Introduction to Java Programming

JAVA was developed by Sun Microsystems Inc in 1991, In 2010, JAVA acquired by Oracle Corporation. It was developed 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.

Java terminology

These are the common java terms-

Java Virtual Machine (JVM) - Before, we discuss about JVM lets see the phases of program execution. Phases are as follows: we write the program, then we compile the program and at last we run the program.
1) Writing of the program is of course done by java programmer like you and me.
2) Compilation of program is done by javac compiler, javac is the primary java compiler included in java development kit (JDK). It takes java program as input and generates java bytecode as output.
3) In third phase, JVM executes the bytecode generated by compiler. This is called program run phase.

So, now that we understood that the primary function of JVM is to execute the bytecode produced by compiler. Each operating system has different JVM, however the output they produce after execution of bytecode is same across all operating systems. That is why we call java as platform independent language. JVM makes java platform independent.

Bytecode - As discussed above, javac compiler of JDK compiles the java source code into bytecode so that it can be executed by JVM. The bytecode is saved in a .class file by compiler.

Java Development Kit(JDK)- Java development kit includes JRE (Java Runtime Environment), compilers and various tools like JavaDoc, Java debugger etc.
In order to create, compile and run Java program you would need JDK installed on your computer.

Java Runtime Environment(JRE) - JRE is a part of JDK which means that JDK includes JRE. When you have JRE installed on your system, you can run a java program however you won't be able to compile it. JRE includes JVM, browser plugins and applets support. When you only need to run a java program on your computer, you would only need JRE.

Main Features of JAVA

Java is a platform independent language - Compiler(javac) converts source code (.java file) to the byte code(.class file). JVM executes the bytecode produced by compiler. 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. Each operating system has different JVM, however the output they produce after execution of bytecode is same across all operating systems. That is why we call java 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.

Four main concepts of Object Oriented programming are: Abstraction, Encapsulation, Inheritance and 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 - Robust means reliable. Java programming language is developed in a way that puts a lot of emphasis on early checking for possible errors, that's why java compiler is able to detect errors that are not easy to detect in other programming languages. The main features of java that makes it robust are garbage collection, Exception Handling and memory allocation.

Secure - We don't have pointers and we cannot access out of bound arrays (you get ArrayIndexOutOfBoundsException if you try to do so) in java. That's why several security flaws like stack corruption or buffer overflow is impossible to exploit in Java.

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. Multithreading is a Java feature that allows concurrent execution of two or more parts of a program for maximum utilization of CPU.

Portable – It follows the WORA (write once run anywhere). 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.

Difference between Compiler and Interpreter

Interpreters and compilers are very similar in structure. The main difference is that an interpreter directly executes the instructions in the source programming language while a compiler translates those instructions into efficient machine code.

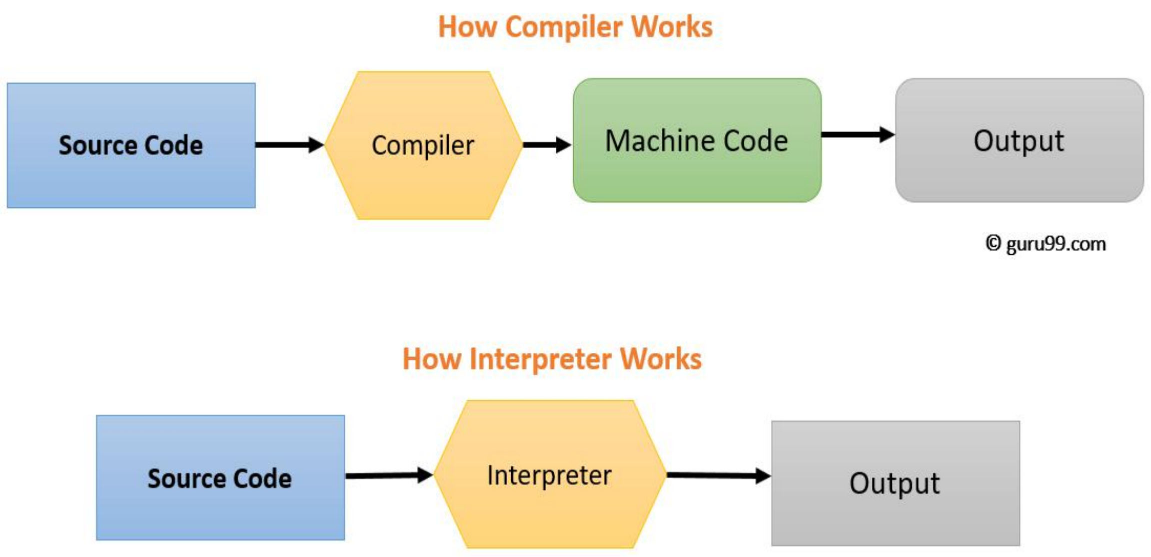
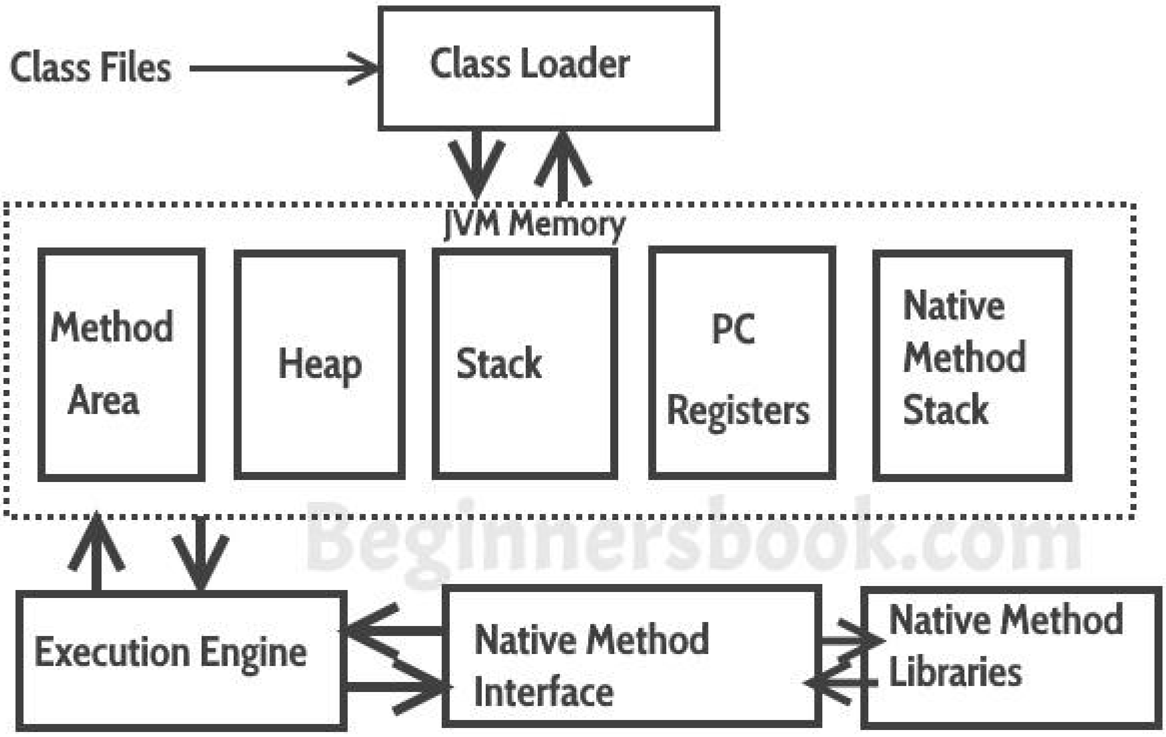


Figure 1

JVM Architecture

The Java Virtual machine (JVM) is the virtual machine that runs on actual machine (your computer) and executes Java byte code. The JVM doesn't understand Java source code, that's why we need to have javac compiler that compiles *.java files to obtain *.class files that contain the byte codes understood by the JVM. JVM makes java portable (write once, run anywhere). Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems.



Let's see how JVM works:

Class Loader- The class loader reads the .class file and save the byte code in the method area.

Method Area- There is only one method area in a JVM which is shared among all the classes. This holds the class level information of each .class file.

Heap- Heap is a part of JVM memory where objects are allocated. JVM creates a Class object for each .class file.

Stack- Stack is a also a part of JVM memory but unlike Heap, it is used for storing temporary variables.

PC Registers- This keeps the track of which instruction has been executed and which one is going to be executed. Since instructions are executed by threads, each thread has a separate PC register.

Native Method Stack- A native method can access the runtime data areas of the virtual machine.

Native Method Interface- It enables java code to call or be called by native applications. Native applications are programs that are specific to the hardware and OS of a system.

Garbage Collection- A class instance is explicitly created by the java code and after use it is automatically destroyed by garbage collection for memory management.

How to Compile and Run your First Java Program

In this note, we will see how to write, compile and run a java program. I will also cover java syntax, code conventions and several ways to run a java program.

First Java Program-

```
public class FirstJavaProgram {
  public static void main(String[] args){
    System.out.println("This is my first program in java");
  }
}
```

Output: This is my first program in java

How to compile and run the above program

Prerequisite- You need to have java installed on your system.

Step 1: Open a text editor, like Notepad or Notepad++ on windows and TextEdit on Mac. Copy the above program and paste it in the text editor.

Step 2: Save the file as FirstJavaProgram.java. We should always name the file same as the public class name. In our program, the public class name is FirstJavaProgram, that's why our file name should be FirstJavaProgram.java.

Step 3: In this step, we will compile the program. For this, open command prompt (cmd) on Windows, if you are Mac OS then open Terminal. To compile the program, type the following command and hit enter.

```
javac FirstJavaProgram.java
```

Step 4: After compilation the .java file gets translated into the .class file(byte code). Now we can run the program using following command and hit enter:

```
java FirstJavaProgram
```

Explanation of the First Java Program

Let's have a closer look at the program we have written above.

```
public class FirstJavaProgram {
```

This is the first line of our java program. Every java application must have at least one class definition that consists of class keyword followed by class name. However the class name can be anything.

I have made the class public by using public access modifier, I will cover access modifier later. All you need to know now that a java file can have any number of classes but it can have only one public class and the file name should be same as public class name.

```
public static void main(String[] args)  {
```

This is our next line in the program, lets break it down to understand it: public: This makes the main method public that means that we can call the method from outside the class, because JVM starts program from main method, which resides in other package.

static: We do not need to create object for static methods to run. They can run itself.

void: It does not return anything.

main: It is the method name. This is the entry point method from which the JVM can run your program.

(String[] args): Used for command line arguments that are passed as strings.

System.out.println("This is my first program in java"); System is predefined class available in java.lang package which is implicitly used in all the java program. Out is predefine object of system class and println() is the predefined method. This method prints the contents inside the double quotes into the console and inserts a newline after.