INTRODUCTION TO JAVA

Lecture Notes

This part is a step-by-step introduction to Java language. The tutorial begins with an overview of the Java platform and language, followed by instructions for setting up a development environment consisting of a Java Development Kit (JDK) and the Eclipse IDE. After you're introduced to your development environment's components, you begin learning basic Java syntax hands-on.

Softech Solutions Inc.

www.softechsolutions.tech

Email: info@softechsolutions.tech

Introduction to Java

Table of Contents	***Page***
Java Platform Overview	
The Java Language	***02***
The Java Compiler	***02***
The JVM	***02***
The Garbage Collector	***02***
The Java Development Kit	***03***
The Java Runtime Environment	***03***
Setting Up Java Development Environment	
Development Environment	***03***
System Requirements	***03***
Install the JDK	***03***
Install Eclipse	***04***
Setup Eclipse	***04***
Getting Started with Eclipse	
The Eclipse Development Environment	***05***
The Java Perspective	***06***
Create a Project	***07***
Highlights	***08***

Java Platform Overview

Java technology is used to develop applications for a wide range of environments, from consumer devices to heterogeneous enterprise systems. In this section, get a high-level view of the Java platform and its components.

The Java Language

Like any programming language, the Java language has its own structure, syntax rules, and programming paradigm. The Java language's programming paradigm is based on the concept of OOP, which the language's features support.

The Java language is a C-language derivative, so its syntax rules look much like C's. For example, code blocks are modularized into methods and delimited by braces ({ and }), and variables are declared before they are used.

Structurally, the Java language starts with packages. A package is the Java language's namespace mechanism. Within packages are classes, and within classes are methods, variables, constants, and more. You learn about the parts of the Java language in this tutorial.

The Java compiler

When you program for the Java platform, you write source code in .java files and then compile them. The compiler checks your code against the language's syntax rules, then writes out ***bytecode*** in .class files. Bytecode is a set of instructions targeted to run on a Java virtual machine (JVM). In adding this level of abstraction, the Java compiler differs from other language compilers, which write out instructions suitable for the CPU chipset the program will run on.

The JVM

At runtime, the JVM reads and interprets .class files and executes the program's instructions on the native hardware platform for which the JVM was written. The JVM interprets the bytecode just as a CPU would interpret assembly-language instructions. The difference is that the JVM is a piece of software written specifically for a particular platform. The JVM is the heart of the Java language's "write-once, run-anywhere" principle. Your code can run on any chipset for which a suitable JVM implementation is available. JVMs are available for major platforms like Linux and Windows, and subsets of the Java language have been implemented in JVMs for mobile phones and hobbyist chips.

The garbage collector

Rather than forcing you to keep up with memory allocation (or use a third-party library to do so), the Java platform provides memory management out of the box. When your Java application creates an object instance at runtime, the JVM automatically allocates memory space for that object from the ***heap***— a pool of memory set aside for your program to use. The Java ***garbage collector*** runs in the background, keeping track of which objects the application no longer needs and reclaiming memory from them. This approach to memory handling is called ***implicit memory management*** because it doesn't require you to write any memory-handling code. Garbage collection is one of the essential features of Java platform performance.

The Java Development Kit

When you download a Java Development Kit (JDK), you get — in addition to the compiler and other tools — complete class libraries of prebuilt utilities that help you accomplish most common application-development tasks. The best way to get an idea of the scope of the JDK packages and libraries is to check out the JDK API documentation.

The Java Runtime Environment

The Java Runtime Environment (JRE; also known as the Java runtime) includes the JVM, code libraries, and components that are necessary for running programs that are written in the Java language. The JRE is available for multiple platforms. You can freely redistribute the JRE with your applications, according to the terms of the JRE license, to give the application's users a platform on which to run your software. The JRE is included in the JDK.

Setting up Java development environment

In this section, you'll download and install the JDK and the current release of the Eclipse IDE, and you'll set up your Eclipse development environment.

If you already have the JDK and Eclipse IDE installed, you might want to skip to the "Getting started with Eclipse" section

Development Environment

The JDK includes a set of command-line tools for compiling and running your Java code, including a complete copy of the JRE. Although you can use these tools to develop your applications, most developers appreciate the additional functionality, task management, and visual interface of an IDE.

Eclipse is a popular open source IDE for Java development. Eclipse handles basic tasks, such as code compilation and debugging, so that you can focus on writing and testing code. In addition, you can use Eclipse to organize source code files into projects, compile and test those projects, and store project files in any number of source repositories. You need an installed JDK to use Eclipse for Java development. If you download one of the Eclipse bundles, it will come with the JDK already.

Install the JDK

Follow these steps to download and install the JDK:

1. Browse to Java SE Downloads and click the Java Platform (JDK) box to display the download page for the latest version of the JDK.
2. Agree to the license terms.
3. Under Java SE Development Kit, choose the download that matches your operating system and chip architecture.

Windows

1. Save the file to your hard drive when prompted.
2. When the download is complete, run the install program. Install the JDK to your hard drive in an easy-to-remember location such as C:\home\Java\jdk1.8.0_60. It's a good idea to encode the update number in the name of the install directory that you choose.

OS X

1. When the download is complete, double-click it to mount it.
2. Run the install program. You do not get to choose where the JDK is installed. You can run /usr/libexec/java_home -1.8 to see the location of JDK 8 on your Mac. The path is display similar to /Library/Java/JavaVirtualMachines/ jdk1.8.0_60.jdk/ Contents/ Home.
   See JDK 8 and JRE 8 Installation for more information.
   You now have a Java environment on your computer. Next, you'll install the Eclipse IDE.

Install Eclipse

To download and install Eclipse, follow these steps:

1. Browse to the Eclipse IDE downloads page.
2. Click Eclipse IDE for Java Developers.
3. Under Download Links on the right side, choose your platform (the site might already have sniffed out your OS type).
4. Click the mirror you want to download from; then, save the file to your hard drive.
5. Extract the contents of the .zip file to a location on your hard drive that you'll be able to remember easily (such as C:\home\eclipse on Windows or ~/home/eclipse on Mac or Linux).

Set up Eclipse

The Eclipse IDE sits atop the JDK as a useful abstraction, but it still needs to access the JDK and its various tools. Before you can use Eclipse to write Java code, you must tell it where the JDK is located.

To set up your Eclipse development environment:

1. Launch Eclipse by double-clicking eclipse.exe (or the equivalent executable for your platform).
2. The Workspace Launcher opens, allowing you to choose a root folder for your Eclipse projects. Use a folder you can easily remember, such as C:\home\workspace on Windows or ~/home/workspace on Mac or Linux.
3. Close the Welcome to Eclipse window.
4. Click Window > Preferences > Java > Installed JREs. Figure 1 show this selection highlighted in the Eclipse setup window for the JRE.

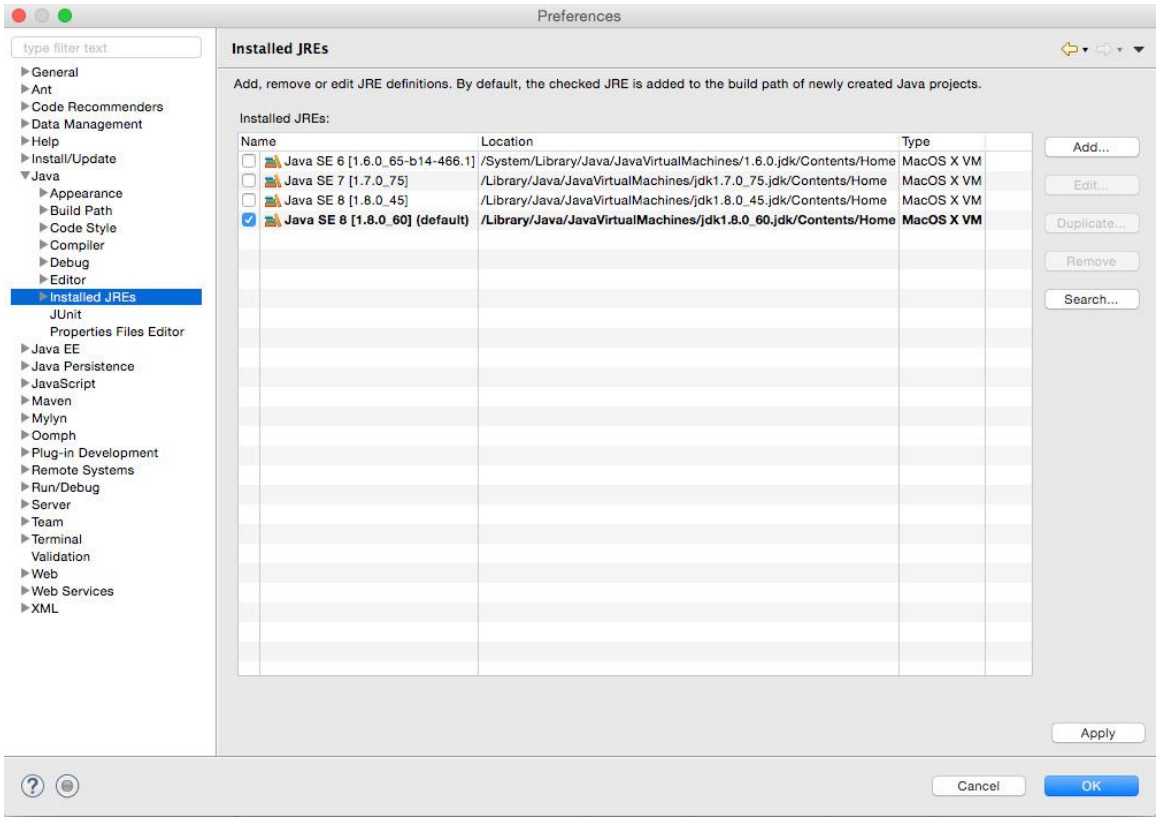


Figure 1: Configuring the JDK that Eclipse uses

5. Eclipse points to an installed JRE. You must use the JRE that you downloaded with the JDK. If Eclipse does not automatically detect the JDK you installed, click Add..., and in the next dialog box, click Standard VM and then click Next.
6. Specify the JDK's home directory (such as C:\home\jdk1.8.0_60 on Windows), and then click Finish.
7. Confirm that the JDK that you want to use is selected and click OK.
   Eclipse is now set up and ready for you to create projects, and compile and run Java code. The next section familiarizes you with Eclipse.

Getting Started with Eclipse

Eclipse is more than an IDE; it's an entire development ecosystem. This section is a brief hands-on introduction to using Eclipse for Java development.

The Eclipse Development Environment

The Eclipse development environment has four main components:

- Workspace
- Projects
- Perspectives
- Views

The primary unit of organization in Eclipse is the ***workspace***. A workspace contains all of your ***projects***. A ***perspective*** is a way of looking at each project (hence the name), and within a perspective are one or more ***views***.

The Java perspective

Figure 2 shows the Java perspective, which is the default perspective for Eclipse. You should see this perspective when you start Eclipse.

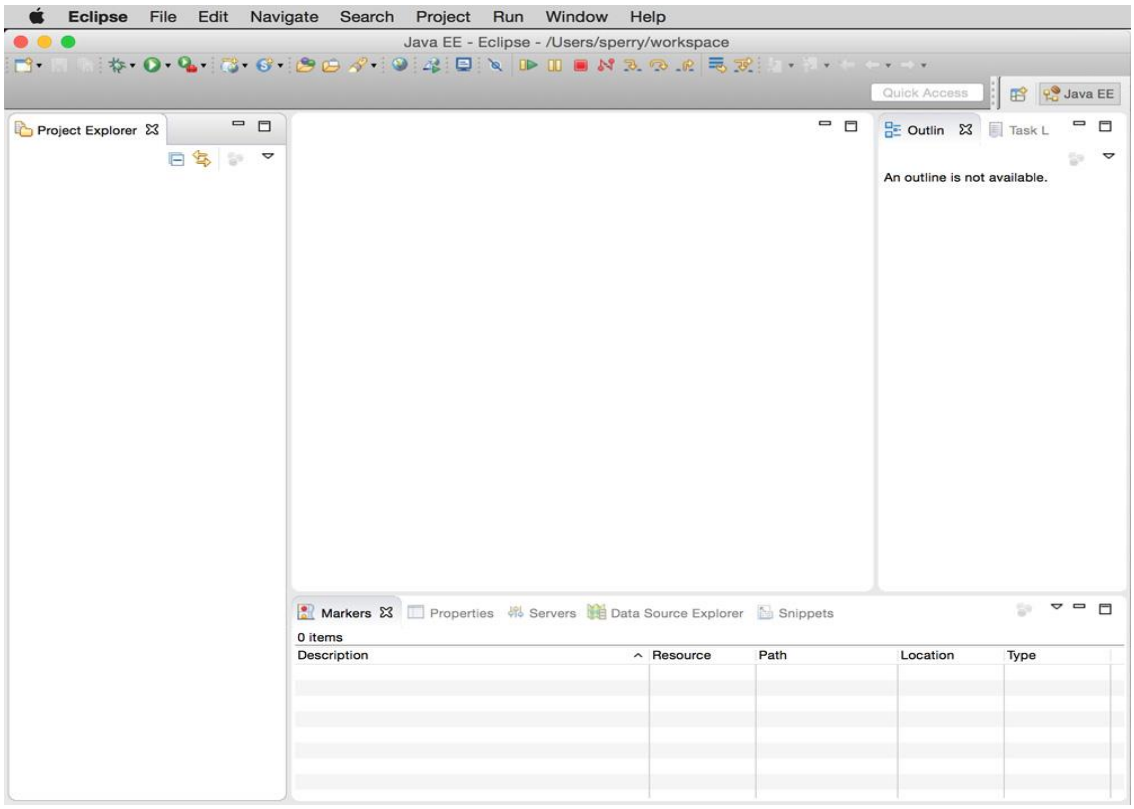


Figure 2: Eclipse Java perspective

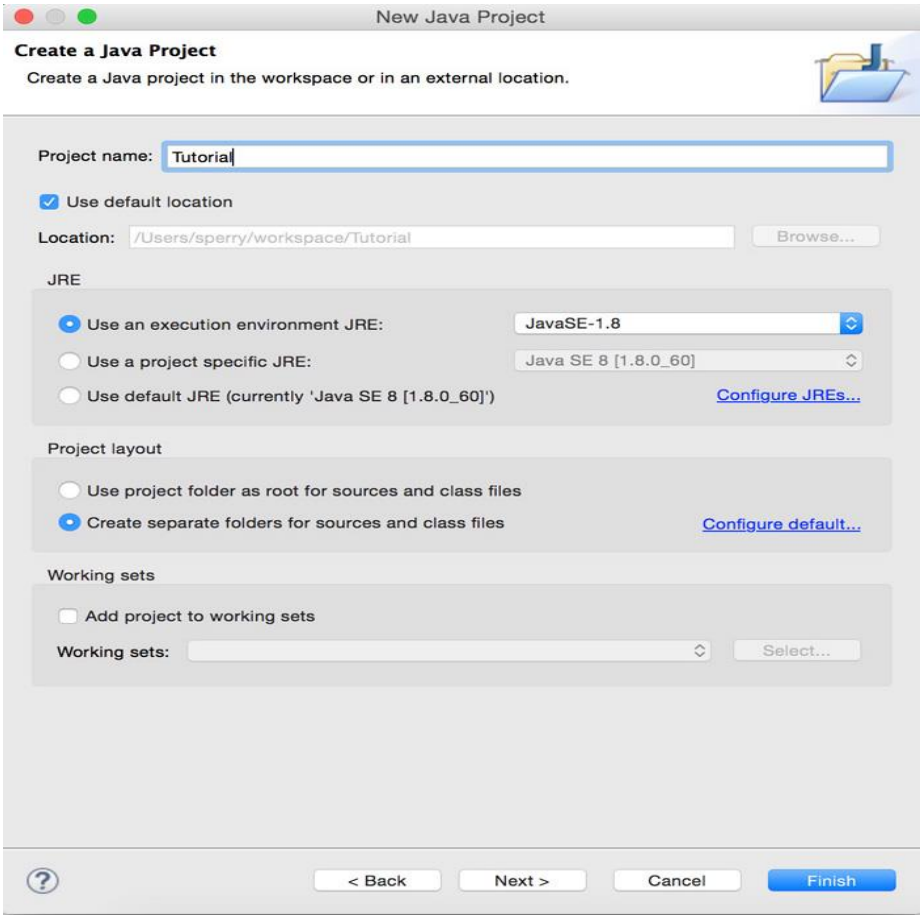
The Java perspective contains the tools that you need to begin writing Java applications. Each tabbed window shown in Figure 2is a view for the Java perspective. Package Explorer and Outline are two particularly useful views.

The Eclipse environment is highly configurable. Each view is dockable, so you can move it around in the Java perspective and place it where you want it. For now, though, stick with the default perspective and view setup.

Create a project

Follow these steps to create a new Java project:

1. Click File > New > Java Project... to start the New Java Project wizard, shown in Figure 3.



2. Enter Tutorial as the project name and click Finish.
3. If you want to modify the default project settings, click Next (recommended ***only*** if you have experience with the Eclipse IDE).
4. Click Finish to accept the project setup and create the project.

You have now created a new Eclipse Java project and source folder. Your development environment is ready for action.

Highlights

- ***Java Platform***: Java technology is used to develop applications for a wide range of environments, from consumer devices to heterogeneous enterprise systems.
- ***Java Compiler***: The compiler checks your code against the language's syntax rules, then writes out bytecode in .class files. Bytecode is a set of instructions targeted to run on a Java virtual machine (JVM).
- ***JVM***: Java Virtual Machin reads and interprets .class files and executes the program's instructions on the native hardware platform for which the JVM was written.
- ***Garbage Collector***: Garbage collection is one of the essential features of Java platform performance.
- ***JRE***: Java Runtime Environment includes the JVM, code libraries, and components that are necessary for running programs that are written in the Java language