Introduction to Java programming

Java is an **object-oriented programming language** developed by Sun Microsystems, a company best known for its high-end Unix workstations in 1991, later acquired by Oracle Corporation. It was conceived by James Gosling and Patrick Naughton. It is a simple programming language. Writing, compiling and debugging a program is easy in java. It helps to create modular programs and reusable code.

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 virtual Machine (JVM) on whichever platform it is being run. *Platform-independence* is a program's capability of moving easily from one Computer system to another. Java is platform-independent at both the source and the binary level

Simple: Java is designed to be easy to learn. If you understand the basic concept of OOP Java 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.

Architectural-neutral: Java compiler generates an architecture-neutral object file format which makes the compiled code to be executable on many processors, with the presence of Java runtime system. The primary motivation of Java was the need for a platform-independent (that is, architecture-neutral) language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls.

How Java Changed the Internet

Java innovated a new type of networked program called the **applet** that changed the way the online world thought about content. Java also addressed some of the thorniest issues associated with the Internet: **portability and security**

Portable: Being architectural-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary which is a POSIX subset. Java also provides for portable programming with applets. Applets appear in a Web page much in the same way as images do, but unlike images, applets are dynamic and interactive. Applets can be used to create animations, figures, or areas that can respond to input from the reader, games, or other interactive effects on the same Web pages among the text and graphics.

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 do many tasks simultaneously. This design feature allows developers to construct smoothly running interactive applications.

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.

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.

Java Applets

An ***applet*** is a special kind of Java program that is designed to be transmitted over the Internet and automatically executed by a Java-compatible web browser. Furthermore, an applet is downloaded on demand, without further interaction with the user. If the user clicks a link that contains an applet, the applet will be automatically downloaded and run in the browser. Applets are intended to be small programs. They are typically used to display data provided by the server, handle user input, or provide simple functions, such as a loan calculator, that execute locally, rather than on the server. In essence, the applet allows some functionality to be moved from the server to the client.
The creation of the applet changed Internet programming because it expanded the universe of objects that can move about freely in cyberspace**. In general, there are two very broad categories of objects that are transmitted between the server and the client: passive information and dynamic, active programs.** For example, when you read your e-mail, you are viewing passive data. Even when you download a program, the program's code is still only passive data until you execute it. By contrast, the applet is a dynamic, self-executing program.

Such a program is an active agent on the client computer, yet it is initiated by the server. As desirable as dynamic, networked programs are, they also present serious problems in the areas of security and portability. Obviously, a program that downloads and executes automatically on the client computer must be prevented from doing harm. It must also be able to run in a variety of different environments and under different operating systems

```
// java program starts with class class_name which should be the name of the java file

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

To write your Java programs, you will need a text editor. There are even more sophisticated IDEs available in the market. But for now, you can consider one of the following:

- **Notepad:** On Windows machine you can use any simple text editor like Notepad (Recommended for this tutorial), TextPad.
- **Netbeans:** is a Java IDE that is open-source and free which can be downloaded from http://www.netbeans.org/index.html.
- **Eclipse:** is also a Java IDE developed by the eclipse open-source community and can be downloaded from http://www.eclipse.org

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 MyFirstJavaClass*

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.

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

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

Structure of Java Source code

1) It essentially consists of a main() method
2) This method is public and thus can be called by any object
3) This method is also static and so can be called without instantiating the object of the class
4) It does not return any value (therefore void keyword is used)
5) The controlling class of every Java application usually contain a main method
6) This can be avoided to allow the class to be tested in a stand-alone mode.
7) Other methods can subsequently be called in main()

```
public class StringDemo{

   public static void main(String args[]){
      char[] helloArray = { 'h', 'e', 'l', 'l', 'o', '.'};
      String helloString = new String(helloArray);
      System.out.println( helloString );
   }
}
```

O/P

hello.

```
public class StringDemo {

   public static void main(String args[]) {
      String palindrome = "Dot saw I was Tod";
      int len = palindrome.length();
      System.out.println( "String Length is : " + len );
   }
}
```

This would produce the following result:

```
String Length is : 17
```

```
public class StringDemo{

   public static void main(String args[]) {
      String string1 = "saw I was";
      System.out.println("Dot " + string1 + "Tod");
   }
}
```

O/P

Dot saw I was Tod

Main Features of JAVA

Java is a platform independent language

To understand the meaning of platform independent, we must need to understand the meaning of platform first. A platform is a pre-existing environment in which a program runs, obeying its constraints, and making use of its facilities.
Lets back to the point. During compilation, the compiler converts java program to its byte code. This byte code can run on any platform such as Windows, Linux, Mac/OS etc. Which means a program that is compiled on windows can run on Linux and vice-versa. This is why java is known as platform independent language.

Java is an Object Oriented language

Object oriented programming is a way of organizing programs as collection of objects, each of which represents an instance of a class.

4 main concepts of Object Oriented programming are:

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Simple

Java is considered as one of simple language because it does not have complex features like Operator overloading, Multiple inheritance, pointers and Explicit memory allocation.

Robust Language

Two main problems that cause program failures are memory management mistakes and mishandled runtime errors. Java handles both of them efficiently.
1) Memory management mistakes can be overcome by garbage collection. Garbage collection is automatic de-allocation of objects which are no longer needed.
2) Mishandled runtime errors are resolved by Exception Handling procedures.

Secure

It provides a virtual firewall between the application and the computer. Java codes are confined within Java Runtime Environment (JRE) thus it does not grant unauthorized access on the system resources.

Java is distributed

Using java programming language we can create distributed applications. RMI(Remote Method Invocation) and EJB(Enterprise Java Beans) are used for creating distributed applications in java. In simple words: The java programs can be distributed on more than one systems that are connected to each other using internet connection. Objects on one JVM (java virtual machine) can execute procedures on a remote JVM.

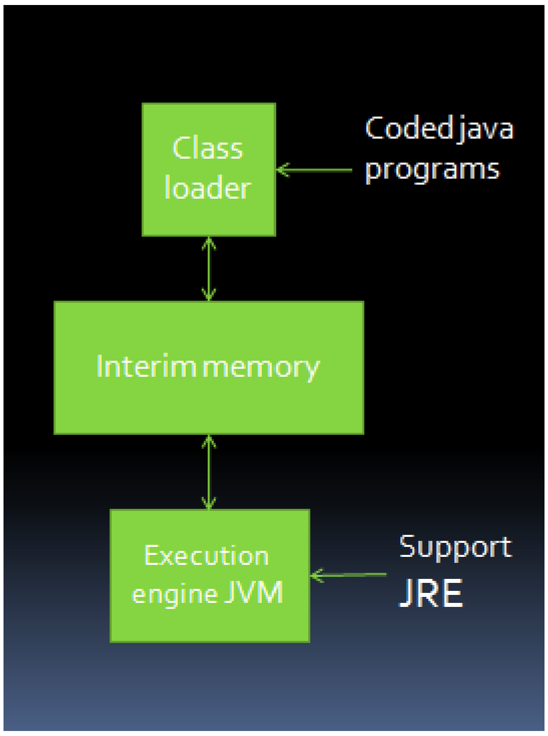
Multithreading

Java supports multithreading. It enables a program to perform several tasks simultaneously.

Portable

As discussed above, java code that is written on one machine can run on another machine. The platform independent byte code can be carried to any platform for execution that makes java code portable.

JVM (Java Virtual Machine)



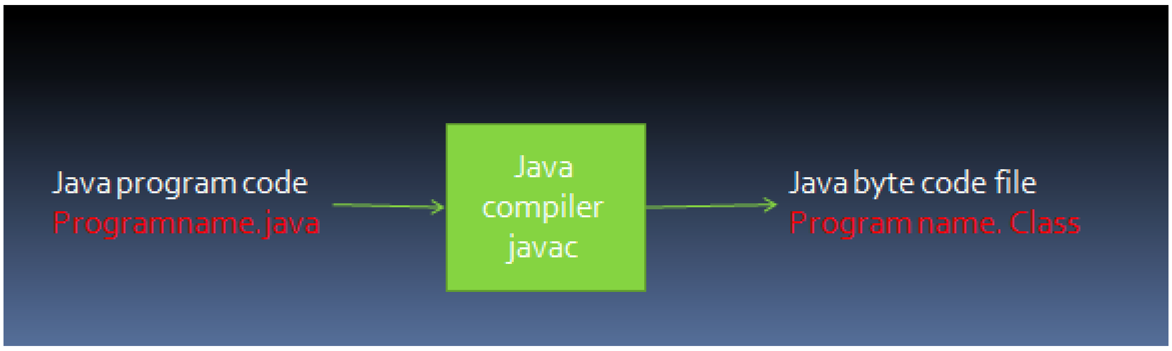
1) Class loader accepts class files
2) Compilation creates class files
3) The interim memory is required during execution
4) It consists of heaps, stacks and registers to store data
5) JRE has native methods and libraries
6) JVM runs two main threads
    a) demon
    b) Non-demon threads

Execution on JVM

1) JVM executes Java byte codes
2) Other programming language codes if converted to adequate Java byte code can be executed on JVM
3) JVM is different for different platforms and can also act as a platform itself
4) JVM supports automatic error handling by intercepting the errors which can be controlled
5) This feature is useful in platform independency and multi user ability of Java.

Compilation

1) The compiler requires to know the TYPE of every CLASS used in the program source code
2) This is done by setting a default user environment variable CLASSPATH
3) The Javac (Java Compiler) reads the program and converts it into byte code files called as class files



Some programs

```
import java.util.Scanner;

class CircleDemo
{
   static Scanner sc = new Scanner(System.in);
   public static void main(String args[])
   {
      System.out.print("Enter the radius: ");
      /*We are storing the entered radius in double
       * because a user can enter radius in decimals
       */
      double radius = sc.nextDouble();
      //Area = PI*radius*radius
      double area = Math.PI * (radius * radius);
      System.out.println("The area of circle is: " + area);
      //Circumference = 2*PI*radius
      double circumference= Math.PI * 2*radius;
      System.out.println( "The circumference of the circle is:"+circumference) ;
   }
}
```

Output:

```
Enter the radius: 1
The area of circle is: 3.141592653589793
The circumference of the circle is:6.283185307179586
```

```
class CircleDemo2
{
   public static void main(String args[])
   {
      int radius = 3;
      double area = Math.PI * (radius * radius);
      System.out.println("The area of circle is: " + area);
      double circumference= Math.PI * 2*radius;
      System.out.println( "The circumference of the circle is:"+circumference) ;
   }
}
```

Output:

```
The area of circle is: 28.274333882308138
The circumference of the circle is:18.84955592153876
```

```
import java.util.Scanner;
class SumDemo{
   public static void main(String args[]){
      Scanner scanner = new Scanner(System.in);
      int[] array = new int[10];
      int sum = 0;
      System.out.println("Enter the elements:");
      for (int i=0; i<10; i++)
      {
            array[i] = scanner.nextInt();
      }
      for( int num : array) {
         sum = sum+num;
      }
      System.out.println("Sum of array elements is:"+sum);
   }
}
```

Output:

```
Enter the elements:
1
2
3
4
5
6
7
8
9
10
Sum of array elements is:55
```

```
import java.util.Scanner;

class GetInputData
{
  public static void main(String args[])
  {
     int num;
     float fnum;
     String str;

     Scanner in = new Scanner(System.in);

     //Get input String
     System.out.println("Enter a string: ");
     str = in.nextLine();
     System.out.println("Input String is: "+str);

     //Get input Integer
     System.out.println("Enter an integer: ");
     num = in.nextInt();
     System.out.println("Input Integer is: "+num);

     //Get input float number
     System.out.println("Enter a float number: ");
     fnum = in.nextFloat();
     System.out.println("Input Float number is: "+fnum);
  }
}
```

Output:

```
Enter a string:
Chaitanya
Input String is: Chaitanya
Enter an integer:
27
Input Integer is: 27
Enter a float number:
12.56
Input Float number is: 12.56
```