Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

Outline

1. Introduction to Java
2. History of Java
3. Features of Java
4. Applications of Java
5. Types of Java Application

Introduction to Java

- Java is a general purpose programming language.
- It is a High Level programming language.
- Java was originally developed by James Gosling at Sun Microsystems in 1995.
- Java is a language that is platform independent.
- A platform is the hardware and software environment in which a program runs.
- Java has its own Java Runtime Environment (JRE) and Application Programming Interface (API).

Introduction to Java (Cont…)

- Java code is once compiled, it can run on any platform without recompiling or any kind of modification.
  - "Write Once Run Anywhere"
- The aforementioned feature of Java is supported by Java Virtual Machine (JVM).
  - First, java code is complied into bytecode. This bytecode gets interpreted on different machines. Java bytecode is the resultant of the compilation of a java program, an intermediate representation of that program that is machine independent.

Introduction to Java (Cont…)

Java Virtual Machine

- Basically, there are three phases to execute a program:
  1. **Writing:** Writing of the program is the first phase.
  2. **Compilation:** Compilation of a program is done by javac compiler. This compiler is included in Java Development Kit (JDK). It takes a java program as input, and generates java bytecode as output.
  3. **Execution:** JVM executes the bytecode generated by compiler.
- Thus, the primary function of JVM is to execute the bytecode generated by compiler.
- Each Operating System (OS) has different JVM. However, the output they produce after execution of bytecode is same across all OSs. So, we call java as platform independent language.

Introduction to Java (Cont…)

Java Virtual Machine

- Implementation of JVM is based on stack.
- It is called a virtual machine because it doesn't physically exist.
- JVMs are available for many hardware and software platforms.
- JVM, Java Runtime Environment (JRE) and JDK are platform dependent as the configuration of each OS is different from each other. However, java is platform independent.
- JVM performs following operation:
  1. Loads code
  2. Verifies code
  3. Executes code
  4. Provides runtime environment

Introduction to Java (Cont…)

- **Java Virtual Machine (Cont…)**
  - **Architecture of JVM**

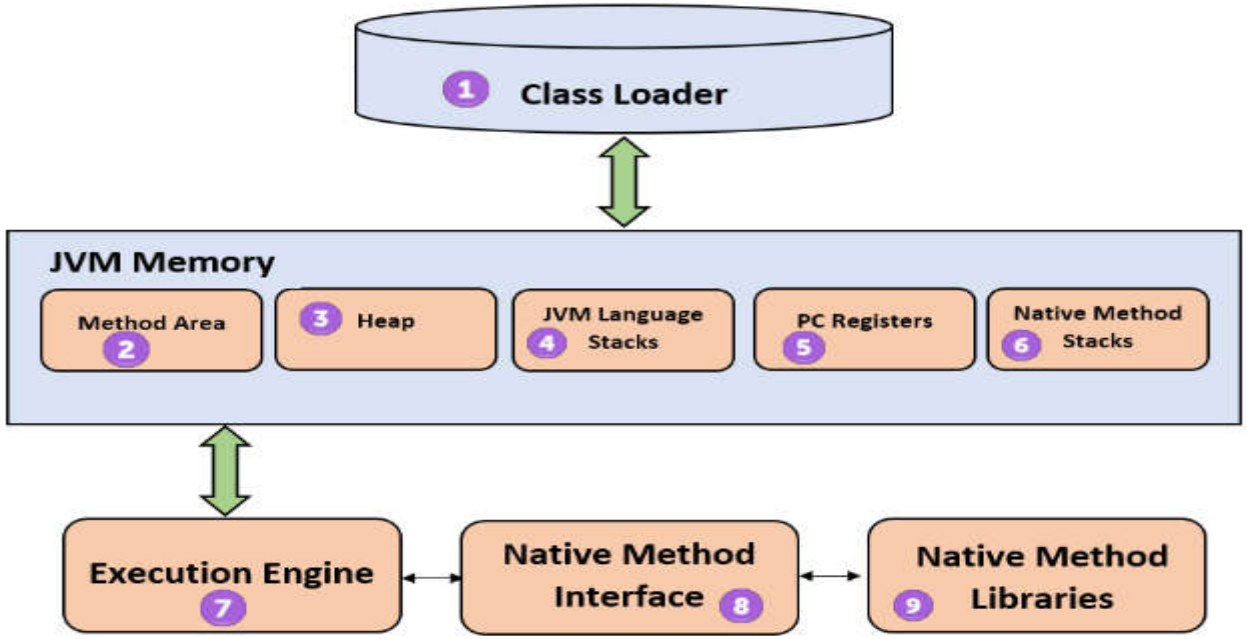

Fig. 1: JVM architecture

Introduction to Java (Cont…)

Java Virtual Machine (Cont…)

Architecture of JVM (Cont…)

1. **Classloader:** It is a subsystem of JVM, which is used to load class files. Whenever we run a java program, it is loaded first by the classloader. There are three built-in classloaders in java:
   1. **Bootstrap ClassLoader:** This is the first classloader, which is the super class of Extension classloader. It loads the rt.jar file, which contains all class files of Java Standard Edition.
   2. **Extension ClassLoader:** This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside $JAVA_HOME/jre/lib/ext directory.
   3. **System/Application ClassLoader:** It loads the classfiles from classpath. By default, classpath is set to the current directory.

Introduction to Java (Cont…)

Java Virtual Machine (Cont…)

Architecture of JVM (Cont…)

2. **Class or Method Area:** There is only one method area per JVM, and it is a shared resource. Here, all class level information like class name, immediate parent class name, methods and variables information, etc. are stored, including static variables.
3. **Heap:** Information of all objects is stored in the heap area. There is also one heap area per JVM. It is also a shared resource.
4. **Stack:** JVM creates one run-time stack for every thread, which is stored here. Each block of this stack is called activation record or stack frame that stores methods calls. All local variables of that method are stored in their corresponding frame. When a thread is terminated, its run-time stack is destroyed by JVM. It is not a shared resource.

Introduction to Java (Cont…)

Java Virtual Machine (Cont…)

Architecture of JVM (Cont…)

5. **Program Counter (PC) Register:** This register contains the address of the JVM instruction currently being executed. Obviously, each thread has separate PC registers.
6. **Native Method Stack:** It contains all the native methods.
7. **Java Native Interface (JNI):** It provides an interface to communicate with another application written in another language. It mainly interacts with the Native Method Libraries and provides the native libraries (C, C++) required for the execution. This framework sends output to the console or interact with OS library.
8. **Native Method Libraries:** It is a collection of the Native Libraries (C, C++), which are required by the Execution Engine.

Introduction to Java (Cont…)

Java Virtual Machine (Cont…)

Architecture of JVM (Cont…)

9. **Execution Engine:** It executes the bytecode. It reads the byte-code line by line, uses data and information present in various memory area and executes instructions. It can be classified into three parts:
   1. **Interpreter:** It interprets the bytecode line by line and then executes. The disadvantage here is that when one method is called multiple times, every time interpretation is required.
   2. **Just-In-Time (JIT) Compiler:** JIT compiler reduces the amount of time needed for compilation by compiling parts of the byte code that have similar functionality at the same time. Thus, it is used to improve the performance. Here, the term "compiler" refers to a translator from the instruction set of a JVM to the instruction set of a specific CPU.
   3. **Garbage Collector:** It destroys un-referenced objects.

Introduction to Java (Cont…)

- **Bytecode**
  - When we write a program in java, at first, the compiler compiles the program and a bytecode is generated.
  - If we wish to run this .class file on any other platform, we can do so.
  - After the first compilation, the generated bytecode is now run by the JVM and the processor is not in consideration (only java installation is required).
  - The bytecode is saved in a .class file by the compiler.

Introduction to Java (Cont…)

- **Bytecode (Cont…)**

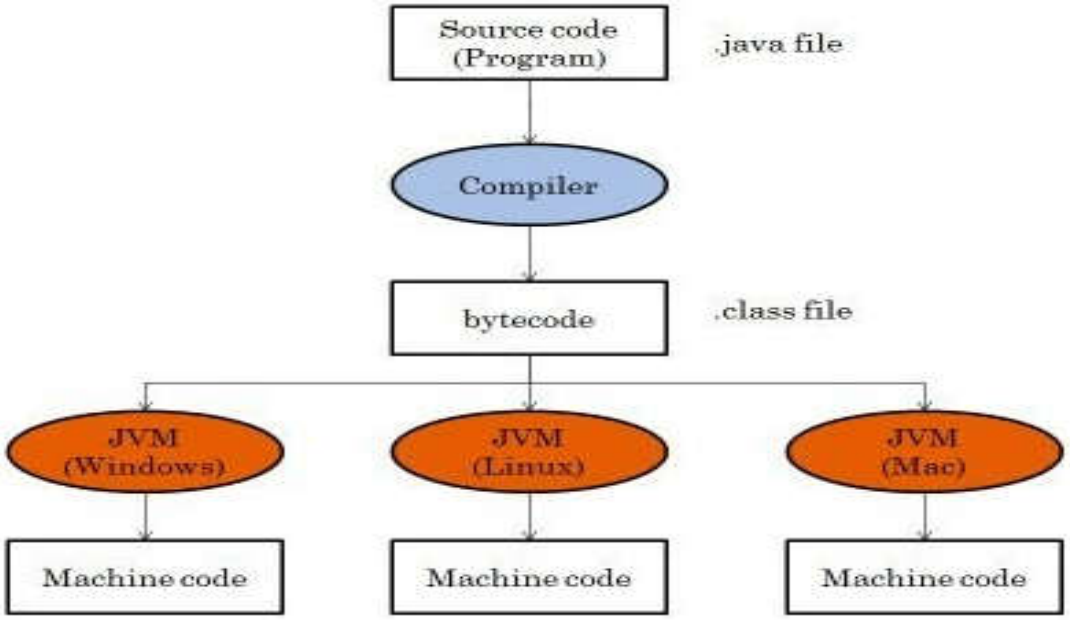

Fig. 2: Bytecode

Introduction to Java (Cont…)

- **Java Development Kit (JDK)**
  - As the name suggests, this is the complete java development kit that is used to develop java applications and applets.
  - It physically exists.
  - JDK contains a private JVM and a few other resources, such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a java application.
  - In order to create, compile and run a java program, we need JDK installed on our system.

Introduction to Java (Cont…)

Java Runtime Environment (JRE)

- JRE is a set of software tools, which are used for developing java applications by providing the runtime environment.
- It is the implementation of JVM.
- It physically exists.
- It contains a set of libraries and other files that JVM uses at runtime.
- JRE is a part of JDK, which means that JDK includes JRE.
- JRE includes JVM, browser plugins and applets support.
- When we have JRE installed on our system, we can run a java program. However, we cannot compile it.
- When we only need to run a java program on our computer, we only need JRE.

Introduction to Java (Cont…)

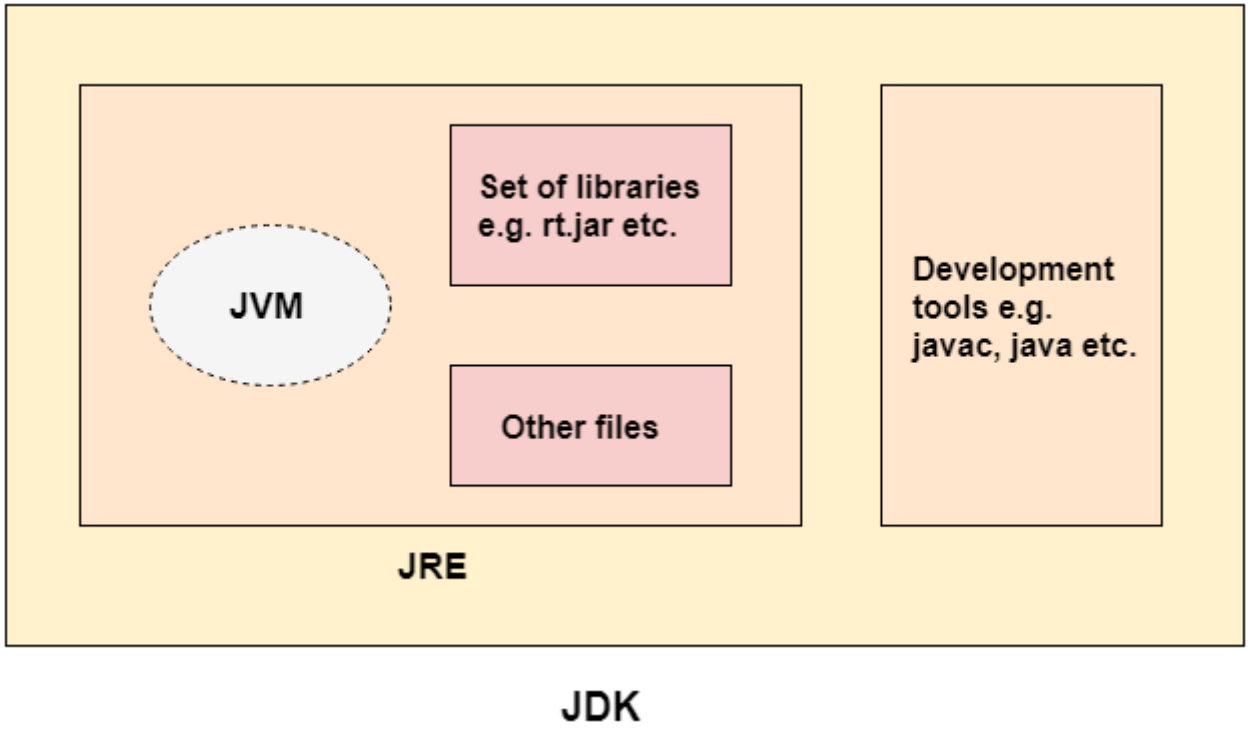

Fig. 3: Relationship among JDK, JRE and JVM

History of Java

- In June 1991, J. Gosling, M. Sheridan, and P. Naughton (Team is Green Team) initiated the java language project.
- At first, it was called “Greentalk” by J. Gosling and file extension was .gt.
- After that, it was called Oak and developed as a part of the Green project.
- In 1995, Oak was renamed as “Java”.
- On 23 Jan 1996, JDK 1.0 was released.
- On 13 Nov 2006, Sun released much of its JVM as free.
- On 6 May 2007, Sun made all its JVM’s core code available under open-source distribution terms.

Features of Java

- There are many features of Java:
  - Object-oriented
  - Platform independent
  - Secure
  - Robust
  - Portable
  - Dynamic
  - High performance
  - Multithread
  - Distribute

Applications of Java

- Approx. 3 Billion devices run java. Some of them are as follows:
  - Desktop applications
  - Web applications
  - Enterprise applications (banking)
  - Mobile
  - Embedded system
  - Smart card
  - Robotics
  - Games

Types of Java Application

- There are 4 type of java applications:
  - **Standalone Application:** An application that install on each system (AWT and Swing are used).
  - **Web Application:** An application that runs on the server side and creates dynamic page (servlet, jsp, etc. are used).
  - **Enterprise Application:** An application that is distributed in nature, such as banking applications. It has the advantage of high level security and load balancing.
  - **Mobile Application:** An application that is created for mobile devices (Android and Java ME are used).

thank you . . .