Java – Overview

Java is **a high-level programming language** originally developed by Sun Microsystems and released in 1995. Java **runs on a variety of platforms**, such as Windows, Mac OS, and the various versions of UNIX.

The latest release of the Java Standard Edition is **Java SE 8** (Java8). With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively.

Java is –

- **Object Oriented** – In Java, everything is an Object. Java can be easily extended since it is based on the Object model
- **Platform Independent** – Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- **Simple** – Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- **Secure** – With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- **Architecture-neutral** – Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- **Robust** – Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.

- **Multithreaded** – With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

- **Interpreted** – Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.

- **High Performance** – With the use of Just-In-Time compilers, Java enables high performance.

- **Distributed** – Java is designed for the distributed environment of the internet.

- **Dynamic** – Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

History of Java

James Gosling initiated Java language project in June 1991 for use in one of his many set-top box projects. The language, initially called 'Oak' after an oak tree that stood outside Gosling's office, also went by the name 'Green' and ended up later being renamed as Java, from a list of random words.

Sun released the first public implementation as Java 1.0 in 1995. It promised **Write Once, Run Anywhere** (WORA), providing no-cost run-times on popular platforms.

On 13 November, 2006, Sun released much of Java as free and open source software under the terms of the GNU General Public License (GPL).

On 8 May, 2007, Sun finished the process, making all of Java's core code free and open-source, aside from a small portion of code to which Sun did not hold the copyright.

Tools You Will Need

For performing the Java examples, you will need a Pentium 200-MHz computer with a minimum of 64 MB of RAM.

You will also need the following softwares –

- Linux 7.1 or Windows xp/7/8 operating system
- Java JDK 8
- Microsoft Notepad or any other text editor

Local Environment Setup

Java SE is freely available from the link *www.oracle.com*. You can download a version based on your operating system.

Follow the instructions to download Java and run the **.exe** to install Java on your machine. Once you installed Java on your machine, you will need to set environment variables to point to correct installation directories –

Setting Up the Path for Windows

Assuming you have installed Java in *c:\Program Files\java\jdk* directory

- ❖ Right-click on 'My Computer' and select 'Properties'.
- ❖ Click the 'Environment variables' button under the 'Advanced' tab.
- ❖ Now, alter the 'Path' variable so that it also contains the path to the Java executable. Example, if the path is currently set to 'C:\WINDOWS\SYSTEM32', then change your path to read 'C:\WINDOWS\SYSTEM32;**C*:\Program Files\java\jdk\bin***'.

Java - Basic Syntax

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into what do class, object, methods, and instance variables mean.

- **Object** – Objects have states and behaviors.
  Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.

- **Class** – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.

- **Methods** – A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.

- **Instance Variables** – Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

First Java Program

Let us look at a simple code that will print the words ***Hello World***.

example:

```
public class MyFirstJavaProgram {

   /* This is my first java program.
    * This will print 'Hello World' as the output
    */

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

Let's look at how to **save** the file, **compile**, and **run** the program. Please follow the subsequent steps –

- Open notepad and add the code as above.
- Save the file as: **MyFirstJavaProgram.java**
- Open a command prompt window and go to the directory where you saved the class. Assume it's C:\.
- Type **javac MyFirstJavaProgram.java** and press enter to compile your code. If there are no errors in your code, the command prompt will take you to the next line.
- Now, type **java MyFirstJavaProgram** to run your program.
- You will be able to see **Hello World** printed on the window.

Output

```
C:\> javac MyFirstJavaProgram.java
C:\> java MyFirstJavaProgram
Hello World
```

Basic Syntax

About Java programs, it is very important to keep in mind the following points.

- **Case Sensitivity –** Java is case sensitive, which means identifier ***Hello*** and ***hello*** would have different meaning in Java.

- **Class Names –** For all class names the **first letter** should be in **Upper Case**. If several words are used to form a name of the class, **each inner word's first letter** should be in **Upper Case**.
  Example: class **M**y**F**irst**J**ava**C**lass

- **Method Names –** All method names should start with a **Lower Case** letter. If several words are used to form the name of the method, then **each inner word's first letter** should be in **Upper Case**
  Example: public void **myM**ethod**N**ame()

- **Program File Name –** Name of the program file should **exactly match** the **class name**
  Example: Assume 'MyFirstJavaProgram' is the class name.
  Then the file should be saved as 'MyFirstJavaProgram**.java**'

- **public static void main(String args[]) –** Java program processing starts from the **main()** method which is a mandatory part of every Java program.

Java Keywords

The following list shows the reserved words in Java. These reserved words may not be used as constant or variable or any other identifier names.

Abstract | assert | **boolean** | **break** | **byte** | **case** | **catch** | **char** | **class** | const | continue | default | **do** | **double** | **else** | **enum** | **extends** | **final** | **finally** | **float** | **for** | goto | **if** | **implements** | **import** | instanceof | **int** | **interface** | **long** | native | **new** | **package** | **private** | **protected** | **public** | **return** | **short** | **static** | strictfp | **super** | **switch** | synchronized | **this** | **throw** | **throws** | transient | **try** | **void** | volatile | **while**

What is For loop?

It executes a block of statements repeatedly until the specified condition returns false.

Syntax of for loop:

```
for (initialization; condition; increment/decrement) {
    statement(s) //block of statements
}
```

Mind the semicolon (;) after initialization and condition in the above syntax.

Initialization expression executes only once during the beginning of loop **Condition (Boolean Expression)** gets evaluated each time the loop iterates. Loop executes the block of statement repeatedly until this condition returns false. **Increment/Decrement** It executes after each iteration of loop.

For loop example:

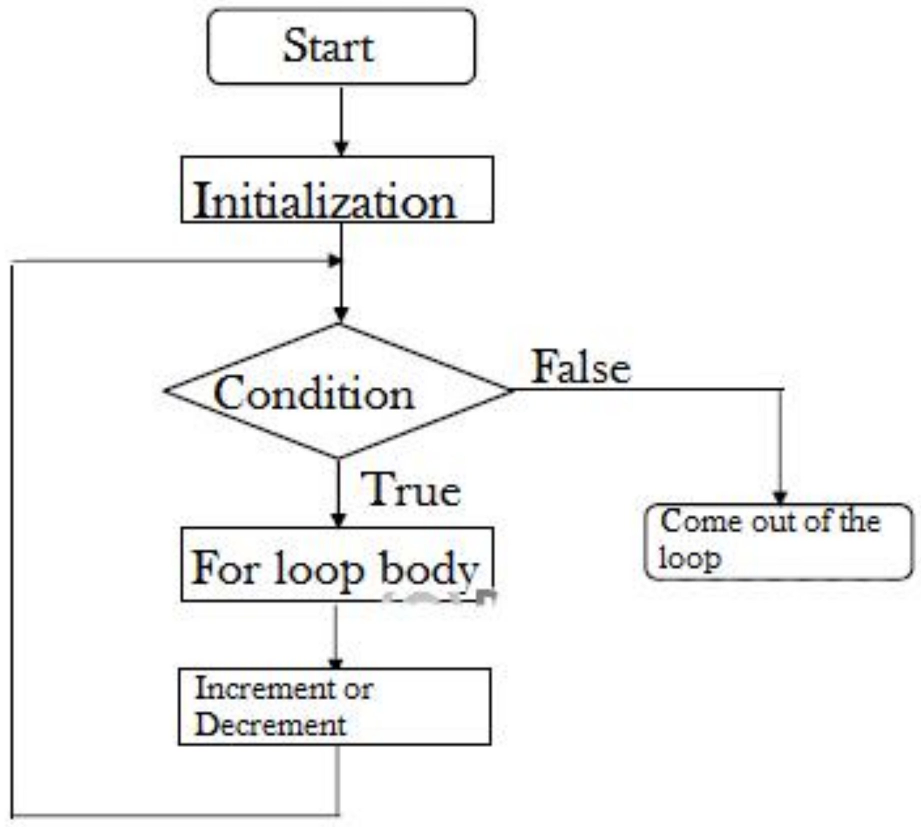
```
class ForLoopExample {
    public static void main(String args[]){
         for(int i=10; i>1; i--){
              System.out.println("The value of i is: "+i);
         }
    }
}
```

The output of this program is:

```
The value of i is: 10
The value of i is: 9
The value of i is: 8
The value of i is: 7
The value of i is: 6
The value of i is: 5
The value of i is: 4
The value of i is: 3
The value of i is: 2
```

In the above program:
int i=1 is initialization expression
i>1 is condition(Boolean expression)
i– Decrement operation

Flow diagram of for loop


Start
Initialization
Condition
False
True
For loop body
Come out of the loop
Increment or Decrement