	55.63	`out.print....`

یک Account که خودش را نمایش میدهد

در جدول 2-7 هر شیءِ چهار چیز دارد: -- , , balance , address , nameروشی برای نمایش خودش— وقتی شما از برنامه نوبسی سنتی به برنامه نویسی شیءِ گرا کوچ میکنید دیگر هیچ وقت به گذشته و برنامه نویسی سنتی بر نخواهید گشت (چون نیازی به این کار نخواهید داشت) : به مثال ۳-۷ دقت کنید :

```
import static java.lang.System.out;
public class Account {
   String name;
   String address;
double balance;
public void display() {
out.print(name);
out.print(" (");
out.print(address);
out.print(") has $");
out.print(balance);
   }
}
```

استفاده از کلاس Account بهبود یافته :در مثال ٤-٧ در زیر :

```
public class UseAccount {
public static void main(String args[]) {
      Account myAccount = new Account();
      Account yourAccount = new Account();
      myAccount.name = " HadiKhodapanah";
myAccount.address = "222 Cyberspace Lane";
myAccount.balance = 24.02;
      yourAccount.name = "Ali Alizadeh ";
yourAccount.address = "111 Consumer Street";
yourAccount.balance = 55.63;
myAccount.display();
System.out.println();
yourAccount.display();
   }
}
```

در مثال ۳-۷ کلاس Account چهار چیز دارد— ***name , address , balance ,*** متد ***display*** -- هر مثالی از کلاس Account یک name یک address یک balance و یک روش برای نمایش خود دارد. راهی که شما برای صدا زدن و فراخوانی این چیز ها استفاده میکنید بسیار زیبا و منحصر به فرد است . برای رجوع به nameذخیره شده در myAccount میتوانید چنین بنویسید

```
myAccount.name
```

یا برای آنکه myAccount خودش را روی صفحه ی نمایش چاپ کند میتوانید بنویسید:

```
myAccount.display()
```

توجه : زمانی که شما یک method را فراخوانی میکنید، بعد از نام متد یک جفت پارانتز قرار می دهید.

فرستادن و گرفتن مقدار به Method ها :

هنگامی که شما هنگام فراخوانی متد ها، مقادیری را داخل پارنتز قرار میدهید و با این کار آن مقادیر را به متد مورد نظر ارسال می کنید. و به این مقادیری هنگام فراخوانی متد داخل پارانتز های متد قرار می دهید پارامتر (parameters) نامیده می شوند.

و به مقادیر که به شما برگشت داده میشوند مقادیر برگشتی یا return value نامیده میشوند، و عموما به انواعی که توسط متد برگردانده میشوند انواع برگشتی یا return type گفته میشود. مثال ۵-۷ :

```
import static java.lang.System.out;
public class Account {
   String name;
   String address;
double balance;
public void display() {
out.print(name);
out.print(" (");
out.print(address);
out.print(") has $");
out.print(balance);
   }
public double getInterest(double percentageRate) {
return balance * percentageRate / 100.00;
   }
}
```

در ادامه همین مبحث ،، به مثال ۶-۷ توجه نمایید.

مثال ۶-۷ :

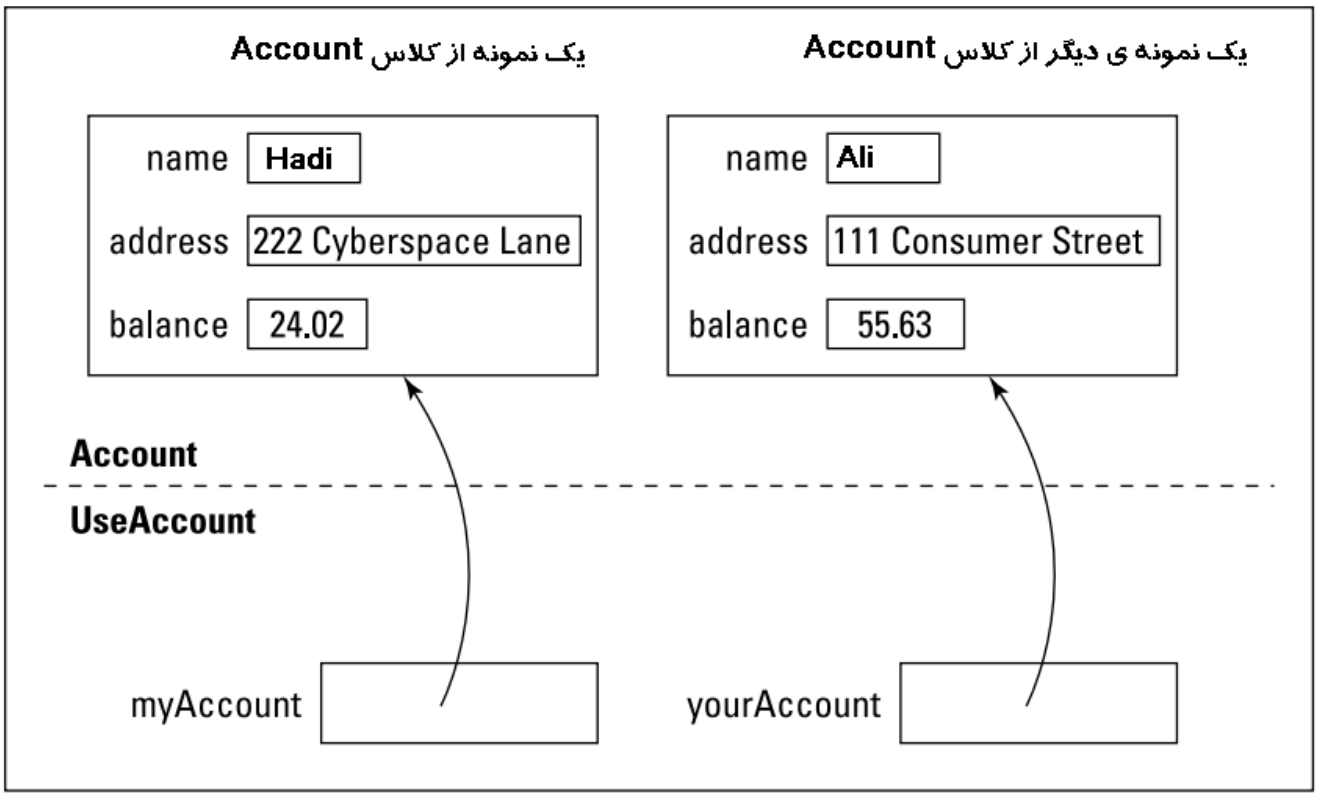
```
import static java.lang.System.out;
public class UseAccount {
public static void main(String args[]) {
      Account myAccount = new Account();
      Account yourAccount = new Account();
      myAccount.name = " HadiKhodapanah";
myAccount.address = "222 Cyberspace Lane";
myAccount.balance = 24.02;
      yourAccount.name = " AliAlizadeh";
yourAccount.address = "111 Consumer Street";
yourAccount.balance = 55.63;
myAccount.display();

out.print(" plus $");
out.print(myAccount.getInterest(5.00));
out.println(" interest ");
yourAccount.display();

doubleyourInterestRate = 7.00;
out.print(" plus $");
doubleyourInterestAmount =
yourAccount.getInterest(yourInterestRate);
out.print(yourInterestAmount);
out.println(" interest ");
   }
}
```

در مثال ۵-۷ ، کلاس Account دارای متد getInterest است. این متد getInterest دو بار از داخل متد اصلی(main) در مثال ۶-۷ فراخوانی شده است. که اولین فراخوانی لیترال ۵,۰۰ را بعنوان پارامتر داراست، و دومین فراخوانی نیز متغیر yourInterestRate را به عنوان پارامتر داراست. خروجی حاصل از اجرای مثالهای ۵-۷ و ۶-۷ :

```
Hadi Khodapanah (222 Cyberspace Lane) has $24.02 plus $1.2009999999999998 interest
Ali Alizadeh (111 Consumer Street) has $55.63 plus $3.8941000000000003 interest
```

در اولین فراخوانی ، نرخ balance برابر با ۲٤,۰۲ و نرخ interest برابر با ۵,۰۰ میباشد. و در دومین فراخوانی نرخ balance برابر ۵۵,۶۳ و نرخ interest برابر با ۷,۰۰ میباشد.

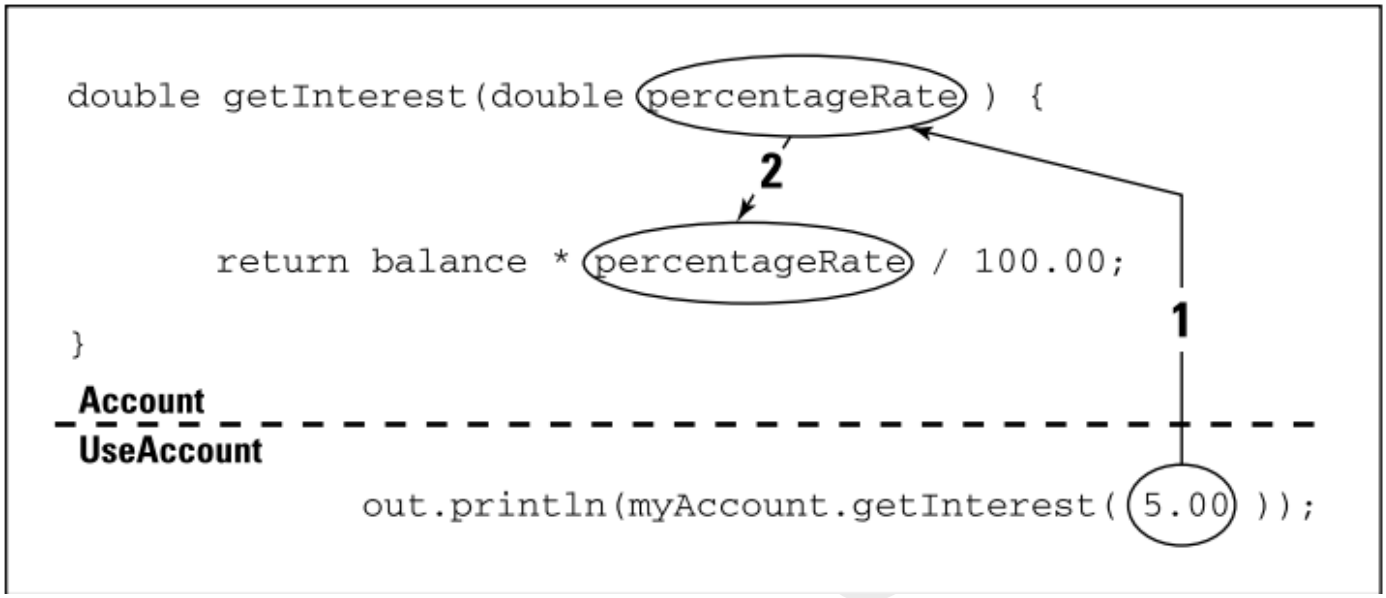
شکل زیر مطالب گفته شده را به صورت گرافیکی نشان می دهد .



کلمه ی getInterest نام یک متد بوده و پارانتز های آن شامل تمام چیزهایی میشود که شما هنگام فراخوانی متد قصد دارید به آ« ارسال کنید. و توجه کنید که ، کلمه ی double به کامپیوتر می گوید که

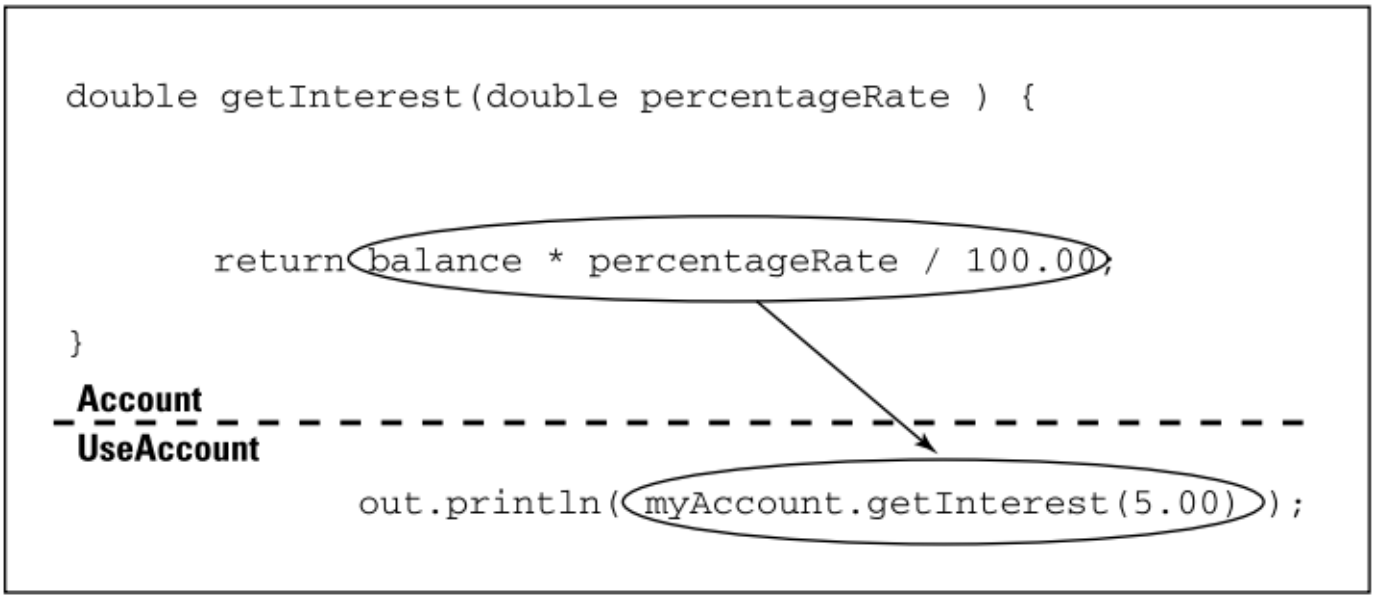
زمانی که متد getInterest فراخوانی شد . متد getInterest یک مقدار double به محلی که فراخوانی شده است ، بر میگرداند.

در فرستادن یک مقدار به یک متد به شکل ۶-۷ که در زیر آورده شده دقت کنید.



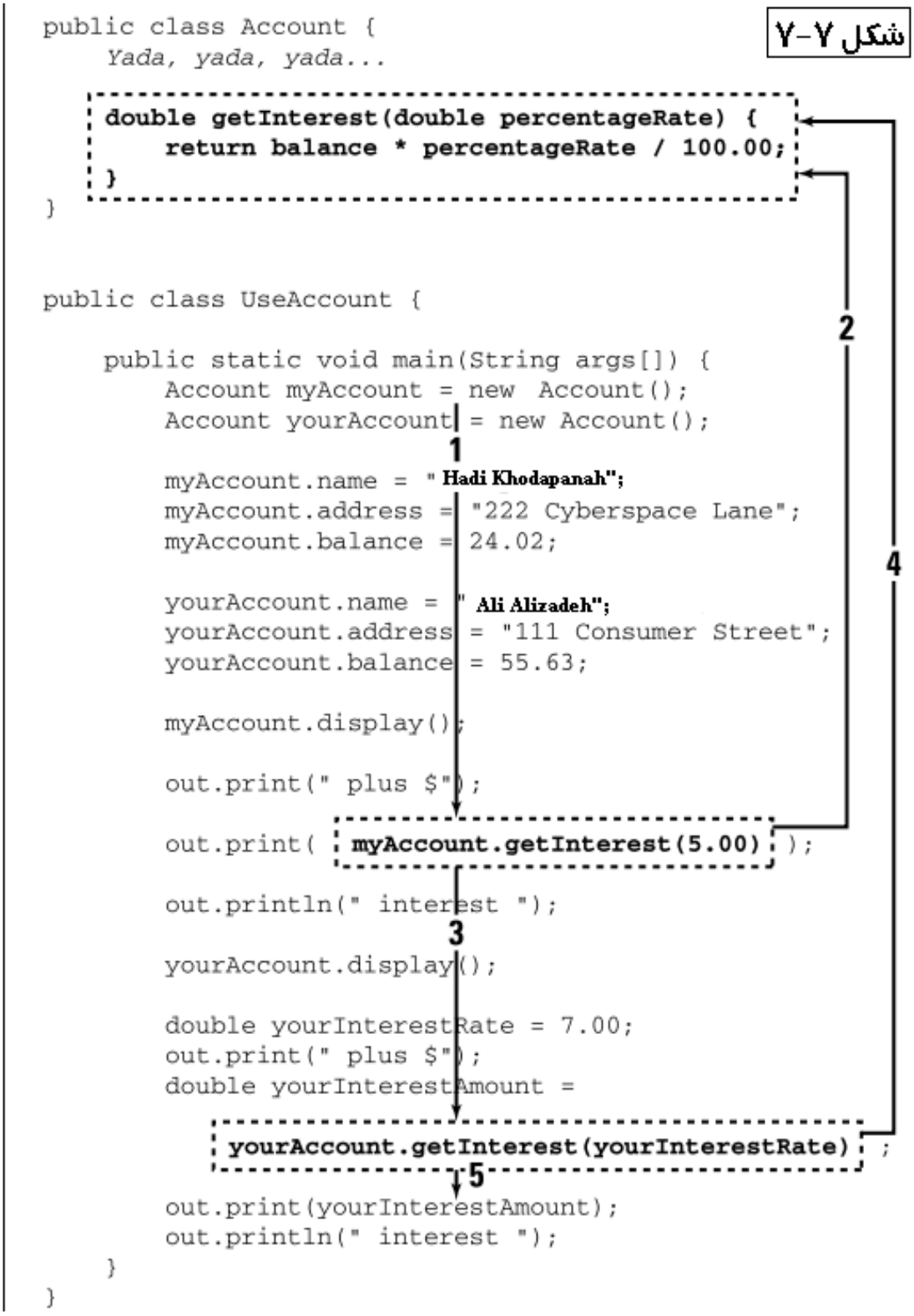
زمانی که شما مثال های ۵-۷ و ۶-۷ را اجرا می کنید، جریان اجزا شدن برنامه از بالا به طرف پایین نیست، بلکه به این صورت است که برنامه ابتدا از main به getInterest میرود و سپس به main برگشته و باز به getInterest میرود و در نهایت به main برمیگردد. . شکل ۷-۷ این موضوع را بهتر نشان میدهد.

و اما در مورد برگشت یک مقدار از یک متد: کل کاری که انجام میشود پایین شکل ۸-۷ توضیح داده شده :



مثلا وقتی متد getInterest فراخوانی میشود ، متد دستور return که داخل بدنه ی متد هست را اجرا میکند. ابتدا مقدار عبارت balance * percentageRate / 100.00 را محاسبه شده و سپس دستور return اجرا میشود، کاری که دستور return انجام میدهد این است که مقدار محاسبه شده را به جایی از main که در آنجا getInterest فراخوانی شده بود بر میگرداند. دوباره به شکل بالا دقت کنید.

شکل ۷-۷


public class Account {
Yada, yada, yada...
double getInterest(double percentageRate) {
return balance * percentageRate / 100.00;
}
}
public class UseAccount {
public static void main(String args[]) {
Account myAccount = new Account();
Account yourAccount = new Account();
myAccount.name = " Hadi Khodapanah";
myAccount.address = "222 Cyberspace Lane";
myAccount.balance = 24.02;
yourAccount.name = " Ali Alizadeh";
yourAccount.address = "111 Consumer Street";
yourAccount.balance = 55.63;
myAccount.display();
out.print(" plus $");
out.print(myAccount.getInterest(5.00));
out.println(" interest ");
yourAccount.display();
double yourInterestRate = 7.00;
out.print(" plus $");
double yourInterestAmount =
yourAccount.getInterest(yourInterestRate) ;
out.print(yourInterestAmount);
out.println(" interest ");
}
}
1
2
3
4
5

خطا

دوباره به خروجی حاصل از اجرای مثال های ۵-۷ و ۶-۷ نگاه کنید: مقدار interest حساب من فقط $۱,۲۰۰۹۹۹۹۹۹۹۹۹۹۸ میباشد. در حالی که این مقدار میبایست ۱,۲۰۱ میشد، واضح است که خطایی رخ داده است . این خطا به این خاطر ایجاد شده است که کامپیوتر از صفر و یک استفاده میکند و فضای بینهایت برای انجام محاسبات در اختیار ندارد. سریع ترین راحل برای به دست آوردن اعدادی بدون این گونه خطا ها استفاده از روش و طریقه ی حساس تری است.شما می توانید اعداد را گرد کنید و فقط دو رقم بعد از ممیز را نشان دهید، یا اینکه از برخی ابزار های موجود در API(Application Programming Interface) می توانید استفاده کنید مثال ۷-۷ موضوع گفته شده را روشن تر می سازد :

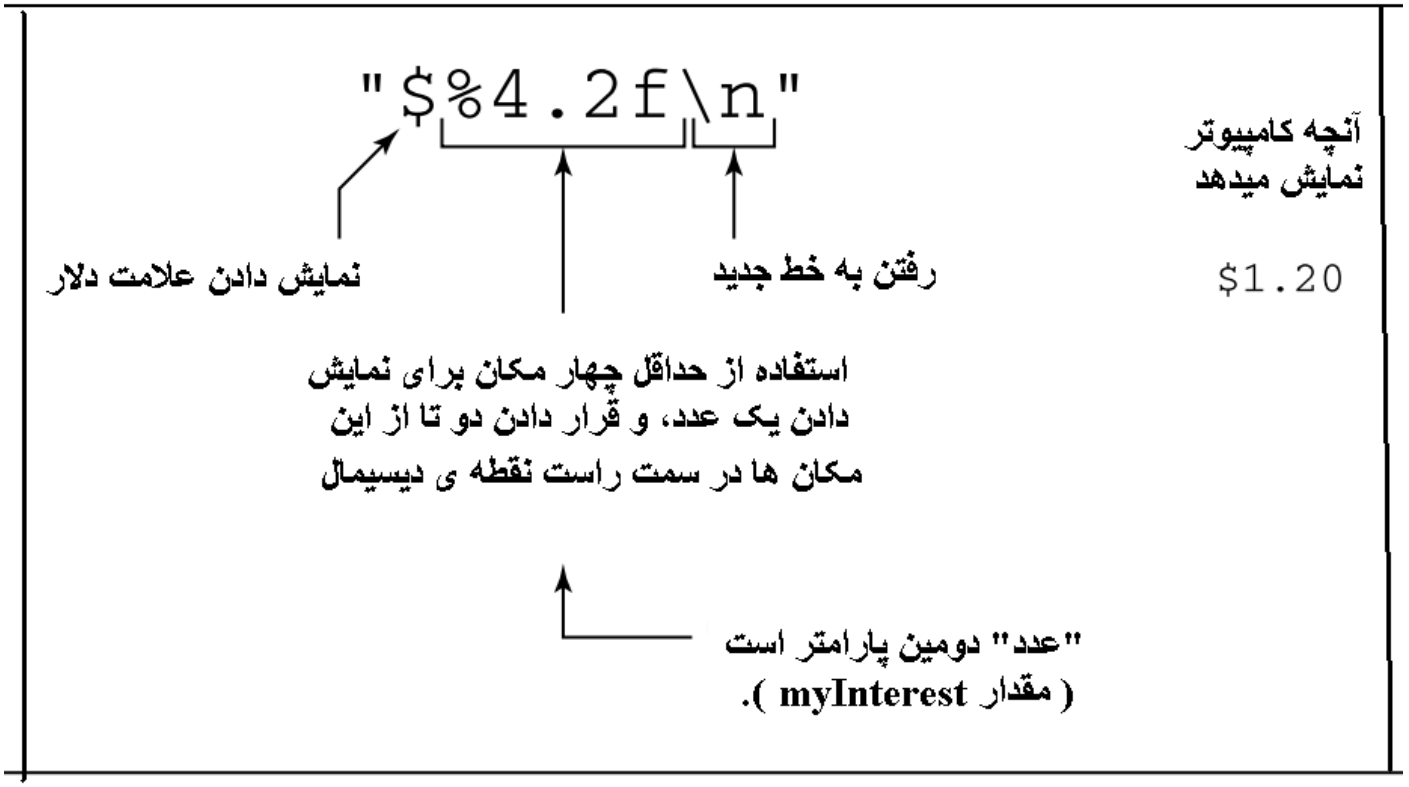
```
import static java.lang.System.out;
public class UseAccount {
public static void main(String args[]) {
      Account myAccount = new Account();
      Account yourAccount = new Account();
myAccount.balance = 24.02;
yourAccount.balance = 55.63;
doublemyInterest = myAccount.getInterest(5.00);
doubleyourInterest = yourAccount.getInterest(7.00);
out.printf("$%4.2f\n", myInterest);
out.printf("$%5.2f\n", myInterest);
out.printf("$%.2f\n",  myInterest);
out.printf("$%3.2f\n", myInterest);
out.printf("$%.2f $%.2f",myInterest, yourInterest);
   }
}
```

خروجی :

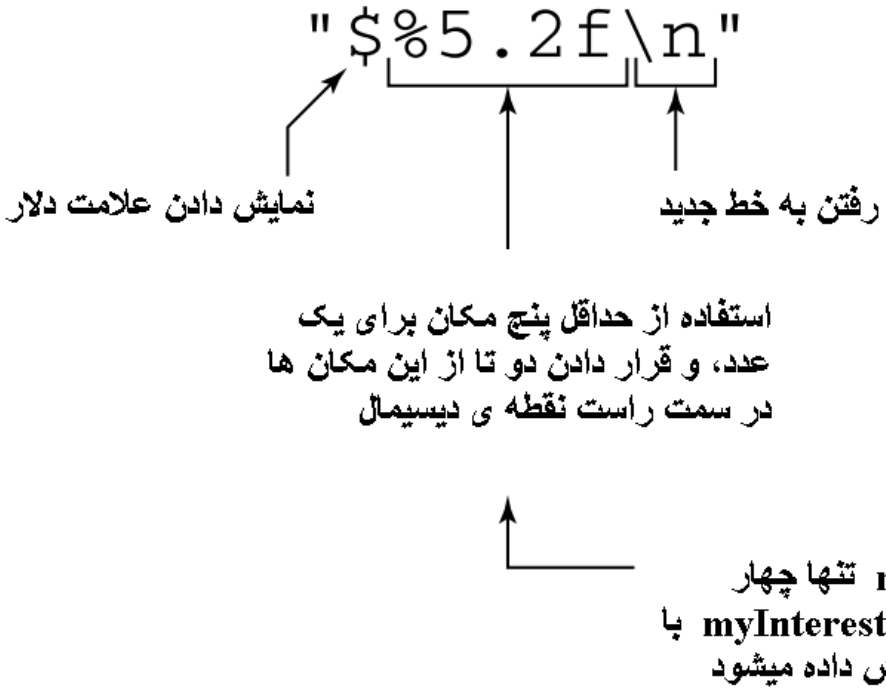
در مثال بال شما از متد printf استفاده کردید، وقتی که شما printf را فراخوانی میکنید، همیشه باید حداقل دو پارامتر داخل پارانتزهای آن قرار دهید. پارامتر اول فرمت رشته (format string) است. فرمت رشته دقیقا چگونگی نمایش پارامتر های دیگر را تعیین میکند. همه ی پارامتر های دیگر(بعد از اولین پارامتر) مقادیری هستند که نمایش داده میشوند.

به آخرین فراخوانی printf دا مثال ۷-۷ دقت کنید، اولین پارامتر دارای دو مکان نگه دارنده (placeholder) است. اولین placeholder(%.2f) نمایش دادن myInterest را توصیف میکند. و دومین placeholder (%.2f) نیز نمایش دادن yourInterest دا توصیف میکند. برای اینکه بفهمید این فرمت رشته ها دقیقا چگونه کار میکنند به شکل ۷-۱۰ تا ۷-۱٤ دقت کنید:

استفاده از format string :



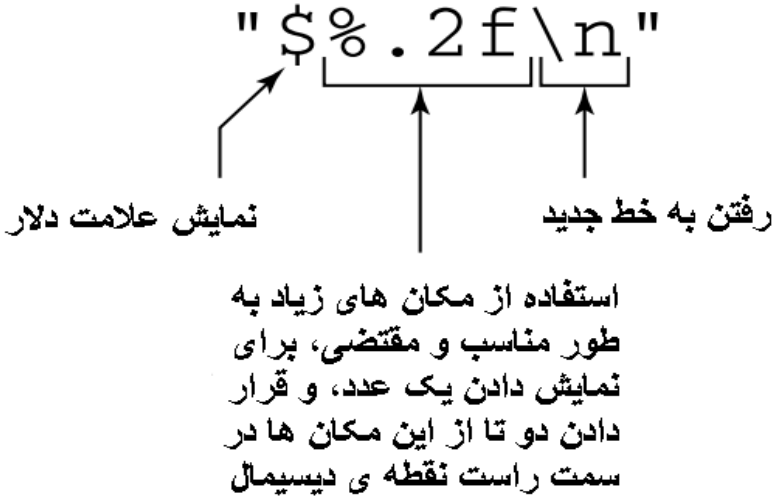
اضافه کردن فضای اضافی برای نمایش مقدار :



آنچه کامپیوتر نمایش میدهد:

```
$ 1.20
```

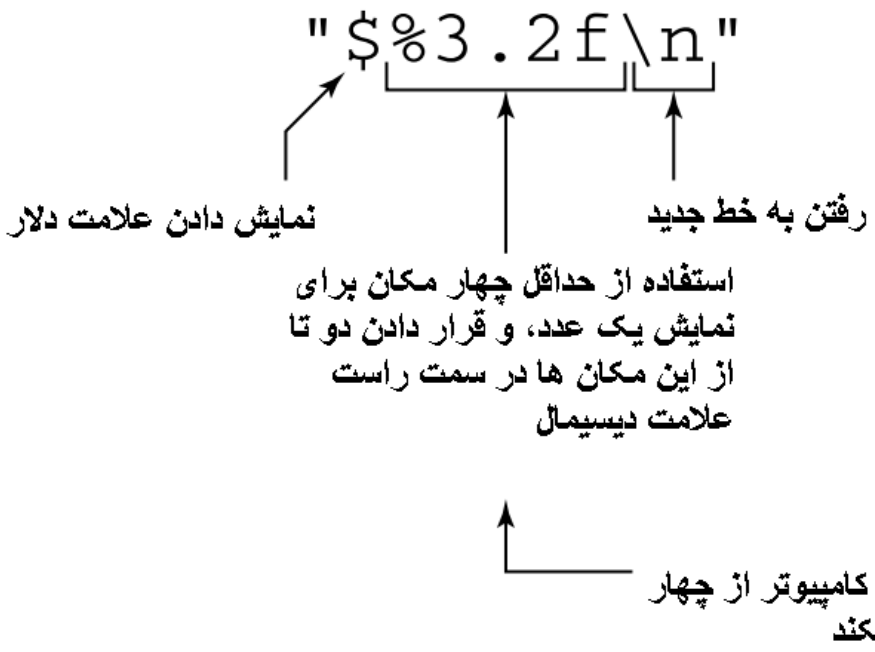
نمایش دادن یک مقدار، بدون مشخص کردن مقدار دقیق مکان های تخصیص داده شده به آن :



آنچه کامپیوتر نمایش میدهد:

```
$1.20
```

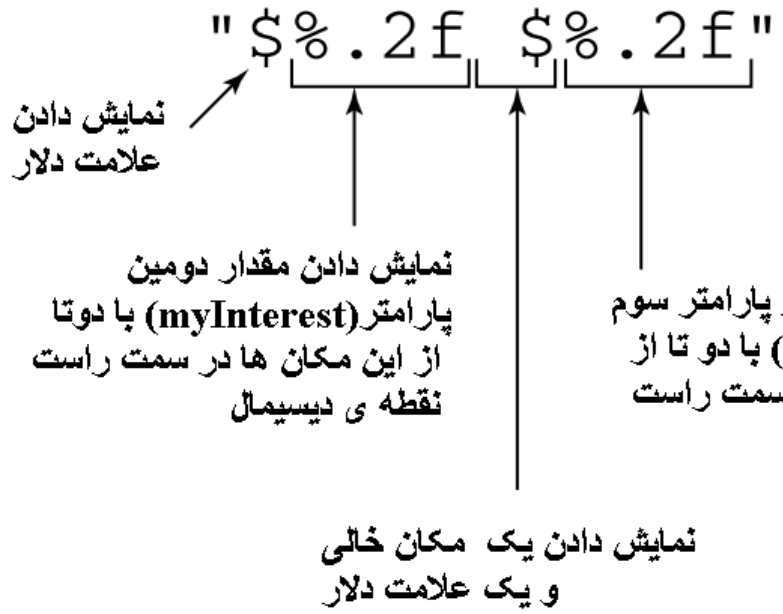
اختصاص دادن دو مکان کمتر ، برای نمایش یک مقدار :



آنچه کامپیوتر نمایش میدهد :

```
$1.20
```

نمایش دادن بیش از یک مقدار :



آنچه کامپیوتر نمایش میدهد:

```
$1.20  $3.89
```

توجه داشته باشید که یک فراخوانی printf شیوه ای که اعداد برای محاسبات ذخیره شده اند را تغییر نمیدهد، کاری که در تصاویر بالا آمده است، این است که دسته ای از کاراکتر ها عددی را را زیبا تر(nice-looking) کنیم.، تا بتوانند رو مانیتور نمایش داده شوند.

برنامه نویسی خوب (Good programming):

فرض کنید شما در حال نوشتن کد برای کلاس UseAccount هستید.آیا شما بدون نوشتن myAccount.name یا yourAccount.balance ، آیا میتوانید کاری را انجام دهید؟؟ جواب استفاده از چیزی به نام متدهای دسترسی دارنده (accessor methods) میباشد . مثال های ۸-۷ و ۹-۷ این متد ها را شرح میدهند :

مخفی کردن فیلد ها :

```
public class Account {
private String name;
private String address;
private double balance;
public void setName(String n) {
name = n;
   }
public String getName() {
return name;
   }
public void setAddress(String a) {
address = a;
   }
public String getAddress() {
return address;
   }
public void setBalance(double b) {
balance = b;
   }
public double getBalance() {
return balance;
   }
}
```

فراخوانی Accessor Methods :

به نظر میرسد حاصل اجرای مثال های ۷-۸ و ۷-۹ که هیچ تفاوتی با اجرای مثال های ۷-۱ و ۷-۲ ندارد. حاصل اجرای هر دو برنامه یکی ستوتفوات بزرگ این است که در مثال ۷-۸ کلاس Account به دقت اجرای فیلدهای name و address و balance را کنترل می کند.

دوباره به مثال ۷-۸ رجوع کرده و به متد setName نگاهی بیاندازید. تصور کنید که عبارت تخصیص داخل متد را درون یک if قرار دهیم :

```
1  public void setName(String n) {
2      if (!n.equals("")) {
3          name = n;
4      }
5  }
```

حال برنامه نویس متصدی نوشتن کلاس UseAccount کدی مانند myAccount.setName("") را مینویسد، فراخوانی setName هیچ تاثیر با نتیجه ای ندارد ندارد.. از این گذشته، چون فیلد name یک فیلد خصوصی (private) است ، عبارت زیر در کلاس UseAccount مجاز نیست::

```
import static java.lang.System.out;
public class UseAccount {
public static void main(String args[]) {
      Account myAccount = new Account();
      Account yourAccount = new Account();
myAccount.setName("HadiKhodapanah");
myAccount.setAddress("222 Cyberspace Lane");
myAccount.setBalance(24.02);
yourAccount.setName("Ali Alizadeh");
yourAccount.setAddress("111 Consumer Street");
yourAccount.setBalance(55.63);
out.print(myAccount.getName());
out.print(" (");
out.print(myAccount.getAddress());
out.print(") has $");
out.print(myAccount.getBalance());
out.println();
out.print(yourAccount.getName());
out.print(" (");
out.print(yourAccount.getAddress());
out.print(") has $");
out.print(yourAccount.getBalance());
   }
}
```

البته فراخوانی هایی همچون myAccount.setName("Joe Schmoe") هنوز کار میکنند زیرا "Joe Schmoe" برابر با رشته ی خالی "" نیست.با استفاده از فیلد های خصوصی(private) و متدهای accessor شما قادر خواهید بود از اینکه کسی یک رشته ی خالی برای فیلد nameتخصیص دهد جلوگیری کنید. یا حتی میتوانید با استفاده از دستورات if بیشتر ، قواعد بیشتری را روی آنچه کاربر وارد میکند اعمال کنید.

فصل ۶

با فکر کردن درباره ی داده ها(data) ، شما شروع به نوشتن برنامه ای شیءگرا میکنید. وقتی درباره ی حساب ها(accounts) کدی مینویسید، سوالی مطرح میشود که: اساسا یک account چیست؟؟ شما کدی برای مدیریت کلیک روی دکمه ها(button) مینویسید، سوال این است که : دکمه چیست؟؟ زمانی که شما برنامه ای در مورد فرستادن لیست حقوق و دستمزد به کارکنان(employees) مینویسید، کارمند در برنامه چیست؟؟

در مثال ۱-۸ یک کارمند کسی هست با یک نام و یک عنوان شغلی. یقینا یک کارمند مشخصات و خصوصیات دیگری هم دارد، ولی ما برای ساده شدن کار از آنها صرف نظر میکنیم، مثال ۱-۸ معین میکند که یک کارمند بودن چه معنی ای میدهد:

```
import static java.lang.System.out;
public class Employee {
private String name;
private String jobTitle;
public void setName(String nameIn) {
name = nameIn;
   }
public String getName() {
return name;
   }
public void setJobTitle(String jobTitleIn) {
jobTitle = jobTitleIn;
   }
public String getJobTitle() {
returnjobTitle;
   }
public void cutCheck(double amountPaid) {
out.printf("Pay to the order of %s ", name);
out.printf("(%s) ***$", jobTitle);
out.printf("%,.2f\n", amountPaid);
   }
}
```

هر کارمند(employee) خصیصه هایی دارد، که دوتا از این خصیصه ها بدون تردید بسیار ساده اند. هر کارمند یک نام(name) و یک عنوان شغلی(jobTitle) دارد.(در مثال ۱-۸ کلاس Employee یک فیلد name و یک فیلد jobTitle دارد).

یک کارمند (employee) چهار متد برای مدیریت مقادیر نام کارمند و عنوان شغلی دارد، که این متد ها عبارت اند از : setName و getName و setJobTitle و getJobTitle .

بعلاوه هر کارمند (employee) یک متد cutCheck دارد. مقصود این است که متدی که حقوق و دستمزد کارمندان را مینویسد بایستی به یک کلاس تعلق داشته باشد. دلیل این امر آن است که ، بیشتر اطلاعاتی که داخل لیست حقوق و دستمزد نوشته میشود، برای هر کارمند متفاوت است. جزئیات این مثال در ادامه توضیح داده خواهد شد.

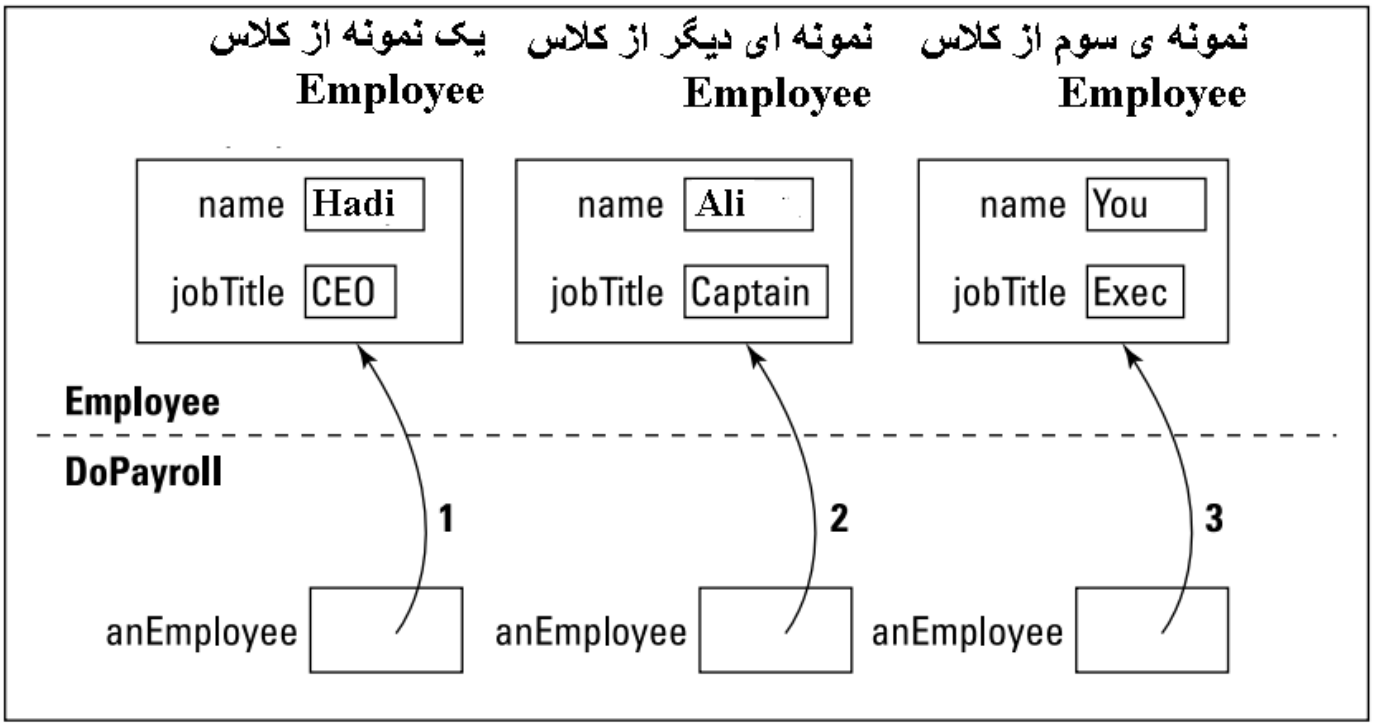
کلاس Employee در مثال ۱-۸ هیچ متد main ای نداشت. پس هیچ نقطه ی شروعی برای اجرای کد وجود ندارد. برای برطرف کردن این نقص برنامه نویس یک برنامه ی جداگانه با یک متد main مینویسد، و از آن برنامه برای ایجاد یک شی ء نمونه از Employee استفاده میکند. مثال ۲-۸ یک کلاس دارای متد main را نشان میدهد.

```
importjava.util.Scanner;
importjava.io.File;
importjava.io.IOException;
public class DoPayroll {
public static void main(String args[])
throwsIOException {
      Scanner diskScanner =
new Scanner(new File("EmployeeInfo.txt"));
for (intempNum = 1; empNum<= 3; empNum++) {
payOneEmployee(diskScanner);
      }
diskScanner.close();
   }
static void payOneEmployee(Scanner aScanner) {
      Employee anEmployee = new Employee();
anEmployee.setName(aScanner.nextLine());
anEmployee.setJobTitle(aScanner.nextLine());
anEmployee.cutCheck(aScanner.nextDouble());
aScanner.nextLine();
   }
}
```

برای اجرا کردن مثال ۲-۸ هارد دیسک شما بایستی دارای فایلی به نام EmployeeInfo.txt داشته باشد.

کلاس DoPayroll در مثال ۲-۸ دارای دو متد میباشد. یکی از متد ها(main) متد دیگر(payOneEmployee) را سه بار فراخوانی میکند. در هر بار متد payOneEmployee یک سری داده ی اولیه از EmployeeInfo.txt دریافت کرده و آن ها را به متد های کلاس Employee میفرستد.

در زمان اولین فراخوانی payOneEmployee عبارت anEmployee = new Employee() باعث میشود که anEmployee به شی جدیدی منسوب شده و به آن اشاره کند. زمانی که payOneEmployee برای دومین بار فراخوانی میشود، کامپیوتر همان عبارت را دوباره اجرا میکند .این اجرای دوباره باعث ایجاد یک تجسم خارجی جدیدی از متغیر anEmployeeشده ، که به شیء کاملا جدید ی منسوب و به آن شیء اشاره میکند. در اجرای سوم همه چیز مثل قبل دوباره تکرار میشود. یک متغیر anEmployee جدید به شی سوم منسوب شده و به آن رجوع میکند. ممکن است . این موضوع ممکن است کمی گیج کننده به نظر برسد ولی شکل ۱-۸ همه چیز به خوبی روشن میسازد:



مثال ۱-۸ سه فراخوانی printf دارد. هر printf فرمت رشته ای (format string) خود را صدا میزند (مانند ("$*** (s%)") و یک متغیر (مانند jobTitle) . هر فرمت رشته ای یک نگه دارنده مکان (placeholder) مانند %s دارد که مشخص میکند که کجا و چگونه مقادی ر متغیر ها نمایش داده شوند.

برای مثال دومین فراخوانی printf ، فرمت رشته دارای یک نگه دارنده ی جاء %s میباشد. این %s

```
Pay to the order of Hadi Khodapanah (CEO) ***$5,000.00
Pay to the order of Ali Alizadeh (Captain) ***$7,000.00
Pay to the order of Your Name Here (Honorary Exec of the Day) ***$10,000.00
```

مکانی را برای مقدار متغیر jobTitle نگه میدارد.در مثال ۱-۸ بر اساس قواعد جاوا ، نماد %s همیشه مکان ای را برای رشته نگه میدارد و میتوان بقین کرد که متقیر jobTitle از نوع String اعلان شده است. پارانتز ها در اطراف هر عنوان شغلی در خروجی برنامه وجود دارد . شکل ۲-۸ :

در مثال ۱-۸ به ویرگول داخل نگه دارنده ی مکان ٪,.۲ توجه کنید. ویرگول یه برنامه می گوید که از جداکننده های گروه بند(grouping separators) استفاده کند. به همین دلیل است که شما در شکل ۲-۸ ، ۵،۰۰۰,۰۰$ و ۷،۰۰۰$ و ۱۰،۰۰۰,۰۰$ را به جای ۵۰۰۰,۰۰$ و ۷۰۰۰,۰۰$ و ۱۰۰۰۰,۰۰$میبینید.

جداکننده های گروه بند در کشور های مختلف متفاوت اند. برای مثال در فرانسه برای نوشتن یک هزار (مایل) شما ۱۰۰۰,۰۰ را مینویسید. جاوا میتوانید به طور اتوماتیک اعداد شما را فرانسوی کند البته با کمک عباراتی همچون :

```
out.print(new,((java.util.Formatter().format(java.util.Locale.FRANCE,"%,.2f", 1000.00 ))
```

برای جزئیات بیشتر به (Application Programming) InterfaceAPIبخش مربوط به کلاس های Formatter و Locale رجوع کنید.

ذخیره کردن داده ها در فایل

کد های در مثال ۲- ۸ اجرا نمیشوند مگر آنکه شما بعضی اطلاعات کارمند(employee) را داخل یک فایل داشته باشید. مثال ۲-۸ میگوید این فایل EmployeeInfo.txt نام دارد. پس قبل از اجرای مثال ۲-۸ یک فایل کوچک EmployeeInfo.txt ایجاد کنید. شکل ۳-۸ :

```
Hadi Khodapanah
CEO
5000.00
Ali Alizadeh
Captain
7000.00
Your Name Here
Honorary Exec of the Day
10000.00
```

در مثال ۲-۸ ، من بر این نکته تکیه کرده ام که، زمانی که شما کاراکتر ها را در(مانند) شکل ۳-۸ تایپ میکنید، شما فایل را با تایپ کردن ۱۰۰۰۰,۰۰ به پایان میبرید.

و سپس Enter را فشار میدهید(دوباره به شکل ۳-۸ نگاه کنید چگونه مکان نما در ابتدای خط جدید قرار دارد). اگر شما فشار دادن Enter را فراموش کنید، آنگاه زمانی که تلاش کنید مثال ۲-۸ را اجرا کنید با مشکلاتی مواجه خواهید شد.

گروه بندی جدا کننده ها ، در کشور های مختلف، متفاوت است. فایل نشان داده شده در شکل ۳-۸ روی یک کامپیوتری که در کشور ایالات متحده پیکره بندی شده است، کار میکند. جایی که ۵۰۰۰,۰۰ به معنی پنج هزار است. ولی در بعضی کشورها که ۵۰۰۰,۰۰ به معنی پنج هزار است، به درستی کار نخواهد کرد.(دقت کتید که در عدد اول نقطه . ما بین صفر ها قرار گرفته درحالی که در عدد دوم ویرگول).

اگر شما در چنین کشوری زندگی میکنید. و از فایلی که در شکل ۳-۸ نشان داده شده است استفاده میکنید، زمانی که سعی در اجرای مثال این بخش داشته باشید احتمالا یک پیام خطا دریافت خواهید کرد(یک InputMismatchException). برای حل این مشکل اعداد موجود در فایل داده را ، به فرمتی رایج در کشور خود تغییر دهید ، تا خطا مورد بحث را رفع نمایید.

Copy و Past کردن کد :

تقریبا در بیشتر زبان های برنامه نویسی نوشتن و خواندن داده ها رو ی فایل می تواند دشوار و نیازمند مهارت باشد. شما خطوط کد اضافی را به برنامه اضافه میکنید تا به کامپیوتر بگویید چه کاری را انجام دهد. در برخی مواقه شما میتوانید این کد ها را از کدهای برنامه های دیگر مردم کپی کنید. برای مثال میتوانید از الگوی ارائه شده در مثال ۲-۸ پیروی کنید :

شما میخواهید از روی فایل اطلاعاتی را بخوانبد. با تصور اینکه دارید از روی کیبورد اطلاعاتی را میخوانید شروع کنید. کد دستورات Scanner ها و نیز کد دستورات next معمول را داخل برنامه ی خود قرار دهید. سپس آیتم های اضافی را به الگوی مثال ۲-۸ اضافه کنید:

دو اعلان import جدید اضافه کنید. یکی برای java.io.File و دیگری برای java.io.IOException .

در هدر(header) متد خود ، **throws IOException** را تایپ کنید.

```
/*
 * الگوی استفاده شده در مثال 8-2 *
 */
importjava.util.Scanner;
importjava.io.File;
importjava.io.IOException;
classSomeClassName {
public static void main(String args[])
throwsIOException {
    Scanner scannerName =
new Scanner(new File("SomeFileName"));
    // محل قرار دادن قسمتی ار کد ها
scannerName.nextInt();
scannerName.nextDouble();
scannerName.next();
scannerName.nextLine();
    //محل قرار دادن قسمت دیگری از کدها
scannerName.close();
  }
}
```

در فراخوانی new Scanner عبارت new File("") را تایپ کنید.

فایلی که هم اکنون داخل کامپیوتر شما وجود دارد را در نظر بگیرید و نام آن را داخل علامت های کوتیشن بنویسید.

کلمه ای که شما برای اسم اسکنر(scanner) تان به کار بردید را به خاطر داشته باشید. از همان کلمه برای فراخوانی next و nextInt و nextDouble و غیره استفاده کنید..در ضمن از همان کلمه برای فراخوانی close استفاده کنید.

البته بعضی اوقات، کپی کردن کدها و جاگذاری آن ها در برنامه ی خودتان میتواند شما را به دردسر بیاندازد.شاید شما دارید برنامه ای میتویسید که با الگوی مثال ۸-۲ مطابقت ندارد و شما نیاز دارید که کمی در الگوی برنانه را دستکاری کرده و تغییر دهید.اما برای تغییر الگو شما باید ایده ی پشت الگو را دقیقا درک کرده باشید.

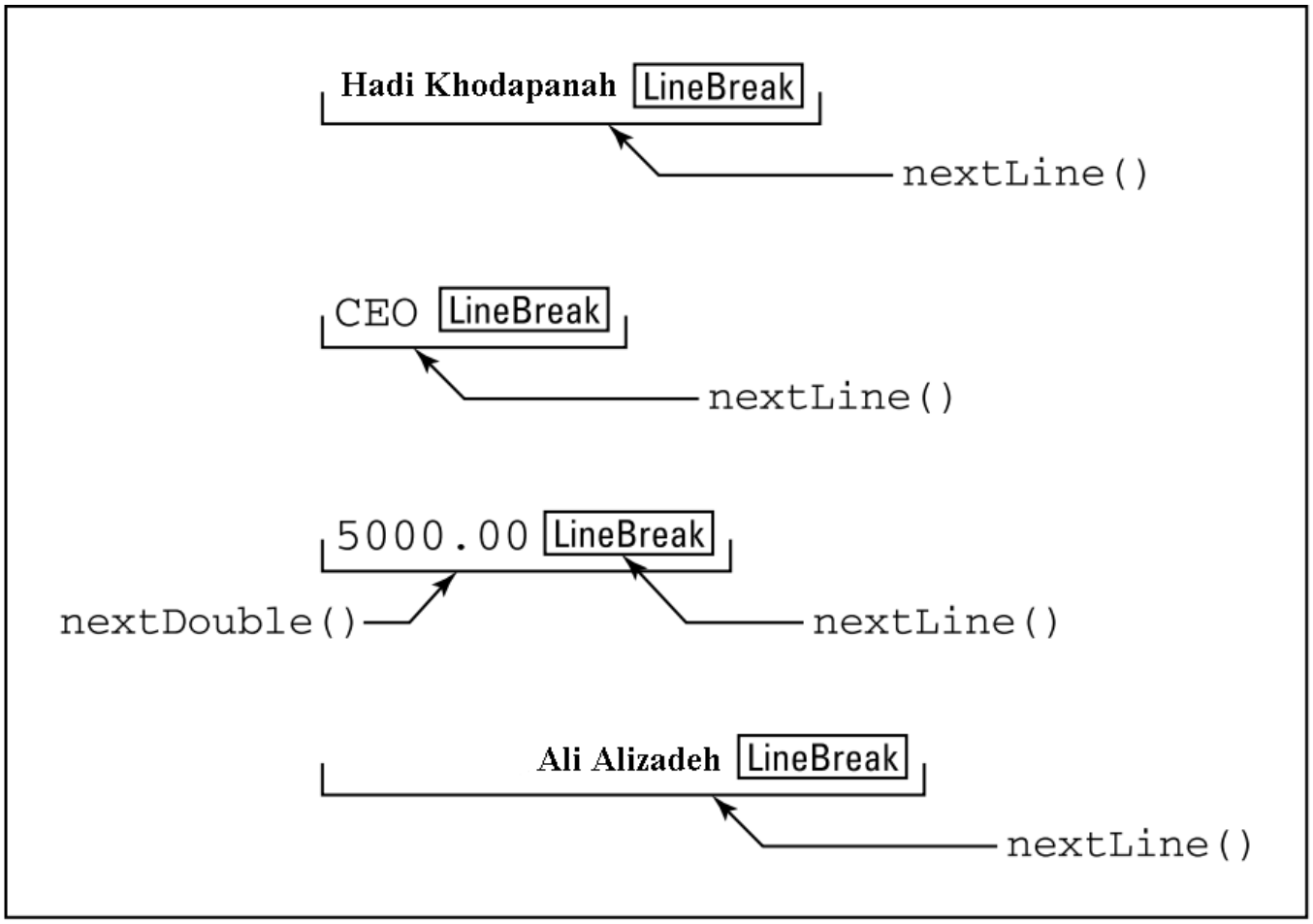
خواندن یک خط در هر دفعه

در مثال ۲-۸ متد payOneEmployee بعضی فنون مفید خواندن داده ها را توضیح میدهد. در عمل، هر اسکنری که شما ایجاد کرده اید یک متد nextLine دارد.(شما ممکن است از این متد nextLine استفاده نکنید، ولی به هر حال این متد در دسترس است.). زمانی که شما متد nextLine اسکنر را فراخوانی میکنید، این متد تا تمام شدن خط کنونی ، هر چیزی را گرفته و نگه میدارد. در مثال ۲-۸ یه فراخوانی به nextLine میتواند کل یک خط را، از فایل EmployeeInfo.txt بخواند.(در برنامه ی دیگر، فراخوانی nextLine اسکنر، امکان دارد که هر هر چیزی که کاریر روی کیبورد تایپ میکند را تا فشار دادن دکمه ی Enter بخواند). دوباره به کلمه ی nextLine دقت کنید: nextLine هر چبزی را تا انتهای خط کنونی میخواند. متاسفانه آن چیزی که خواندن تا انتهای خط کنونی همیشه آن چیزی که شما فکر میکنید، معنی نمی دهد

با آمیختن فراخوانی های nextInt و nextDouble و nextLine ممکن است برنامه کمی شلوغ و آشفته به نظر آید، پس شما بایدآن چیزی که در برنامه انجام میدهید را زیر نظر داشته باشید. و خروجی برنامه را به دقت چک کنید.

برای فهمیدن همه ی اینها،شما باید از چگونگی جدا شدن خطوط فایل داده آگاهی داشته باشید. جدا کننده ی یک خط از یک خط دیگر را به عنوان یک کاراتر اضافی در نظر بگیرید که بین دو خط جدا قرار میگیرد. در این صورت فراخوانی nextLine به معنی خواندن هر چیزی در خط کنونی فایل داده، تا انتهای خط (یعنی تا رسیدن به کاراکتر جداکننده است) میباشد.

حال به شکل ٤-٨ نگاه کنید:



اگر فراخوانی nextLine ، Hadi Khodapanah[LineBreak] را بخواند فراخوانی nextLine بعدی CEO[LineBreak] را خواهد خواند.

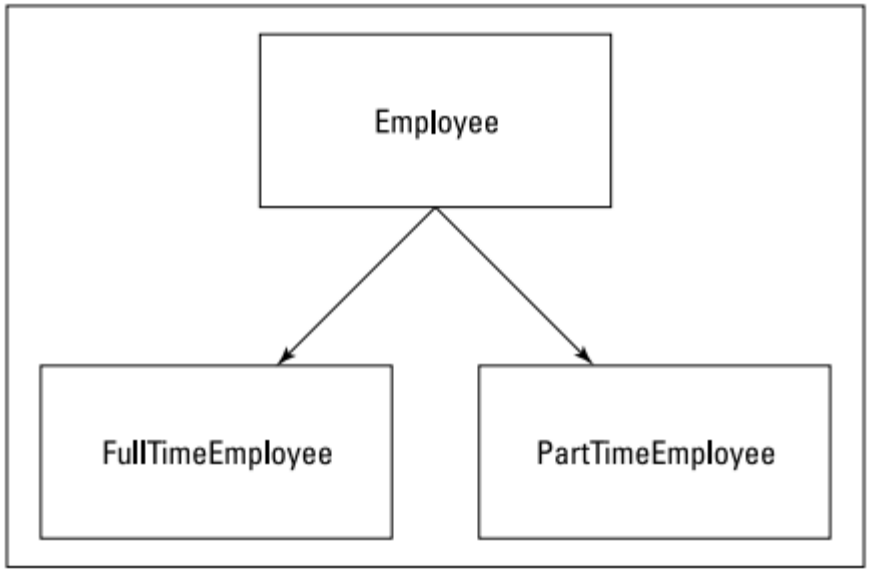
اگر فراخوانی nextDouble عدد ۵۰۰۰,۰۰ را بخواند ، فراخوانی بعدی nextLine ، [LineBreak] را خواهد خواند.

اگر فراخوانی nextLine ، [LineBreak] را بخواند. بعد از عدد ۵۰۰۰,۰۰ فراخوانی بعدی nextLine ، Ali Alizadeh[LineBreak] را خواهد خواند.

پس بعد از خواندن عدد ۵۰۰۰,۰۰ شما به دو فراخوانی nextLine ، به منظور جمع کردن نام Ali Alizadeh نیاز دارید، اشتباهی که معمولا دانشجویان مرتکب میشوند این است که اولین فراخوانی از این دو فراخوانی ر ا فراموش میکنند.

دوباره به شکل ۳-۸ نگاهی بیاندازید، برای مثال به درستی کار کند شما نیاز دارید که بعد از آخرین ۱۰۰۰۰,۰۰ یک جدا کننده ی خطوط (line break) داشته باشید. از آخرین فراخوانی nextLine را انجام ندهید ممکن است برنامه ی شما خراب شده و پیام خطایی به مضمون NoSuchElementException: No line found. دریافت نمایید

توجه به این نکته الزامی است که ، اولین nextLine که از فایل شکل ۳-۸ ، [Hadi Khodapanah[LineBreak را خوانده ، اما همین فراخوانی Hadi Khodapanah را بدون هیچگونه line break به کد اجرایی تحویل میدهد. بنابراین nextLine در جستجوی یک line break بوده و پس از پیدا کردن line break ، آن از دست میدهد. توجه نکردن به این نکته میتواند برنامه شما را با مشکلات جدی مواجه کند.



ایجاد یک زیر کلاس

در مثال ۱-۸ یک کلاس Employee تعریف شده است. شما میتوانید از آنچه که در مثال ۱-۸ تعریف شده استفاده کرده و یا آن را گسترش دهید. با این کار میتوانیدکلاس های جدیدتر و مخصوص تری را ایجاد کنید.بر این اساس در مثال ۳-۸ تصمیم گرفتن یک کلاس جدید به اسم FullTimeEmployee تعریف کنم :

```
public class FullTimeEmployee extends Employee {
private double weeklySalary;
private double benefitDeduction;
public void setWeeklySalary(double weeklySalaryIn) {
weeklySalary = weeklySalaryIn;
  }
public double getWeeklySalary() {
returnweeklySalary;
  }
public void setBenefitDeduction(double benefitDedIn) {
benefitDeduction = benefitDedIn;
  }
public double getBenefitDeduction() {
returnbenefitDeduction;
  }
public double findPaymentAmount() {
returnweeklySalary - benefitDeduction;
  }
}
```

به مثال ۳-۸ نگاه کنید. شما میتوانید ببینید که هر نمونه از کلاس FullTimeEmployee دارای دو فیلد weeklySalary و benefitDeduction میباشد. اما آیا آن ها تنها فیلد هایی هستند که هر نمونه از کلاس FullTimeEmployee دارد؟؟ نه ...خط اول از برنامه ۳-۸ میگوید که کلاس FullTimeEmployee توسعه یافته ی کلاس Employee میباشد. این بدین معنی است که علاوه بر داشتن دو فیلد weeklySalary و benefitDeduction هر نمونه از کلاس FullTimeEmployee دارای دو فیلد دیگر name و jobTitle میباشد. این دو فیلد از تعریف کلاس Employee حاصل میشوند.

زمانی که یک کلاس بر اساس گسترش دادن یک کلاسی که از قبل موجود است ایجاد میکنید، کلاس گسترش یافته به طور اتوماتیک (وظایف اساسی کارکردی) functionality ای در کلاس موجود قبلی

تعریف شده را به ارث میبرد. بنابراین کلاس FullTimeEmployee فیلد های name و jobTitle را به ارث میبرد. کلاس FullTimeEmployee همچنین تمام متد های در کلاس Employee اعلان گردیده اند را نیز به ارث می برد.(setJobTitle, getJobTitle, and cutCheck,setName, getName)....در واقع FullTimeEmployee بک زیر کلاس از کلاس Employee میباشد این بدان معنی است که کلاس Employee یک سوپر کلاس(superclass) برای کلاس FullTimeEmployee میباشد. شما همچنین میتوانید بگویید که کلاس FullTimeEmployee فرزند کلاس Employee میباشد و کلاس Employee والد کلاس FullTimeEmployee است. مثال ٤-٨ تقلبی و جعلی است اما میتواند آموزنده باشد.

ولی چرا کد مثال ٤-٨ تقلبی ومعرفی شد. در واقع تفاوت اصلی مابین مثال ٤-٨ و وضعیت .اراثت در مثال های ١-٨ و ٣-٨ این است که : کلاس فرزند نمیتواند مستقیم به فیلد های خصوصی کلاس والد رجوع داشته باشد. برای اینکه کلاس فرزند به فیلد های خصوصی کلاس والد دسترسی داشته باشد، کلاس فرزند بایستی ابتدا متد های دسترسی دارنده(accessor) ی کلاس والد را فراخوانی کند. به مثال ٣-٨ برگردید فراخوانی setName("Rufus") مجاز و قانونی است ولی دستور name="Rufus" مجاز نخواهد بود.

نکته : شما به فایل Employee.java روی هارد خود برای نوشتن کلاسی که توسعه یافته ی Employee باشد نیاز ندارید. تمام نیاز شما فایل Employee.class می باشد.

مثال ٤-٨ :

```
import static java.lang.System.out;
public class FullTimeEmployee {
private String name;
private String jobTitle;
private double weeklySalary;
private double benefitDeduction;
public void setName(String nameIn) {
name = nameIn;
   }
public String getName() {
return name;
   }
public void setJobTitle(String jobTitleIn) {
jobTitle = jobTitleIn;
   }
public String getJobTitle() {
returnjobTitle;
   }
public void setWeeklySalary(double weeklySalaryIn) {
weeklySalary = weeklySalaryIn;
   }
public double getWeeklySalary() {
returnweeklySalary;
   }
public void setBenefitDeduction(double benefitDedIn) {
benefitDeduction = benefitDedIn;
   }
public double getBenefitDeduction() {
returnbenefitDeduction;
   }
public double findPaymentAmount() {
returnweeklySalary - benefitDeduction;
   }
public void cutCheck(double amountPaid) {
out.printf("Pay to the order of %s ", name);
out.printf("(%s) ***$", jobTitle);
out.printf("%,.2f\n", amountPaid);
   }
}
```

ایجاد زیر کلاس ها شکل گیری عادت هاست

زمانی که شما با ایجاد زیر کلاس ها(کلاس های توسعه یافته) خوگرفتید. خلق زیر کلاس باعث ایجاد راحتی در برنامه هایی که می نویسید خواهد شد. اگر شما کلاس FullTimeEmployee را ایجاد کرده اید. پس میتوانید کلاس دیگری به اسم PartTimeEmployee را نیز ایجاد کنید که در شکل ۵-۸ ایجاد شده است.

```
public class PartTimeEmployee extends Employee {
private double hourlyRate;
public void setHourlyRate(double rateIn) {
hourlyRate = rateIn;
   }
public double getHourlyRate() {
returnhourlyRate;
   }
public double findPaymentAmount(int hours) {
returnhourlyRate * hours;
   }
}
```

بر خلاف کلاس FullTimeEmployee ، کلاس PartTimeEmployee هیچ حقوق و دستمزد(salary) و یا کسر (deduction)ندارد. در عوض PartTimeEmployee یک فیلد hourlyRate دارد.(بعلاوه ی یک فیلد numberOfHoursWorked که ممکن است در برنامه قرار داده شود..البته توجه داشته باشید که تعداد ساعات کاری یک کارمند پاره وقت از یک هفته به هفته ی دیگر (بسته به زیادی کار موجود) ممکن است متفاوت باشد.

استفاده از زیر کلاس ها

بخش های قبلی نحوه ی ایجاد زیر کلاس ها را توضیح می دهد، البته ی نکته ی مهم این است : ایجاد زیر کلاس ها به خودی خود مزیتی ندارد مگر آنکه شما کدی برای استفاده از آنها بنویسید. در این بخش کدی که از این زیرکلاس ها(subclasses) را توضیح خواهیم داد.

مثال ۶-۸ شما یه برنامه ی ساده ، که از زیر کلاس های FullTimeEmployee و PartTimeEmployee استفاده میکند، را نشان میدهد. و شکل ۶-۸ نیز خروجی این برنامه را نشان میدهد.

```
public class DoPayrollTypeF {
public static void main(String args[]) {
FullTimeEmployeeftEmployee = new FullTimeEmployee();
ftEmployee.setName("HadiKhodapanah");
ftEmployee.setJobTitle("CEO");
ftEmployee.setWeeklySalary(5000.00);
ftEmployee.setBenefitDeduction(500.00);
ftEmployee.cutCheck(ftEmployee.findPaymentAmount());
System.out.println();
PartTimeEmployeeptEmployee = new PartTimeEmployee();
ptEmployee.setName("Ali Alizadeh");
ptEmployee.setJobTitle("Driver");
ptEmployee.setHourlyRate(7.53);
ptEmployee.cutCheck   (ptEmployee.findPaymentAmount(10));
  }
}
```

```
Pay to the order of Hadi Khodapanah(CEO)  ***$4,500.00

Pay to the order of    Ali Alizadeh   (Driver)  ***$75.30
```

برای اینکه کاملا مثال ۶-۸ را متوجه شوید بایستی دوباره به زیرکلاس های Employee و FullTimeEmployee و PartTimeEmployee نگاهی بیاندازید.

نیمه ی نخست مثال ۶-۸ به یک کارمند تمام وقت می پردازد. توجه کنید که چگونه متد های زیادی با متغیر ftEmployee برای استفاده در دسترس هستند. برای مثال شما میتوانید ftEmployee.setWeeklySalary را فراخوانی کنیدزیرا که ftEmployee نوع FullTimeEmployee را داراست. شما همچنین میتوانید ftEmployee.setName را فراخوانی کنید، چونکه کلاس FullTimeEmployee کلاس Employee را گسترش میدهد.

برای اینکه cutCheck داخل کلاس Employee اعلان شده است شما میتوانید ftEmployee.cutCheck را فراخوانی کنید. شما همچنینی میتوانید ftEmployee.findPaymentAmount را نیز فراخوانی کنید زیراکه متدfindPaymentAmount داخل کلاس FullTimeEmployee میباشد.

تطابق انواع(types match)

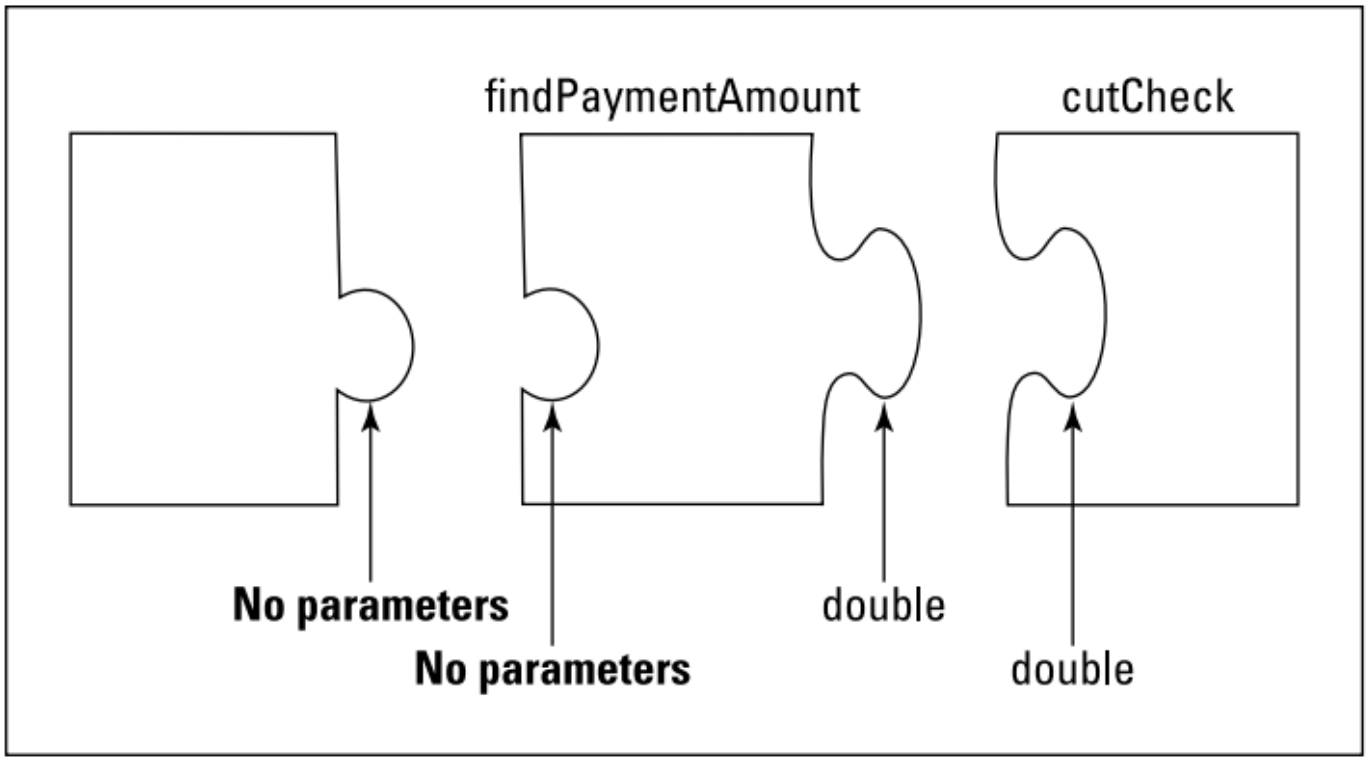
دوباره به نیمه ی نخست مثال ۶-۸ نگاه ی انداخته و به آخرین عبارت توجه توجه کنید:

متد ftEmployee.findPaymentAmount با لیست پارامتر خالی فراخوانی میشود. برای اینکه متد findPaymentAmount هیچ پارامتری را نمیپذیرد.(مثال ۳-۸) .

متد findPaymentAmount نوع داده ای double را برمیگرداند.

مقدار double ای که ftEmployee.findPaymentAmount برمیگرداند به متد ftEmployee.cutCheck پاس داده میشود. الیته متد cutCheck یک پارامتر از نوع double میگیرد.(مثال ۱-۸).

به شکل زیر دقت کنید:



تغییر دادن متد های قبلی

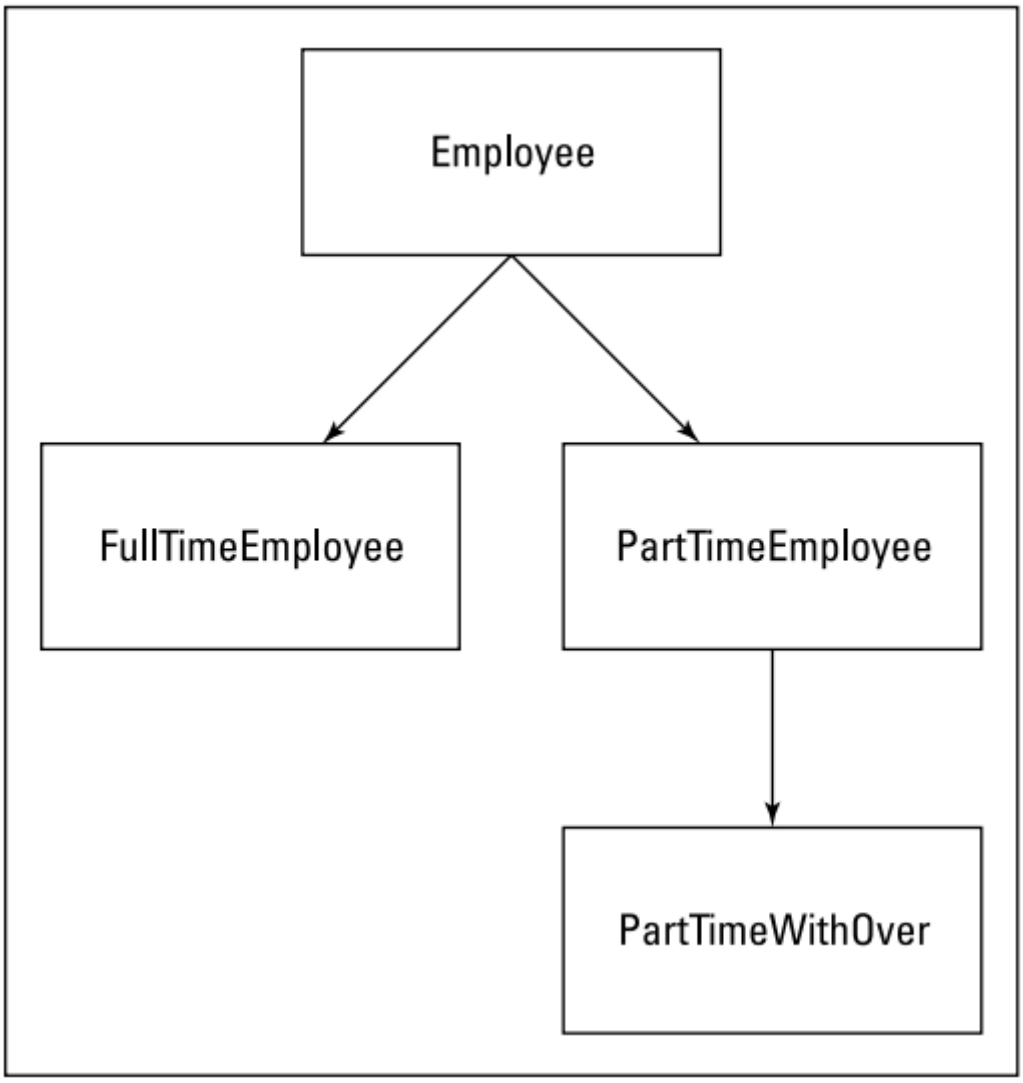
خوب ، شما میتوانید کدهای کلاس PartTimeEmployee را کند و کاو کرده ، و روی آنها کمی تغییرات اعمال کنید.(به نظر من ایده ی خوبی نیست).

شما میتوانید از آموزش های بخش قبلی استفاده کرده و یک زیر کلاس ، از کلاس موجود PartTimeEmployee ایجاد نمایید. اما کلاس PartTimeEmployee قبلا متد findPaymentAmount را داشت. آیا ما به راهی برای دور زدن و نادیده گرفتن متد findPaymentAmount نیاز داریم ؟؟

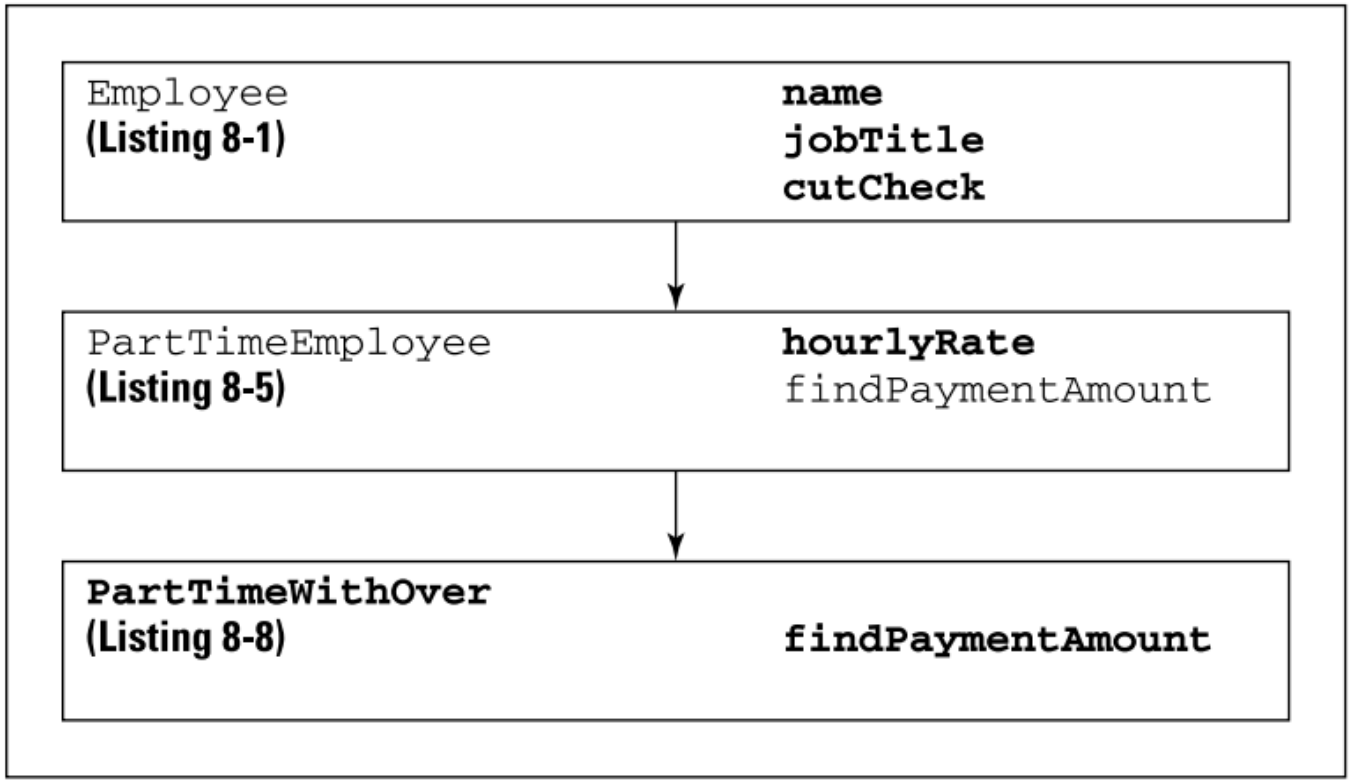
در اینجا شما خوش شانس هستید که از زبان شیءگرای جاوا استفاده میکنید. با برنامه نویسی شیء گرا شما میتوانید زیرکلاس هایی خلق نمایید که در کارکرد اساسی کلاس والد خود را دارا نیستند. مثال ۷-۸ چنین زیر کلاسی را نشان میدهد:

```
public class PartTimeWithOver extends PartTimeEmployee {
   @Override
public double findPaymentAmount(int hours) {
if(hours <= 40) {
returngetHourlyRate() * hours;
      } else {
                                        returngetHourlyRate() * 40 +
getHourlyRate() * 2 * (hours - 40);
      }
   }
}
```

شکل ۸-۸ رابطه ی مابین مثال ۷-۸ و سایر تکه کدهای موجود در این فصل را نشان میدهد. اگر دقیق شویم PartTimeWithOver یک زیرکلاس از یک زیرکلاس است. در برنامه نویسی شیءگرا زنجیره هایی اینچنین غیر معمول نیست. بلکه حتی شاهد زنجیره هایی طولانی تر از این هم هستیم.



کلاس PartTimeWithOver کلاس PartTimeEmployee را گسترش میدهد، اما PartTimeWithOver هر آنچه که میخواهد از کلاس از کلاس PartTimeEmployee به ارث ببرد را انتخاب میکند. زیراکه PartTimeWithOver دارای اعلان خودش برای متد findPaymentAmount بوده، و کلاس PartTimeWithOver متد findPaymentAmount از والد خود به ارث نمیبرد. به شکل ۹-۸ که در زیر آمده دقت کنید:



اگر شما یک شیء از کلاس PartTimeWithOver ایجاد کنید. شیء ایجاد شده دارای name و jobTitle و hourlyRate و cutCheck از کلاس PartTimeEmployee میباشد.. اما شیء دارای مند findPaymentAmount که در مثال ۷-۸ تعریف شده نیز میباشد.

استفاده از متد ها از درون زیرکلاس ها و کلاس ها

```
public class DoPayrollTypeF {
public static void main(String args[]) {
FullTimeEmployeeftEmployee = new FullTimeEmployee();
ftEmployee.setName("HadiKhodapanah");
ftEmployee.setJobTitle("CEO");
ftEmployee.setWeeklySalary(5000.00);
ftEmployee.setBenefitDeduction(500.00);
ftEmployee.cutCheck(ftEmployee.findPaymentAmount());
PartTimeEmployeeptEmployee = new PartTimeEmployee();
ptEmployee.setName("Ali Alizadeh");
ptEmployee.setJobTitle("Computer Book Author");
ptEmployee.setHourlyRate(7.53);
ptEmployee.cutCheck  (ptEmployee.findPaymentAmount(50));
PartTimeWithOverptoEmployee =    new PartTimeWithOver();
ptoEmployee.setName("Kouroshkhashayari");
ptoEmployee.setJobTitle("Driver");
ptoEmployee.setHourlyRate(7.53);
ptoEmployee.cutCheck (ptoEmployee.findPaymentAmount(50));
 }
}
```

ابتدا به مثال ۸-۸ دقت کرده سپس از روی این مثال ، مبحث توضیح داده خواهد شد

خروجی :

```
Pay to the order of Hadi Khodapanah  (CEO)  ***$4,500.00
Pay to the order of Ali Alizadeh (Computer Book Author)  ***$376.50
Pay to the order of Kourosh khashayari (Driver)  ***$451.80
```

با زیرکلاس ها، هر سه تای این کارمندان، در مثال ۸-۸ باهم وجود دارند. مطمئنا یک زیرکلاس ، از کلاس قدیمی PartTimeEmployee می آید. اما این بدین معنی نیست که شما نمیتوانید شیء ای از کلاس PartTimeEmployee ایجاد کنید. در واقع، جاوا خیلی هوشمندانه عمل میکند، مثال ۸-۸ سه فراخوانی به متد findPaymentAmount دارد و هر فراخوانی به ورژن متفاوتی از متد میرسد.

در اولین فراخوانی، ftEmployee.findPaymentAmount متغیر ftEmployee یک نمونه از کلاس FullTimeEmployee میباشد.پس متدی که فراخوانی شده مربوط به مثال ۳-۸ میباشد.

در دومین فراخوانی، ptEmployee.findPaymentAmount متغیر ptEmployee یک نمونه ای از کلاس PartTimeEmployee میباشد. پس متدی که فراخوانی شده مربوط به مثال ۵-۸ میباشد.

در سومین فراخوانی ptoEmployee.findPaymentAmount متغیر ptoEmployee نمونه ای از کلاس PartTimeWithOver میباشد. پس متد فراخوانی شده مربوط به مثال ۷-۸ میباشد.

فصل ۷

مقیاس دما (temperature scale) چیست؟

(انواع enumجاوا)

جاوا راه های گوناگونی را، به منظور گروه بندی چیز های(things)، برای شما فراهم می آورد. در این فصل شما چیز هایی را داخل یک نوع enum گروه بندی خواهید نمود.

ایجاد یک enum پیچیده ، آسان نیست ولی برای ایجاد یک enum ساده ،کافی ست یک گروه از کلمات داخل یک جفت آکولاد بنویسید. مثال ۱-۹ یک نوع enum تعریف میکند که نوع enum در این مثال TempScale نام دارد.

```
publicenumTempScale {
   CELSIUS, FAHRENHEIT, KELVIN, RANKINE,
   NEWTON, DELISLE, RÉAUMUR, RØMER, LEIDEN
};
```

وقتی شما نوع enum را تعریف میکنید دو اتفاق مهم می افتد:

* شما مقادیری را ایجاد میکنید،، همانطور که ۱۳ و ۱۵ مقادیر int هستند. CELSIUS و FAHRENHEIT نیز مقادیری TempScale هستند.

*شما میتوانیدمتغیر هایی را برای ارجاع به آن مقادیر ایجاد کنید،، در مثال ۲-۹ من فیلد های number و scale را اعلان نموده ام. همانگونه که doublenumber متغیر number را از نوع double اعلان میکند.

TempScalescale نیز متغیر scale از نوع TempScale اعلان میکند

از نوع TempScale بودن به این معناست که شما مقادیر .....CELSIUS,FAHRENHEIT, KELVIN را دارا میباشید.

یک نوع enum ، یک کلاس جاوا در استتار هست. به همین دلیلی است که مثال ۱-۹ شامل تمام فایلی که یک ، به یک چیز اختصاص داده شده، میشود، یعنی اعلان نوع TempScale enum . همانند اعلان یک

کلاس ، در اعلان نوع enum به فایل مخصوص خودش متعلق است. در مثال ۱-۹ به فایلی به نام TempScale.java متعلق است.

Temperature چیست؟

هر temperature (درجه حرارت) دو چیز در نظر گرفته میشود: یک عدد و یک مقیاس حرارت. مثال ۲-۹ در زیر این حقیقت را روشن میسازد.

```
public class Temperature {
private double number;
privateTempScale scale;
public Temperature() {
number = 0.0;
scale = TempScale.FAHRENHEIT;
   }
public Temperature(double number) {
this.number = number;
scale = TempScale.FAHRENHEIT;
   }
public Temperature(TempScale scale) {
number = 0.0;
this.scale = scale;
   }
public Temperature(double number, TempScale scale) {
this.number = number;
this.scale = scale;
   }
public void setNumber(double number) {
this.number = number;
   }
public double getNumber() {
return number;
   }
public void setScale(TempScale scale) {
this.scale = scale;
   }
publicTempScalegetScale() {
return scale;
   }
}
```

مثال ۲-۹ متد های قرار دهنده (setter) و دریافت کننده(getter) معمول را داراست.(متد های دسترسی دارنده به فیلد های number و scale .

نکته ای که باید متوجه آن شده باشید این است که، مثال ۲-۹ ، چهار قسمت ، شبیه به متد وجود دارد..هر یک از این بخش های شبیه به متد، Temperature نام دارند.مشابه با نام کلاس را داشت باشد.هیچ یک از این چیزهای Temperature شبیه به متد، یک نوع return از هیچ نوعی را دارا نمیباشند. نه حتی void که نوع returnhcop-out میباشد. هر یک از این چیزهای شبیه به متد سازنده (constructor) نام دارند.

یک سازنده شبیه یک متد است ، به استثنای اینکه یک سازنده، هدف بسیار مخصوصی دارد-ایجاد اشیاءِ جدید.

هرزمانی که اشیاءِ جدیدی ایجاد می کند، کامپیوتر دستورات درون سازنده ر ا اجرا میکند.

با سازنده(statements) چه کاری میتوان انجام داد

در مثال ۳-۹ شما سازنده ای که در مثال ۲-۹ اعلان شده را فراخوانی میکنید. تصویر شماره ۱-۹ نشان میدهد که وقتی شما تمام این کد را اجرا کردید چه اتفاقی خواهد افتاد.

```
import static java.lang.System.out;
public class UseTemperature {
public static void main(String args[]) {
final String format = "%5.2f degrees %s\n";
    Temperature temp = new Temperature();
temp.setNumber(70.0);
temp.setScale(TempScale.FAHRENHEIT);
out.printf(format, temp.getNumber(),
temp.getScale());
temp = new Temperature(32.0);
out.printf(format, temp.getNumber(),
temp.getScale());
temp = new Temperature(TempScale.CELSIUS);
out.printf(format, temp.getNumber(),
temp.getScale());
temp = new Temperature(2.73, TempScale.KELVIN);
out.printf(format, temp.getNumber(),
temp.getScale());
  }
}
```

```
 70.00 degrees FAHRENHEIT
 32.00 degrees FAHRENHEIT
  0.00 degrees CELSIUS
  2.73 degrees KELVIN
```

در مثال ۳-۹ هر دستور مانند :

temp = new Temperature(blah,blah,blah);

یک سازنده را از مثال ۲-۹ فراخوانی میکند. پس همانطور که مثال ۳-۹ را اجرا میشود، آن چهار شیء از کلاس Temperature ایجاد میکند. هر شیء با فراخوانی یک سازنده ی متفاوت از مثال ۲-۹ ایجادمیشود.

در مثال ۳-۹ آخرین سازنده دارای دو پارامتر ۲,۷۳ و TempScale.KELVIN بود. این مخصوص فراخوانی سازنده ها نیست.فراخوانی یک متد ی فراخوانی یک سازنده هم نیز میتواند یک لیست از پارامتر ها را داشته باشد.شما پارامتر ها را با کاما از یکدیگر جدا میکنید.

تنها قاعده ای که بایستی بدان توجه کنید این است که: پارامتر های فراخوانی با پارامترهایی که اعلان شده اند تطابق داشته باشند. برای مثال در مثال ۳-۹ هنگام فراخوانی چهامین و آخرین سازنده :

```
new Temperature(2.73, TempScale.KELVIN)
```

دو پارامتر دارد.اولی از نوع double و دومی از نوع TempScale . جاوا با این فراخوانی موافقت خواهذ نمود زیراکه که آن با هدر(header) تعریف شده در مثال ۲-۹ تطابق دارد :

```
public Temperature(double number, TempScale scale)
```

دو پارامتر دارد اولی از نوع double و دومی از نوع TempScale میباشد. که اگر این تطابق را رعایت نکنید با مشکل مواجه خواهید شد.

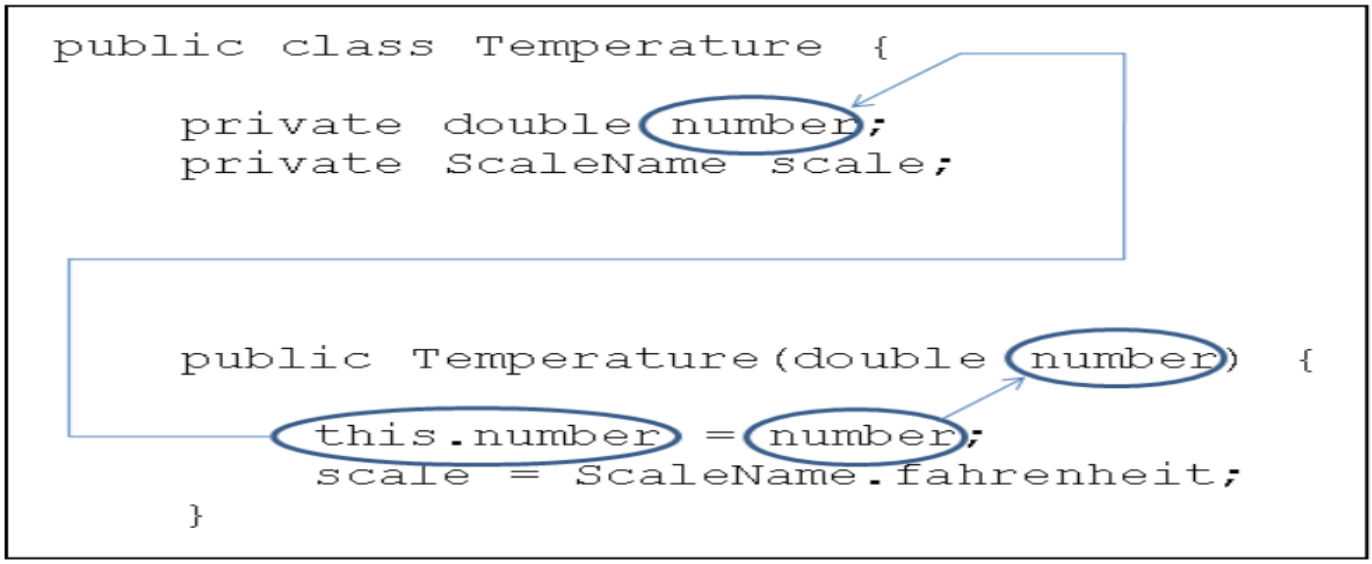
دو راه برای انجام یک کار

مثال ٤-٩ دارای دو سازنده درون خود میباشد. من از دو نام متفاوت استفاده میکنم : number و whatever. در دومین سازنده ، من به دو نام احتیاج ندارم.به جای ایجاد یک نام جدید برای پارامتر سازنده، من تصمیم گرفتم از نام موجود با نوشتن this.number دوباره استفاده کنم.

```
//Use this constructor . . .
public Temperature(double whatever) {
number = whatever;
scale = TempScale.FAHRENHEIT;
    }
//. . . or use this constructor . . .
public Temperature(double number) {
this.number = number;
scale = TempScale.FAHRENHEIT;
    }
//. . . but don't put both constructors in your code.
```

در دستور this.number = number اسم this.number به فیلد number شیءِ جدید ارجاع میکند.- فیلدی که در بالای مثال ۲-۹ تعلان شده است.

در دستور this.number = number اسم number سمت چپ، به پارامتر سازنده ارجاع دارد.



در حالت کلی، this.someName به فیلدی که که حاوی کد ارجاع میکند..به عبارت ساده someName به نزدیکترین مکان که someName اعلان شده،رجاع میکند. در دستور this.number=number که نزدیکترین مکان لیست پارامتر سازنده ی Temperature میباشد.

ساخت temperatures بهتر

مثال ۵-۹

```
import static java.lang.System.out;
public class TemperatureNice extends Temperature {
publicTemperatureNice() {
super();
   }
publicTemperatureNice(double number) {
super(number);
   }
publicTemperatureNice(TempScale scale) {
super(scale);
   }
publicTemperatureNice(double number, TempScale scale)
   {
super(number, scale);
   }
public void display() {
out.printf("%5.2f degrees %s\n",
getNumber(), getScale());
   }
}
```

در متدdisplay مثال ۵-۹ ،، به فراخوانی کلاس Temperature متد های getNumber و getScale توجه کنید.چرا چنین کاری را انجام دادیم؟؟؟ خوب در واقع درون کد کلاس TemperatureNice هیچ ارجاع مستقیمی به فیلدهای number و number پیام خطایی تولید خواهد نمود. درست است ، هر شیء TemperatureNice فیدهای number و scale مخصوص خود را دارد.(در کنار این همه TemperatureNice یک زیر کلاس از کلاس Temperature میباشد و کد کلاس Temperature فیلد های

number و scale را تعریف میکند.). اما چون number و scale به صورت خصوصی درون کلاس Temperature اعلان گردیده اند.تنها کدی که درست درون کلاس Temperature باشد میتواند از این فیلد ها استفاده کند.

دستور super() در مثال ۵-۹پارامتر های سازنده ی Temperature() را در مثال ۲-۹ فراخوانی میکند. که سازنده ی بدون پارامتر ۰,۰ را به فیلد number و TempScale اختصاص میدهد. FAHRENHEIT به فیلد field..

دستور super(number, scale) در مثال ۵-۹ سازنده ی

Temperature(double number, TempScale scale) را که در مثال ۲-۹ آورده شده فراخوانی میکند.

به صورتی مشابه، دستورات super(number) و super(scale) در مثال ۵-۹ سازنده ها از مثال ۲-۹ فراخوانی میکند.

Using all this stuff

```
public class UseTemperatureNice {
public static void main(String args[]) {
TemperatureNice temp = new TemperatureNice();
temp.setNumber(70.0);
temp.setScale(TempScale.FAHRENHEIT);
temp.display();
temp = new TemperatureNice(32.0);
temp.display();
temp = new TemperatureNice(TempScale.CELSIUS);
temp.display();
temp = new TemperatureNice(2.73,
TempScale.KELVIN);
temp.display();
   }
}
```

کد مثال ۶-۹ در مثال بالا خیلی شبیه به کد ۳-۹ میباشد.تنها تفاوت آنها عبارت اند از :

درمثال ۶-۹ نمونه ای از کلاسTemperatureNice ایجادمیشود. . به همین دلیل است مثال ۶-۹ سازنده ها را از کلاس TemperatureNice فراخوانی میکند نه از کلاس Temperature.

در مثال ۶-۹ از امتیاز متد display در کلاس TemperatureNice بهره میگیرد. بنابراین کد مثال ۶-۹ منظم و آراسته تر از همتایش در مثال ۳-۹ است.

یک سازنده ی پیش فرض

```
publicFullTimeEmployee() {
super();}
```

سازنده ای که در مثال ۷-۹ در بالا استفاده کردیم هیچ پارامتری نمیگیرید.و تنها دستور موجود در آن، سازنده ی هرکلاسی که شما توسعه داده اید را فراخوانی میکند.(در صورتی که کلاسی که شما توسعه داده اید سازنده ی بدون پارامتر نداشته باشد ، اشتباه رخ خواهد داد).

به مثال ۸-۹ در زیر توجه نمایید:

```
public class FullTimeEmployee extends Employee {
private double weeklySalary;
private double benefitDeduction;
publicFullTimeEmployee(double weeklySalary) {
this.weeklySalary = weeklySalary;
  }
public void setWeeklySalary(double weeklySalaryIn) {
weeklySalary = weeklySalaryIn;
  }
public double getWeeklySalary() {
returnweeklySalary;
  }
public void setBenefitDeduction(double benefitDedIn) {
benefitDeduction = benefitDedIn;
  }
public double getBenefitDeduction() {
returnbenefitDeduction;
  }
public double findPaymentAmount() {
returnweeklySalary - benefitDeduction;
  }
}
```

اگر شما مانند زیر از کد FullTimeEmployee در مثال ۸-۹ استفاده کنید، برنامه ی شما کار نخواهد کرد :

```
FullTimeEmployeeftEmployee = new FullTimeEmployee;()
```

این برنامه کار نخواهد کرد زیراکه سازنده ی FullTimeEmployee اعلان شده که یک پارامتر double بگیرد، (یک سازنده ی بدون پارامتر پیش فرض را قبول نمیکند).

سازنده ای که کارهای بیشتری انجام میدهد

در این بخش میخواهیم مثالی از یک سازنده بزنم که کاری بیشتر از اختصاص مقادیری به چند فیلد انجام میدهد.مثال های ۹-۹ و ۱۰-۹ و نتیجه ی خروجی نیز در مثال ۳-۹ نشان داده شده است.

تعریف Frame

```
importjava.awt.FlowLayout;
importjavax.swing.JFrame;
importjavax.swing.JButton;
@SuppressWarnings("serial")
public class SimpleFrame extends JFrame {
publicSimpleFrame() {
setTitle("Don't click the button!");
setLayout(new FlowLayout());
setDefaultCloseOperation(EXIT_ON_CLOSE);
add(new JButton("Panic"));
setSize(300, 100);
setVisible(true);
   }
}
```

نشان دادن Frame

```
public class ShowAFrame {
public static void main(String args[]) {
newSimpleFrame();
   }
}
```

شکل ۳-۹

کد مثال ۹-۹ عمدتا از متدهای API های جاوا شکل گرفته است.(قصد ما از این مثال توضیح دادن تمام ایم متد ها نیست). به هر حال متد اصلی مثال ۱۰-۹ فقط یک دستور دارد: یک فراخوانی به سازنده در کلاس SimpleFrame. توجه کنید که چگونه شیءِ ای که این فراخوانی ایجاد میکند، حتی به یک متغیر هم تخصیص داده نشده است. همه چیز درست است زیرا که ما در این کد نیازی به ارجاع به شیءِ مورد نظر در هیچ جای دیگری نداریم.

در بالا، کلاس SimpleFrame فقط تنها یک اعلان سازنده(constructor) موجود است. این سازنده متدی بعد از متد دیگر را از API جاوا فراخوانی میکند.

تمام متد های فراخوانی شده در کلاس سازنده ی SimpleFrame ، از کلاس والد JFrame میآیند. کلاس JFrame در داخل پکیج javax.swing موجود است. این پکیج . دیگر پکیج های موجود در java.awt به شما کمک میکنند که تصاویر، پنجره ، نقاشی ،...و دیگر اسباب و ابزار را روی صفحه ی نمایش کامپیوتر قرار دهید.

setTitle :فراخوانی setTitle کلمات را در نوار عنوان فریم قرار میدهد.(شیءِ جدید SimpleFrame ، متد setTitle خودش را فراخوانی کیکند).

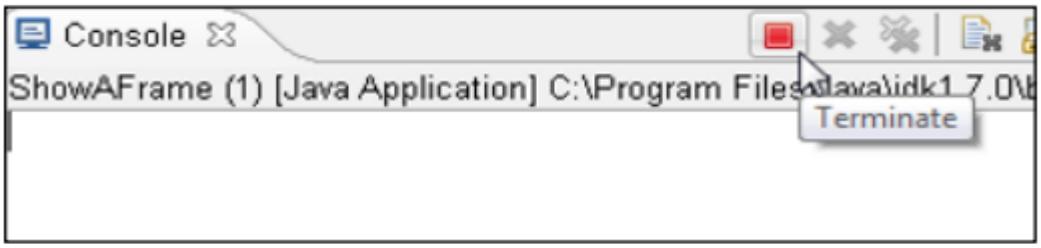
FlowLayout : یک نمونه از کلاس FlowLayout ، اشیاءِ روی فریم رابه سبک ماشین تحریر درمرکزیت قرار میدهد. زیرا که فریم تصویر ۳-۹ تنها یک دکمه در خود دارد.آن دکمه در مرکزیت و نزدیک بالای فریم قرار داده شده است.اگر فریم هشت دکمه می داشت. پنج تای آن ها ممکن بود در خط بالایی فریم قرار گرفته و سه تای باقی مانده در مرکزیت خط پایین فریم قرار بگیرند.

setLayout : فراخوانی setLayout شیءِ FlowLayout جدید را متصدی مرتب ساختن اجزاءِ میکند. مانند دکمه ها یا فریم.(شیءِ جدید SimpleFrame متد setLayout خودش را فراخوانی میکند).

setDefaultCloseOperation:فراخوانی setDefaultCloseOperation به جاوا میگوید که وقتی شما روی × در گوشه بالا-راست کلیک میکنید، چه انجام دهد. بدون فراخوانی این متد، فریم خودش ناپدید میشود. اما ماشین مجازی جاوا Java Virtual Machine (JVM) به فعالیت ادامه میدهد. اگر شما از Eclipse استفاده میکنید شما بایستی JVMرا با کلیک روی مربع چهر گوش بالای کنسول ، متوقف کنید.(شکل ٤-٩ را ببینید).

فراخوانی setDefaultCloseOperation(EXIT_ON_CLOSE) به جاوا میگوید که هنگامی که شما روی × در گوشه بالا-سمت راست کلیک میکنید، خودش را ببندد.

شکل ٤-٩



JButton : کلاس JButton در پکیج javax.swing موجود است. یکی از سازنده های کلاس، برای پارامتر هایش نمونه ای از String را میگیرد.(مانند "Panic")..فراخوانی این سازنده باعث میشود که نمونه(instance) ی String روی دکمه ی جدید برچسب زده شود.

Add: شیءِ SimpleFrame جدید متد add خودش را فراخوانی میکند. فراخوانی متد add ، دکمه را را روی سطح ظاهری شیءِ قرار میدهد.(در این مثال سطح ظاهری فریم).

setSize : فریم ۳۰۰ پیکسل عرض ۱۰۰ پیکسل ارتفاع دارد.(در پکیج javax.swing هنگامی که شما ابعاد دو بعد را مشخص کنید، عدد مشخص کننده ی عرض(پهنا) قبل از عدد مشخص کننده ی ارتفاع میآید.

setVisible : زمانی نخست که آن ایجاد میشود.یک فریم جدید ناپدید میشود.اما زمانی که فریم جدید setVisible(true) را فراخوانی میکند. روی صفحه ی نمایش کامپیوتر شما ظاهر میشود.

منابع و ماخذ :

Java How to Program (9th Edition) ,By Paul Deitel , Harvey Deitel, *Publisher : Prentice Hall*

Java 8: The Fundamentals, By Dane Cameron, Publisher: Cisdal Publishing

Head First Java, By Kathy Sierra , Bert Bates, Publisher: O'Reilly Media

Java The Complete Reference, 8th Edition,By Herbert Schildt, Publisher: McGraw-Hill Osborne

Java 8 Pocket Guide, By Robert Liguori , Patricia Liguori, Publisher: O'Reilly Media

Learn Java for Android Development: Java 8 and Android 5 Edition,By Jeff Friesen, Publisher: Apress

Pro Java 8 Programming, By Terrill Brett Spell, Publisher: Apress

Java 8: The Fundamentals, By Dane Cameron, Publisher: Cisdal Publishing

Java SE8 for the Really Impatient: A Short Course on the Basics 1st Edition,By Cay S. Horstmann, Publisher: Addison-Wesley Professional

Beginning Java, Author : Ivor Horton,Publication : Wrox