Hours per Week : Theory - 4   **Credit: 3**

Objectives:
 To review Object Oriented Programming concepts.
 Learn features of Java programming
 To develop s kill in java programming.

Module I
Introduction to Java programming : Java technology; history; java as a new paradigm; features of java; Applications and applets (Simple examples); Java Development Kit , Java Language fundamentals : Building blocks; Data types; variable declarations; wrapper classes; Operators and assignment; control structures; arrays; strings; String buffer classes.

Module II
Java as an OOP Language: Defining classes; Modifiers; Packages; Interfaces.

Module III
Exception handling: Basics; handling exceptions in java; (Try, catch, finally, multiple catch, nested try, throw); Exception and inheritance; Throwing user defined exceptions; Advantages of exception handling.Multithreading: Overview; Creating threads; thread life cycle; Priorities and scheduling; synchronization; Thread groups; communication of threads; Sample programs.

Module IV
Files and I/O streams: Overview; Java I/O; file streams; FileInputStram and FileOutputStream; Filter Streams; RandomAccessFile; Serialization.Applets : Introduction; Application vs. applets; Applet lifecycle; Working with Applets; The HTML APPLET tag; the java.Applet Package; Sample programs.

Module V
The Abstract Window Toolkit:- Basic classes in AWT; Drawing with Graphics class; Class hierarchy; Event handling; AWT controls (Labels, Buttons, checkbox, radio buttons; choice control; list, textbox, scroll bars); Layout Managers. The menu component hierarchy; Creating menus ; Handling events from menu items ; Enabling keyboard operation ; Bringing up a popup menu ; Customizing menu layout; The Menu API

Text book :
1. Object Oriented Programming through JAVA, Radha Krishna, University Press.
Reference:
    1.   Programming with java: A primer, 3rd Edn; E. Balaguruswami;

Introduction

Java is an object oriented programming language developed at Sun Microsystems .It is easier to learn than many other OOP languages, since it has collected the best parts of existing ones. A language that is standardized enough so that executable applications can run on any computer that contains a virtual machine .Virtual machine can be embedded in web browsers and operating systems. It contains a standard sets of class libraries that support creating Graphical User Interface, controlling multimedia data and communicating over networks. It supports the most recent features in computer science at the programming language level.

History of Java

Java was conceived by James Gosling ,Patrick Naughton,Ed Frank and Mike Sheridan at Sun Microsytems,Inc in 1991.It took 18 months to develop the first working version.This was initially called "Oak" but was renamed " Java" in 1995.The primary motivation was a platform independent language that could be used to create a software to be embedded in various consumre electronic devices such as micro wave ovesns,toasters ,remote controllers etc.

Features of Java

The primary objective of Java programming language creation was to make it portable, simple and secure programming language. Apart from this, there are also some excellent features which play an important role in the popularity of this language. The features of Java are also known as java *buzzwords*.

A list of most important features of Java language is given below.

1. Simple
2. Object-Oriented
3. Portable
4. Platform independent
5. Secured
6. Robust
7. Architecture neutral
8. Interpreted
9. High Performance
10. Multithreaded
11. Distributed
12. Dynamic

Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand. According to Sun, Java language is a simple programming language because:

- Java syntax is based on C++ (so easier for programmers to learn it after C++).
- Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.
- There is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java.

Object-oriented

Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.

Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.

Basic concepts of OOPs are:

1. Object
2. Class
3. Inheritance
4. Polymorphism
5. Abstraction
6. Encapsulation

Platform Independent

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language. A platform is the hardware or software environment in which a program runs.

There are two types of platforms software-based and hardware-based. Java provides a software-based platform.

The Java platform differs from most other platforms in the sense that it is a software-based platform that runs on the top of other hardware-based platforms. It has two components:

1. Runtime Environment
2. API(Application Programming Interface)

Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a

platform-independent code because it can be run on multiple platforms, i.e., Write Once and Run Anywhere(WORA).
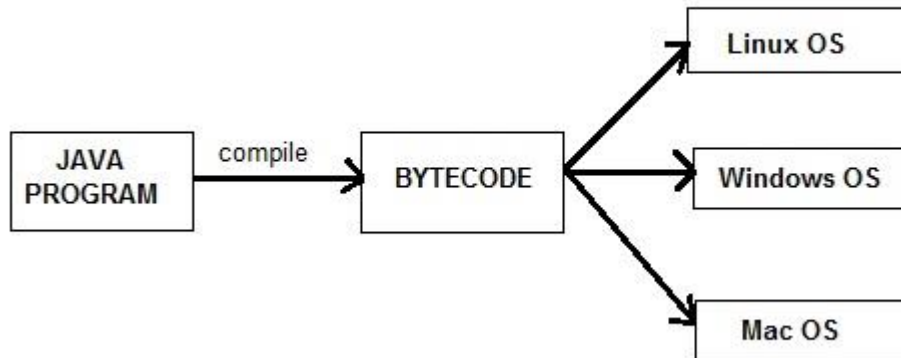
Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

- **No explicit pointer**
- **Java Programs run inside a virtual machine sandbox**

- **Classloader:**Classloader in Java is a part of the Java Runtime Environment(JRE) which is used to load Java classes into the Java Virtual Machine dynamically. It adds security by separating the package for the classes of the local file system from those that are imported from network sources
- **Byte code Verifier:** It checks the code fragments for illegal code that can violate access right to objects.
- **Security Manager:** It determines what resources a class can access such as reading and writing to the local disk.

Java language provides these securities by default. Some security can also be provided by an application developer explicitly through SSL, JAAS, Cryptography, etc.

Robust

Robust simply means strong. Java is robust because:

- It uses strong memory management.
- There is a lack of pointers that avoids security problems.

- There is automatic garbage collection in java which runs on the Java Virtual Machine to get rid of objects which are not being used by a Java application anymore.
- There are exception handling and the type checking mechanism in Java. All these points make Java robust.

Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

In C programming, int data type occupies 2 bytes of memory for 32-bit architecture and 4 bytes of memory for 64-bit architecture. However, it occupies 4 bytes of memory for both 32 and 64-bit architectures in Java.

Portable

Java is portable because it facilitates you to carry the Java bytecode to any platform. It doesn't require any implementation.

High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++). Java is an interpreted language that is why it is slower than compiled languages, e.g., C, C++, etc.

Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

Multi-threaded

A thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

Dynamic

Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Java supports dynamic compilation and automatic memory management (garbage collection).

Comparison of C++ & Java

	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming. It was an extension of C programming language.	Java was designed and created as an interpreter for printing systems but later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in java.
Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread	Java has built-in thread support.

	support.	
Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.
Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

The JDK contains a private Java Virtual Machine (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.

JVM

VM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent. There are three notions of the JVM: *specification*, *implementation*, and *instance*.

The JVM performs the following main tasks:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JVM Architecture
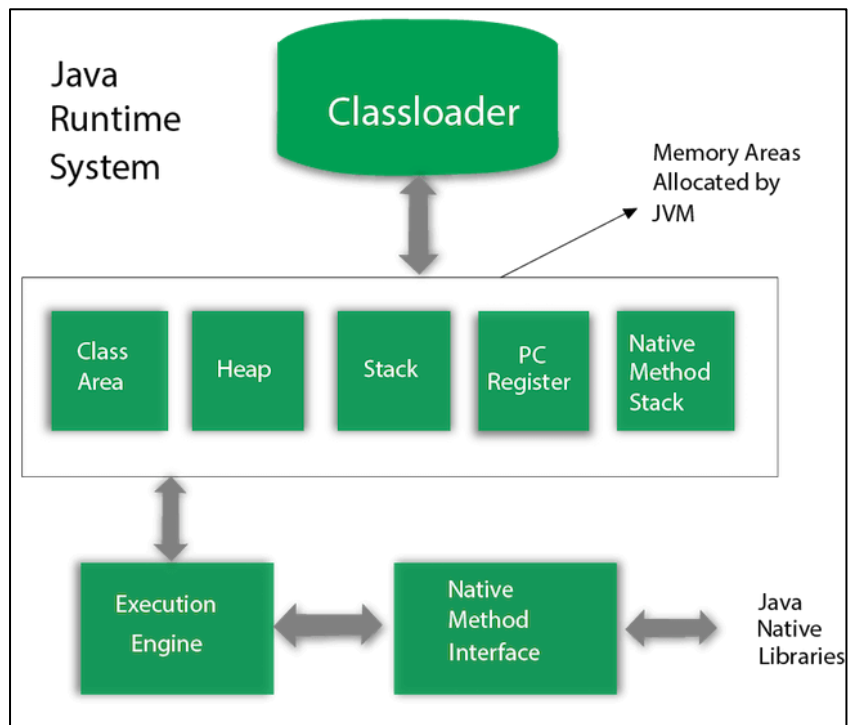
1) Classloader

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader.

2) Class(Method) Area

Class(Method) Area stores per-class structures such as the runtime constant pool, field and method data, the code for methods.

3) Heap

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

4) Stack

Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return.

Each thread has a private JVM stack, created at the same time as thread.

A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.

5) Program Counter Register

PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.

6) Native Method Stack

It contains all the native methods used in the application.

### 7) Execution Engine

It contains:

1. **A virtual processor**
2. **Interpreter:** Read bytecode stream then execute the instructions.
3. **Just-In-Time(JIT) 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 needed for compilation. Here, the term "compiler" refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

8) Java Native Interface

Java Native Interface (JNI) is a framework which provides an interface to communicate with another application written in another language like C, C++, Assembly etc. Java uses JNI framework to send output to the Console or interact with OS libraries.

Bytecode

Bytecode is the compiled format for <u>Java</u> programs. Once a Java program has been converted to bytecode, it can be transferred across a network and executed by <u>Java Virtual Machine (JVM)</u>. Bytecode files generally have a *.class* <u>extension</u>.
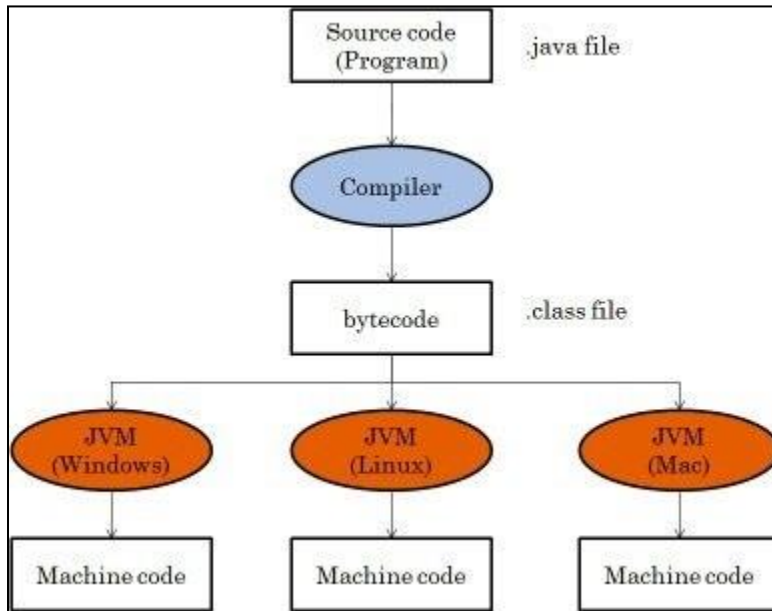
Difference between

String	StringBuffer
The length of String object is fixed	length of an object of StringBuffer can be increased when required
String object is immutable i.e. it's object can't be reassigned	object of StringBuffer is mutable
String object is slower in performance	StringBuffer object is faster
String object consumes more memory	StringBuffer objects consumes less memory
String objects are stored in a constant pool	StringBuffer objects are stored on heap memory.

Java application v/s Java Applet

	APPLET	APPLICATION
	It is small program uses another application	An application is the programs executed on

	program for its execution	the computer independently.
	Do not use the main method	Uses the main method for execution
	Cannot run independently require API's (Ex. Web API).	Can run alone but require JRE
	The files cannot be read and write on the local computer through applet	Applications are capable of performing those operations to the files on the local computer
	Cannot communicate with other servers.	Can communicate with other servers.
	Cannot access files residing on the local computer.	Can access files residing on the local computer.