Overview of Java

- Java programming language was originally developed by Sun Microsystems which was initiated by **James Gosling** and released in **1995** as core component of Sun Microsystems' Java platform **(Java 1.0 [J2SE]).**

Java is a high-level object-oriented programming language, which provides developers with the means to create powerful applications, which are very small in size, platform independent, secure and robust.

- Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions ofUNIX.
- Java is mainly used for Internet Programming.
- Java is related to the languages C and C++. From C, Java inherits its syntax and from C++,Java inherits its OOP concepts.
- Ancestors of Java: - C, C++, B, BCPL.

Five primary goals in the creation of the Java language:

1. It should use the object-oriented programming methodology.
2. It should allow the same program to be executed on multiple operating systems.
3. It should contain built-in support for using computer networks.
4. It should be designed to execute code from remote sources securely.
5. It should be easy to use.

BASIC JAVA TERMINOLOGIES:

1. **BYTECODE:**

   Byte code is an intermediate code generated from the source code by java compiler and it is platform independent.

2. **JAVA DEVELOPMENT KIT (JDK):**

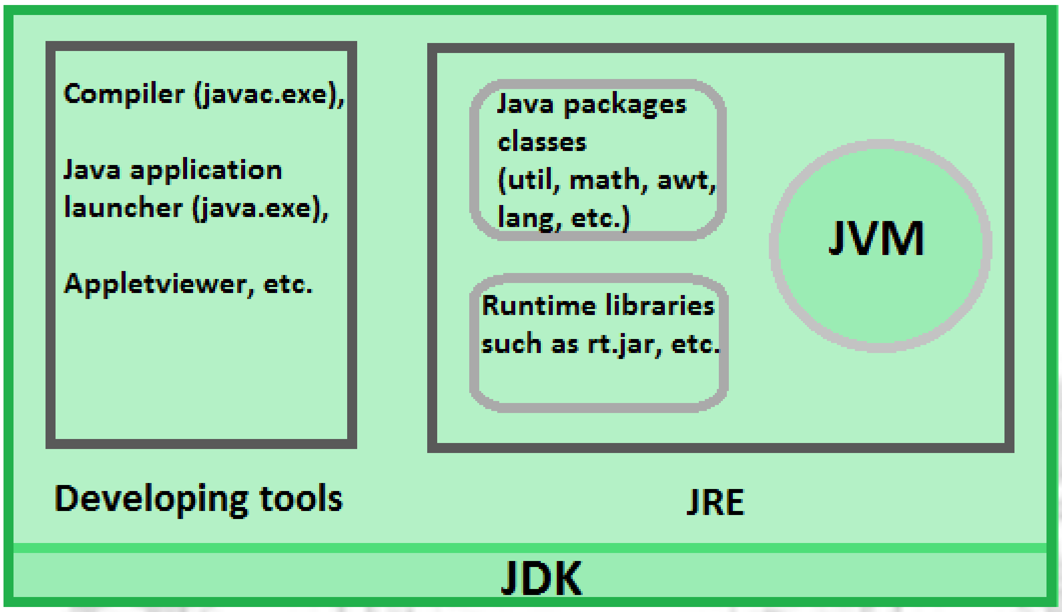
- The Java Development Kit (JDK) is a software development environment usedfor developing Java applications and applets.
- It includes the Java Runtime Environment (JRE), an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) and other tools needed in Java development.

3. **JAVA RUNTIME ENVIRONMENT (JRE):**

   JRE is used to provide runtime environment for JVM. It contains set of libraries +other files that JVM uses at runtime.

4. **JAVA VIRTUAL MACHINE (JVM):**

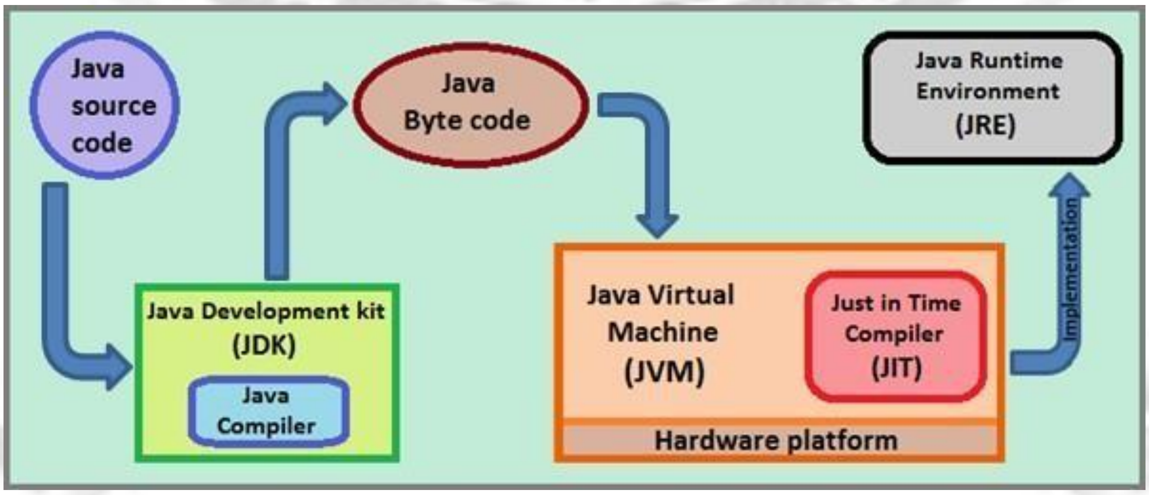
- JVM is an interpreter that converts a program in Java bytecode (intermediate language) into native machine code and executes it.
- JVM needs to be implemented for each platform because it will differ from platform to platform.


Compiler (javac.exe),
Java application launcher (java.exe),
Appletviewer, etc.
Java packages classes (util, math, awt, lang, etc.)
Runtime libraries such as rt.jar, etc.
JVM
Developing tools
JRE
JDK

➢ The JVM performs following main tasks:
- Loads code
- Verifies code
- Executes code
- Provides runtime environment

5. **JIT (JUST IN TIME) COMPILER:**

It is used to improve the performance. JIT compiles parts of the byte code that have similar functionality at the same time, and hence reduces the amount of time neededfor compilation.


Java source code
Java Byte code
Java Runtime Environment (JRE)
Java Development kit (JDK)
Java Compiler
Java Virtual Machine (JVM)
Just in Time Compiler (JIT)
Hardware platform
Implementation

Types of Java program:

In Java, there are two types of programs namely,

1. Application Program
2. Applet Program

1. Application Programs

Application programs are stand-alone programs that are written to carry out certain tasks on local computer such as solving equations, reading and writing files etc. The application programs can be executed using two steps:

1. Compile source code to generate Byte code using javac compiler.
2. Execute the byte code program using Java interpreter.

2. Applet programs:

Applets are small Java programs developed for Internet applications. An applet located in distant computer can be downloaded via Internet and executed on a local computer using Java capable browser. The Java applets can also be executed in the command line using appletviewer, which is part of the JDK.

JAVA SOURCE FILE - STRUCTURE – COMPILATION

THE JAVA SOURCE FILE:

A Java source file is a plain text file containing Java source code and having .java extension. The .java extension means that the file is the Java source file. Java source code file contains source code for a class, interface, enumeration, or annotation type. There are some rules associated to Java source file.

Java Program Structure:

Java program may contain many classes of which only one class defines the main method.
A Java program may contain one or more sections.

Documentation Section
Package Statement
Import Statements
Interface Statements
Class Definitions

main Method Class
{
Main Method Definition
}

Of the above Sections shown in the figure, the Main Method class is Essential part, Documentation Section is a suggested part and all the other parts are optional.

Documentation Section

✓ It Comprises a Set of comment lines giving the name of the program, the author andother details.

- ✓ Comments help in Maintaining the Program.
- ✓ Java uses a Style of comment called *documentation comment.*

  /** …… */
- ✓ This type of comment helps is generating the documentation automatically.
- ✓ **Example:**

```
/*
 * Title: Conversion of Degrees
 * Aim: To convert Celsius to Fahrenheit and vice versa
 * Date: 31/08/2000
 * Author: tim
*/
```

Package Statement

- ✓ The first statement allowed in a Java file is a package statement.
- ✓ It declares the package name and informs the compiler that the classes defined belong to this package.
- ✓ **Example :**

```
package student;
package basepackage.subpackage.class;
```

- ✓ It is an optional declaration.

Import Statements

- ✓ The statement instructs the interpreter to load a class contained in a particular package.
- ✓ Example :

```
import student.test;
```

  Where, student is the package and test is the class.

Interface Statements

- ✓ An interface is similar to classes which consist of group of method declaration.
- ✓ Like classes, interfaces contain methods and variable.
- ✓ To link the interface to our program, the keyword **implements** is used.
- ✓ **Example:**

```
public class xx extends Applet implements ActionListener
```

  where, xx – class name (subclass of Applet)Applet – Base class name ActionListener – interface Extends & implements - keywords
- ✓ It is used when we want to implement the feature of Multiple Inheritance in Java
- ✓ It is an optional declaration.

Class Definitions

- ✓ A Java Program can have any number of class declarations.

- ✓ The number of classes depends on the complexity of the program.

Main Method Class

- ✓ Every Java Standalone program requires a main method as its starting point.
- ✓ A Simple Java Program will contain only the main method class.
- ✓ It creates objects of various classes and uses those objects for performing various operations.
- ✓ When the end of main is reached the program terminates and the control transferred back to the Operating system.
- ✓ **Syntax for writing main:**

  public static void main(String arg[])

  where,

  public – It is an access specifier to control the visibility of class members. main() must be declared as public, since it must be called by code outside of its class when the program is started.

  static – this keyword allows main() method to be called without having to instantiate the instance of the class.

  void – this keyword tells the compiler that main() does not return any value.

  main() – is the method called when a Java application begins.

  String arg[] – arg is an string array which receives any command-line arguments present when the program is executed.

<u>Rules to be followed to write Java Programs:</u>

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 hellowould 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 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.

  Example public void myMethodName()
- ***Program File Name - Name of the program file should exactly match the classname.***

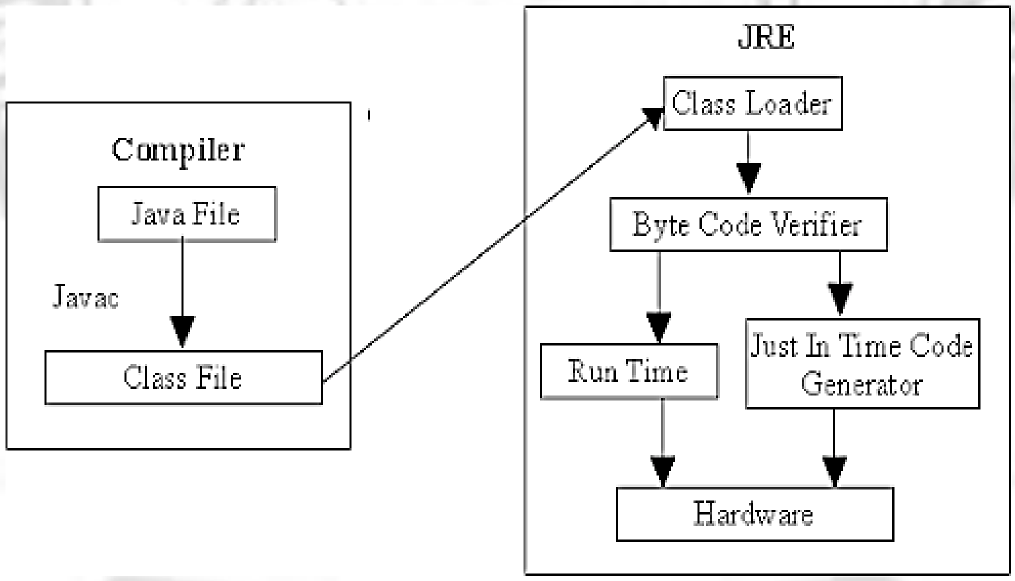
  When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name donot match your program will not compile).

  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.***

Compiling and running a java program in command promptSTEPS:

1. Set the path of the compiler as follows (type this in command prompt):
   Set path="C:\Program Files\Java\jdk1.6.0_20\bin";
2. To create a Java program, ensure that the name of the class in the file is the sameas the name of the file.
3. Save the file with the extension .java (Example: HelloWorld.java)
4. To compile the java program use the command javac as follows:
   javac HelloWorld.java
   This will take the source code in the file HelloWorld.java and create the javabytecode in a file HelloWorld.class
5. To run the compiled program use the command java as follows:
   java HelloWorld
   (Note that you do not use any file extension in this command.)

   At compile time, java file is compiled by Java Compiler (It does not interact with OS) andconverts the java code into bytecode.



Class Loader : is the subsystem of JVM that is used to load class files.
Bytecode Verifier : checks the code fragments for illegal code that can violate access right to objects
Interpreter : read bytecode stream then execute the instructions.

Example 1: A First Java Program:

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

Save: HelloWorld.java **Compile:** javac HelloWorld.java**Run:** java HelloWorld
Output: Hello World

Program Explanation:

public is the access specifier, **class** is a keyword and **HelloWorld** is the class name. **{** indicates the start of program block and **}** indicates the end of the program block. **System.out.println()** – is the output statement to print some message on the screen. Here, **System** is a predefined class that provides access to the system, **out** is the output stream that is connected to the console and **println()** is method to display the given string.

Example 2: A Second Java Program:

```
import java,util.Scanner; // Scanner is a class which contains necessary methods
                            to provide a user an access to the i/p console.
public class Example2                    // class declaration
{                                        // class definition starts
public static void main(String args[])
{
//main() definition starts
    int num=0,res;  // declares two integer with initial value 0
    Scanner in=new Scanner(System.in); //creating object of Scanner class toaccess the i/p stream.
    System.out.println("Enter a Number : ");
     num=in.nextInt();                      // to read the next integer value from the i/pstream
    res=num*2;                   // manipulation of the data
     System.out.println("The value of "+num+" * 2 = "+res); //displaysresult
}
}
```

Save: Example2.java **Compile:** javac Example2.java**Run:** java Example2

Output:

```
Enter a Number: 25
The value of 25 * 2 = 50
```