1. Introduction to Java programming

Java is a high-level, third generation programming language, like C,C++, FORTRAN, Smalltalk, Perl, and many others. You can use Java to write computer applications that play games, store data or do any of the thousands of other things computer software can do. Compared to other programming languages, Java is most similar to C++.

What's most special about Java in relation to other programming languages is that it lets you write special programs called applets that can be downloaded from the Internet and played safely within a web browser.

1.1 History

Java is an object-oriented programming language created by James Gosling from Sun Microsystems (Sun) in 1991. The first publicly available version of Java (Java 1.0) was released in 1995.

Sun Microsystems was acquired by the Oracle Corporation in 2010. Oracle has now the steermanship for Java.

Over time new enhanced versions of Java have been released. The current version of Java is Java 1.7 which is also known as Java 7.

From the Java programming language the Java platform evolved. The Java platform allows software developers to write program code in other languages than the Java programming language which still runs on the Java virtual machine. The Java platform is usually associated with the Java virtual machine and the Java core libraries.

In 2006 Sun started to make Java available under the GNU General Public License (GPL). Oracle continues this project called OpenJDK.

1.2 The Java compiler

A compiler is a program which converts a program from one level of language to another. Example conversion of C++ program into machine code.

When you program for the Java platform, you write source code in .java files and then compile them. The java compiler converts high level java code into bytecode (which is also a type of machine code).

The compiler checks your code against the language's syntax rules, then writes out bytecodes in .class files. Bytecodes are standard 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.

An interpreter is a program which directly executes programs source code (instructions written in a high-level language). No compilation is necessary before execution occurs.

In Java, the Just In Time Code generator converts the bytecode into the native machine code which are at the same programming levels. Hence java is both compiled as well as interpreted language.

1.3. Java virtual machine

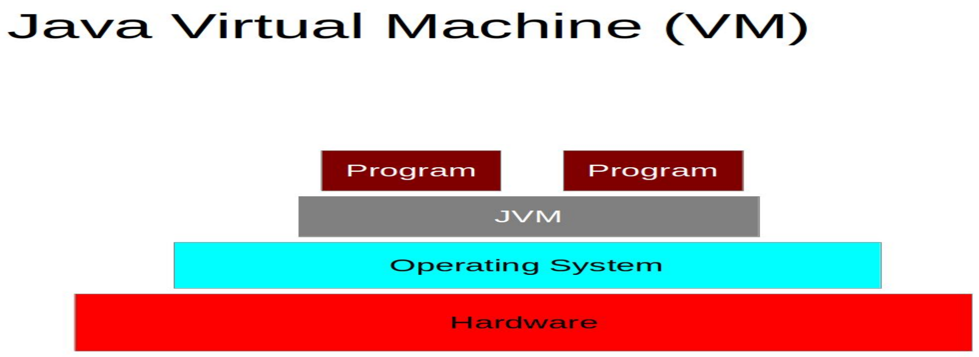
Java programs are compiled by the Java compiler into bytecode. The Java virtual machine interprets this bytecode and executes the Java program.

The Java virtual machine (JVM) is a software implementation of a computer that executes programs like a real machine.

At run time, 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 bytecodes just as a CPU would interpret assembly-language instructions.

The Java virtual machine is written specifically for a specific operating system, e.g., for Linux a special implementation is required as well as for Windows. Java programs are compiled by the Java compiler into bytecode. The Java virtual machine interprets this bytecode and executes the Java program.

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.



Roles and functions of JVM

JVM loads class files

Class files are verified to determine whether the .class file contain valid byte code

Interpretation of byte code and running the application

Garbage Collection – the process of reclaiming memory space by removing orphan objects.

Java platform

The **Java platform** is the name given to the computing platform from Oracle that helps users to *run* and *develop* Java applications. The platform does not just enable a user to run and develop Java application, but also features a wide variety of tools that can help developers work efficiently with the Java programming language.

The platform consists of two essential components:

- **Java Runtime Environment (JRE)**, which is needed to *run* Java applications and applets; and,
- **Java Development Kit (JDK)**, which is needed to *develop* those Java applications and applets. If you have installed the JDK, you should know that it comes equipped with a JRE as well. So, for all the purposes of this book, you would only require the JDK.

When installed on a computer the JRE provides the operating system with the means to run Java programs, whereas the JDK is a collection of tools used by a programmer to create Java applications.

The JDK has as its primary components a collection of programming tools, including:

- ✓ appletviewer – this tool can be used to run and debug Java applets without a web browser
- ✓ java – the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
- ✓ javac – the Java compiler, which converts source code into Java bytecode
- ✓ javadoc – the documentation generator, which automatically generates documentation from source code comments
- ✓ jar – the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.

Java Runtime Environment contains JVM, class libraries, and other supporting files. It does not contain any development tools such as compiler, debugger, etc. Actually JVM runs the program, and it uses the class libraries, and other supporting files provided in JRE. If you want to run any java program, you need to have JRE installed in the system.**Discuss the difference between JRE, JDK and JVM.**

The Java platform editions contain additional Java APIs for creating different types of applications:

Java Standard Edition

The Java Standard Edition (Java SE) is for building desktop applications and applets. These applications typically serve only a small number of users at one time.

Java Enterprise Edition

The Java Enterprise Edition (Java EE) is tailored for more complex applications to suit medium to large businesses. Typically they will be server based applications focusing on serving the needs of lots users at one time.

Note: The Java EE contains many of the Java APIs found in the Java SE.

Java Micro Edition

The Java Micro Edition is for applications used on mobile (e.g., cell phone, PDA) and embedded devices (e.g., TV tuner box, printers).

Just in Time Compiler (JIT)

Initially Java has been accused of poor performance because it’s both compiles andinterpret instruction. Since compilation or Java file to class file is independent of execution of Java program do not confuse. Here compilation word is used for byte code to machine instruction translation. JIT are advanced part of Java Virtual machine which optimize byte code to machine instruction conversion part by compiling similar byte codes at same time and thus reducing overall execution time. JIT is part of Java Virtual Machine and also performs several other optimizations such as in-lining function.

1.4. Java Runtime Environment vs. Java Development Kit

A Java distribution typically comes in two flavors, the Java Runtime Environment (JRE) and the Java Development Kit (JDK).

The Java runtime environment (JRE) consists of the JVM and the Java class libraries. Those contain the necessary functionality to start Java programs.

The JDK additionally contains the development tools necessary to create Java programs. The JDK therefore consists of a Java compiler, the Java virtual machine and the Java class libraries.

1.5. Characteristics of Java

The target of Java is to write a program once and then run this program on multiple operating systems. Java has the following properties:

a) **Platform independent:** Java programs use the Java virtual machine as abstraction and do not access the operating system directly. This makes Java programs highly portable. A Java program (which is standard-compliant and follows certain rules) can run unmodified on all supported platforms, e.g., Windows or Linux.

b) **Object-orientated programming language:** Except the primitive data types, all elements in Java are objects.

c) **Strongly-typed programming language:** Java is strongly-typed, e.g., the types of the used variables must be pre-defined and conversion to other objects is relatively strict, e.g., must be done in most cases by the programmer.

d) **Interpreted and compiled language:** Java source code is transferred into the bytecode format which does not depend on the target platform. These bytecode instructions will be interpreted by the Java Virtual machine (JVM). The JVM contains a so called Hotspot-Compiler which translates performance critical bytecode instructions into native code instructions.

e) **Automatic memory management:** Java manages the memory allocation and de-allocation for creating new objects. The program does not have direct access to the memory. The so-called garbage collector automatically deletes objects to which no active pointer exists.

1.6. Development Process with Java

Java source files are written as plain text documents. The programmer typically writes Java source code in an Integrated Development Environment (IDE) for programming. An IDE supports the programmer in the task of writing code, e.g., it provides auto-formating of the source code, highlighting of the important keywords, etc.

At some point the programmer (or the IDE) calls the Java compiler (javac). The Java compiler creates the bytecode instructions. These instructions are stored in .class files and can be executed by the Java Virtual Machine.

1.7. Garbage collector

The JVM automatically re-collects the memory which is not referred to by other objects. The Java garbage collector checks all object references and finds the objects which can be automatically released.

While the garbage collector relieves the programmer from the need to explicitly manage memory, the programmer still need to ensure that he does not keep unneeded object references, otherwise the garbage collector cannot release the associated memory. Keeping unneeded object references are typically called memory leaks.

1.8. Classpath

The classpath defines where the Java compiler and Java runtime look for .class files to load. These instructions can be used in the Java program. For example, if you want to use an external Java library you have to add this library to your classpath to use it in your program. The Java syntax is similar to C++. Java is case-sensitive, e.g., variables called myValue and myvalue are treated as different variables.

Java Development Tools

Java source files are written as plain text documents. The programmer typically writes Java source code in an *Integrated Development Environment* (IDE) for programming.
An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Several modern IDEs integrate with Intelli-sense coding features.
An IDE also supports the programmer in the task of writing code, e.g., it provides auto-formatting of the source code, highlighting of the important keywords, etc.

Common Java IDE Tools include:

- ✓ Eclipse Java development tools (JDT)
- ✓ JCreator — Java IDE
- ✓ IntelliJ IDEA
- ✓ Java-Editor
- ✓ NetBeans IDE among others

2. Installation of Java

2.1. Check installation

To run Java programs on your computer you must at least have the Java runtime environment (JRE) installed. This might already be the case on your machine. You can test is the JRE is installed and in your current path by opening a console (if you are using Windows: Win+R, enter cmd and press Enter) and by typing in the following command:

```
java -version
```

If the JRE is installed and within your path, this commands print information about your Java installation. If the command line returns the information that the program could not be found, you have to install Java.

2.2. Install Java on Ubuntu

On Ubuntu you can install Java 7 via the following command on the command line.

```
sudo apt-get install openjdk-7-jdk
```

2.3. Install Java on MS Windows

For Microsofts Windows, Oracle provides a native installer which can be found on the Oracle website. The central website for installing Java is located under the following URL and also contains instructions how to install Java for other platforms. java.com

2.4. Installation problems and other operating systems

If you have problems installing Java on your system, search via Google for How to install JDK on YOUR_OS . This should result in helpful links. Replace YOUR_OS with your operating system, e.g., Windows, Ubuntu, Mac OS X, etc.

2.5. Validate installation

Switch again to the command line and run the following command.

```
java -version
```

The output should be similar to the following output.

```
java version "1.7.0_25"
OpenJDK Runtime Environment (IcedTea 2.3.10) (7u25-2.3.10-1ubuntu0.13.04.2)
OpenJDK 64-Bit Server VM (build 23.7-b01, mixed mode)
```

2.6. How can you tell you are using a 32 bit or 64 bit version of Java?

You can run a 32 bit or a 64 bit version of Java on a 64 bit system. If you use java -version and the output contains the "64-Bit" string you are using the 64 bit version of Java otherwise your are using the 32 bit version. The following is the output of a 64-bit version.

```
java version "1.7.0_25"
OpenJDK Runtime Environment (IcedTea 2.3.10) (7u25-2.3.10-1ubuntu0.13.04.2)
OpenJDK 64-Bit Server VM (build 23.7-b01, mixed mode)
```

2.6. How can you tell you are using a 32 bit or 64 bit version of Java?

You can run a 32 bit or a 64 bit version of Java on a 64 bit system. If you use java - version and the output contains the "64-Bit" string you are using the 64 bit version of Java otherwise your are using the 32 bit version. The following is the output of a 64-bit version.

```
java version "1.7.0_25"
OpenJDK Runtime Environment (IcedTea 2.3.10) (7u25-2.3.10-1ubuntu0.13.04.2)
OpenJDK 64-Bit Server VM (build 23.7-b01, mixed mode)
```

3. Writing, compiling and running a Java program

3.1. Write source code

To create java programs any text editor such as Notepad, WordPad or gedit in ubux/linux. After creation, save the file with. Java file extension. Example: Create file Welcome.java then write the following source code:

The following Java program can be developed under Linux/Windows using a text editor and the command line. The process on other operating system should be similar.

Select or create a new directory in Linux which will be used for your Java development. In this description the path \home\john\javalessons is used. On Microsoft Windows you might want to use c:\javalessons. This path is called javadir in the following description.

Open a text editor which supports plain text, e.g., gedit under Linux or Notepad under Windows and write the following source code.

```
// Simplest Java program

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

Save the source code in your javadir directory with the HelloWorld.java filename. The name of a Java source file must always equal the class name (within the source code) and end with the .java extension. In this example the filename must be HelloWorld.java, because the class is called HelloWorld.

3.2. Compile and run your Java program

Compiling and Running Java Programs

Open a shell for command line access. Switch to the javadir directory with the command cd javadir, for example, in the above example via the cd \home\john\javalessons command. Use the ls command (dir under Microsoft Windows) to verify that the source file is in the directory.

To compile the source code file into a class file, use the javac command. For example, the following command compiles Welcome.java:
javac Welcome.java
Afterwards list again the content of the directory with the ls or dir command. The directory contains now a file HelloWorld.class. If you see this file, you have successfully compiled your first Java source code into bytecode.

You can now start to run your compiled Java program. Ensure that you are still in the jardir directory.

To run the class, use the java command. For example, the following command runs Welcome: **java Welcome**
The system should write "Hello World" on the command line.

3.3. Using the classpath

Common Error when compiling programs is:

```
'javac' is not recognized as an internal or external command,
operable program or batch file.
```

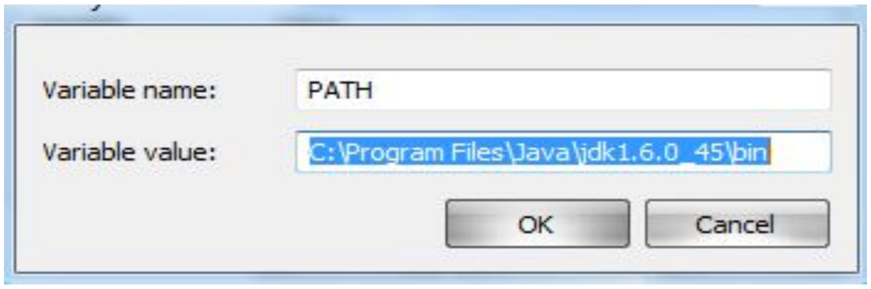
To solve the problem, set the class path in the environmental variables as follows:

Classpath

The *classpath* defines where the Java compiler and Java runtime look for .class files to load. These instructions can be used in the Java program. You can use the classpath to run the program from another place in your directory.

Configure classpath in Java environment on Windows (Windows 7)

1. Select Computer from the Start menu
2. Choose System Properties from the context menu
3. Click Advanced system settings > Advanced tab
4. Click on Environment Variables, under System Variables, find **PATH**, and click on it.
5. In the Edit windows, modify **PATH** by adding the location of the class to the value for **PATH**. If you do not have the item **PATH**, you may select to add a new variable and add **PATH** as the name and the location of the class as the value.e.g PATH= C:\Program Files\Java\jdk1.6.0_45\bin


Variable name: PATH
Variable value: C:\Program Files\Java\jdk1.6.0_45\bin

6. Reopen Command prompt window, and run your java code using javac command.

Another common error message is: Exception in thread "main" java.lang.NoClassDefFoundError

To use the class, type the following command. Replace "mydirectory" with the directory which contains the test directory. You should again see the "HelloWorld" output.

```
java -classpath "mydirectory" HelloWorld
```

Java Development Tools

Java source files are written as plain text documents. The programmer typically writes Java source code in an *Integrated Development Environment* (IDE) for programming.
An integrated development environment (IDE) or interactive development environment is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Several modern IDEs integrate with Intelli-sense coding features.
An IDE also supports the programmer in the task of writing code, e.g., it provides auto-formatting of the source code, highlighting of the important keywords, etc.

Common Java IDE Tools include:

- ✓ Eclipse Java development tools (JDT)
- ✓ JCreator — Java IDE
- ✓ IntelliJ IDEA
- ✓ Java-Editor
- ✓ NetBeans IDE among others