جاوا به زبان ساده

نویسنده : یونس ابراهیمی

1394/04/10

Younes.ebrahimi.1391@gmail.com

منبع : w3-farsi.com

جاوا چیست؟

NetBeans و JDK

نصب NetBeans و JDK

ساخت یک برنامه ساده در JAVA

استفاده از IntelliSense در NetBeans

رفع خطاها

کاراکترهای کنترلی

متغیر

انواع ساده

استفاده از متغیرها

ثابت

تبدیل ضمنی

تبدیل صریح

عبارات و عملگرها

عملگرهای ریاضی

عملگرهای تخصیصی

عملگرهای مقایسه ای

عملگرهای منطقی

عملگرهای بیتی

تقدم عملگرها

گرفتن ورودی از کاربر

ساختارهای تصمیم

دستور if

دستور if تو در تو

عملگر شرطی

دستور if چندگانه

استفاده از عملگرهای منطقی

دستور switch

تکرار

حلقه While

حلقه do While

حلقه for

خارج شدن از حلقه با استفاده از break و continue

آرایه ها

حلقه foreach

آرایه های چند بعدی

متد

مقدار برگشتی از یک متد

پارامترها و آرگومان ها

نامیدن آرگومان ها

ارسال آرگومان ها به روش ارجاع

پارامترهای out

ارسال آرایه به عنوان آرگومان

کلمه کلیدی params

محدوده متغیر

پارامترهای اختیاری

سربارگذاری متدها

بازگشت (Recursion)

نماینده ها(Delegates)

شمارش (Enumeration)

تبدیل انواع شمارشی

ساختار (Struct)

برنامه نویسی شیء گرا (Object Oriented Programming)

کلاس

سازنده ها (Constructors)

مخرب ها (Destructors)

فیلدهای فقط – خواندنی

سطح دسترسی

کپسوله سازی (Encapsulation)

خواص (Properties)

فضای نام

تفاوت ساختار و کلاس

کتابخانه کلاس

وراثت

سطح دسترسی Protect

اعضای Static

کلاس آبجکت(System.Object Class)

رابط ها (Interfaces)

کلاسهای انتزاعی (Abstract Class)

کلاس مهر و موم شده (Sealed Class)

چند ریختی (Polymorphism)

مدیریت استثناءها و خطایابی

استثناءهای اداره نشده

دستورات try و catch

استفاده از بلوک finally

ایجاد استثناء

خواص Exception

تعریف یک استثناء توسط کاربر

کلکسیون ها (Collections)

کلاس ArrayList

ایجاد یک کلکسیون

ساخت دیکشنری

Hashtable در سی شارپ

پیمایشگر (Iterator)

جنریک ها (Generics)

متدهای جنریک

کلاس جنریک

Initializer

عبارات لامبدا (Lambda expressions)

جاوا چیست؟

جاوا یک زبان برنامه نویسی و همچنین یک پلتفرم می باشد. این زبان جزء زبان های سطح بالا و شیءگرا محسوب می شود. برای نخستین بار توسط جیمز گاسلینگ در شرکت سان مایکروسیستمز ایجاد گردید و در سال ۱۹۹۵ به عنوان بخشی ازسکوی جاوا منتشر شد. زبان جاوا شبیه به C++ است اما مدل شیءگرایی آسان‌تری دارد و از قابلیت‌های سطح پایین کمتری پشتیبانی می‌کند .یکی از قابلیت‌های بنیادین جاوا این است که مدیریت حافظه را بطور خودکار انجام می‌دهد. ضریب اطمینان عملکرد برنامه‌های نوشته‌شده به این زبان بالا است و وابسته به سیستم‌عامل خاصی نیست، به عبارت دیگر می‌توان آن را روی هر رایانه با هر نوع سیستم‌عاملی اجرا کرد. در زیر یک نمونه برنامه جاوا نشان داده شده است:

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

در باره جزییات برنامه بالا در درس های آینده بیشتر توضیح می دهیم. جاوا برای نوشتن انواع برنامه‌های کاربردی مناسب است. با جاوا می‌توان انواع برنامه‌های زیر را نوشت:

- برنامه‌های تحت وب : برنامه های تحت وب همانطور که از نام شان پیداست برنامه هایی هستند که در سمت سرور اجرا می شوند و صفحات وب داینامیک را به وجود می آورندServlet ،. jsp، struts، jsf تکنولوژی های مورد استفاده در جاوا برای ایجاد صفحات وب هستند.
- برنامه‌نویسی سیستم‌های کوچک : برنامه های موبایل در این دسته قرار می گیرند. از JAVA ME برای ساخت این نوع برنامه ها استفاده می شود.
- برنامه‌های کاربردی بزرگ :(Enterprise) از این نوع برنامه می توان به برنامه های بانکی اشاره کرد که در آنها به امنیت بسیار بالا نیاز است. از EJBدر ساخت این نوع برنامه ها در جاوا استفاده می شود.
- برنامه‌های رومیزی :(Desktop) این برنامه ها برنامه هایی هستند که بر روی ویندوز نصب می شوند. برنامه هایی مانند Media Playerو آنتی ویروس ها جزو این دسته محسوب می شوند. در جاوا از AWT و Swing برای ساخت برنامه های ویندوزی استفاده می شود.

یکی از ویژگی‌های جاوا قابل حمل بودن آن است. یعنی برنامهٔ نوشته شده به زبان جاوا باید به طور مشابهی در کامپیوترهای مختلف با سخت‌افزارهای متفاوت اجرا شود؛ و باید این توانایی را داشته باشد که برنامه یک بار نوشته شود، یک بار کامپایل شود و در همه کامپیوترها اجرا گردد. به علت تشابه بین این زبان و زبان های خانواده C و همچنین درخواست کاربران عزیز مبنی بر آموزش این زبان قدرتمند، یک دوره کامل آموزشی از این زبان را در سایت قرار می دهم و امیدوارم که مورد استقبال شما عزیزان قرار گیرد.

JDKو

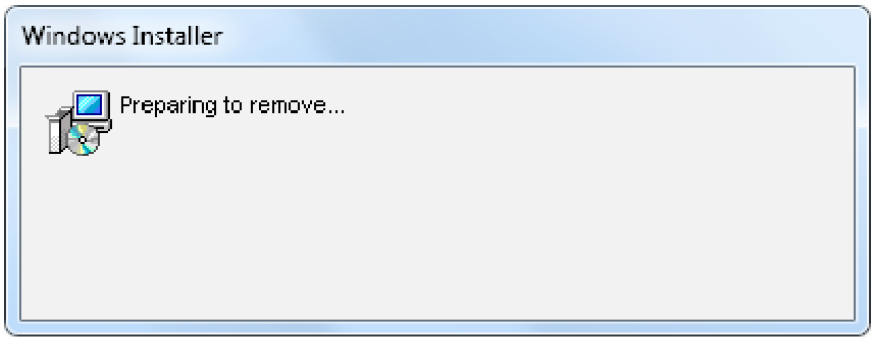
Netbeansمحیط توسعه یکپارچه ای است که دارای ابزارهایی برای کمک به شما برای توسعه برنامه های JAVA می باشد. توصیه می کنیم که از محیط Netbeansبرای ساخت برنامه استفاده کنید، چون این محیط دارای ویژگی های زیادی برای کمک به شما جهت توسعه برنامه های JAVA می باشد. توسط Netbeans می توان در استانداردهای مختلف جاوا مانند J2SE، J2EEو J2ME برنامه نویسی کرد. همچنین از محیط زبان هایPHP ، HTML، Cو نیز Groovy پشتیبانی می کند .قبل از نصب Netbeans می بایست JDK را نصب نمایید،در غیر این صورت برای نصب دچار مشکل خواهید شد JDK .که مخفف عبارت Java Development Toolkitمی باشد ترکیبی از کامپایلر زبان جاوا، کلاس های کتابخانه ای (Java Class Libraries)و JVM و فایل های راهنمای آنها می باشد. برای اینکه ما بتوانیم با استفاده از زبان برنامه نویسی جاوا، برنامه بنویسیم به این مجموعه نیاز داریم . تعداد زیادی از پردازش ها که وقت شما را هدر می دهند به صورت خودکار توسط NetBeansانجام می شوند. یکی از این ویژگی ها اینتل لایسنس (Intellisense) است که شما را در تایپ سریع کدهایتان کمک می کند NetBeans .برنامه شما را خطایابی می کند و حتی خطاهای کوچک (مانند بزرگ یا کوچک نوشتن حروف) را برطرف می کند .با این برنامه های قدرتمند بازدهی شما افزایش می یابد و در وقت شما با وجود این ویژگیهای شگفت انگیز صرفه جویی می شود NetBeans آزاد است و می توان ان را دانلود و از آن استفاده کرد.این برنامه ویژگیهای کافی را برای شروع برنامه نویسی JAVA در اختیار شما قرار می دهد.در آموزش ها از NetBeansنسخه 8.0.2 استفاده شده است و استفاده از این نسخه برای انجام تمرینات این سایت کافی می باشد. برای دانلود نرم افزارهای مورد نیاز بر روی لینک زیر کلیک کنید:

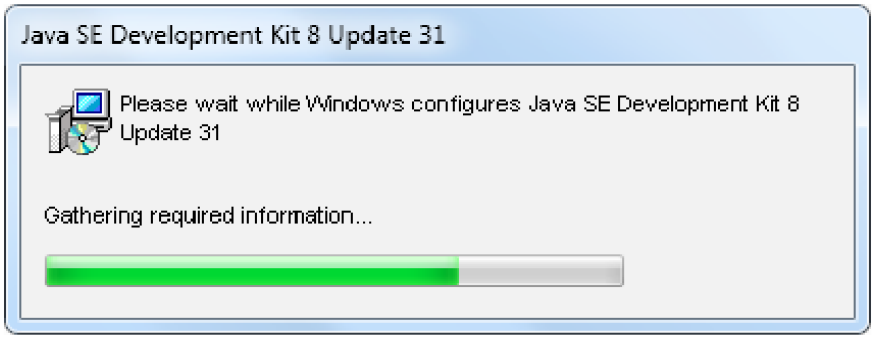
دانلود JDK وNetBeans

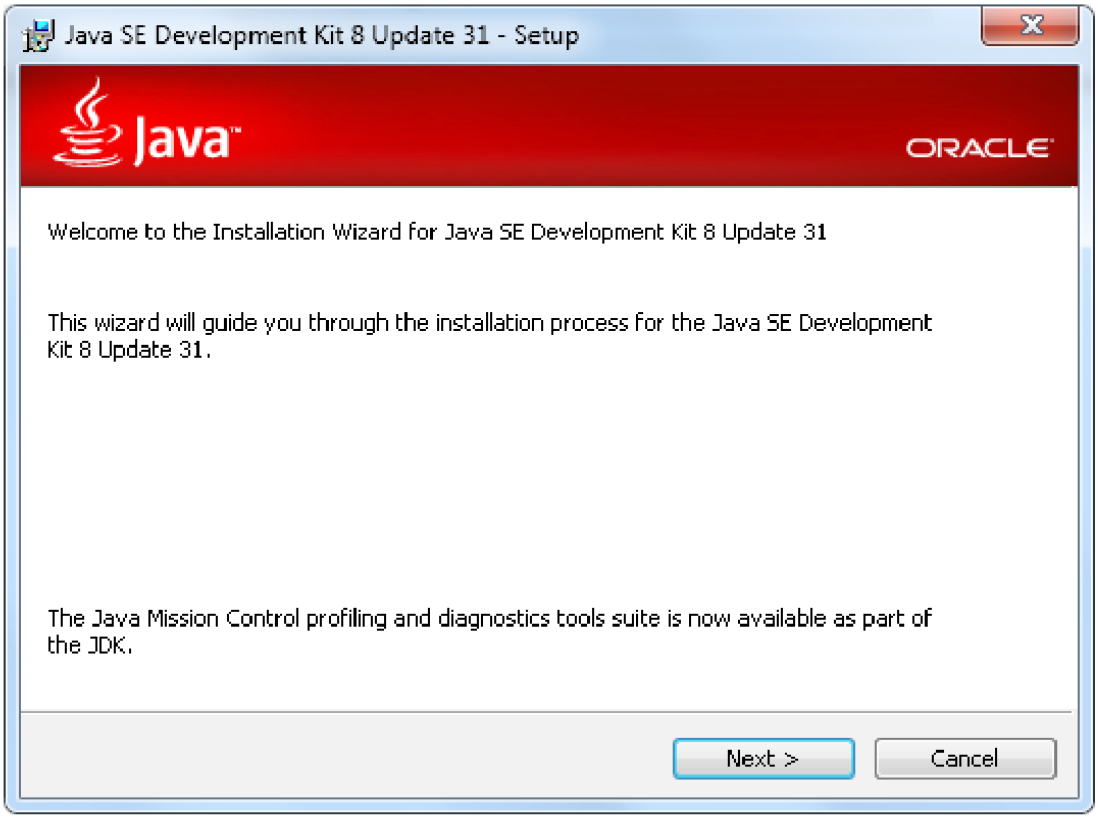
در درس آینده مراحل نصب و راه اندازی دو نرم افزار JDK و NetBeans را توضیح می دهیم.

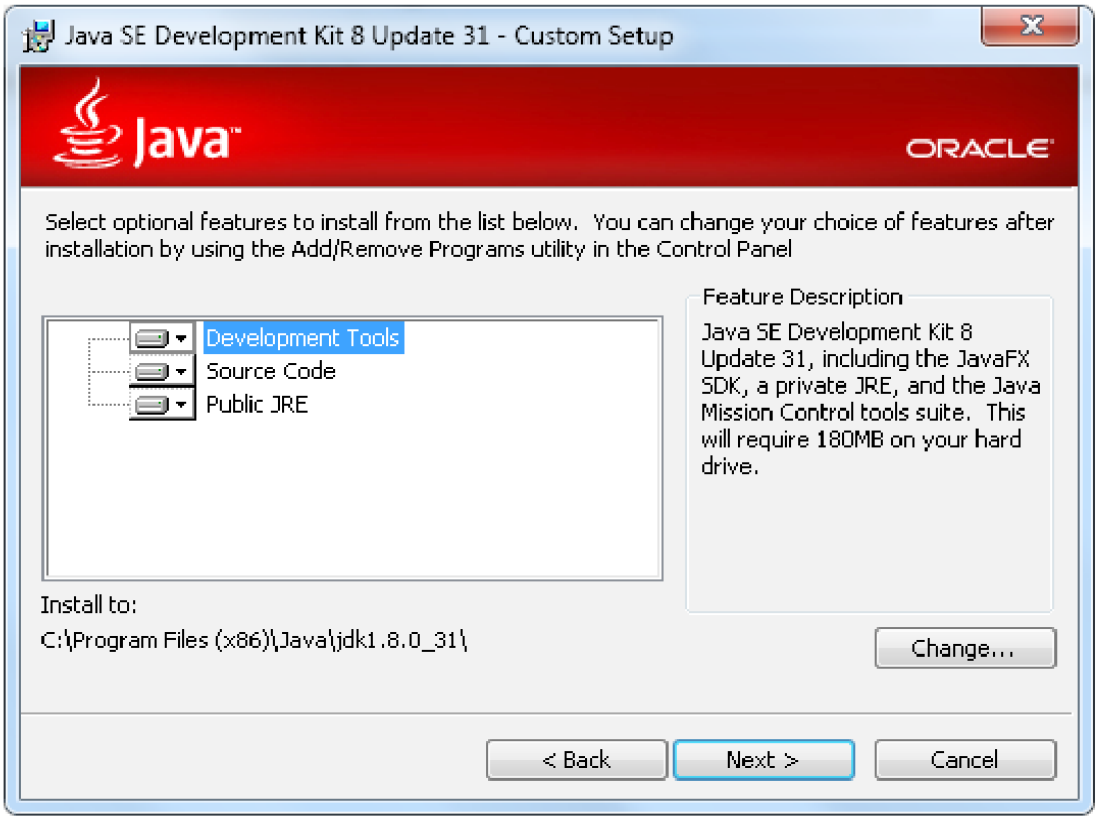
نصب NetBeans وJDK

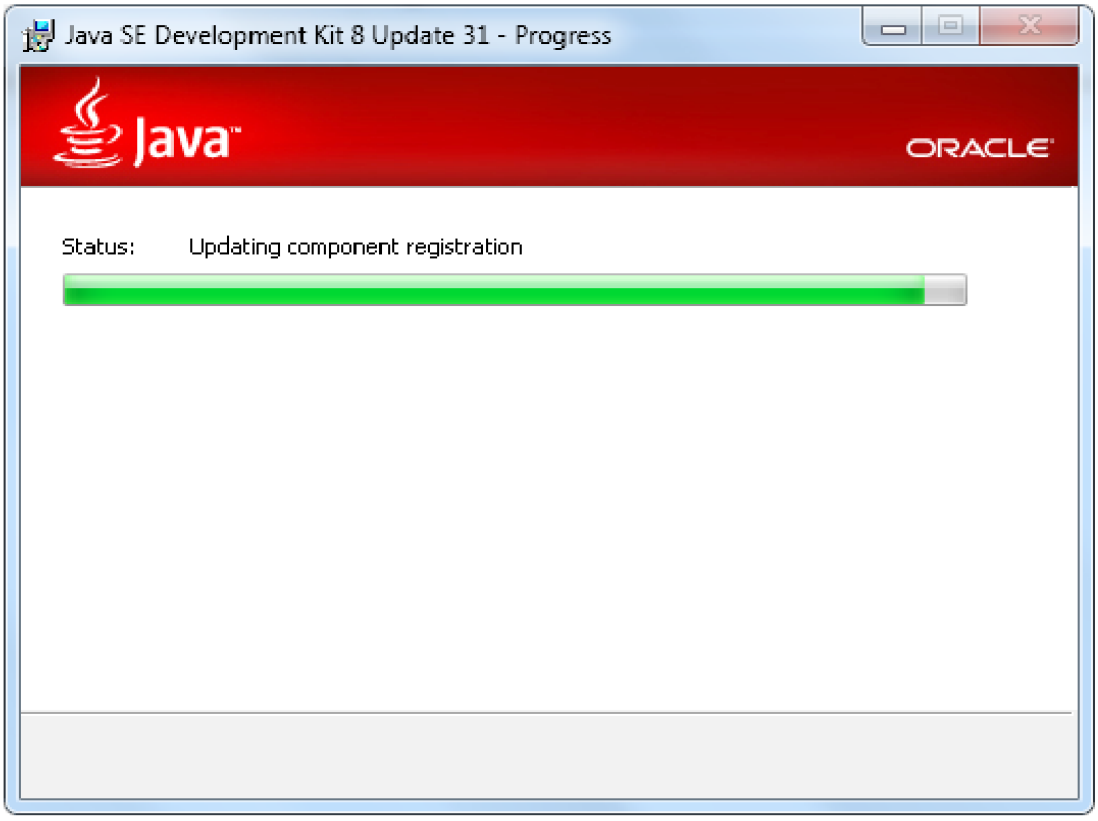
در درس قبل در مورد نرم افزارهای NetBeans و JDK توضیحات مختصری ارائه دادیم. در این درس می خواهیم شما را با نحوه نصب این دو نرم افزار آشنا کنیم. نصب این نرم افزارها مانند اکثر نرم افزارهای دیگر بسیار آسان بود و بعد از زدن چند دکمه Nextنصب می شوند. در زیر مراحل تصویری نصب این دو نرم افزار نشان داده شده است.


Windows Installer
Preparing to remove...


Java SE Development Kit 8 Update 31
Please wait while Windows configures Java SE Development Kit 8 Update 31
Gathering required information...


Java SE Development Kit 8 Update 31 - Setup
Java
ORACLE
Welcome to the Installation Wizard for Java SE Development Kit 8 Update 31
This wizard will guide you through the installation process for the Java SE Development Kit 8 Update 31.
The Java Mission Control profiling and diagnostics tools suite is now available as part of the JDK.
Next >
Cancel

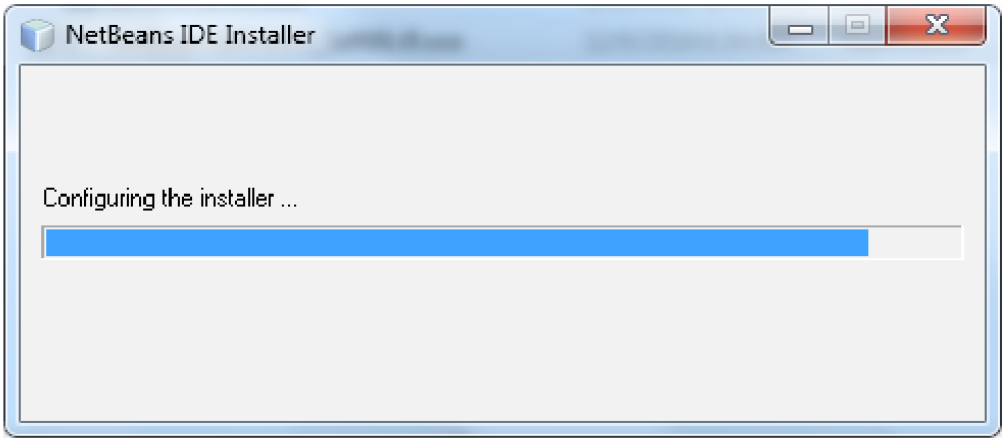

Java SE Development Kit 8 Update 31 - Custom Setup
Java
ORACLE
Select optional features to install from the list below. You can change your choice of features after installation by using the Add/Remove Programs utility in the Control Panel
Development Tools
Source Code
Public JRE
Feature Description
Java SE Development Kit 8 Update 31, including the JavaFX SDK, a private JRE, and the Java Mission Control tools suite. This will require 180MB on your hard drive.
Install to:
C:\Program Files (x86)\Java\jdk1.8.0_31\
Change...
< Back
Next >
Cancel

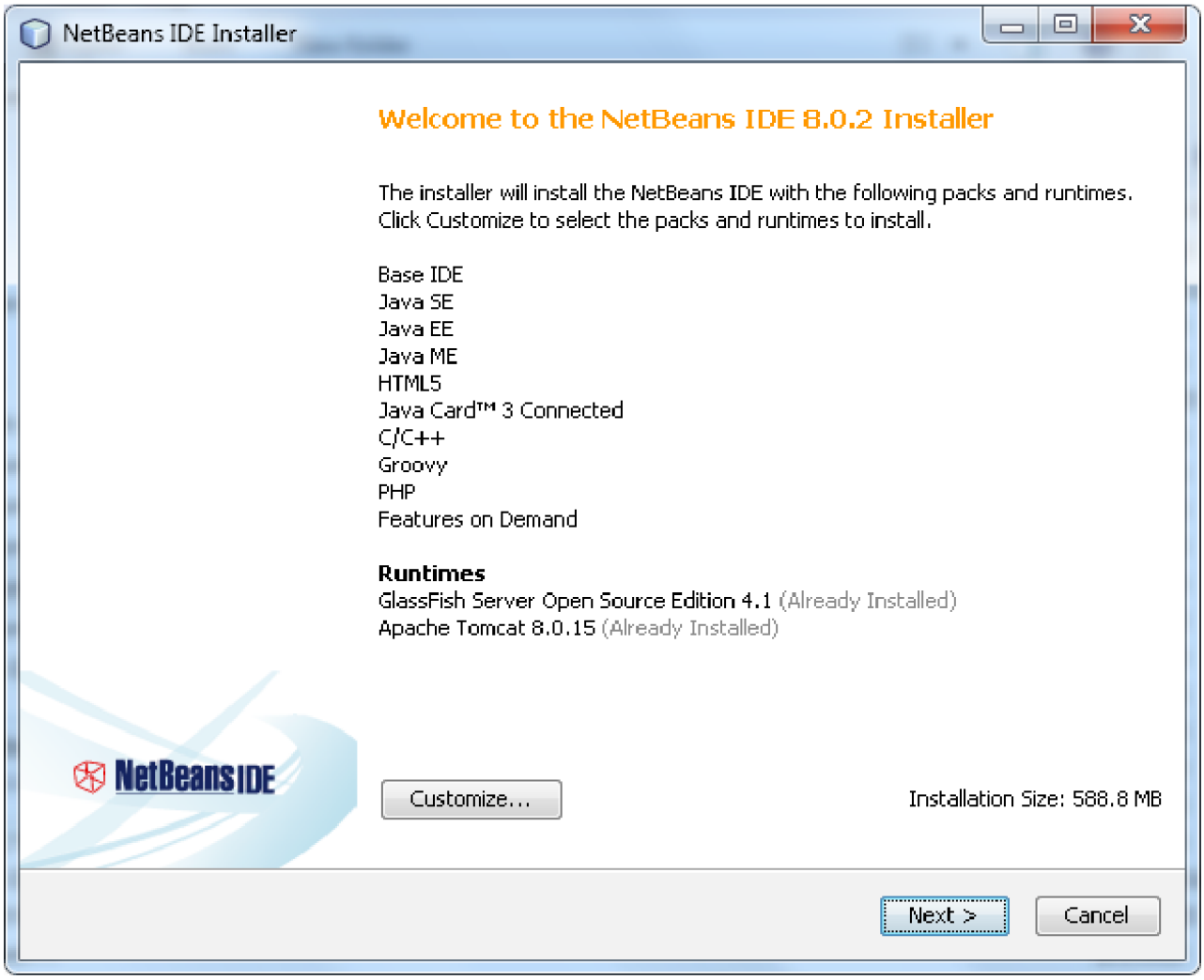

Java SE Development Kit 8 Update 31 - Progress
Java
ORACLE
Status:
Updating component registration

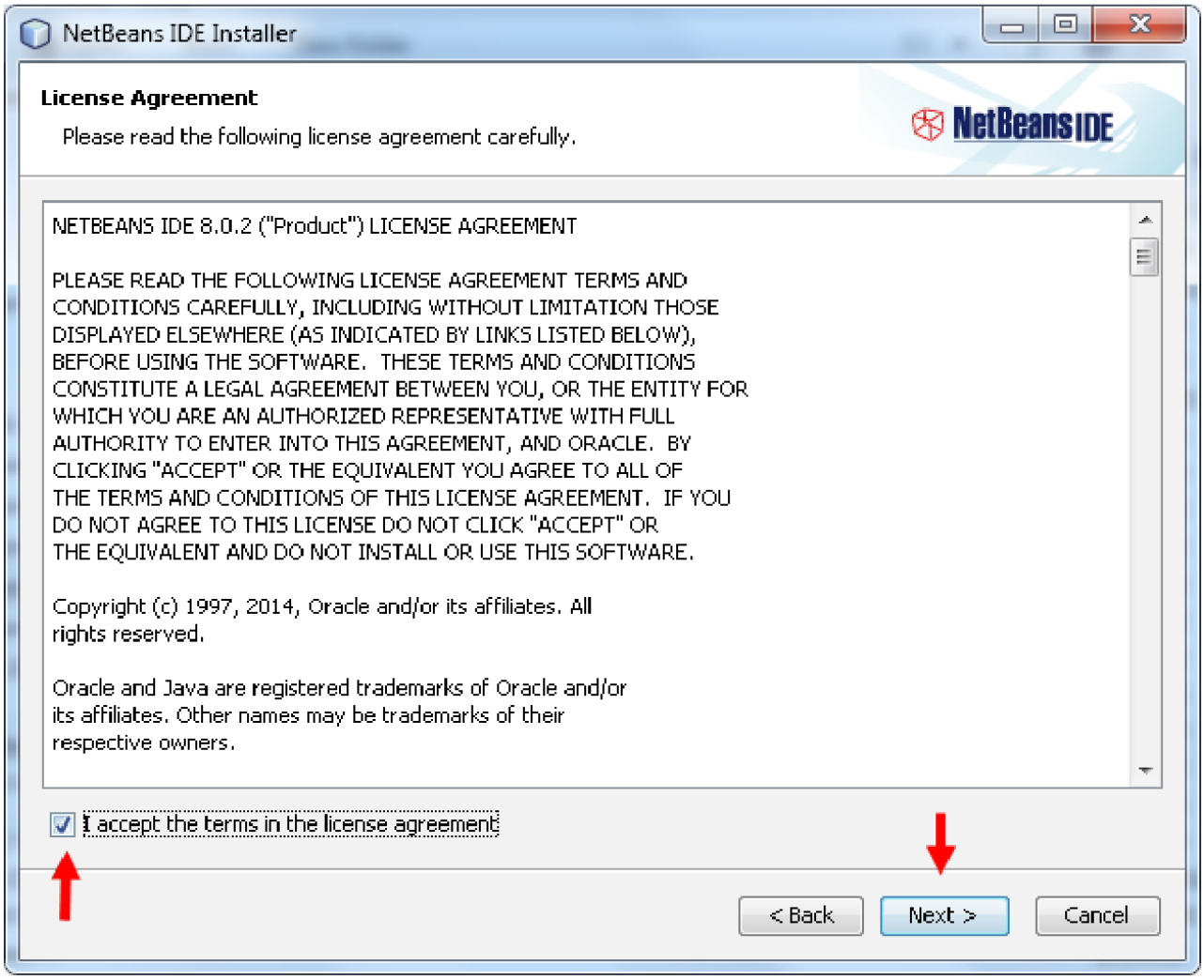

Java Setup - Destination Folder
Java
ORACLE
Destination Folder
Click "Change" to install Java to a different folder.
Install to:
C:\Program Files (x86)\Java\jre1.8.0_31\
Change...
Next >

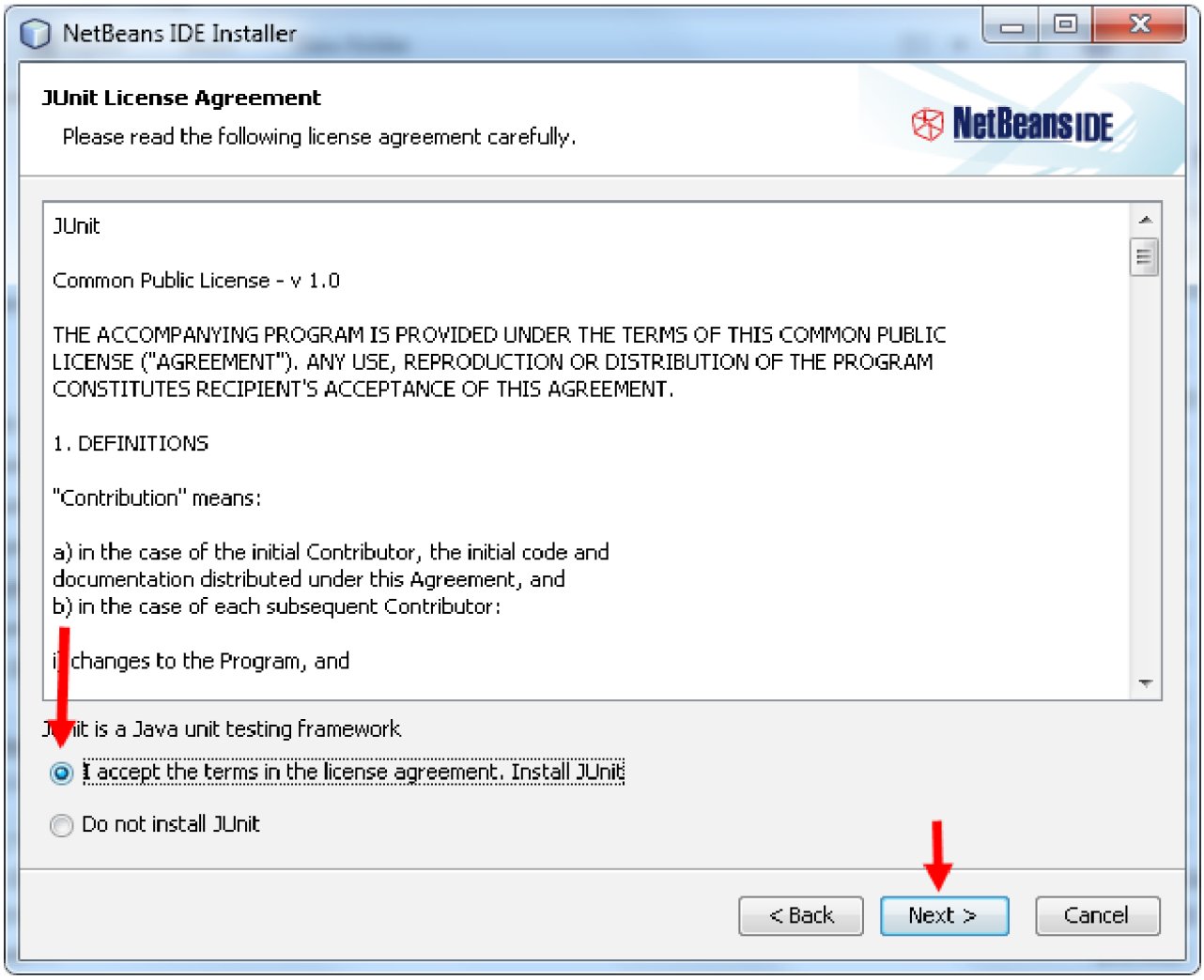

Java Setup - Progress
Java
ORACLE
Status: Installing Java
3 Billion Devices Run Java
Computers, Printers, Routers, Cell Phones, BlackBerry, Kindle, Parking Meters, Public Transportation Passes, ATMs, Credit Cards, Home Security Systems, Cable Boxes, TVs...
ORACLE

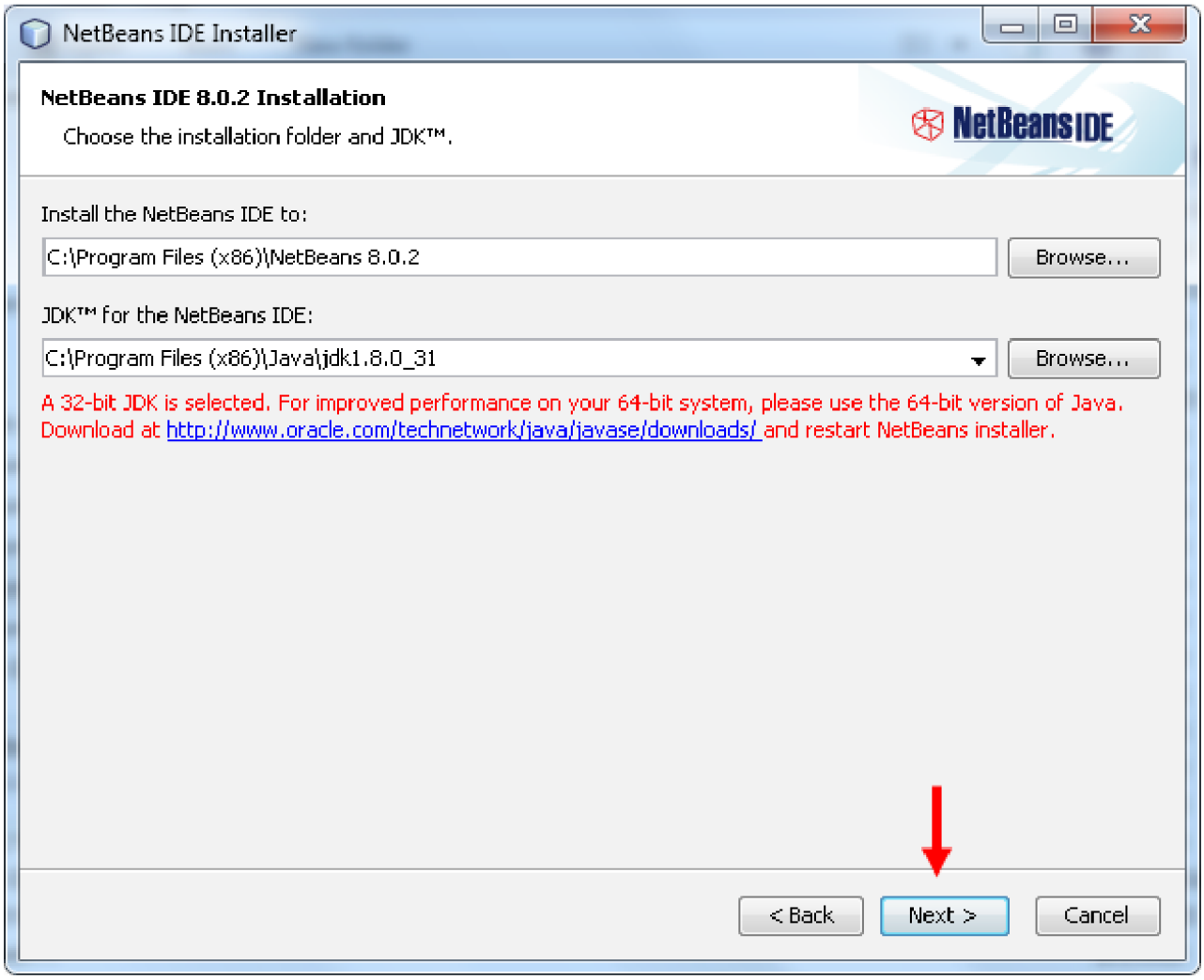
نصب NetBeans

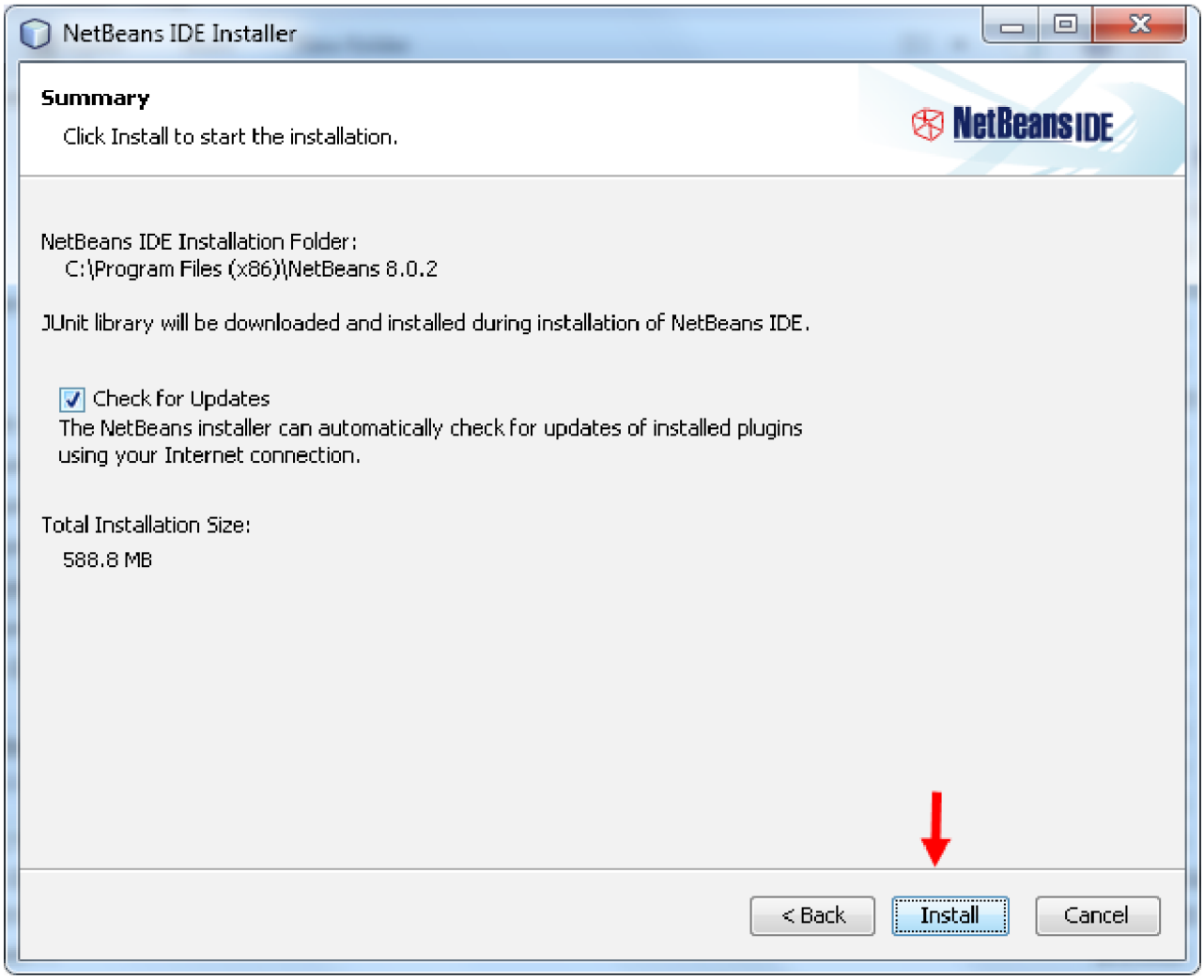


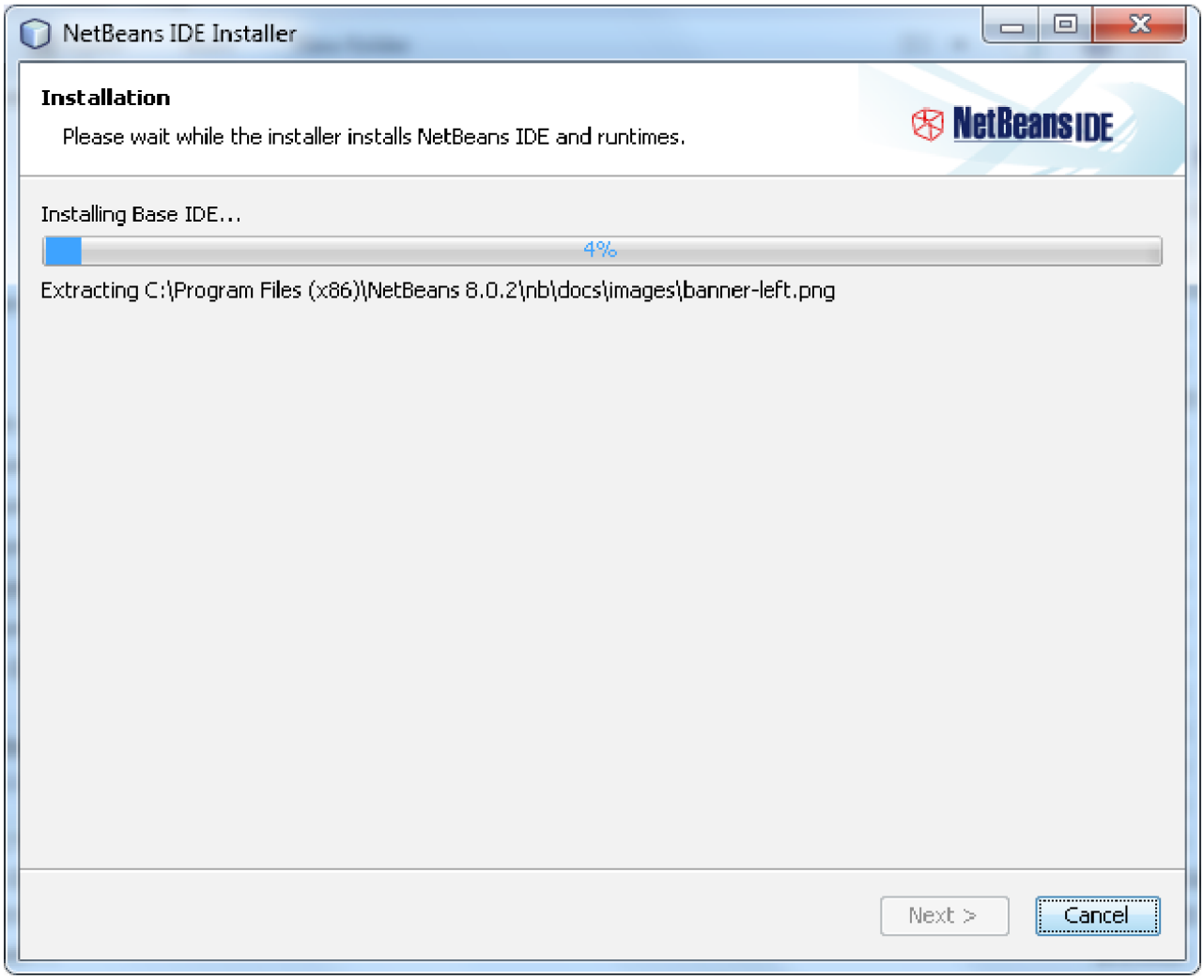

NetBeans IDE Installer
Welcome to the NetBeans IDE 8.0.2 Installer
The installer will install the NetBeans IDE with the following packs and runtimes.
Click Customize to select the packs and runtimes to install.
Base IDE
Java SE
Java EE
Java ME
HTML5
Java Card™ 3 Connected
C/C++
Groovy
PHP
Features on Demand
Runtimes
GlassFish Server Open Source Edition 4.1 (Already Installed)
Apache Tomcat 8.0.15 (Already Installed)
NetBeans IDE
Customize...
Installation Size: 588.8 MB
Next >
Cancel

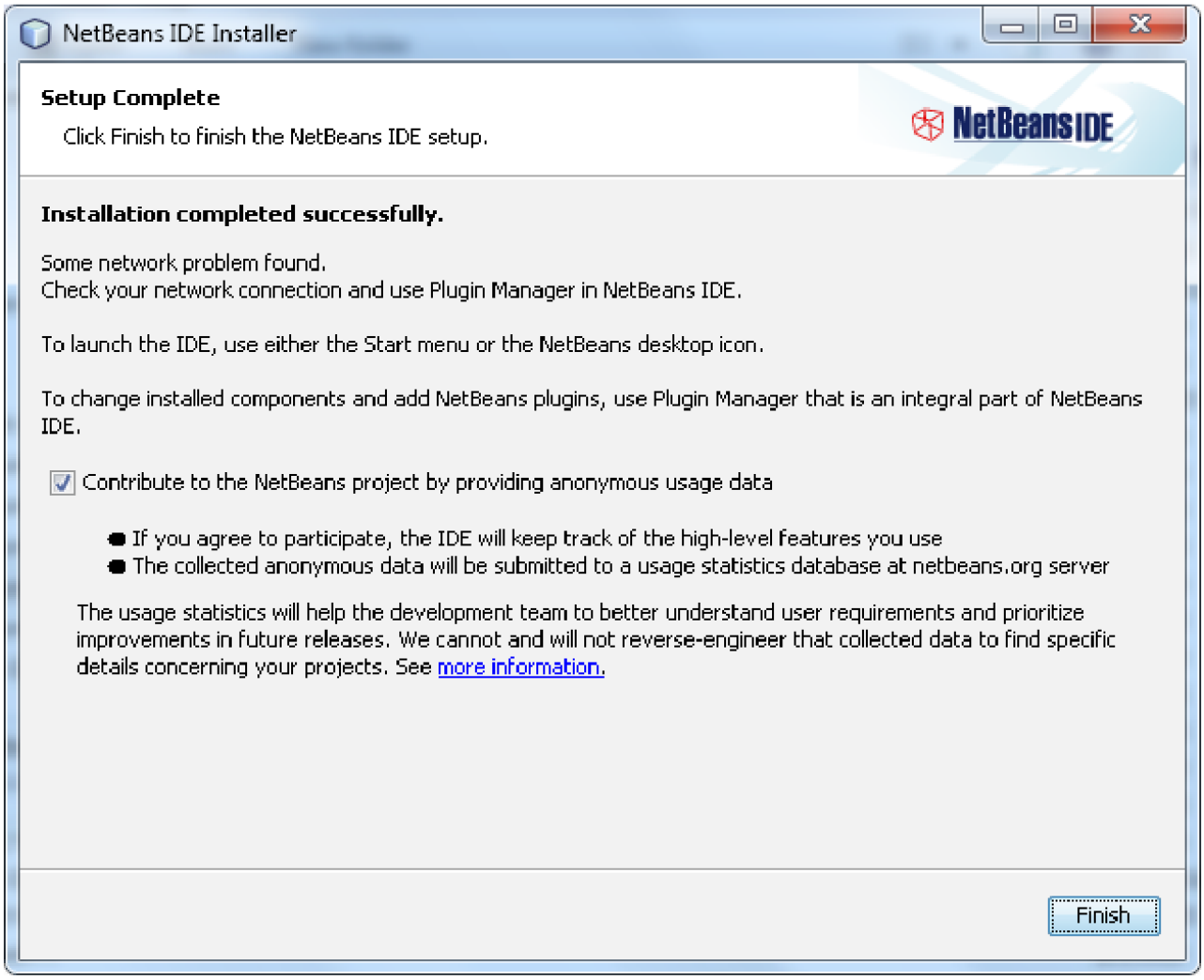

NetBeans IDE Installer
License Agreement
Please read the following license agreement carefully.
NetBeans IDE
NETBEANS IDE 8.0.2 ("Product") LICENSE AGREEMENT
PLEASE READ THE FOLLOWING LICENSE AGREEMENT TERMS AND CONDITIONS CAREFULLY, INCLUDING WITHOUT LIMITATION THOSE DISPLAYED ELSEWHERE (AS INDICATED BY LINKS LISTED BELOW), BEFORE USING THE SOFTWARE. THESE TERMS AND CONDITIONS CONSTITUTE A LEGAL AGREEMENT BETWEEN YOU, OR THE ENTITY FOR WHICH YOU ARE AN AUTHORIZED REPRESENTATIVE WITH FULL AUTHORITY TO ENTER INTO THIS AGREEMENT, AND ORACLE. BY CLICKING "ACCEPT" OR THE EQUIVALENT YOU AGREE TO ALL OF THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT. IF YOU DO NOT AGREE TO THIS LICENSE DO NOT CLICK "ACCEPT" OR THE EQUIVALENT AND DO NOT INSTALL OR USE THIS SOFTWARE.
Copyright (c) 1997, 2014, Oracle and/or its affiliates. All rights reserved.
Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.
I accept the terms in the license agreement
< Back
Next >
Cancel


NetBeans IDE Installer
JUnit License Agreement
Please read the following license agreement carefully.
NetBeans IDE
JUnit
Common Public License - v 1.0
THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.
1. DEFINITIONS
"Contribution" means:
a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and
b) in the case of each subsequent Contributor:
i) changes to the Program, and
Junit is a Java unit testing framework
I accept the terms in the license agreement. Install JUnit
Do not install JUnit
< Back
Next >
Cancel


NetBeans IDE Installer
NetBeans IDE 8.0.2 Installation
Choose the installation folder and JDK™.
NetBeans IDE
Install the NetBeans IDE to:
C:\Program Files (x86)\NetBeans 8.0.2
Browse...
JDK™ for the NetBeans IDE:
C:\Program Files (x86)\Java\jdk1.8.0_31
Browse...
A 32-bit JDK is selected. For improved performance on your 64-bit system, please use the 64-bit version of Java.
Download at http://www.oracle.com/technetwork/java/javase/downloads/ and restart NetBeans installer.
< Back
Next >
Cancel

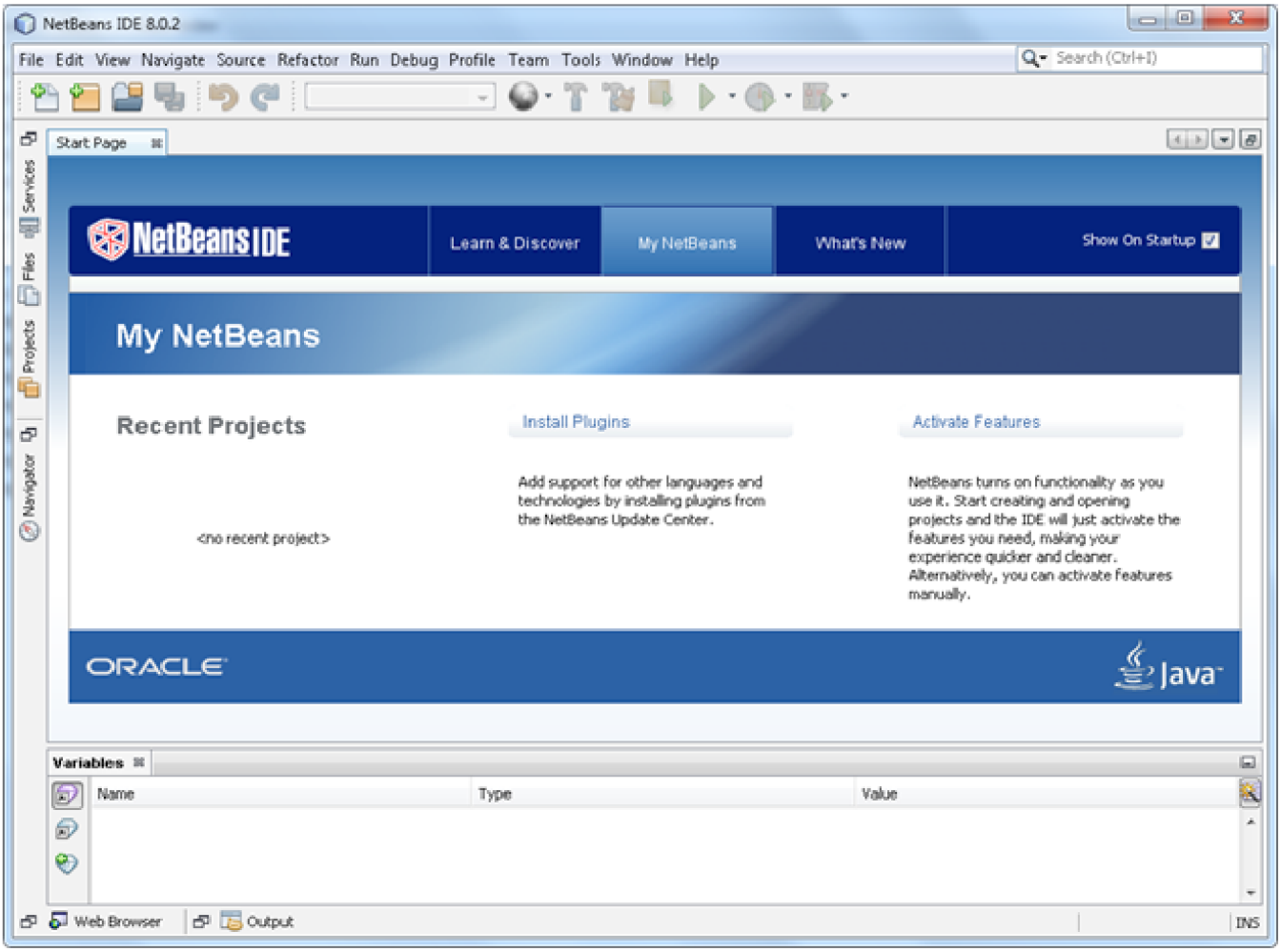

NetBeans IDE Installer
Summary
Click Install to start the installation.
NetBeans IDE
NetBeans IDE Installation Folder:
C:\Program Files (x86)\NetBeans 8.0.2
JUnit library will be downloaded and installed during installation of NetBeans IDE.
Check for Updates
The NetBeans installer can automatically check for updates of installed plugins using your Internet connection.
Total Installation Size:
588.8 MB
< Back
Install
Cancel


NetBeans IDE Installer
Installation
Please wait while the installer installs NetBeans IDE and runtimes.
NetBeans IDE
Installing Base IDE...
4%
Extracting C:\Program Files (x86)\NetBeans 8.0.2\nb\docs\images\banner-left.png
Next >
Cancel



وقتی برای اولین بار بر روی آیکون NetBeans بر روی دسکتاپ کلیک کرده و آن را اجرا می کنید، صفحه اول برنامه به صورت

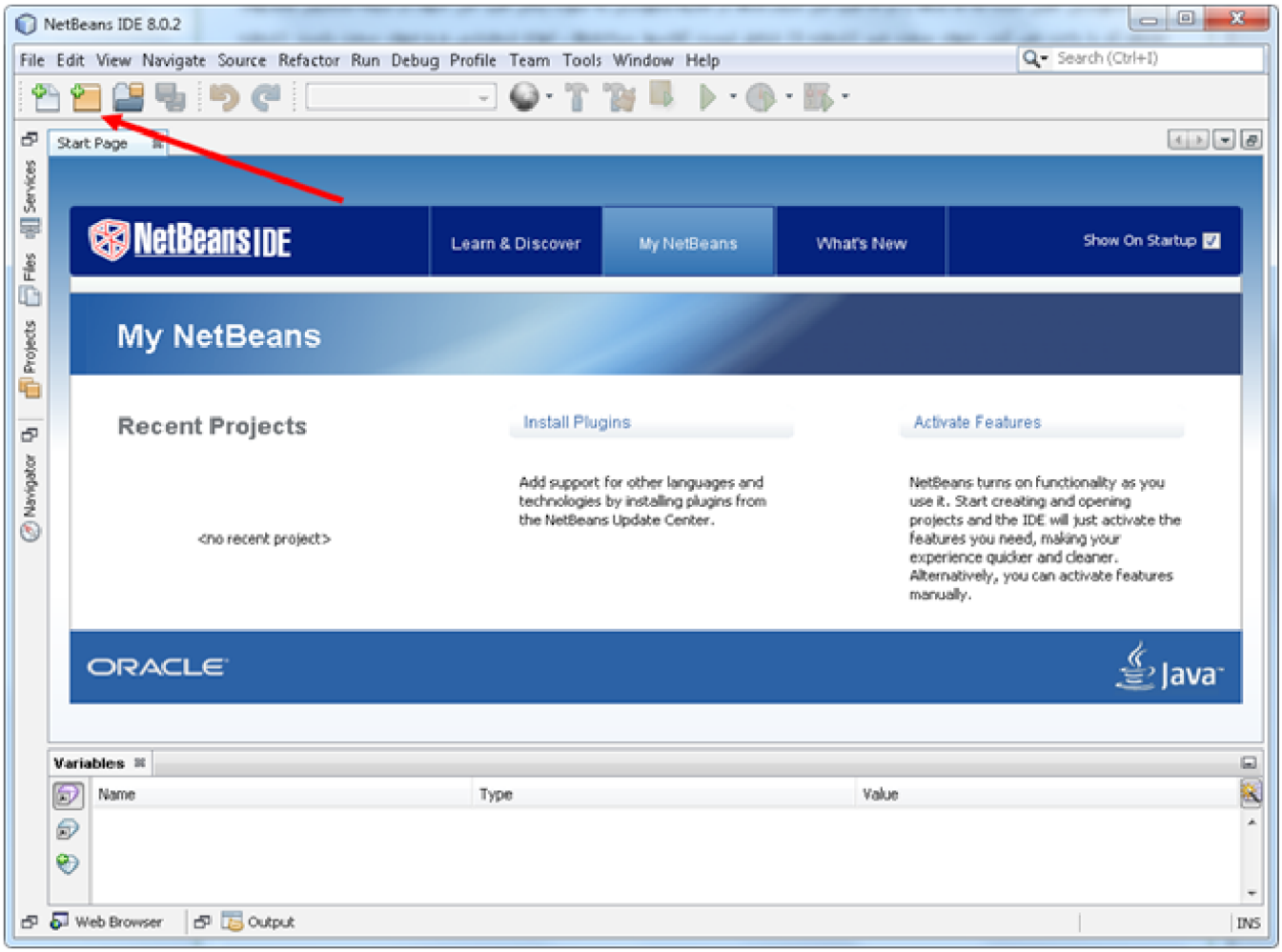
زیر نمایش داده می شود که نشان دهنده نصب کامل آن است:



در درس آینده درباره ایجاد پروژه در NetBeans توضیح می دهیم.

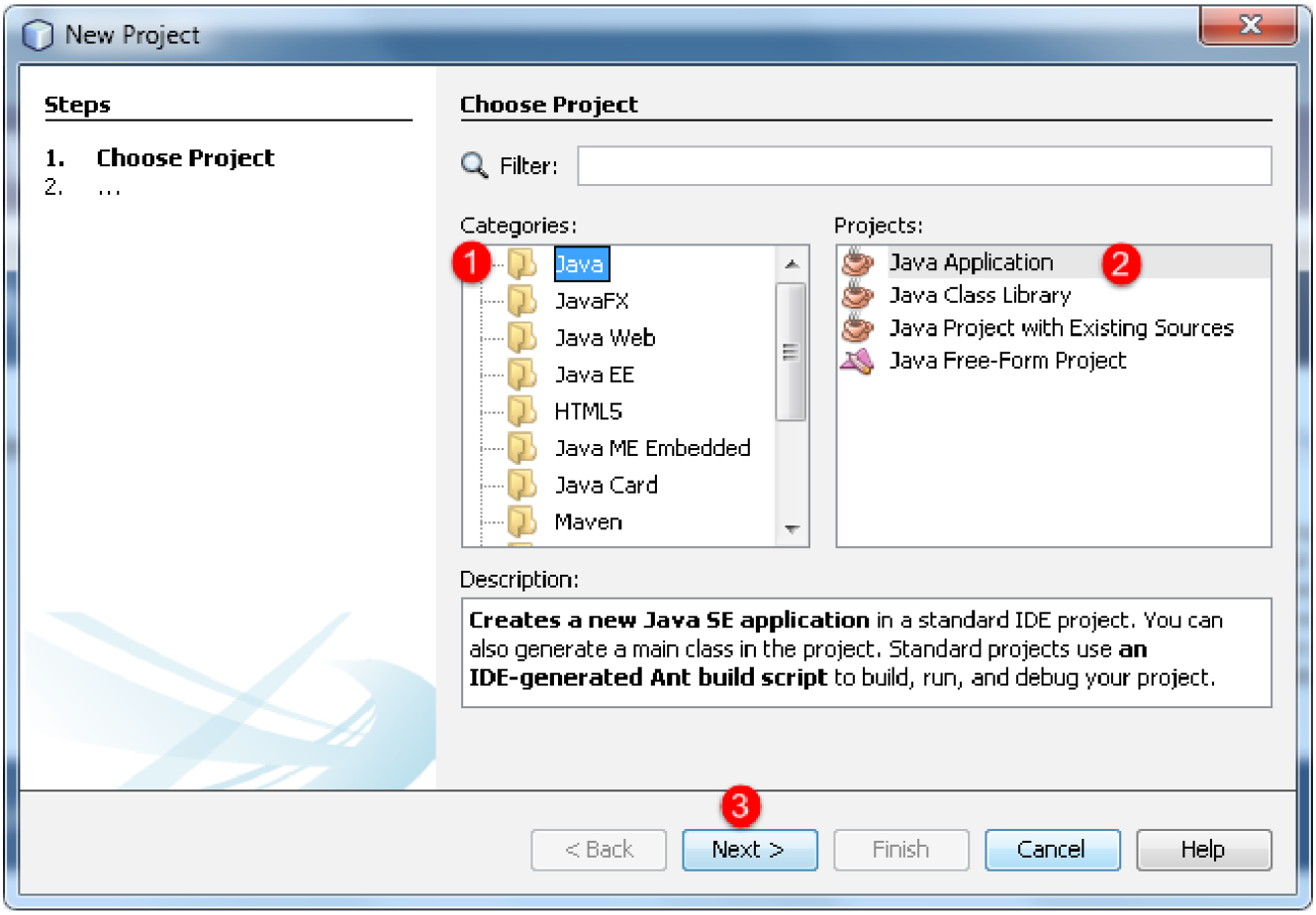
ساخت یک برنامه ساده درJAVA

اجازه بدهید یک برنامه بسیار ساده به زبان جاوا بنویسیم.این برنامه یک پیغام را نمایش می دهد.در این درس می خواهم ساختار و دستور زبان یک برنامه ساده جاوا را توضیح دهم. برنامه NetBeans را اجرا کنید. از مسیری که در شکل زیر نشان داده شده است یک پروژه جدید ایجاد کنید:

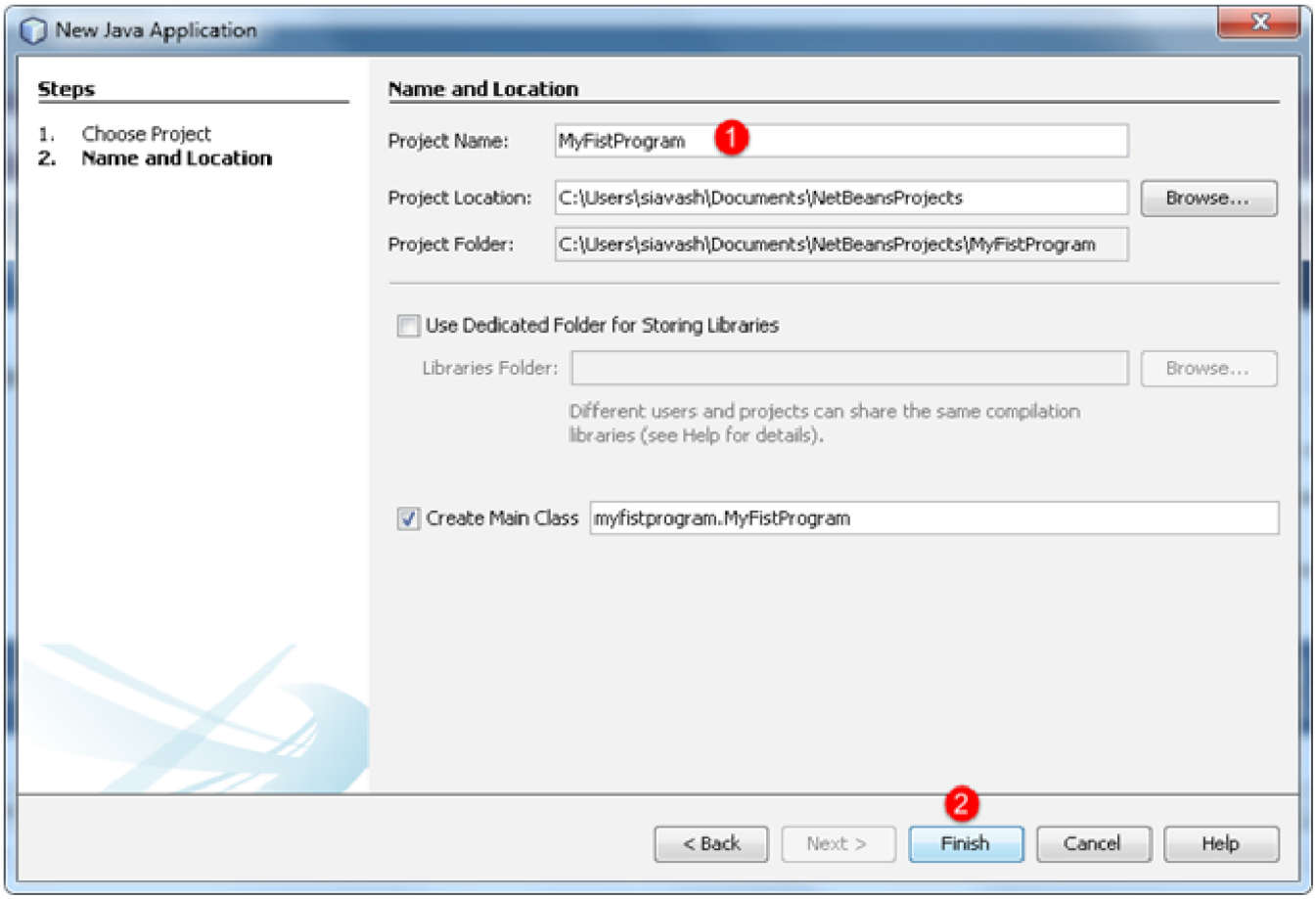


حال با یک صفحه مواجه می شوید. طبق شماره هایی که در شکل زیر نمایش داده شده اند گزینه ها را انتخاب کرده و به مرحله بعد

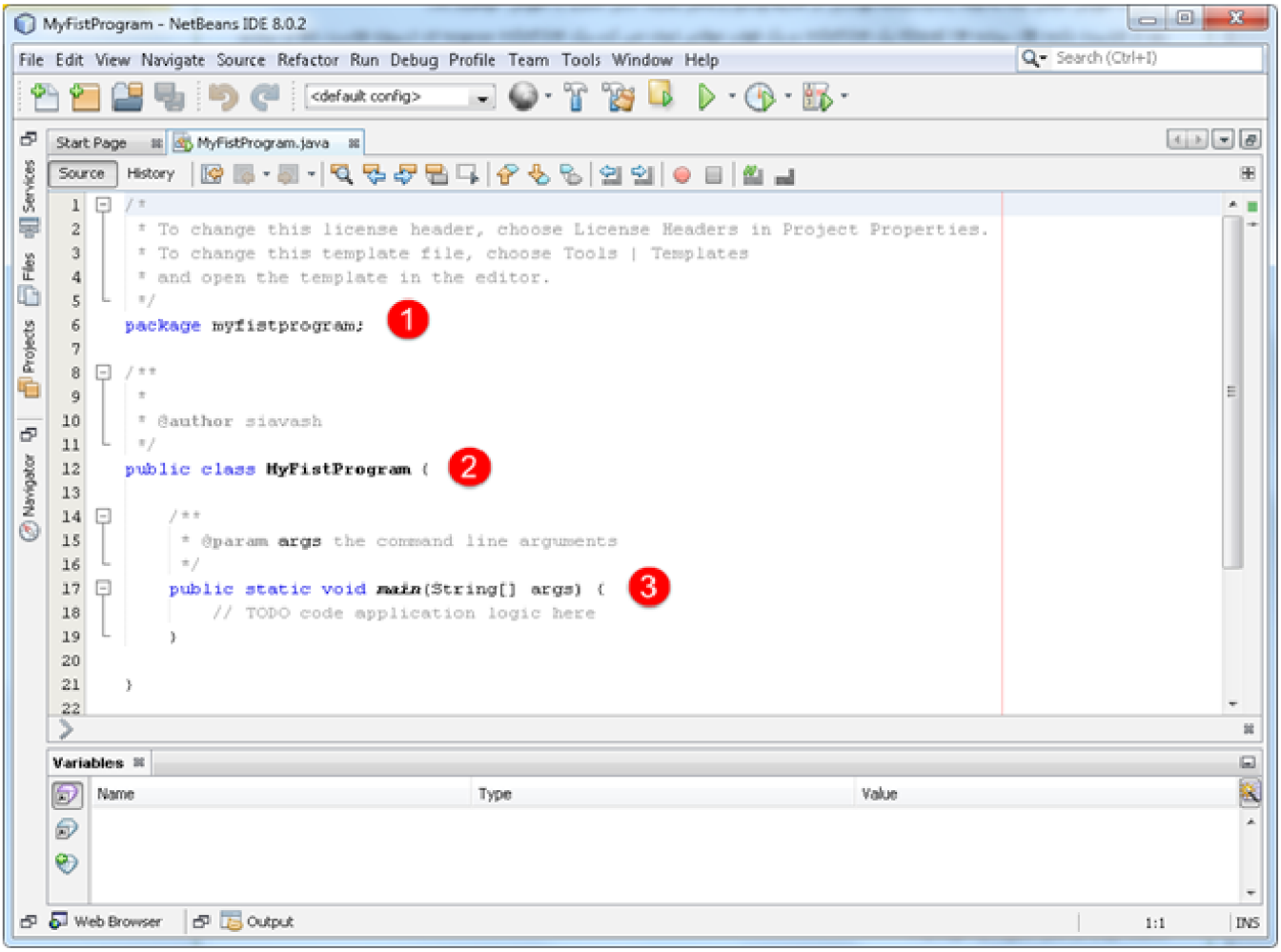
برویدِ:



با زدن دکمه Next صفحه ای به صورت زیر نمایش داده می شود. در این پنجره نام پروژه تان (MyFirstProgram) را نوشته و سپس بر روی دکمه Finish کلیک کنید:


New Java Application
Steps
1. Choose Project
2. Name and Location
Name and Location
Project Name:
MyFistProgram
1
Project Location:
C:\Users\siavash\Documents\NetBeansProjects
Browse...
Project Folder:
C:\Users\siavash\Documents\NetBeansProjects\MyFistProgram
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Browse...
Different users and projects can share the same compilation libraries (see Help for details).
Create Main Class
myfistprogram.MyFistProgram
2
< Back
Next >
Finish
Cancel
Help

بعد از فشردن دکمه Finish ، وارد محیط کدنویسی برنامه به صورت زیر می شویم:



محیط کدنویسی جایی است که ما کدها را در آن تایپ می کنیم.کدها در محیط کدنویسی به صورت رنگی تایپ می شوند در نتیجه تشخیص بخشهای مختلف کد را راحت می کند. همانطور که در شکل بالا مشاهده می کنید ما کدهای پیشفرض را به سه قسمت تقسیم کرده ایم. قسمت اولPackage ، قسمت دوم کلاس و قسمت سوم متد .mainنگران اصطلاحاتی که به کار بردیم نباشید آنها را در فصول بعد توضیح خواهم داد. در محل کد نویسی کدهایی از قبل نوشته شده که برای شروع شما آنها را پاک کنید و کدهای زیر را در محل کدنویسی بنویسید:

```
package myfirstprogram;
public class MyFirstProgram {
  public static void main(String[] args) {
    System.out.println("Welcome to JAVA Tutorials!");
  }
}
```

ساختار یک برنامه در جاوا

مثال بالا ساده ترین برنامه ای است که شما می توانید در جاوا بنویسید .هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است.هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. اجازه بدهید هر خط کد را در مثال بالا توضیح بدهیم. در خط اول Packageتعریف شده است که شامل کدهای نوشته شده توسط شما است و از تداخل نامها جلوگیری می کند.در باره Package در درسهای آینده توضیح خواهیم داد.

در خط دوم آکولاد ({) نوشته شده است.آکولاد برای تعریف یک بلوک کد به کار می رود.جاوا یک زبان ساخت یافته است که شامل کدهای زیاد و ساختارهای فراوانی می باشد.هر آکولاد باز({) در جاوا باید دارای یک آکولاد بسته (}) نیز باشد.همه کدهای نوشته شده از خط 2 تا خط 6 یک بلوک کد است.

در خط 2 یک کلاس تعریف شده است.در باره کلاسها در فصلهای آینده توضیح خواهیم داد.در مثال بالا کدهای شما باید در داخل یک کلاس نوشته شود.بدنه کلاس شامل کد های نوشته شده از خط 2 تا 6 می باشد. خط 3 متد Main یا متد اصلی نامیده می شود. هر متد شامل یک سری کد است که وقتی اجرا می شوند که متد را صدا بزنیم. در باره متد و نحوه صدا زدن آن در فصول بعدی توضیح خواهیم داد.متد Main نقطه آغاز اجرای برنامه است.این بدان معناست که ابتدا تمام کدهای داخل متد Main و سپس بقیه کدها اجرا می شود.در باره متد Main در فصول بعدی توضیح خواهیم داد.متد Main و سایر متدها دارای آکولاد و کدهایی در داخل آنها می باشند و وقتی کدها اجرا می شوند که متدها را صدا بزنیم.هر خط کد در جاوا به یک سیمیکولن (;) ختم می شود.اگر سیمیکولن در آخر خط فراموش شود برنامه با خطا مواجه می شود. مثالی از یک خط کد در جاوا به صورت زیر است:

```
System.out.println("Welcome to JAVA Tutorials!");
```

این خط کد پیغام Welcome to JAVA Tutorials! را در صفحه نمایش نشان می دهد.از متد println() برای چاپ یک رشته استفاده می شود.یک رشته گروهی از کاراکترها است که به وسیله دابل کوتیشن (") محصور شده است. مانند "Welcome to Visual C# Tutorials!"

یک کاراکتر می تواند یک حرف،عدد، علامت یا ....باشد.در کل مثال بالا نحوه استفاده از متد println() نشان داده شده است. این متد یک متد از کلاس PrintStreamبوده و از آن برای چاپ مقدیر استفاده می شود .یک فیلد استاتیک کلاس System و کلاس System هم هم یک کلاس از پیش تعریف شده در جاوا می باشد.جاوا فضای خالی و خطوط جدید را نادیده می گیرد.بنابراین شما می توانید همه برنامه را در یک خط بنویسید.اما اینکار خواندن و اشکال زدایی برنامه را مشکل می کند. یکی از خطاهای معمول در برنامه نویسی فراموش کردن سیمیکولن در پایان هر خط کد است.به مثال زیر توجه کنید:

```
System.out.println("Welcome to JAVA Tutorials!");
```

جاوا فضای خالی بالا را نادیده می گیرد و از کد بالا اشکال نمی گیرد. اما از کد زیر ایراد می گیرد.

```
System.out.println(;
        "Welcome to JAVA Tutorials!");
```

به سیمیکولن آخر خط اول توجه کنید.برنامه با خطای نحوی مواجه می شود چون دو خط کد مربوط به یک برنامه هستند و شما فقط باید یک سیمیکولن در آخر آن قرار دهید. همیشه به یاد داشته باشید که جاوا به بزرگی و کوچکی حروف حساس است.یعنی به

طور مثال MAN و man در جاوا با هم فرق دارند. رشته ها و توضیحات از این قاعده مستثنی هستند که در درسهای آینده توضیح خواهیم داد .مثلا کدهای زیر با خطا مواجه می شوند و اجرا نمی شوند:

```
system.out.println("Welcome to JAVA Tutorials!");
SYSTEM.OUT.PRINTLN("Welcome to JAVA Tutorials!");
sYsTem.oUt.pRinTLn("Welcome to JAVA Tutorials!");
```

تغییر در بزرگی و کوچکی حروف از اجرای کدها جلوگیری می کند. اما کد زیر کاملا بدون خطا است:

```
System.out.println("Welcome to JAVA Tutorials!");
```

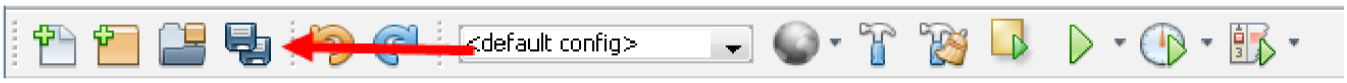
همیشه کدهای خود را در داخل آکولاد بنویسید.

```
{
  statement1;
}
```

این کار باعث می شود که کدنویسی شما بهتر به چشم بیاید و تشخیص خطاها راحت تر باشد. یکی از ویژگیهای مهم جاوا نشان دادن کدها به صورت تو رفتگی است بدین معنی که کدها را به صورت تو رفتگی از هم تفکیک می کند و این در خوانایی برنامه بسیار موثر است.

ذخیره پروژه و اجرای برنامه

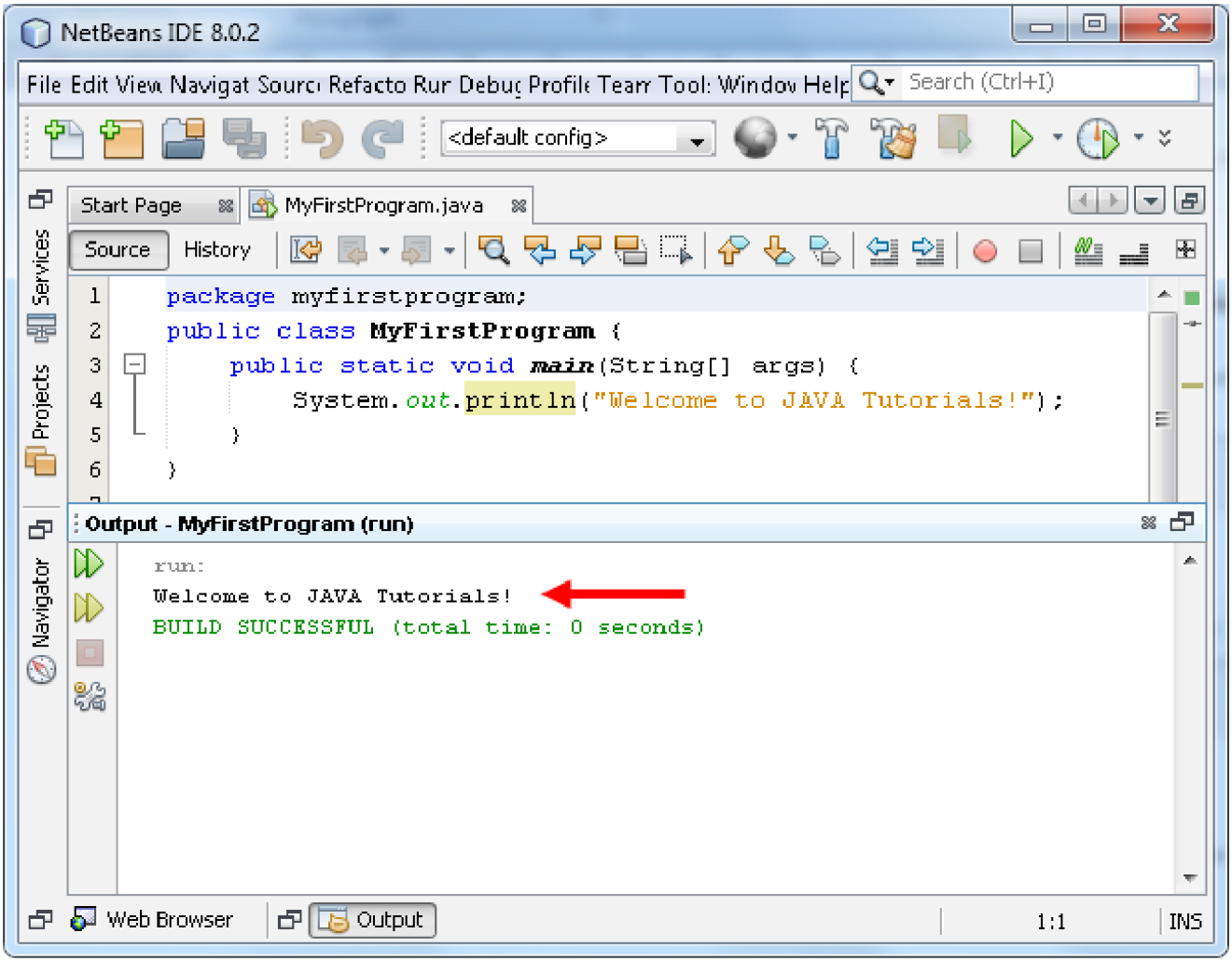
برای ذخیره پروژه و برنامه می توانید به مسیر File > Save All بروید یا از کلیدهای میانبر Ctrl+Shift+S استفاده کنید. همچنین می توانید از قسمت Toolbarبر روی شکل زیر کلیک کنید:



و برای اجرای برنامه هم از فلش سبزرنگ موجود در Toolbar و یا دکمه F6 استفاده کنید:

با اجرای برنامه بالا مشاهده می کنید که رشته Welcome to JAVA Tutorials! در خروجی برنامه به صورت زیر نمایش داده می

شود:



وجود خط سبز در پایین فلش قرمز در شکل بالا نشان دهنده اجرای بدون نقص برنامه می باشد. حال که با خصوصیات و ساختار اولیه جاوا آشنا شدید در دسهای آینده مطالب بیشتری از این زبان برنامه نویسی قدرتمند خواهید آموخت.

استفاده از Package

برای دسته بندی کلاس ها و قرار دادن کلاس های مرتبط با هم در یک مکان، جاوا از مفهومی به نام بسته یا package استفاده می کند Package .معادل فضای نام در سی شارپ هستند.یک دلیل برای گروه بندی کلاس ها در package این است که امکان دارد دو برنامه نویس از دو کلاس هم نام استفاده کنند.با این کار از چنین برخوردهایی جلوگیری به عمل می آید.یعنی اگر دو کلاس هم نام در دو Packageغیر همنام باشند مشکلی به وجود نمی آید. همانطور که در مثال بالا دیدید به طور پیشفرض NetBeatns هنگام ایجاد برنامه یک Package با همنام یا اسمی که برای برنامه انتخاب کرده ایم با حروف کوچک و در داخل این Package هم کلاسی به همین اسم ایجاد می کند:

```
package myfirstprogram;

public class MyFirstProgram
{
    ...
}
```

برای وارد کردن کلاس یک Package در داخل Package دیگر از کلمه کلیدی import به صورت زیر استفاده می شود:

```
import PackageName.ClassName
```

همانطور که در مثال بالا مشاهده می کنید برای استفاده از کلاسی که در یک Packageقرار دارد در Package دیگر ابتدا کلمه importسپس نام Package ، بعد علامت نقطه و در آخر نام کلاس را می نویسیم. مثلا برای استفاده از کلاس MyFirstProgram مربوط به پکیج myfirstprogram به صورت زیر عمل می شود:

```
import myfirstprogram.MyFirstProgram;
```

بسته ها را می توان به صورت تو در تو تعریف کرد. در این حالت در تعریف بسته یک کلاس، از بیرونی ترین بسته شروع کرده و هر بسته را با نقطه (.) به بسته بعدی متصل می کنیم:

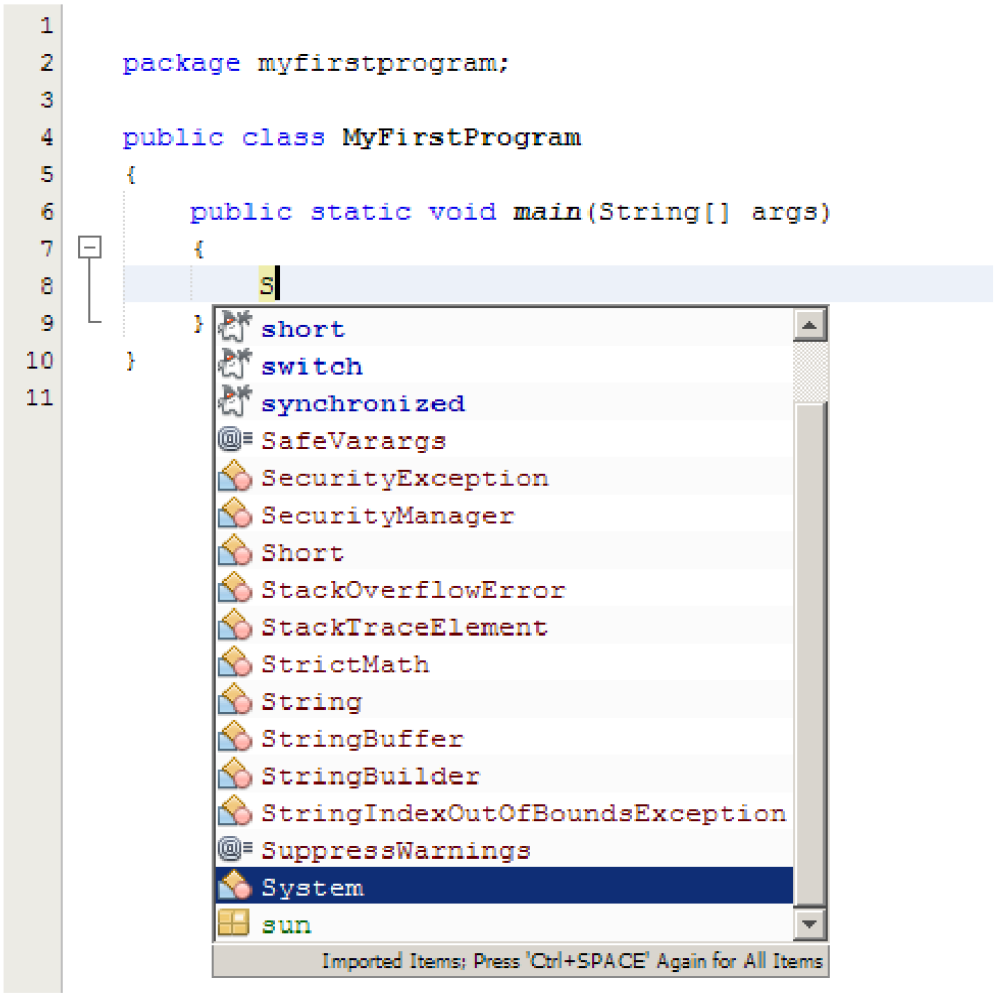
```
import firstPackage.secondPackage.ClassName
```

استفاده از IntelliSense درNetBeans

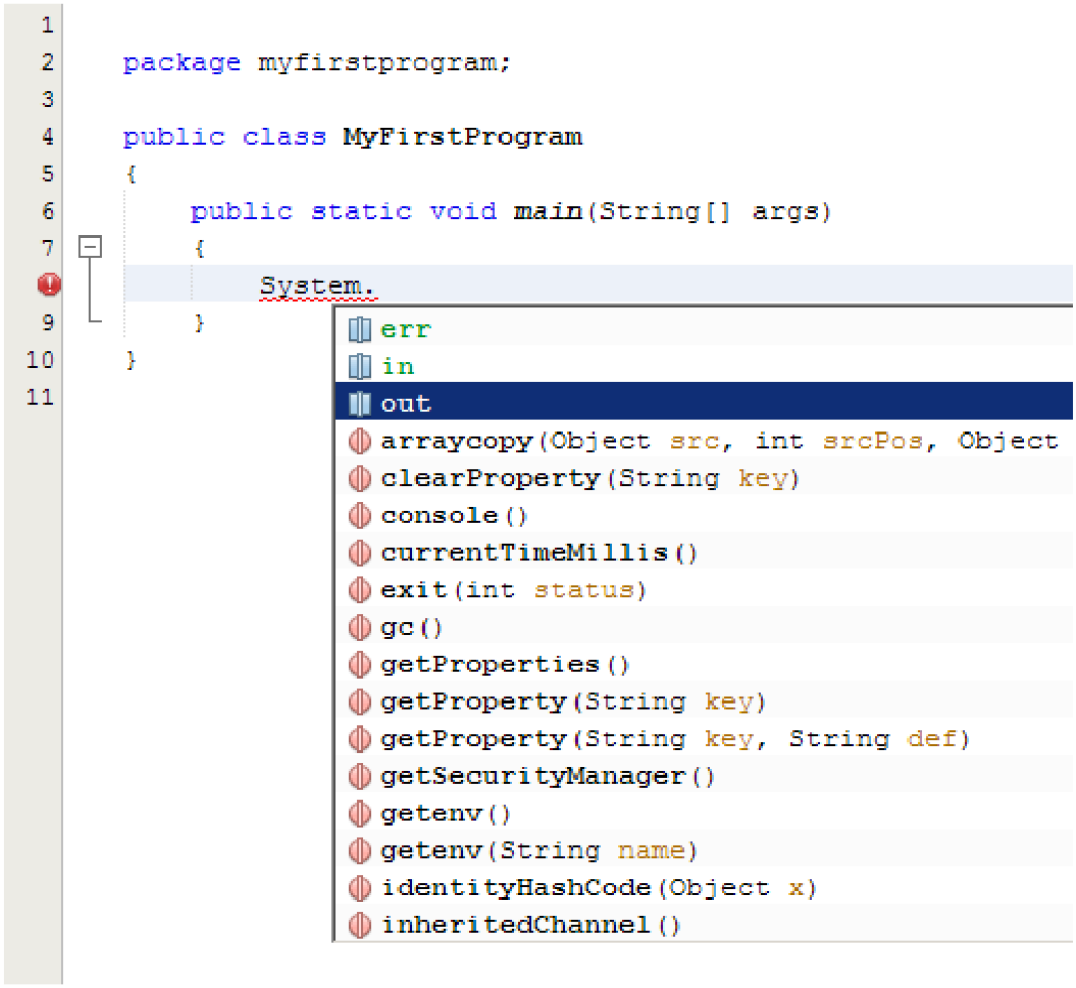
شاید یکی از ویژگیهای مهمNetBeans ، اینتل لایسنس باشد IntelliSense .ما را قادر می سازد که به سرعت به کلاسها و متدها و ....دسترسی پیدا کنیم.وقتی که شما در محیط کدنویسی حرفی را تایپ کنید IntelliSenseفورا فعال می شود. کد زیرا را در داخل متد main() بنویسید.

```
System.out.println("Welcome to JAVA Tutorials!");
```

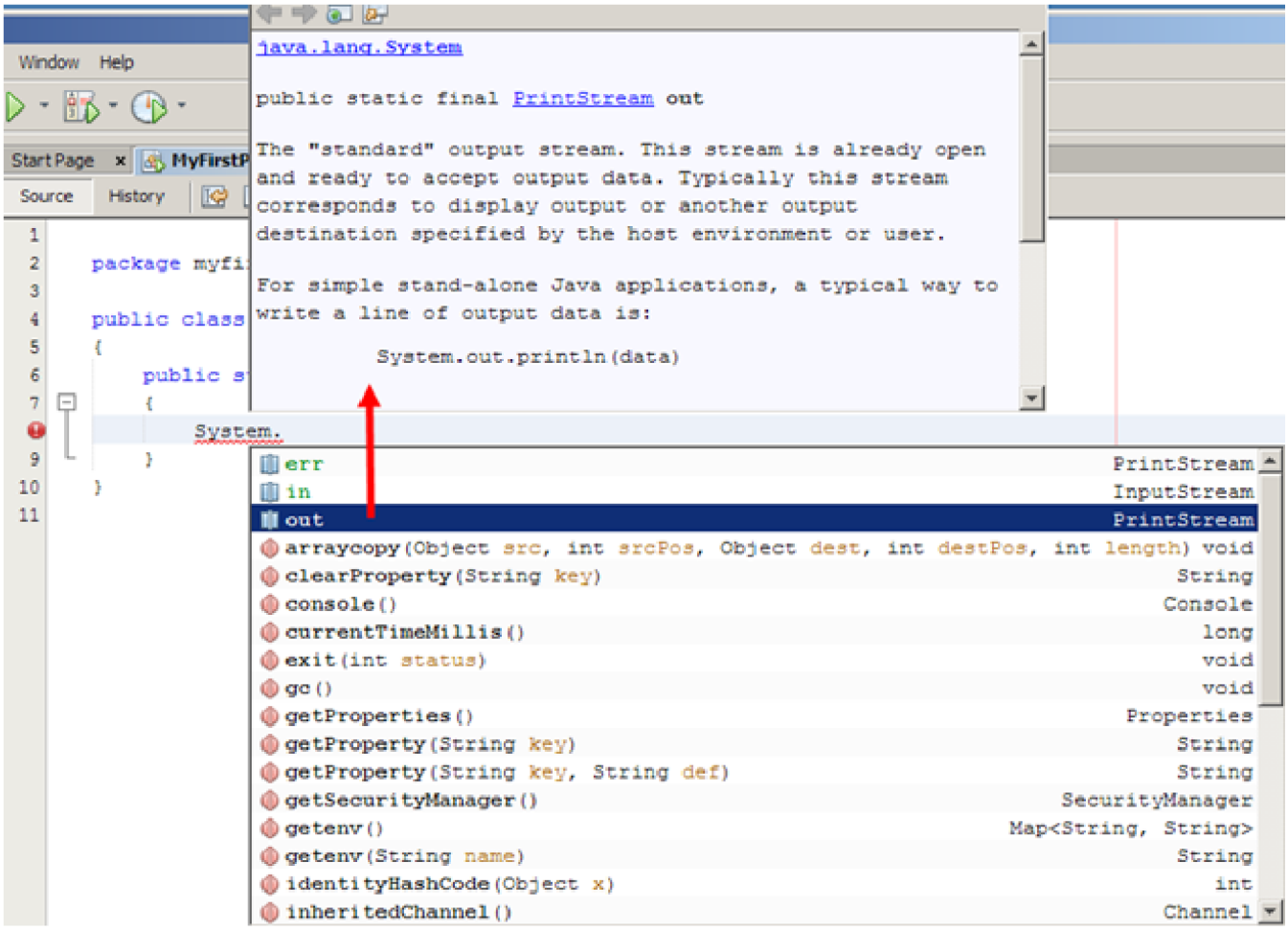
اولین حرف را تایپ کرده و سپس دکمه های ترکیبی Ctrl+Space را فشار دهید تا IntelliSense فعال شود:



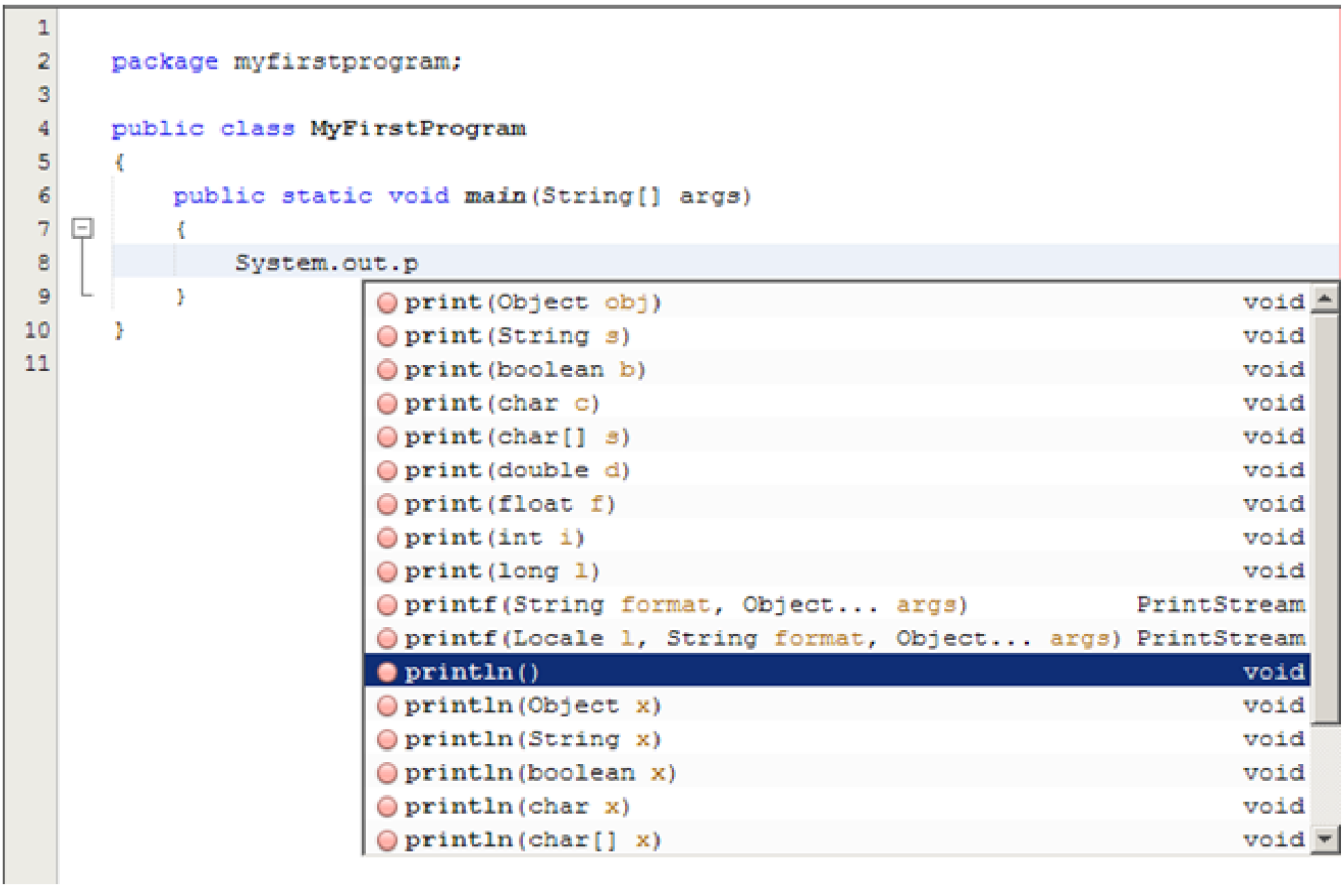
IntelliSenseلیستی از کلمات به شما پیشنهاد می دهد که بیشترین تشابه را با نوشته شما دارند.شما می توانید با زدن دکمه tab گزینه مورد نظرتان را انتخاب کنید. با تایپ نقطه (.) شما با لیست پیشنهادی دیگری مواجه می شوید:



اگر بر روی گزینه ای که می خواهید انتخاب کنید لحظه ای مکث کنید توضیحی در رابطه با آن مشاهده خواهید کرد مانند شکل زیر:

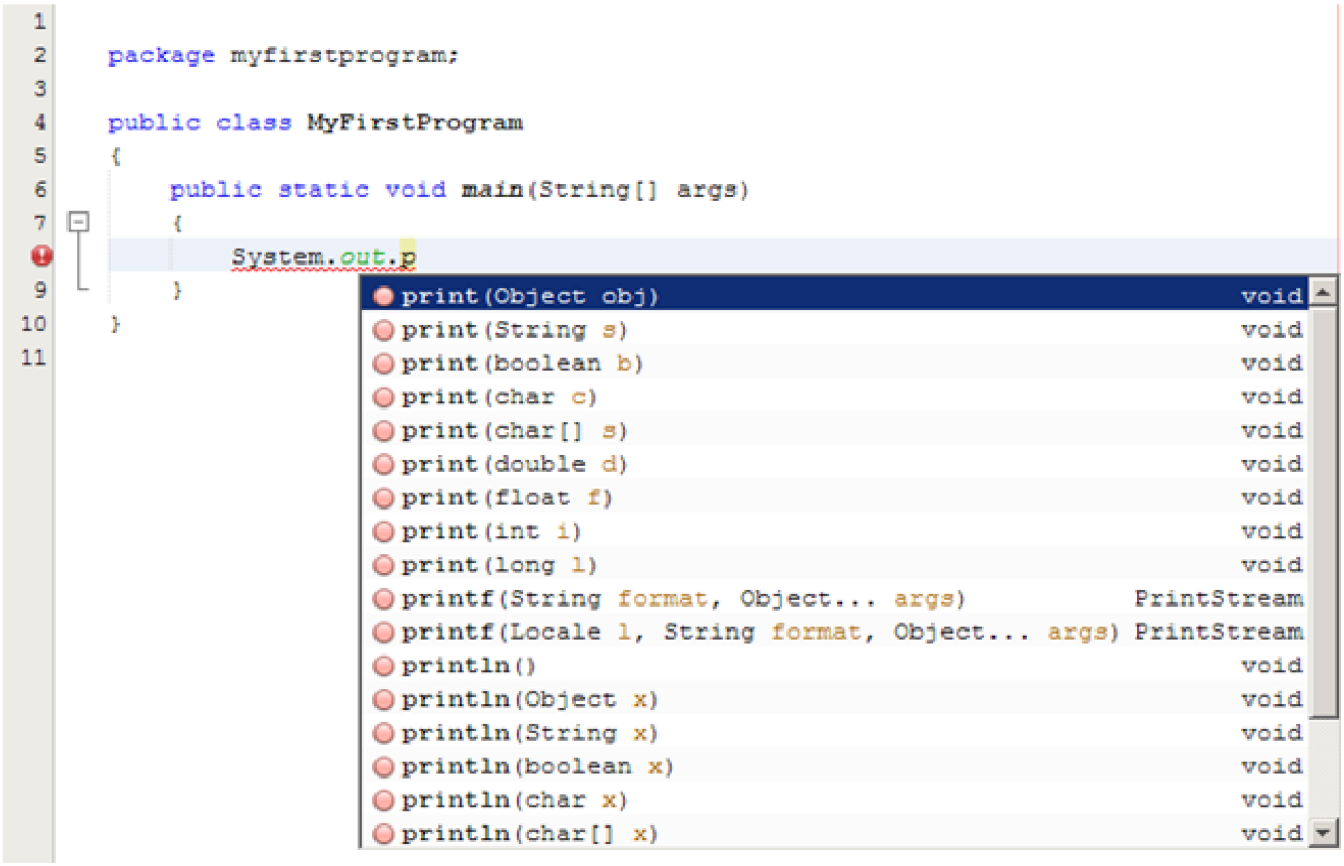


هر چه که به پایان کد نزدیک می شوید لیست پیشنهادی محدود تر می شود.برای مثال با تایپ p ، اینتل لایسنس فقط کلماتی را که دارای حرف p هستند را نمایش می دهد:



با تایپ حرف های بیشتر لیست محدودتر می شود. اگر IntelliSense نتواند چیزی را که شما تایپ کرده اید پیدا کند هیچ چیزی را نمایش نمی دهد. برای ظاهر کردن IntelliSense کافیست دکمه ترکیبی Ctrl+Space را فشار دهید.برای انتخاب یکی از متدهایی که دارای چند حالت هستند، می توان با استفاده از دکمه های مکان نما (بالا و پایین) یکی از حالت ها را انتخاب کرد. مثلا متد

println()همانطور که در شکل زیر مشاهده می کنید دارای چندین حالت نمایش پیغام در صفحه است:



IntelliSenseبه طور هوشمند کدهایی را به شما پیشنهاد می دهد و در نتیجه زمان نوشتن کد را کاهش می دهد.

رفع خطاها

بیشتر اوقات هنگام برنامه نویسی با خطا مواجه می شویم. تقریبا همه برنامه هایی که امروزه می بینید حداقل از داشتن یک خطا رنج می برند.خطا ها می توانند برنامه شما را با مشکل مواجه کنند .در جاوا سه نوع خطا وجود دارد :

خطای کامپایلری

این نوع خطا از اجرای برنامه شما جلوگیری می کند. این خطا ها شامل خطای دستور زبان می باشد.این بدین معنی است که شما قواعد کد نویسی را رعایت نکرده اید.یکی دیگر از موارد وقوع این خطا هنگامی است که شما از چیزی استفاده می کنید که نه وجود

دارد و نه ساخته شده است. حذف فایلها یا اطلاعات ناقص در مورد پروژه ممکن است باعث به وجود آمدن خطای کامپایلری شود.استفاده از برنامه بوسیله برنامه دیگر نیز ممکن است باعث جلوگیری از اجرای برنامه و ایجاد خطای کامپایلری شود.

خطاهای منطقی

این نوع خطا در اثر تغییر در یک منطق موجود در برنامه به وجود می آید.رفع این نوع خطاها بسیار سخت است چون شما برای یافتن آنها باید کد را تست کنید.نمونه ای از یک خطای منطقی برنامه ای است که دو عدد را جمع می کند ولی حاصل تفریق دو عدد را نشان می دهد.در این حالت ممکن ات برنامه نویس علامت ریاضی را اشتباه تایپ کرده باشد.

استثناء

این نوع خطاها هنگامی رخ می دهند که برنامه در حال اجراست. این خطا هنگامی روی می دهد که کاربر یک ورودی نامعتبر به برنامه بدهد و برنامه نتواند آن را پردازش کند.

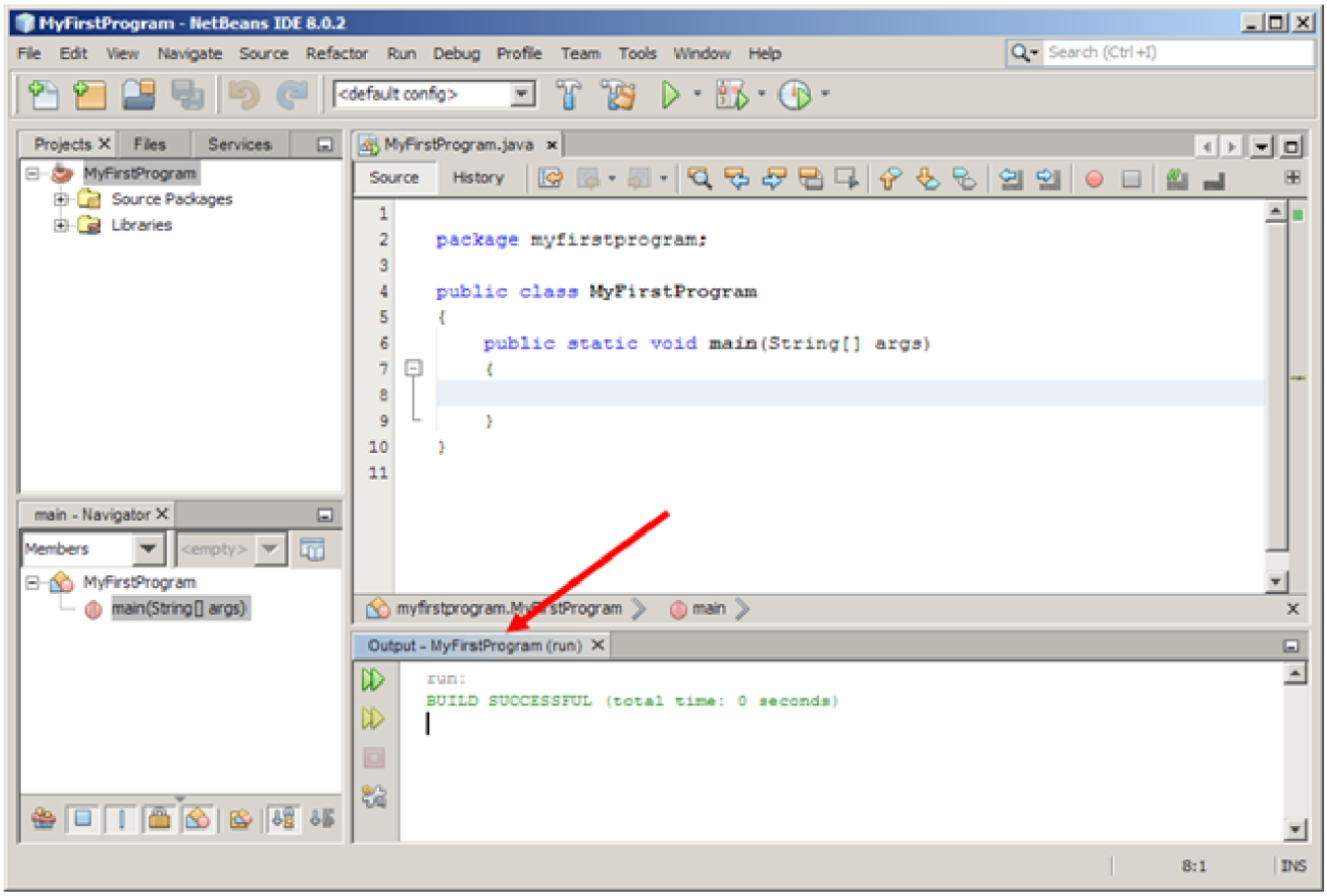
NetBeans دارای ابزارهایی برای پیدا کردن و برطرف کردن خطاها هستند.وقتی در محیط کدنویسی در حال تایپ کد هستیم یکی از ویژگیهای NetBeans تشخیص خطاهای ممکن قبل از اجرای برنامه است.زیر کدهایی که دارای خطای کامپایلری هستند خط قرمز کشیده می شود.

```
package myfirstprogram;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
        System.oUt.println();
        System.out.println()
    }
}
```

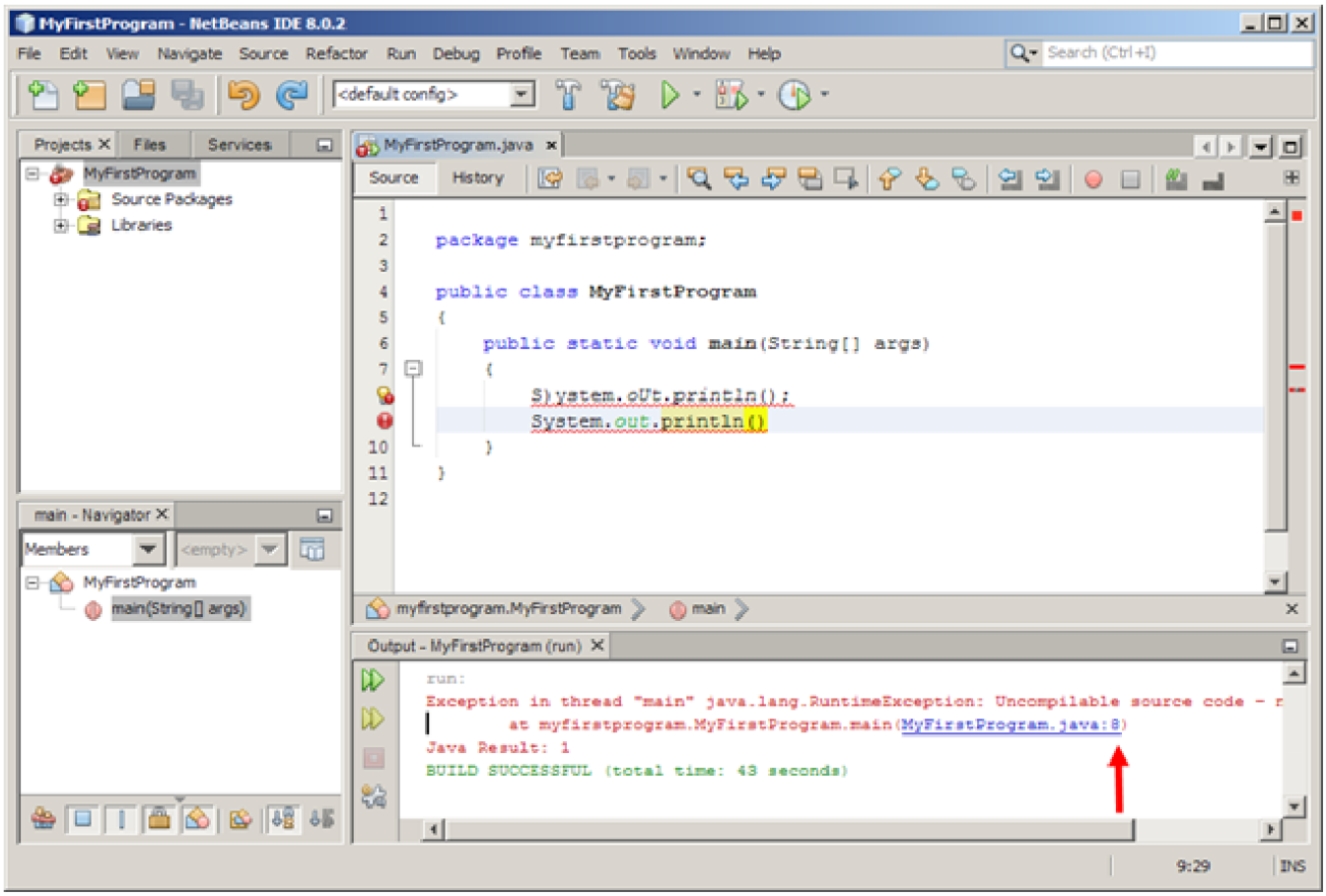
هنگامی که شما با موس روی این خطوط توقف کنید توضیحات خطا را مشاهده می کنید.شما ممکن است با خط سبز هم مواجه شوید که نشان دهنده اخطار در کد است ولی به شما اجازه اجرای برنامه را می دهند.به عنوان مثال ممکن است شما یک متغیر را تعریف کنید ولی در طول برنامه از آن استفاده نکنید. (در درس های آینده توضیح خواهیم داد).

```
package myfirstprogram;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
        int number;
    }
}
```

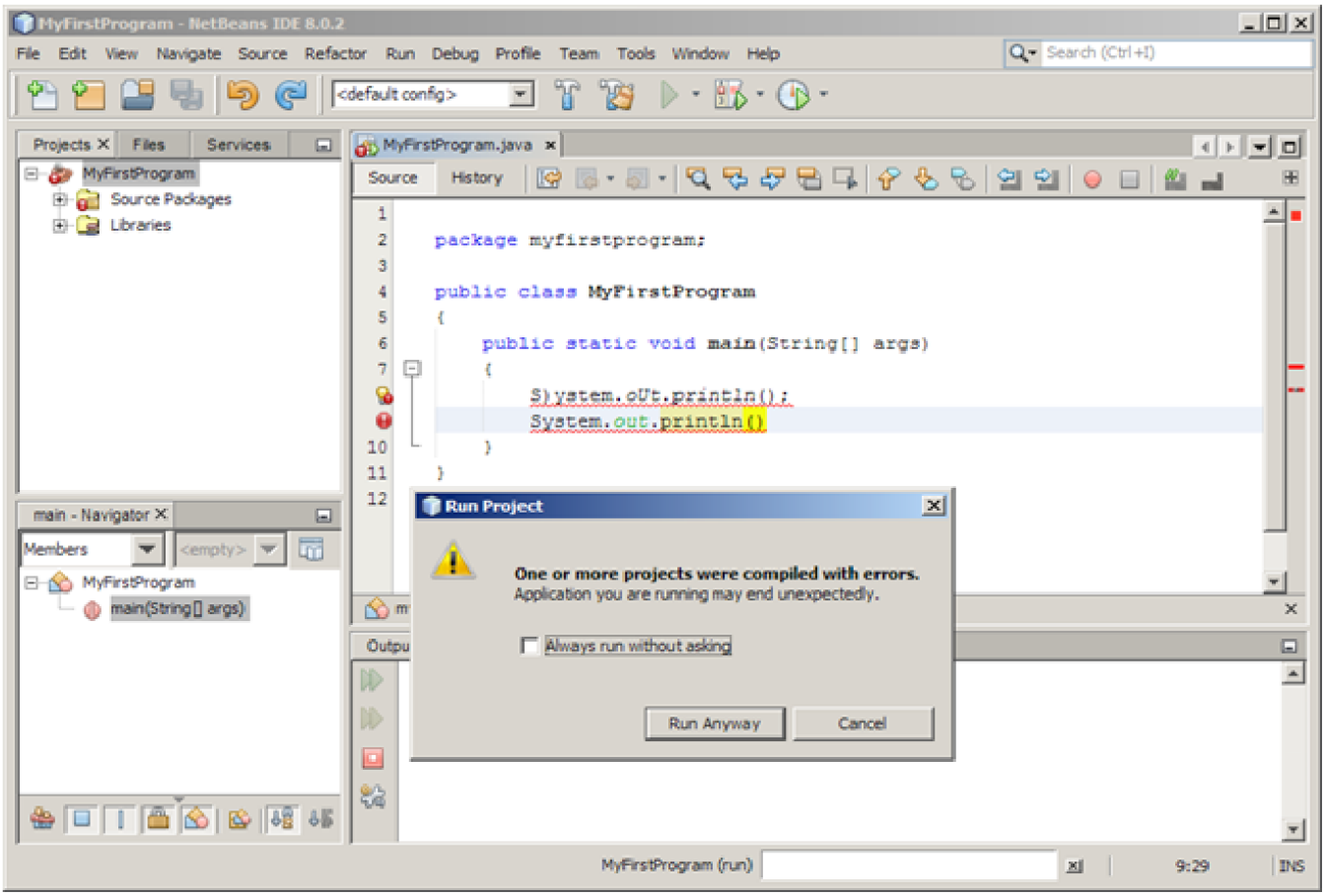
در باره رفع خطاها در آینده توضیح بیشتری می دهیم. پنجره Output که در شکل زیر با فلش قرمز نشان داده شده است به شما امکان مشاهده خطاها ، هشدارها و رفع آنها را می دهد.



همانطور که در شکل زیر مشاهده می کنید هرگاه برنامه شما با خطا مواجه شود لیست خطاها در پنجره Output نمایش داده می شود.



در شکل بالا علت به وجود آمدن خطا و شماره خطی که خطا در آن رخ داده است، نمایش داده شده است. اگر برنامه شما دارای خطا باشد و آن را اجرا کنید با پنجره زیر روبرو می شوید :



مربع کوچک داخل پنجره بالا را تیک زنید چون دفعات بعد که برنامه شما با خطا مواجه شود دیگر این پنجره به عنوان هشدار نشان داده نخواهد شد. با کلیک بر رو ی دکمه Run Anyway برنامه با وجود خطا نیز اجرا می شود.اما با کلیک بر روی دکمه Cancel اجرای برنامه متوقف می شود و شما باید خطاهای موجود در پنجره Output را بر طرف نمایید.

کاراکترهای کنترلی

کاراکترهای کنترلی کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می شوند و به دنبال آنها یک حرف یا عدد می آید و یک رشته را با فرمت خاص نمایش می دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می توان از کاراکتر کنترلی \n استفاده کرد:

```
System.out.println("Hello\nWorld!");
```

```
Hello
World
```

مشاهده کردید که کامپایلر بعد از مواجهه با کاراکتر کنترلی \nنشانگر موس را به خط بعد برده و بقیه رشته را در خط بعد نمایش می دهد. متد println()هم مانند کاراکتر کنترلی \nیک خط جدید ایجاد می کند ، البته بدین صورت که در انتهای رشته یک کاراکتر کنترلی \nاضافه می کند:

```
System.out.println("Hello World!");
```

کد بالا و کد زیر هیچ فرقی با هم ندارند:

```
System.out.print("Hello World!\n");
```

متد print()کارکردی شبیه به println()دارد با این تفاوت که نشان گر موس را در همان خط نگه می دارد و خط جدید ایجاد نمی کند. جدول زیر لیست کاراکترهای کنترلی و کارکرد آنها را نشان می دهد:

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\’
خط جدید	\n	چاپ دابل کوتیشن	\”
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	حرکت به عقب	\b

ما برای استفاده از کاراکترهای کنترلی از بک اسلش (\) استفاده می کنیم.از آنجاییکه \معنای خاصی به رشته ها می دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم:

```
System.out.println("We can print a \\ by using the \\\\ escape sequence.");
```
```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\،نشان دادن مسیر یک فایل در ویندوز است:

```
System.out.println("C:\\Program Files\\Some Directory\\SomeFile.txt");
```
```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (“) برای نشان دادن رشته ها استفاده می کنیم برای چاپ آن از \”استفاده می کنیم:

```
System.out.println("I said, \"Motivate yourself!\".");
```
```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (‘) از \’استفاده می کنیم:

```
System.out.println("The programmer\'s heaven.");
```
```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از t\استفاده می شود:

```
System.out.println("Left\tRight");
```
```
Left    Right
```

هر تعداد کاراکتر که بعد از کاراکتر کنترلی r\بیایند به اول سطر منتقل و جایگزین کاراکترهای موجود می شوند:

```
System.out.println("Mitten\rK");
```
```
K
```

مثلا در مثال بالا کاراکتر Kبعد از کاراکتر کنترلی r\آمده است.کاراکتر کنترلی حرف Kرا به ابتدای سطر برده و جایگزین Mittenمی کند .

برای مشاهده لیست مقادیر مبنای 16 برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید :

http://www.ascii.cl/htmlcodes.htm

اگر کامپایلر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می دهد.بیشترین خطا زمانی اتفاق می افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می کند.

متغیر

متغیر مکانی از حافظه است که شما می توانید مقادیری را در آن ذخیره کنید.می توان آن را به عنوان یک ظرف تصور کرد که داده های خود را در آن قرار داده اید.محتویات این ظرف می تواند پاک شود یا تغییر کند.هر متغیر دارای یک نام نیز هست.که از طریق آن می توان متغیر را از دیگر متغیر ها تشخیص داد و به مقدار آن دسترسی پیدا کرد.همچنین دارای یک مقدار می باشد که می تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد.مقدار متغیر می تواند تهی نیز باشد.متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده ای که در آن ذخیره می شود یکی است. متغیر دارای عمر نیز هست که از روی آن می توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد.و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است.ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می کنیم. هنگامی که یک برنامه ایجاد می کنیم احتیاج به یک مکان برای ذخیره داده ،مقادیر یا داده هایی که توسط کاربر وارد می شوند داریم.ایم مکان همان متغیر است. برای این از کلمه متغیر استفاده می شود چون ما می توانیم بسته به نوع شرایط هر جا که لازم باشد مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می گیرند که برنامه در حال اجراست و

وقتی شما برنامه را می بندید محتویات متغیر ها نیز پاک می شود. قبلا ذکر شد که به وسیله نام متغیر می توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد:

1. نام متغیر باید با یک از حروف الفبا (a-z or A-Z) شروع شود.
2. نمی تواند شامل کاراکترهای غیرمجاز مانند .$, ^, ?, #باشد.
3. نمی توان از کلمات رزرو شده در جاوا برای نام متغیر استفاده کرد.
4. نام متغیر نباید دارای فضای خالی (spaces) باشد.
5. اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند. در جاوا دو حرف مانند a و Aدو کاراکتر مختلف به حساب می آیند.

دو متغیر با نامهای myNumber و MyNumber دو متغیر مختلف محسوب می شوند چون یکی از آنها با حرف کوچک m و دیگری با حرف بزرگ M شروع می شود. شما نمی توانید دو متغیر را که دقیق شبیه هم هستند را در یک scope (محدوده) تعریف کنید.Scope به معنای یک بلوک کد است که متغیر در آن قابل دسترسی و استفاده است. در مورد Scope در فصلهای آینده بیشتر توضیح خواهیم داد. متغیر دارای نوع هست که نوع داده ای را که در خود ذخیره می کند را نشان می دهد. معمولترین انواع داده int, short, long, byte, double, float, char, Boolean می باشند. برای مثال شما برای قرار دادن یک عدد صحیح در متغیر باید از نوع int استفاده کنید.

انواع ساده

انواع ساده انواعی از داده ها هستند که شامل اعداد، کاراکترها و مقادیر بولی می باشند.به انواع ساده انواع اصلی نیز گفته می شود چون از آنها برای ساخت انواع پیچیده تری مانند کلاس ها و ساختارها استفاده می شود. انواع ساده دارای مجموعه مشخصی از مقادیر هستند و محدوده خاصی از اعداد را در خود ذخیره می کنند.در جدول زیر انواع ساده و محدود آنها آمده است :

نوع	دامنه		
byte	-128	-	127
short	-32768	-	32767
int	-2147483648	-	2147483647
long	-9223372036854775808	-	9223372036854775807

جدول زیر انواعی که مقادیر با ممیز اعشار را می توانند در خود ذخیره کنند را نشان می دهد :

نوع	دامنه تقریبی	دقت
float	±1.5E-45 to ±3.4E38	7 رقم
double	±5.0E-324 to ±1.7E308	15 – 16 رقم

برای به خاطر سپردن آنها باید از نماد علمی استفاده شود. نوع دیگری از انواع ساده برای ذخیره داده های غیر عددی به کار می روند و در جدول زیر نمایش داده شده اند :

نوع	مقادیر مجاز
char	کاراکترهای یونیکد
boolean	مقدار true یا false

نوع char برای ذخیره کاراکترهای یونیکد استفاده می شود.کاراکترها باید داخل یک کوتیشن ساده قرار بگیرند مانند ('a').

نوع bool فقط می تواند مقادیر درست (true) یا نادرست (false) را در خود ذخیره کند و بیشتر در برنامه هایی که دارای ساختار تصمیم گیری هستند مورد استفاده قرار می گیرد.

استفاده از رشته ها

از رشته برای ذخیره گروهی از کاراکترها مانند یک پیغام استفاده می شود.مقادیر ذخیره شده در یک رشته باید داخل دابل کوتیشن قرار گیرند تا توسط کامپایلر به عنوان یک رشته در نظر گرفته شوند، مانند .("massage") جاوا دارای نوعی به نام رشته نیست، بلکه رشته ها اشیایی هستند که از روی کلاس) Stringحرف S به صورت بزرگ نوشته می شود) ساخته می شوند. با مفاهیم شیء و کلاس در درس های آینده آشنا می شوید. فقط در همین حد کافی است که بدانید که از رشته ها برای نمایش متن استفاده می شود مثلا برای نمایش متن Hello World می توان به صورت زیر عمل کرد :

```
String str ="Hello World";
```

دلیل اینکه در این قسمت درباره رشته ها مختصری توضیح دادیم این است که ممکن است در آموزش های بعدی با آنها سر و کار داشته باشیم. در آینده به طور مفصل در مورد رشته ها توضیح می دهیم.

استفاده از متغیرها

در مثال زیر نحوه تعریف و مقدار دهی متغیرها نمایش داده شده است :

```
package myfirstprogram;

import java.text.MessageFormat;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
        //Declare variables
        int num1;
        int num2;
        double num3;
        double num4;
        boolean boolVal;
        char myChar;

        //Assign values to variables
        num1 = 1;
        num2 = 2;
        num3 = 3.54;
        num4 = 4.12;
        boolVal = true;
        myChar = 'R';

        //Show the values of the variables
        System.out.println(MessageFormat.format("num1 = {0}", num1));
        System.out.println(MessageFormat.format("num3 = {0}", num3));
        System.out.println(MessageFormat.format("num4 = {0}", num4));
        System.out.println(MessageFormat.format("boolVal = {0}", boolVal));
        System.out.println(MessageFormat.format("num2 = {0}", num2));
        System.out.println(MessageFormat.format("myChar = {0}", myChar));
    }
}
```

```
num1 = 1
num2 = 2
num3 = 3.54
num4 = 4.12
boolVal = true
myChar = R
```

تعریف متغیر

در خطوط 11-16 متغیرهایی با نوع و نام متفاوت تعریف شده اند. ابتدا باید نوع داده هایی را که این متغیر ها قرار است در خود ذخیره کنند را مشخص کنیم و سپس یک نام برای آنها در نظر بگیریم و در آخر سیمیکولن بگذاریم.همیشه به یاد داشته باشید که قبل از مقدار دهی و استفاده از متغیر باید آن را تعریف کرد.

```
int num1;
int num2;
double num3;
double num4;
bool boolVal;
char myChar;
```

نحوه تعریف متغیر به صورت زیر است :

```
data_type identifier;
```

date_type همان نوع داده است مانند double،int و ..... .

Identifier نیز نام متغیر است که به ما امکان استفاده و دسترسی به مقدار متغیر را می دهد.

برای تعریف چند متغیر از یک نوع می توان به صورت زیر عمل کرد :

```
data_type identifier1, identifier2, ... indentifierN;
```

مثال

```
int num1, num2, num3, num4, num5;
```

در مثال بالا 5 متغیر از نوع صحیح تعریف شده است.توجه داشته باشید که بین متغیر ها باید علامت کاما (,) باشد.

نامگذاری متغیرها

- **نام متغیر باید با یک حرف یا زیرخط و به دنبال آن حرف یا عدد شروع شود.**
- **نمی توان از کاراکترهای خاص مانند & ,% ,#یا عدد برای شروع نام متغیر استفاده کرد مانند .2numbers**
- **نام متغیر نباید دارای فاصله باشد.برای نام های چند حرفی میتوان به جای فاصله از علامت زیرخط یا _استفاده کرد.**

نامهای مجاز :

```
num1  myNumber  studentCount  total       first_name    _minimum
num2  myChar    average      amountDue   last_name     _maximum
name  counter   sum          isLeapYear  color_of_car  _age
```

نامهای غیر مجاز :

```
123        #numbers#  #ofstudents  1abc2
123abc     $money     first name   ty.np
my number  this&that  last name    1:00
```

اگر به نامهای مجاز در مثال بالا توجه کنید متوجه قراردادهای به کار رفته در نامگذاری آنها خواهید شد.یکی از روشهای نامگذاری،نامگذاری کوهان شتری است.در این روش که برای متغیرهای دو کلمه ای به کار می رود،اولین حرف اولین کلمه با حرف کوچک و اولین حرف دومین کلمه با حرف بزرگ نمایش داده می شود مانند : myNumber توجه کنید که اولین حرف کلمه Number با حرف بزرگ شروع شده است. مثال دیگر کلمه numberOfStudents است.اگز توجه کنید بعد از اولین کلمه حرف اول سایر کلمات با حروف بزرگ نمایش داده شده است.

محدوده متغیر

متغیرها در داخل متد main() تعریف می شوند.این متغیرها فقط در داخل متد main() قابل دسترسی هستند. محدوده یک متغیر مشخص می کند که متغیر در کجای کد قابل دسترسی است. هنگامیکه برنامه به پایان متد main() می رسد متغیرها از محدوده خارج و بدون استفاده می شوند تا زمانی که برنامه در حال اجراست.محدوده متغیرها انواعی دارد که در درسهای بعدی با آنها آشنا می شوید. تشخیص محدوده متغیر بسیار مهم است چون به وسیله آن می فهمید که در کجای کد می توان از متغیر استفاده کرد. باید یاد آور شد که دو متغیر در یک محدوده نمی توانند دارای نام یکسان باشند. مثلا کد زیر در برنامه ایجاد خطا می کند :

```
int num1;
int num1;
```

از آنجاییکه جاوا به بزرگی و کوچک بودن حروف حساس است می توان از این خاصیت برای تعریف چند متغیر هم نام ولی با حروف متفاوت(از لحاظ بزرگی و کوچکی) برای تعریف چند متغیر از یک نوع استفاده کرد مانند :

```
int num1;
int Num1;
int NUM1;
```

مقداردهی متغیرها

می توان فورا بعد از تعریف متغیرها مقادیری را به آنها اختصاص داد.این عمل را مقداردهی می نامند. در زیر نحوه مقدار دهی متغیر ها نشان داده شده است :

```
data_type identifier = value;
```

به عنوان مثال :

```
int myNumber = 7;
```

همچنین می توان چندین متغیر را فقط با گذاشتن کاما بین آنها به سادگی مقدار دهی کرد :

```
data_type variable1 = value1, varaible2 = value2, ... variableN, valueN;
int num1 = 1, num2 = 2, num3 = 3;
```

تعریف متغیر با مقدار دهی متغیرها متفاوت است. تعریف متغیر یعنی انخاب نوع و نام برای متغیر ولی مقدار دهی یعنی اختصاص یک مقدار به متغیر.

اختصاص مقدار به متغیر

در زیر نحوه اختصاص مقادیر به متغیرها نشان داده شده است:

```
num1 = 1;
num2 = 2;
num3 = 3.54;
num4 = 4.12;
boolVal = true;
myChar = 'R';
```

به این نکته توجه کنید که شما به مغیری که هنوز تعریف نشده نمی توانید مقدار بدهید. شما فقط می توانید از متغیرهایی استفاده کنید که هم تعریف و هم مقدار دهی شده باشند. مثلا متغیرهای بالا همه قابل استفاده هستند. در این مثال num1 و num2 هر دو تعریف شده اند و مقادیری از نوع صحیح به آنها اختصاص داده شده است. اگر نوع داده با نوع متغیر یکی نباشد برنامه پیغام خطا می دهد.

جانگهدار (Placeholders)

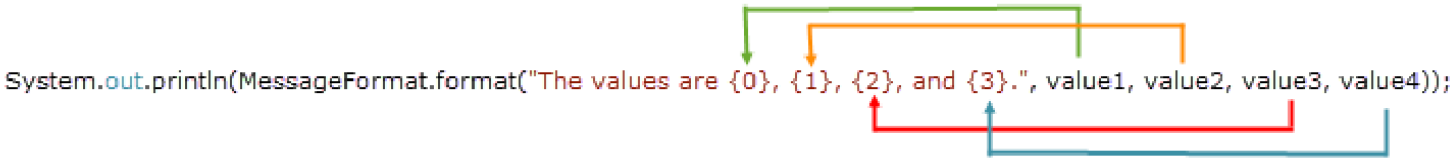
به متد format() از کلاس MessageFormat در خطوط (27-32) توجه کنید.برای استفاده از متد format() و کلاس MessageFormat ابتدا باید Package مربوط به آنها را در برنامه وارد کنید (خط 3) :

```
import java.text.MessageFormat;
```

این متد دو آرگومان قبول می کند. آرگومانها اطلاعاتی هستند که متد با استفاده از آنها کاری انجام می دهد. آرگومانها به وسیله کاما از هم جدا می شوند. آرگومان اول یک رشته قالب بندی شده است و آرگومان دوم مقداری است که توسط رشته قالب بندی شده مورد استفاده قرار می گیرد. اگر به دقت نگاه کنید رشته قالب بندی شده دارای عدد صفری است که در داخل دو آکولاد محصور شده است. البته عدد داخل دو آکولاد می تواند از صفر تا nباشد. به این اعداد جانگهدار می گویند. این اعداد بوسیله مقدار آرگومان بعد جایگزین می شوند. به عنوان مثال جانگهدار {0}به این معناست که اولین آرگومان (مقدار) بعد از رشته قالب بندی شده در آن قرار می گیرد. متد format()عملا می تواند هر تعداد آرگومان قبول کند اولین آرگومان همان رشته قالب بندی شده است که

جانگهدار در آن قرار دارد و دومئین آرگومان مقداری است که جایگزین جانگهدار می شود. در مثال زیر از 4 جانگهدار استفاده شده است:

```
System.out.println(MessageFormat.format("The values are {0}, {1}, {2}, and {3}.", value1,
value2, value3, value4);
```



جانگهدارها از صفر شروع می شوند. تعداد جانگهدارها باید با تعداد آرگومانهای بعد از رشته قالب بندی شده برابر باشد. برای مثال اگر شما چهار جانگهدار مثل بالا داشته باشید باید چهار مقدار هم برای آنها بعد از رشته قالب بندی شده در نظر بگیرید. اولین جا نگهدار با دومین آرگومان و دومین جا نگهدار با سومین آرگومان جایگزین می شود. در ابتدا فهمیدن این مفهوم برای کسانی که تازه برنامه نویسی را شروع کرده اند سخت است اما در درسهای آینده مثالهای زیادی در این مورد مشاهده خواهید کرد.

ثابت

ثابت ها انواعی از متغیرها هستند که مقدار آنها در طول برنامه تغییر نمی کند. ثابت ها حتما باید مقدار دهی اولیه شوند و اگر مقدار دهی آنها فراموش شود در برنامه خطا به وجود می آید. بعد از این که به ثابت ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی توان آن را تغییر داد. برای تعریف ثابت ها باید از کلمه کلیدی **final** استفاده کرد.معمولا نام ثابت ها را طبق قرارداد با حروف بزرگ می نویسند تا تشخیص آنها در برنامه راحت باشد. نحوه تعریف ثابت در زیر آمده است:

```
final data_type identifier = initial_value;
```

مثال :

```
package myfirstprogram;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
        final int NUMBER = 1;

        NUMBER = 10; //ERROR, Cant modify a constant
    }
}
```

در این مثال می بینید که مقدار دادن به یک ثابت ،که قبلا مقدار دهی شده برنامه را با خطا مواجه می کند.نکته ی دیگری که نباید فراموش شود این است که نباید مقدار ثابت را با مقدار دیگر متغیرهای تعریف شده در برنامه برابر قرار داد. مثال :

```
int someVariable;
final int MY_CONST = someVariable;
```

ممکن است این سوال برایتان پیش آمده باشد که دلیل استفاده از ثابت ها چیست؟ اگر مطمئن هستید که مقادیری در برنامه وجود دارند که هرگز در طول برنامه تغییر نمی کنند بهتر است که آنها را به صورت ثابت تعریف کنید.این کار هر چند کوچک کیفیت برنامه شما را بالا می برد.

تبدیل ضمنی

تبدیل ضمنی یا تبدیل بزرگ کننده یا widening conversion یک نوع تبدیل است که به طور خودکار انجام می شود. در این نوع تبدیل در صورتی یک متغیر از یک نوع داده می تواند به یک نوع دیگر تبدیل شود که مقدار آن از مقدار داده ای که می خواهد به آن تبدیل شود کمتر باشد. به عنوان مثال نوع داده ای byte می تواند مقادیر 0 تا 255 را در خود ذخیره کند و نوع داده ای int مقادیر -2147483648 تا 2147483647 را شامل می شود. پس می توانید یک متغیر از نوع byte را به یک نوع int تبدیل کنید:

```
byte number1 = 5;

int number2 = number1;
```

در مثال بالا مقدار number1 برابر 5 است در نتیجه متغیر number2که یک متغیر از نوع صحیح است می تواند مقدار number1 را در خود ذخیره کند چون نوع صحیح از نوع بایت بزرگتر است. پس متغیر number1 که یک متغیر از نوع بایت است می تواند به طور ضمنی به number2 که یک متغیر از نوع صحیح است تبدیل شود. اما عکس مثال بالا صادق نیست.

```
int  number1 = 5;

byte number2 = number1;
```

در این مورد ما با خطا مواجه می شویم. اگر چه مقدار 5 متغیر number1در محدوده مقادیر byte یعنی اعداد بین 255-0 قرار دارد اما متغیر ی از نوع بایت حافظه کمتری نسبت به متغیری از نوع صحیح اشغال می کند. نوع byte شامل 8 بیت یا 8 رقم دودویی است در حالی که نوع int شامل 32بیت یا رقم باینری است.یک عدد باینری عددی متشکل از 0 و 1 است.برای مثال عدد 5 در کامپیوتر به عدد باینری 101 ترجمه می شود. بنابراین وقتی ما عدد 5 را در یک متغیر از نوع بایت ذخیره می کنیم عددی به صورت زیر نمایش داده می شود:

```
00000101
```

و وقتی آن را در یک متغیر از نوع صحیح ذخیره می کنیم به صورت زیر نمایش داده می شود:

```
00000000000000000000000000000101
```

بنابراین قرار دادن یک مقدار int در یک متغیر byte درست مانند این است که ما سعی کنیم که یک توپ فوتبال را در یک سوراخ کوچک گلف جای دهیم. برای قرار دادن یک مقدار int در یک متغیر از نوع byte می توان از تبدیل صریح استفاده کرد که در درسهای آینده توضیح داده می شود. نکته دیگری که نباید فراموش شود این است که شما نمی توانید اعداد با ممیز اعشار را به یک نوع intتبدیل کنید چون این کار باعث از بین رفتن بخش اعشاری این اعداد می شود.

```
double number1 = 5.25;

int number2 = number1; //Error
```

تبدیلاتی که جاوا به صورت ضمنی می تواند انجام دهد در زیر آمده است:

تبدیل صریح

تبدیل صریح یا تبدیل کوچک کننده یا Narrowing Casting نوعی تبدیل است که برنامه را مجبور می کند که یک نوع داده را به نوعی دیگر تبدیل کند اگر این نوع تبدیل از طریق تبدیل ضمنی انجام نشود.در هنگام استفاده از این تبدیل باید دقت کرد چون در این نوع تبدیل ممکن است مقادیر اصلاح یا حذف شوند.ما می توانیم این عملیات را با استفاده از Cast انجام دهیم .فقط نام دیگر تبدیل صریح است و دستور آن به صورت زیر است:

```
datatypeA variableA = value;
datatypeB variableB = (datatypeB)variableA;
```

همانطور که قبلا مشاهده کردید نوع int را نتوانستیم به نوع byte تبدیل کنیم اما اکنون با استفاده از عمل Cast این تبدیل انجام خواهد شد :

```
int number1 = 5;

byte number2 = (byte)number1;
```

حال اگر برنامه را اجرا کنید با خطا مواجه نخواهید شد. همانطور که پیشتر اشاره شد ممکن است در هنگام تبدیلات مقادیر اصلی تغییر کنند.برای مثال وقتی که یک عدد با ممیز اعشار مثلا از نوع double را به یک نوع int تبدیل می کنیم مقدار اعداد بعد از ممیز از بین می روند :

```
double number1 = 5.25;

int number2 = (int)number1;

System.out.println(number2);
```
```
5
```

خروجی کد بالا عدد 5 است چون نوع داده ای int نمی تواند مقدار اعشار بگیرد. حالت دیگر را تصور کنید.اگر شما بخواهید یک متغیر را که دارای مقداری بیشتر از محدوده متغیر مقصد هست تبدیل کنید چه اتفاقی می افتد؟ مانند تبدیل زیر که می خواهیم متغیر number1 را که دارای مقدار 300 است را به نوع بایت تبدیل کنیم که محدود اعداد بین 255-0 را پوشش می دهد.

```
int number1 = 300;

byte number2 = (byte)number1;

System.out.println(MessageFormat.format("Value of number2 is {0}.", number2));
```
```
Value of number2 is 44.
```

خروجی کد بالا عدد 44 است. Byte فقط می تواند شامل اعداد 0 تا 255 باشد و نمی تواند مقدار 300 را در خود ذخیره کند.حال می خواهیم ببینیم که چرا به جای عدد 300 ما عدد 44 را در خروجی می گیریم. این کار به تعداد بیتها بستگی دارد.یک byte دارای 8 بیت است درحالی که int دارای 32 بیت است. حال اگر به مقدار باینری 2 عدد توجه کنید متوجه می شوید که چرا خروجی عدد 44 است.

```
300 = 00000000000000000000000100101100
255 =                         11111111
 44 =                         00101100
```

خروجی بالا نشان می دهد که بیشترین مقدار byte که عدد 255 است می تواند فقط شامل 8 بیت باشد (11111111) بنابراین فقط 8 بیت اول مقدار int به متغیر byte انتقال می یابد که شامل (00101100) یا عدد 44 در مبنای 10 است.

عبارات و عملگرها

ابتدا با دو کلمه آشنا شوید:

- عملگر : نمادهایی هستند که اعمال خاص انجام می دهند.
- عملوند : مقادیری که عملگرها بر روی آنها عملی انجام می دهند.

مثلا X+Y: یک عبارت است که در آن X و Y عملوند و علامت + عملگر به حساب می آیند.

زبانهای برنامه نویسی جدید دارای عملگرهایی هستند که از اجزاء معمول زبان به حساب می آیند. جاوا دارای عملگرهای مختلفی از جمله عملگرهای ریاضی ،تخصیصی ، مقایسه ای ، منطقی و بیتی می باشد. از عملگرهای ساده ریاضی می توان به عملگر جمع و تفریق اشاره کرد. سه نوع عملگر در جاوا وجود دارد:

- یگانی – (Unary) به یک عملوند نیاز دارد
- دودویی – (Binary)به دو عملوند نیاز دارد
- سه تایی – (Ternary) به سه عملوند نیاز دارد

انواع مختلف عملگر که در ای بخش مورد بحث قرار می گیرند عبارتند از:

- عملگرهای ریاضی
- عملگرهای تخصیصی
- عملگرهای مقایسه ای
- عملگرهای منطقی
- عملگرهای بیتی

عملگرهای ریاضی

جاوا از عملگرهای ریاضی برای انجام محاسبات استفاده می کند. جدول زیر عملگرهای ریاضی جاوا را نشان می دهد :

عملگر	دسته	مثال	نتیجه
+	Binary	var1 = var2 + var3;	Var1 برابر است با حاصل جمع var2 و var3
-	Binary	var1 = var2 – var3;	Var1 برابر است با حاصل تفریق var2 و var3
*	Binary	var1 = var2 * var3;	Var1 برابر است با حاصلضرب var2 در var3
/	Binary	var1 = var2 / var3;	Var1 برابر است با حاصل تقسیم var2 بر var3
٪	Binary	var1 = var2 % var3;	Var1 برابر است با باقیمانده تقسیم var2 و var3
+	Unary	var1 = +var2;	Var1 برابر است با مقدار var2
-	Unary	var1 = -var2	Var1 برابر است با مقدار var2 ضربدر 1-

دیگر عملگرهای جاوا عملگرهای کاهش و افزایش هستند.این عملگرها مقدار 1 را از متغیر ها کم یا به آنها اضافه می کنند. از این متغیر ها اغلب در حلقه ها استفاده می شود :

عملگر	دسته	مثال	نتیجه
++	Unary	var1 = ++var2;	مقدار var1 برابر است با var2 بعلاوه 1
–	Unary	var1 = –var2;	مقدار var1 برابر است با var2 منهای 1
++	Unary	var1 = var2++;	مقدار var1 برابر است با var2 به متغیر var2 یک واحد اضافه می شود
–	Unary	var1 = var2–;	مقدار var1 برابر است با var2 از متغیر var2 یک واحد کم می شود

به این نکته توجه داشته باشید که محل قرار گیری عملگر در نتیجه محاسبات تاثیر دارد.اگر عملگر قبل از متغیر var2 بیاید افزایش یا کاهش var1 اتفاق می افتد.چنانچه عملگرها بعد از متغیر var2 قرار بگیرند ابتدا var1 برابر var2 می شود و سپس متغیر var2 افزایش یا کاهش می یابد.

به مثال های زیر توجه کنید:

```
package myfirstprogram;

import java.text.MessageFormat;
```

```
public class MyFirstProgram
{
    public static void main(String[] args)
    {
        int x = 0;
        int y = 1;

        x = ++y;

        System.out.println(MessageFormat.format("x= {0}",x));
        System.out.println(MessageFormat.format("y= {0}", y));
    }
}
```

```
x=2
y=2
```

```
package myfirstprogram;

import java.text.MessageFormat;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
        int x = 0;
        int y = 1;

        x = --y;

        System.out.println(MessageFormat.format("x= {0}",x));
        System.out.println(MessageFormat.format("y= {0}", y));
    }
}
```

```
x=0
y=0
```

همانطور که در دو مثال بالا مشاهده می کنید،درج عملگرهای — و ++ قبل از عملوند y باعث می شود که ابتدا یک واحد از y کم و یا یک واحد به y اضافه شود و سپس نتیجه در عملوند x قرار بگیرد. حال به دو مثال زیر توجه کنید:

```
package myfirstprogram;

import java.text.MessageFormat;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
        int x = 0;
        int y = 1;
```

```
        x = y--;

        System.out.println(MessageFormat.format("x= {0}",x));
        System.out.println(MessageFormat.format("y= {0}", y));
    }
}
```

```
x=1
y=0
```

```
package myfirstprogram;

import java.text.MessageFormat;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
        int x = 0;
        int y = 1;

        x = y++;

        System.out.println(MessageFormat.format("x= {0}",x));
        System.out.println(MessageFormat.format("y= {0}", y));
    }
}
```

```
x=1
y=2
```

همانطور که در دو مثال الا مشاهده می کنید،درج عملگرهای —— و ++ بعد از عملوند y باعث می شود که ابتدا مقدار y در داخل متغیر x قرار بگیرد و سپس یک واحد از y کم و یا یک واحد به ان اضافه شود. حال می توانیم با ایجاد یک برنامه نحوه عملکرد عملگرهای ریاضی در جاوا را یاد بگیریم:

```
package myfirstprogram;

import java.text.MessageFormat;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
       //Variable declarations
       int num1, num2;

       //Assign test values
       num1 = 5;
       num2 = 3;

       System.out.println(MessageFormat.format("The sum of {0} and {1} is {2}.", num1,
num2, (num1 + num2)));
       System.out.println(MessageFormat.format("The difference of {0} and {1} is {2}.",
```

```
num1, num2, (num1 - num2)));
       System.out.println(MessageFormat.format("The product of {0} and {1} is {2}.",
num1, num2, (num1 * num2)));
       System.out.println(MessageFormat.format("The quotient of {0} and {1} is {2}.",
num1, num2, ((double)num1 / num2)));
       System.out.println(MessageFormat.format("The remainder of {0} and {1} is {2}.",
num1, num2, (num1 % num2)));
    }
}
```

```
The sum of 5 and 3 is 8.
The difference of 5 and 3 is 2.
The product of 5 and 3 is 15.
The quotient of 5 and 3 is 1.67.
The remainder of 5 divided by 3 is 2
```

برنامه بالا نتیجه هر عبارت را نشان می دهد. در این برنامه از تد println()برای نشان دادن نتایج در سطرهای متفاوت استفاده شده است. در این مثال با یک نکته عجیب مواجه می شویم و آن حاصل تقسیم دو عدد صحیح است.وقتی که دو عدد صحیح را بر هم تقسیم کنیم حاصل باید یک عدد صحیح و فاقد بخش کسری باشد.اما همانطور که مشاهده می کنید اگر فقط یکی از اعداد را به نوع اعشاری doubleتبدیل کنیم (در مثال می بینید) حاصل به صورت اعشار نشان داده می شود.

عملگرهای تخصیصی

نوع دیگر از عملگرهای جاوا عملگرهای جایگزینی نام دارند.این عملگرها مقدار متغیر سمت راست خود را در متغیر سمت چپ قرار می دهند. جدول زیر انواع عملگرهای تخصیصی در جاوا را نشان می دهد:

عملگر	مثال	نتیجه
=	var1 = var2;	مقدار var1 برابر است با مقدار var2
+=	var1 += var2;	مقدار var1 برابر است با حاصل جمع var1 وvar2
-=	var1 -= var2;	مقدار var1 برابر است با حاصل تفریق var1 وvar2
*=	var1 *= var2;	مقدار var1 برابر است با حاصل ضرب var1 درvar2
/=	var1 /= var2;	مقدار var1 برابر است با حاصل تقسیم var1 برvar2
%=	var1 %= var2;	مقدار var1 برابر است با باقیمانده تقسیم var1 برvar2

از عملگر += برای اتصال دو رشته نیز می توان استفاده کرد.استفاده از این نوع عملگرها در واقع یک نوع خلاصه نویسی در کد است. مثلا شکل اصلی کد var1 += var2 به صورت var1 = var1 + var2 می باشد. این حالت کدنویسی زمانی کارایی خود را

نشان می دهد که نام متغیرها طولانی باشد. برنامه زیر چگونگی استفاده از عملگرهای تخصیصی و تاثیر آنها را بر متغیر ها نشان می دهد.

```
package myfirstprogram;

import java.text.MessageFormat;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
       int number;

       System.out.println("Assigning 10 to number...");
       number = 10;
       System.out.println(MessageFormat.format("Number = {0}", number));

       System.out.println("Adding 10 to number...");
       number += 10;
       System.out.println(MessageFormat.format("Number = {0}", number));

       System.out.println("Subtracting 10 to number...");
       number -= 10;
       System.out.println(MessageFormat.format("Number = {0}", number));
    }
}
```

```
Assigning 10 to number...
Number = 10
Adding 10 to number...
Number = 20
Subtracting 10 from number...
Number = 10
```

در برنامه از 3 عملگر تخصیصی استفاده شده است. ابتدا یک متغیر و مقدار 10با استفاده از عملگر = به آن اختصاص داده شده است.سپس به آن با استفاده از عملگر += مقدار 10 اضافه شده است.و در آخر به وسیله عملگر -= عدد 10 از آن کم شده است.

عملگرهای مقایسه ای

از عملگرهای مقایسه ای برای مقایسه مقادیر استفاده می شود.نتیجه این مقادیر یک مقدار بولی(منطقی) است. این عملگرها اگر نتیجه مقایسه دو مقدار درست باشد مقدار true و اگر نتیجه مقایسه اشتباه باشد مقدار false را نشان می دهند. این عملگرها به طور معمول در دستورات شرطی به کار می روند به این ترتیب که باعث ادامه یا توقف دستور شرطی می شوند. جدول زیر عملگرهای مقایسه ای در جاوا را نشان می دهد:

عملگر	دسته	مثال	نتیجه
==	Binary	var1 = var2 == var3	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر باشد در غیر اینصورت false است
!=	Binary	var1 = var2 != var3	var1 در صورتی true است که مقدار var2 با مقدار var3 برابر نباشد در غیر اینصورت false است
<	Binary	var1 = var2 < var3	var1 در صورتی true است که مقدار var2 کوچکتر از var3 مقدار باشد در غیر اینصورت false است
>	Binary	var1 = var2 > var3	var1 در صورتی true است که مقدار var2بزرگتر ازمقدارvar3 باشد در غیر اینصورت false است
<=	Binary	var1 = var2 <= var3	var1 در صورتی true است که مقدار var2 کوچکتر یا مساوی مقدار var3 باشد در غیر اینصورت false است
>=	Binary	var1 = var2 >= var3	var1 در صورتی true است که مقدار var2 بزرگتر یا مساوی var3 مقدار باشد در غیر اینصورت false است

برنامه زیر نحوه عملکرد ای عملگرها را نشان می دهد :

```
package myfirstprogram;

import java.text.MessageFormat;

public class MyFirstProgram
{
    public static void main(String[] args)
    {
       int num1 = 10;
       int num2 = 5;

       System.out.println(MessageFormat.format("{0} == {1} : {2}", num1, num2, num1 ==
num2));
       System.out.println(MessageFormat.format("{0} != {1} : {2}", num1, num2, num1 !=
num2));
       System.out.println(MessageFormat.format("{0}  < {1} : {2}", num1, num2, num1 <
num2));
       System.out.println(MessageFormat.format("{0}  > {1} : {2}", num1, num2, num1 >
num2));
       System.out.println(MessageFormat.format("{0} <= {1} : {2}", num1, num2, num1 <=
num2));
       System.out.println(MessageFormat.format("{0} >= {1} : {2}", num1, num2, num1 >=
num2));
    }
}
```

```
10 == 5 : False
10 != 5 : True
10 <  5 : False
10 >  5 : True
```

```
10 <= 5 : False
10 >= 5 : True
```

در مثال بالا ابتدا دو متغیر را که می خواهیم با هم مقایسه کنیم را ایجاد کرده و به آنها مقادیری اختصاص می دهیم.سپس با استفاده از یک عملگر مقایسه ای آنها را با هم مقایسه کرده و نتیجه را چاپ می کنیم.

به این نکته توجه کنید که هنگام مقایسه دو متغیر از عملگر == به جای عملگر = باید استفاده شود.

عملگر = عملگر تخصیصی است و در عبارتی مانند x = y مقدار y را در به x اختصاص می دهد.

عملگر == عملگر مقایسه ای است که دو مقدار را با هم مقایسه می کند مانند x==y و اینطور خوانده می شود x برابر است با. y

عملگرهای منطقی

عملگرهای منطقی بر روی عبارات منطقی عمل می کنند و نتیجه آنها نیز یک مقدار بولی است. از این عملگرها اغلب برای شرطهای پیچیده استفاده می شود. همانطور که قبلا یاد گرفتید مقادیر بولی می توانند false یا true باشند. فرض کنید که var2 و var3 دو مقدار بولی هستند.

عملگر	نام	دسته	مثال
&&	منطقی AND	Binary	var1 = var2 && var3;
||	منطقی OR	Binary	var1 = var2 || var3;
!	منطقی NOT	Unary	var1 = !var1;

عملگر منطقی AND(&&)

اگر مقادیر دو طرف عملگر AND ، trueباشند عملگر AND مقدار true را بر می گرداند. در غیر اینصورت اگر یکی از مقادیر یا هر دوی آنها false باشند مقدار false را بر می گرداند. در زیر جدول درستی عملگر AND نشان داده شده است:

X	Y	X && Y

X	Y	X && Y
true	true	true
true	false	false
false	true	false
false	false	false

برای درک بهتر تاثیر عملگر AND یاد آوری می کنم که این عملگر فقط در صورتی مقدار true را نشان می دهد که هر دو عملوند مقدارشان true باشد.در غیر اینصورت نتیجه تمام ترکیبهای بعدی false خواهد شد. استفاده از عملگر ANDمانند استفاده از عملگرهای مقایسه ای است. به عنوان مثال نتیجه عبارت زیر درست (true) است اگر سن (age) بزرگتر از 18 و salary کوچکتر از 1000 باشد.

```
result = (age > 18) && (salary < 1000);
```

عملگر AND زمانی کارامد است که ما با محدود خاصی از اعداد سرو کار داریم. مثلا عبارت 10 <= x <= 100بدین معنی است که xمی تواند مقداری شامل اعداد 10 تا 100 را بگیرد. حال برای انتخاب اعداد خارج از این محدوده می توان از عملگر منطقی AND به صورت زیر استفاده کرد.

```
inRange = (number <= 10) && (number >= 100);
```

عملگر منطقی OR(||)

اگر یکی یا هر دو مقدار دو طرف عملگر OR ،درست (true) باشد،عملگر OR مقدار true را بر می گرداند. جدول درستی عملگر ORدر زیر نشان داده شده است :

X	Y	X || Y
true	true	true
true	false	true
false	true	true
false	false	false

در جدول بالا مشاهده می کنید که عملگر OR در صورتی مقدار false را بر میگرداند که مقادیر دو طرف آن false باشند. کد زیر را در نظر بگیرید.نتیجه این کد در صورتی درست (true) است که رتبه نهایی دانش آموز (finalGrade) بزرگتر از 75 یا یا نمره نهایی امتحان آن 100 باشد.

```
isPassed = (finalGrade >= 75) || (finalExam == 100);
```

عملگر منطقی (!)NOT

برخلاف دو اپراتور OR و AND عملگر منطقی NOT یک عملگر یگانی است و فقط به یک عملوند نیاز دارد. این عملگر یک مقدار یا اصطلاح بولی را نفی می کند .مثلا اگر عبارت یا مقدار true باشد آنرا false و اگر false باشد آنرا true می کند. جدول زیر عملکرد اپراتور NOT را نشان می دهد:

X	!X
true	false
false	true

نتیجه کد زیر در صورتی درست است که) age سن) بزرگتر یا مساوی 18 نباشد.

```
isMinor = !(age >= 18);
```

عملگرهای بیتی

عملگرهای بیتی به شما اجازه می دهند که شکل باینری انواع داده ها را دستکاری کنید.برای درک بهتر این درس توصیه می شود که شما سیستم باینری و نحوه تبدیل اعداد دهدهی به باینری را یاد بگیرید.در سیستم باینری (دودویی) که کامپیوتر از آن استفاده می کند وضعیت هر چیز یا خاموش است یا روشن.برای نشان دادن حالت روشن از عدد 1 و برای نشان دادن حالت خاموش از عدد 0 استفاده می شود.بنابراین اعداد باینری فقط می توانند صفر یا یک باشند.اعداد باینری را اعداد در مبنای 2 و اعداد اعشاری را اعداد در مبنای 10 می گویند. یک بیت نشان دهنده یک رقم باینری است و هر بایت نشان دهنده 8 بیت است. به عنوان مثال برای یک داده از نوع intبه 32 بیت یا 8 بایت فضا برای ذخیره آن نیاز داریم،این بدین معناست که اعداد از 32 رقم 0 و 1 برای ذخیره استفاده می کنند. برای مثال عدد 100 وقتی به عنوان یک متغیر از نوع int ذخیره می شود در کامپیوتر به صورت زیر خوانده می شود:

```
00000000000000000000000001100100
```

عدد 100 در مبنای ده معادل عدد 1100100 در مبنای 2 است.در اینجا 7 رقم سمت راست نشان دهنده عدد 100 در مبنای 2 است و مابقی صفرهای سمت راست برای پر کردن بیتهایی است که عدد از نوع int نیاز دارد.به این نکته توجه کنید که اعداد باینری از سمت راست به چپ خوانده می شوند. عملگرهای بیتی سی شارپ در جدول زیر نشان داده شده اند :

عملگر	نام	دسته	مثال
&	بیتیAND	Binary	x = y & z;
|	بیتیOR	Binary	x = y | z;
^	بیتیXOR	Binary	x = y ^ z;
~	بیتیNOT	Unary	x = ~y;
&=	بیتیAND Assignment	Binary	x &= y;
|=	بیتیOR Assignment	Binary	x |= y;
^=	بیتیXOR Assignment	Binary	x ^= y;

عملگر بیتی AND(&)

عملگر بیتی AND کاری شبیه عملگر منطقی AND انجام می دهد با این تفاوت که این عملگر بر روی بیتها کار می کند.اگر مقادیر دو طرف آن 1 باشد مقدار 1 را بر می گرداند و اگر یکی یا هر دو طرف آن صفر باشد مقدار صفر را بر می گرداند. جدول درستی عمگر بیتی AND در زیر آمده است:

X	Y	X AND Y
1	1	1
1	0	0
0	1	0

X	Y	X AND Y
0	0	0

در زیر نحوه استفاده از عملگر بیتی AND آمده است:

```
int result = 5 & 3;

System.out.println(result);
```
1

همانطور که در مثال بالا مشاهده می کنید نتیجه عملکرد عملگر AND بر روی دو مقدار 5 و 3 عدد یک می شود.اجازه بدهید ببینیم که چطور این نتیجه را به دست می آید:

```
5:  00000000000000000000000000000101
3:  00000000000000000000000000000011
    --------------------------------
1:  00000000000000000000000000000001
```

ابتدا دو عدد 5 و 3 به معادل باینری شان تبدیل می شوند. از آنجاییکه هر عدد صحیح 32 (int)بیت است از صفر برای پر کردن بیتهای خالی استفاده می کنیم. با استفاده از جدول درستی عملگر بیتیAND می توان فهمید که چرا نتیجه عدد یک می شود.

عملگر بیتی OR(|)

اگر مقادیر دو طرف عملگر بیتی OR هر دو صفر باشند نتیجه صفر در غیر اینصورت 1 خواهد شد.جدول درستی این عملگر در زیر آمده است:

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

نتیجه عملگر بیتی OR در صورتی صفر است که عملوند های دو طرف آن صفر باشند.اگر فقط یکی از دو عملوند یک باشد نتیجه یک خواهد شد. به مثال زیر توجه کنید:

```
int result = 7 | 9;

System.out.println(result);
```
```
15
```

وقتی که از عملگر بیتی OR برای دو مقدار در مثال بالا (7 و 9) استفاده می کنیم نتیجه 15 می شود.حال بررسی می کنیم که چرا این نتیجه به دست آمده است؟

```
7: 00000000000000000000000000000111
9: 00000000000000000000000000001001
---------------------------------
15: 00000000000000000000000000001111
```

با استفاده از جدول درستی عملگر بیتی OR می توان نتیجه استفاده از این عملگر را تشخیص داد.عدد 1111 باینری معادل عدد 15 صحیح است.

عملگر بیتی XOR(^)

جدول درستی این عملگر در زیر آمده است:

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

در صورتیکه عملوندهای دو طرف این عملگر هر دو صفر یا هر دو یک باشند نتیجه صفر در غیر اینصورت نتیجه یک می شود.در مثال زیر تاثیر عملگر بیتی XORرا بر روی دو مقدار مشاده می کنید:

```
int result = 5 ^ 7;

System.out.println(result);
```
```
2
```

در زیر معادل باینری اعداد بالا (5 و 7) نشان داده شده است.

```
5: 00000000000000000000000000000101
7: 00000000000000000000000000000111
---------------------------------
2: 00000000000000000000000000000010
```

با نگاه کردن به جدول درستی عملگر بیتی XOR ، می توان فهمید که چرا نتیجه عدد 2 می شود.

عملگر بیتی NOT(~)

این عملگر یک عملگر یگانی است و فقط به یک عملوند نیاز دارد.در زیر جدول درستی این عملگر آمده است:

X	NOT X
1	0
0	1

عملگر بیتی NOT مقادیر بیتها را معکوس می کند.در زیر چگونگی استفاده از این عملگر آمده است:

```
int result = ~7;

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

به نمایش باینری مثال بالا که در زیر نشان داده شده است توجه نمایید.

```
7: 00000000000000000000000000000111
---------------------------------
-8: 11111111111111111111111111111000
```

عملگر بیتی تغییر مکان(shift)

این نوع عملگرها به شما اجازه می دهند که بیتها را به سمت چپ یا راست جا به جا کنید.دو نوع عملگر بیتی تغییر مکان وجود دارد که هر کدام دو عملوند قبول می کنند.عملوند سمت چپ این عملگرها حالت باینری یک مقدار و عملوند سمت راست تعداد جابه جاییبیت ها را نشان می دهد.

مثال	دسته	نام	عملگر

عملگر	نام	دسته	مثال
>>	تغییر مکان به سمت چپ	Binary	x = y << 2;
<<	تغییر مکان به سمت راست	Binary	x = y >> 2;

عملگر تغییر مکان به سمت چپ

این عملگر بیتهای عملوند سمت چپ را به تعداد n مکان مشخص شده توسط عملوند سمت راست، به سمت چپ منتقل می کند. به عنوان مثال:

```
int result = 10 << 2;

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

```
40
```

در مثال بالا ما بیتهای مقدار 10 را دو مکان به سمت چپ منتقل کرده ایم، حال بیایید تاثیر این انتقال را بررسی کنیم:

```
10: 00000000000000000000000000001010
--------------------------------
40: 00000000000000000000000000101000
```

مشاهده می کنید که همه بیت ها به اندازه دو واحد به سمت چپ منتقل شده اند.در این انتقال دو صفر از صفرهای سمت چپ کم می شود و در عوض دو صفر به سمت راست اضافه می شود.

عملگر تغییر مکان به سمت راست

این عملگر شبیه به عمگر تغییر مکان به سمت چپ است با این تفاوت که بیت ها را به سمت راست جا به جا می کند.
به عنوان مثال:

```
int result = 100 >> 4;

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

```
6
```

با استفاده از عملگرتغییر مکان به سمت راست بیت های مقدار 100 را به اندازه 4 واحد به سمت چپ جا به جا می کنیم. اجازه بدهید تاثیر این جا به جایی را مورد بررسی قرار دهیم:

100: 00000000000000000000000001100100

6: 00000000000000000000000000000110

هر بیت به اندازه 4 واحد به سمت راست منتقل می شود،بنابراین 4 بیت اول سمت راست حذف شده و چهار صفر به سمت چپ اضافه می شود.

تقدم عملگرها

تقدم عملگرها مشخص می کند که در محاسباتی که بیش از دو عملوند دارند ابتدا کدام عملگر اثرش را اعمال کند.عملگر ها در جاوا در محاسبات دارای حق تقدم هستند.به عنوان مثال :

```
number = 1 + 2 * 3 / 1;
```

اگر ما حق تقدم عملگرها را رعایت نکنیم و عبارت بالا را از سمت چپ به راست انجام دهیم نتیجه 9 خواهد شد (3=2+1 سپس 9=3×3 و در آخر 9=9/1).اما کامپایلر با توجه به تقدم عملگرها محاسبات را انجام می دهد.برای مثال عمل ضرب و تقسیم نسبت به جمع و تفریق تقدم دارند.بنابراین در مثال فوق ابتدا عدد 2 ضربدر 3 و سپس نتیجه آنها تقسیم بر 1 می شود که نتیجه 6 به دست می آید.در آخر عدد 6 با 1 جمع می شود و عدد 7 حاصل می شود.

در جدول زیر تقدم برخی از عملگرهای جاوا آمده است :

عملگر	تقدم
++, –, (used as prefixes); +, – (unary)	بالاترین
*, /, %	
+, -	
<<, >>	
<, >, <=, >=	
==, !=	
&	
^	
|	
&&	
||	
=, *=, /=, %=, +=, -=	
++, — (used as suffixes)	پایین ترین

ابتدا عملگرهای با بالاترین و سپس عملگرهای با پایین ترین حق تقدم در محاسبات تاثیر می گذارند.به این نکته توجه کنید که تقدم عملگرها ++ و --به مکان قرارگیری آنها بستگی دارد (در سمت چپ یا راست عملوند باشند).

به عنوان مثال :

```
int number = 3;

number1 = 3 + ++number; //results to 7
number2 = 3 + number++; //results to 6
```

در عبارت اول ابتدا به مقدار number یک واحد اضافه شده و 4 می شود و سپس مقدار جدید با عدد 3 جمع می شود و در نهایت عدد 7 به دست می آید.در عبارت دوم مقدار عددی 3 به مقدار number اضافه می شود و عدد 6 به دست می آید.سپس این مقدار در متغیر number2 قرار می گیرد. و در نهایت مقدار number به 4 افزایش می یابد.

برای ایجاد خوانایی در تقدم عملگرها و انجام محاسباتی که در آنها از عملگرهای زیادی استفاده می شود از پرانتز استفاده می کنیم :

```
number = ( 1 + 2 ) * ( 3 / 4 ) % ( 5 - ( 6 * 7 ));
```

در مثال بالا ابتدا هر کدام از عباراتی که داخل پرانتز هستند مورد محاسبه قرار می گیرند.به نکته ای در مورد عبارتی که در داخل پرانتز سوم قرار دارد توجه کنید.در این عبارت ابتدا مقدار داخلی ترین پرانتز مورد محاسبه قرار می گیرد یعنی مقدار 6 ضربدر 7 شده و سپس از 5 کم می شود.

اگر دو یا چند عملگر با حق تقدم یکسان موجود باشد ابتدا باید هر کدام از عملگرها را که در ابتدای عبارت می آیند مورد ارزیابی قرار دهید.

به عنوان مثال :

```
number = 3 * 2 + 8 / 4;
```

هر دو عملگر * و / دارای حق تقدم یکسانی هستند. بنابر این شما باید از چپ به راست آنها را در محاسبات تاثیر دهید.یعنی ابتدا 3 را ضربدر 2 می کنید و سپس عدد 8 را بر 4 تقسیم می کنید. در نهایت نتیجه دو عبارت را جمع کرده و در متغیر number قرار می دهید.