Programming in Java
CIC-212

Unit - 1

Overview and characteristics of Java, Java program compilation and Execution Process, Organization of the Java Virtual Machine, JVM as an interpreter and emulator, Instruction set, class file format, Verification, Class Area, Java stack, Heap, Garbage collection. Security Promises of the JVM, Security Architecture & security Policy. class loaders & security aspects, sandbox model.

Overview of Java

Java is an object-oriented programming language which was developed by James Gosling in 1991 at sun microsystems.

→ Earlier it was called as 'Oak'.
→ First version Java 1.1 was released in 1996.
→ In 2010, Oracle company acquired Java.
→ Java is a popular language - resides in mobile, client m/cs, server m/cs, embedded devices, smartphones, etc.
→ Java is easy, simple, robust, secure.

Java Features

- Compiled & Interpreted
- Platform - Independent & Portable
- Object - Oriented
- Robust & Secure
- Distributed
- Familiar Small & simple

- Multithreaded & Interactive
- High Performance
- Dynamic & Extensible

- Compiled & Interpreted

Java is both a compiled & an interpreted language. First, Java compiler translates source code into what is known as bytecode instructions. Bytecodes are not machine instructions & therefore, in the second stage, Java interpreter generates machine code that can be directly executed by the machine that is running the Java program.

- Platform Independent & Portable

Java programs can be easily moved from one computer systems to another, anywhere & anytime. Changes & upgrades in operating systems, processors & system resources will not force any changes in Java programs. This is the reason why Java has become a popular language for programming on Internet which interconnects different kinds of system worldwide.

Java ensures portability in two ways.

- Java compiler generates bytecode instructions that can be implemented on any machine.
- The size of the primitive data types are machine independent.

- Object-Oriented

Java is a true object-oriented language. All program code & data reside within objects & classes. Java comes with an extensive set of classes, arranged in packages, that we can use in our programs by inheritance.

- Robust & Secure:

Java is a robust language. It provides many safeguards to ensure reliable code. It has strict compile time & run-time checking for data types. It is designed as a garbage-collected language relieving the programmers virtually all memory management problems. Java also incorporates the concept of exception handling which captures serious errors & eliminates any risk of crashing the system.

Security becomes an important issue for a language that is used for programming on Internet. Java systems not only verify all memory access but also ensure that no viruses are communicated with an applet. The absence of pointers in Java ensures that programs cannot gain access to memory locations w/o proper authorization.

- Distributed

Java is designed as a distributed language for creating applications on networks. It has the ability to share both data & programs. Java applications can open & access remote objects on Internet as easily as they can do in a local system. This enables multiple programmers at multiple remote locations to collaborate & work together on a single project.

- Simple, Small & Familiar

Java is a small & simple language. Many redundant or sources of unreliable code are not part of Java. For ex. Java doesnot use pointers, preprocessor header files, goto statement, & many others. It also eliminates operator overloading & multiple inheritance.

Familiarity is another striking feature of Java. To make the language look familiar, it was modelled on C & C++. Java code "looks like a C++" code.

- Multithreaded & Interactive

multithreaded means handling multiple tasks simultaneously. Java supports multithreaded programs. This means that we need not to wait for the application to finish one task before beginning another. For ex- we can listen to an audio clip while scrolling a page & download applet at the same time. This feature greatly improves the interactive performance of graphical applications

- High Performance

Java speed is comparable to the native C/C++. Java architecture is also designed to reduce overheads during runtime. Further, the incorporation of multithreading enhances the overall execution speed of Java programs.

- Dynamic & Extensible

Java is a dynamic language. It is capable of dynamically linking in new class libraries, methods, and objects. Java programs support functions written in other languages such as C & C++. These functions are called native methods. Native functions are linked dynamically at runtime.

Java Program

We can develop two types of Java programs.

- Stand-alone applications
- Web applets.

Stand alone applications are programs written in Java to carry out certain tasks on a stand-alone local computer. Executing a stand-alone Java program involves two steps:

1. Compiling the source code into bytecodes using javac compiler.
2. Executing the bytecode program using java interpreter.

Applets are small Java programs developed for internet applications. Applets are embedded in an HTML document & run inside a web page.

Simple Java program

```
class SampleOne
{
    public static void main (String args[])
    {
        System.out.println ("Java is better");
    }
}
```

→ Class Declaration - Java is a true object oriented language, ∴ everything must be placed in a class.

→ Opening Brace

→ Main line - Java application program must include the main() method. This is the starting point for the interpreter to begin the execution of the program. A Java application can have any no. of classes but only one of them must include main method to initiate the execution.

public - Keyword declaring main() as unprotected & ∴ making it accessible to all other classes.

static - declares this method as one that belongs to the entire class & not a part of any object of the class.
The main must be always declared as static since the interpreter uses this method before any objects are created.

void - main method doesnot return any value.

→ Output line - The println method is a member of the out object, which is a static data member of System class. The method println always append a newline character to the end of the string.

Implementing a Java Program
- Creating the program
- Compiling the program
- Running the program

Create & save the program with class name. For ex. Test.java - This file is called source file

To compile the program, we must run java compiler javac.

javac Test.java

If everything is ok, the javac compiler creates a file called Test.class containing the bytecodes of the program. The compiler automatically names the bytecode file as

<classname>.class.

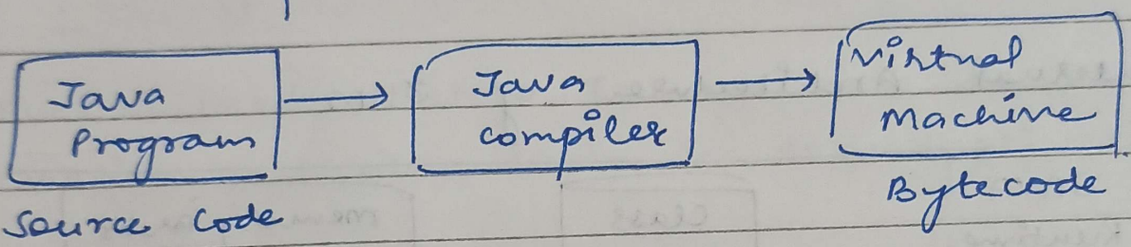
We need to use the Java interpreter to run a stand-alone program.

java Test

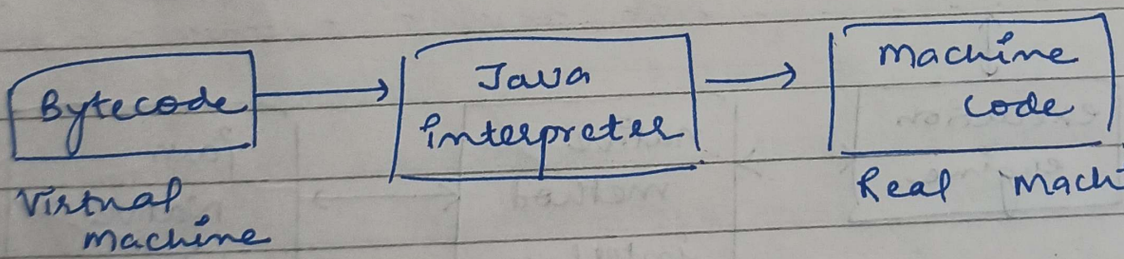
Now interpreter looks for the main method in the program & begins execution from there.

Java Virtual Machine

All language compilers translate source code into machine code for a specific computer. Java compiler produces an intermediate code known as bytecode for a machine does not exist. This machine is called the Java Virtual Machine & it exists only inside the computer memory. It is a simulated computer within the computer & does all major functions of a real computer.



The Virtual machine code is not machine specific. The machine specific code is generated by the Java interpreter by acting as an ~~interpr~~ intermediary between the Virtual machine & the real machine.



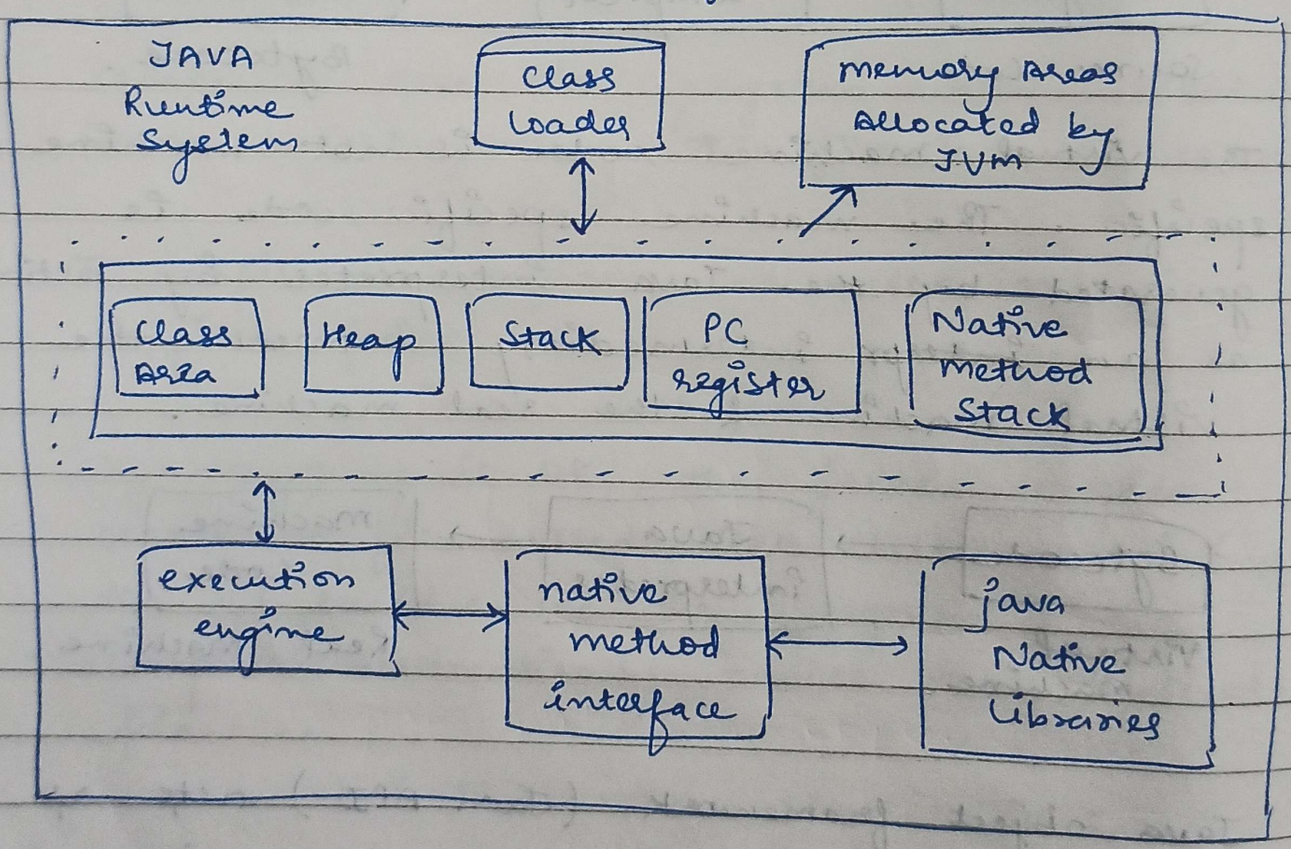
Java object framework (Java API) acts as the intermediary between the user programs & the Virtual machine which in turn acts as the intermediary b/w the operating system & the Java object framework.

JVM is a java virtual machine or a program that provides run-time environment in which java byte code can be executed. JVMs are available for many hardware & software platform.

The JVM performs following operation.

- loads code
- verifies code
- executes code
- provides runtime enviroment.

Internal Architecture of JVM



Class loader - It is a subsystem of JVM that is used to load class files.

Class Area - stores per-class structures such as the run-time constant pool, field & method data, the code for methods.

Heap - It is the runtime data area in which objects are allocated.

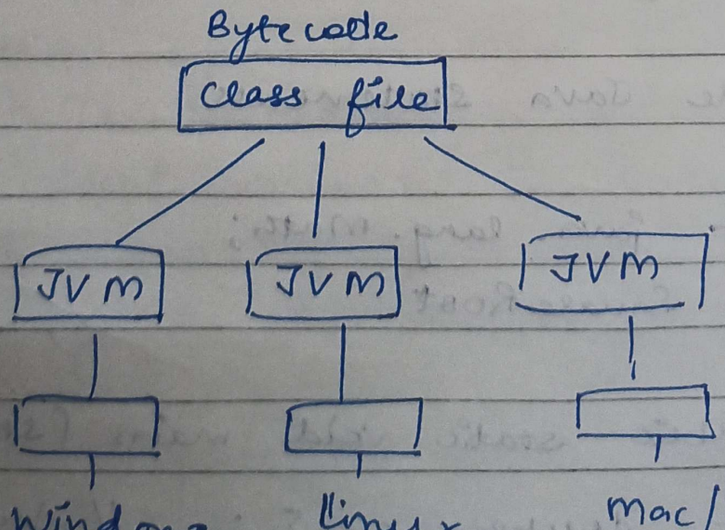
Stack - It holds local variables and partial results, & plays a part in method invocation & return.

Program Counter Register - It contains the address of the java virtual machine instruction currently being executed.

Native method stack - It contains all the native methods used in the application.

Execution Engine - It contains virtual processor, interpreter, Just-In-Time (JIT) compiles.

Bytecode - Java Bytecode is the instruction set of java virtual machine. Bytecode is the compiled format for java programs. Once a java program has been converted to bytecode, it can be transferred across a network and executed by JVM. Bytecode files generally have a .class extension.

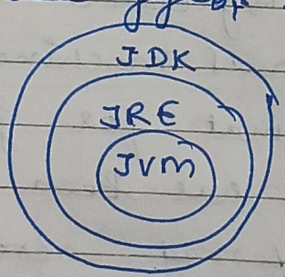


Difference between JVM, JDK & JRE

JVM - runs the bytecode & generates the machine code.

JRE - Java Runtime Environment.
- provides the libraries.
- JVM
- other components to run applets & app^n's.
- java plug-in, which enables applets to run in browsers.
- doesn't contain tools & utilities such as compilers or debuggers.

JDK - Java Development Kit.
- is superset of JRE & contains everyting that is in JRE + tools such as compilers, debuggers.



Java Program with multiple statements.

```
/*
 * More Java statements
 */
import java.lang.Math;
class SquareRoot
 {
   public static void main (string args[])
    { double x = 5 ;    // Declaration & initialization
      double y;         // simple declaration
      y = Math.Sqrt (x);
      System.out.println ("y = " + y);
 } }
```

Garbage collection in Java

Garbage collection in Java is the process by which Java programs perform automatic memory management. When Java program runs on the JVM, objects are created on the heap, which is portion of memory dedicated to the program. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects & deletes them to free up memory.

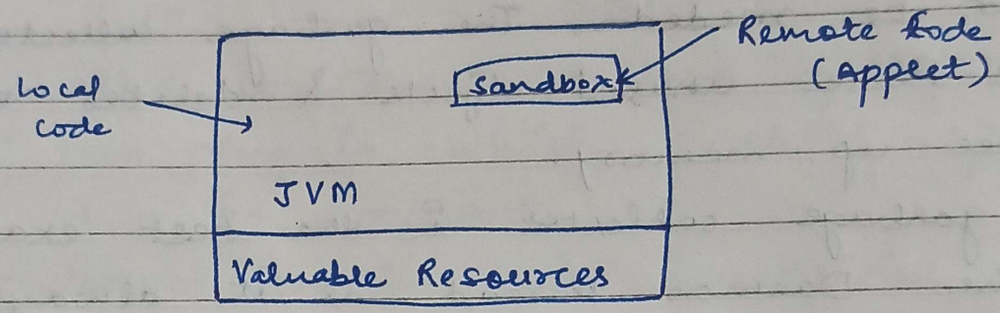
The garbage collector is the best example of the Daemon thread as it always running in the background.

Security Promises of the JVM

- Every object is constructed exactly once before it is used.
- Every object is an instance of exactly one class, which does not change through the life of the object.
- If a field or method is marked private, then the only code that ever accessed it is found within the class itself.
- Fields & methods marked protected are used only by code that participates in the implementation of the class.
- Every local variable is initialized before it is used.
- Every field is initialized before it is used.
- It is impossible to underflow or overflow the stack.
- It is impossible to read or write the end of an array or before the

beginning of the array.
- It is impossible to change the length of an array once it has been created.
- Final methods cannot be overridden & final classes cannot be subclassed.

Java Sandbox Model



- Security model of Java
- Private restricted environment
- Local code is trusted to have full access to vital system resources (such as file system)
- Downloaded remote code (Applet) is not trusted & can access only limited resources provided inside the sandbox.