Lecture 1: Introduction

Software

Java Development Kit (JDK)  http://www.oracle.com/technetwork/java/javase/downloads/index.html

In order to compile and run Java code you need the tools that come with the Java Development Kit. It is likely that you already have it installed on your computer, but if that is not the case go to the Oracle website and download the package appropriate for your operating system. You should be using Java 8 or 7 (see the link above).

Eclipse  http://www.eclipse.org/downloads/

One can write, compile and run Java programs using the JDK, a simple text editor and a command line environment (terminal). Most beginners and many advanced programmers prefer to work with an integrated development environment (IDE). Eclipse is one of several IDEs. We will use Eclipse in class demonstrations and the tutors can help you with it. See section 1.12 in the textbook for help on installing Eclipse and writing your first Java project.

Java

Java language specification - technical definition of the language that includes the syntax and semantics of the Java programming language;
http://docs.oracle.com/javase/specs/

Application program interface (API) - predefined classes and interfaces for developing Java programs;
http://docs.oracle.com/javase/8/docs/api/

Java is a full-fledged and powerful language that can be used in many ways. It comes in three editions:

• Java Standard Edition (Java SE) can be used to develop client-side standalone applications or applets.

• Java Enterprise Edition (Java EE) can be used to develop server-side applications, such as Java servlets and JavaServer Pages. Java Micro Edition (Java ME) can be used to develop applications for mobile devices, such as cell phones.

• Java Development Toolkit (JDK) - a set of separate programs, each invoked from a command line, for developing and testing Java programs.

Integrated development environment (IDE) - software that provides/combines many tools for rapidly developing (Java) programs: editing, compiling, building, debugging, and online help are integrated in one graphical user interface. Examples:
A Simple Java Program

Listing 1: What does this code do: what does each line mean? what is the output?

1 public class Welcome {
2    public static void main(String[] args) {
3        System.out.println("Programming is fun!");
4        System.out.println((10.5 + 3 * 2.5) / (7.5 - 4.5/3));
5    }
6 }

Listing 1, version 2:

1 // name of the class,
2 // every Java program needs at least one class
3 public class Welcome {
4    // main method of the class Welcome,
5    // that’s where the program starts
6    public static void main(String[] args) {
7        // display the string
8        System.out.println("Programming is fun!");
9        // compute and display the number
10       System.out.println((10.5 + 3 * 2.5) / (7.5 - 4.5/3));
11    }
12 }

Output:

Programming is fun!
3.0
Creating, Compiling and Executing a Java Program

<table>
<thead>
<tr>
<th>action</th>
<th>result</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create/Modify source code: use plain text</td>
<td>Source code files, ex.</td>
<td>high level language, human readable</td>
</tr>
<tr>
<td>editor</td>
<td>Welcome.java</td>
<td></td>
</tr>
<tr>
<td>Compile source code:</td>
<td>Bytecode, ex.</td>
<td>if errors occur go back to step 1 (compilation errors)</td>
</tr>
<tr>
<td>javac Welcome.java</td>
<td>Welcome.class</td>
<td>low level language, similar to machine code, runs on JVM, plus: machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>independent, minus: slower</td>
</tr>
<tr>
<td>Run bytecode:</td>
<td></td>
<td>if errors occur go back to step 1 (run-time errors)</td>
</tr>
<tr>
<td>java Welcome</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>