UNIT I – BASIC CONCEPTS **9**

Object Oriented Programming - Benefits of OOP – Applications of OOP. Java fundamentals: Features of java – Java development environment – Bytecode - Data types- Variables -Operators – Expressions – Functions – Static Members – Arrays – Strings – Classes and objects – Constructing objects using constructors.

JAVA DEVELOPMENT ENVIRONMENT

Java environment includes a large number of development tools and hundreds of classes and methods. The development tools are part of the system known as Java Development Kit (JSL), also known as the Application Programming Interface (API).

Java Development Kit

The Java Development Kit comes with a collection of tools that are used for developing and running Java programs. They include:

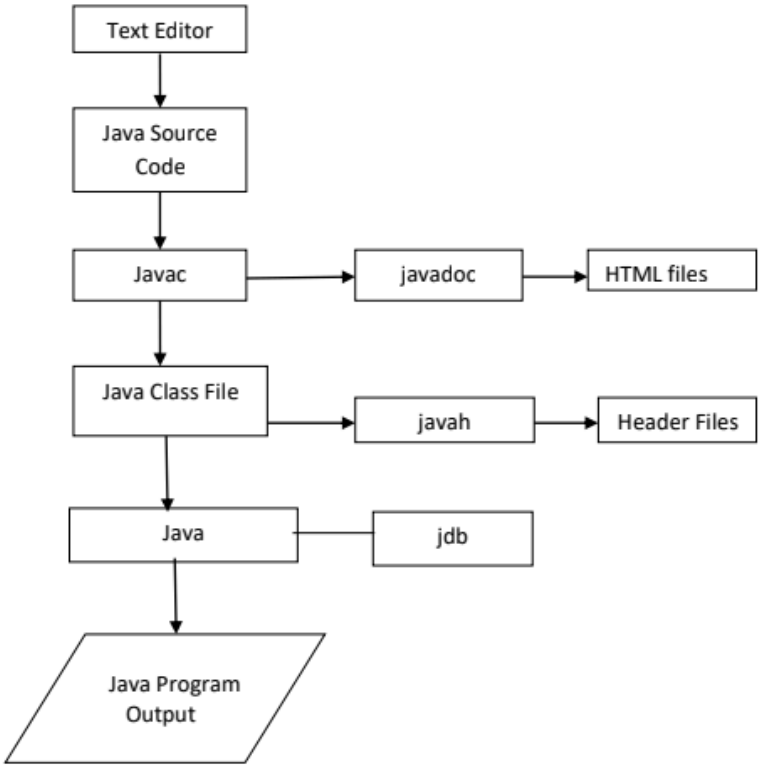
Tool	Java Development Tools Description
Appletviewer (for viewing Java applets)	Enables us to run Java applets(without actually using a Java-compatible browser).
Java (Java interpreter)	Java interpreter, which runs applets and applications by reading and interpreting byte code files.
Javac (Java compiler)	The Java compiler, which translates Java sourcecode to bytecode files that the interpreter can understand.
Javadoc (for creating HTML documents)	Creates HTML format documentation from Java source code files.
Javah (for C header files)	Produces header files for use with native methods.
Javap (Java disassembler)	Java disassembler, which enables us to convert bytecode files into a program description.
Jdb (Java debugger)	Java debugger, which helps us to find errors in our programs.

The way these tools are applied to build and run application programs is illustrated in the figure. To create a Java program, we need to create a source code file using a text editor. The source code is then compiled using the Java compiler javac and executed using the Java interpreter java. The Java debugger jdb is used to find errors, if any, in the source code. A compiled Java program can be converted into a source code with the help of Java disassembler javap.

Application Programming Interface

The Java Standard Library (or API) includes hundreds of classes and methods grouped into several functional packages. Most commonly used packages are:

- Language Support Package: A collection of classes and methods required for implementing basic features of Java.
- Utilities Package: A collection of classes to provide utility functions such as date and time functions.
- Input/Output Package: A collection of classes required for input/output manipulation.
- Networking Package: A collection of classes for communicating with other computers via Internet
- AWT PACKAGE: The Abstract Window Tool Kit package contains classes that implement platform-independent graphical user interface.
- Applet Package: This includes a set of classes that allows us to create Java applets.



Java Byte Code:

Java bytecode is the instruction set of the Java virtual machine (JVM), the language to which Java and other JVM-compatible source code is compiled. Each instruction is represented by a single byte, hence the name bytecode, making it a compact form of data.

Due to the nature of bytecode, a Java bytecode program is runnable on any machine with a compatible JVM; without the lengthy process of compiling from source code.

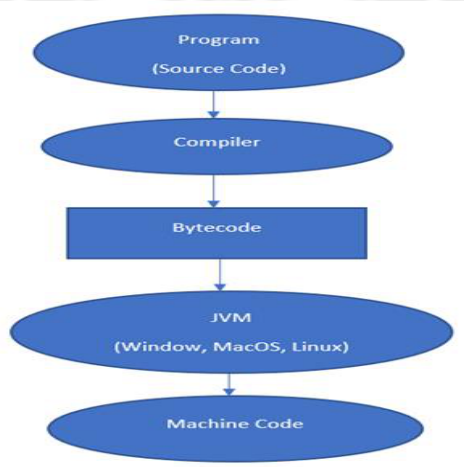
Java bytecode is used at runtime either interpreted by a JVM or compiled to machine code via just-in-time (JIT) compilation and run as a native application.

As Java bytecode is designed for a cross-platform compatibility and security, a Java bytecode application tends to run consistently across various hardware and software configurations.

Byte Code can be defined as an intermediate code generated by the compiler after the compilation of source code(JAVA Program). This intermediate code makes Java a platform-independent language.

How is Byte Code generated?

Compiler converts the source code or the Java program into the Byte Code(or machine code), and secondly, the Interpreter executes the byte code on the system. The Interpreter can also be called JVM(Java Virtual Machine). The byte code is the common piece between the compiler(which creates it) and the Interpreter (which runs it).



Advantage of Java Bytecode

Platform independence is one of the soul reasons for which James Gosling started the formation of java and it is this implementation of bytecode which helps us to achieve this. Hence bytecode is a very important component of any java program.The set of instructions for the JVM may differ from system to system but all can interpret the

bytecode. A point to keep in mind is that bytecodes are non-runnable codes and rely on the availability of an interpreter to execute and thus the JVM comes into play.

Bytecode is essentially the machine level language which runs on the Java Virtual Machine. Whenever a class is loaded, it gets a stream of bytecode per method of the class. Whenever that method is called during the execution of a program, the bytecode for that method gets invoked.Javac not only compiles the program but also generates the bytecode for the program. Thus, we have realized that the bytecode implementation makes Java a platform-independent language. This helps to add portability to Java which is lacking in languages like C or C++. Portability ensures that Java can be implemented on a wide array of platforms like desktops, mobile devices, severs and many more. Supporting this, Sun Microsystems captioned JAVA as "write once, read anywhere" or "WORA" in resonance to the bytecode interpretation.

GENERAL STRUCTURE OF JAVA PROGRAM:

Documentation Section
Package statement # optional
import statement # optional
interface statement # optional
class definition # optional Main Method Class
{
    Main Method Definition € Essential
}

Documentation section:

Documentation section comprises of comment lines giving the name of the program. Java uses a comment style as /**.....*/ known as documentation section

Package statement:

The first statement allowed in Java file is package statement. The statement declares a package name and informs the compiler that the classes defined here belong to the package.

Package name;
Where package keyword for implementing packages
Package name - name of the package used in program where the classes resides
Package student;
Where student is the name of the package used in program

Import statement:

This import statement in Java is similar to the #include statements in C.

Syntax:

import packagename.classname

where package name is the name of the package used in the program. Classname is the name of the class inside the package name used in the program.

Example:

import student.test

Here student is the package name used in the program test is the class name present in package name student. The above statement instructs the interpreter to load test class contained in the package student.

Interface Statements:

The interface is like a class but includes a group of method declarations. This is used in the program when multiple inheritances are done in the program.

Class definitions:

Java programs may contain many class definitions. Classes are the primary and essential elements of a Java program. Classes with main method definition is essential and other classes used are optional

Main method class:

Since every Java program requires the main method as its starting point, this class is an essential part of the Java program. Simple Java programs may have only this method class only. The main method class creates objects of various classes and establishes communication between them.

Implementing a Simple Java Program:

Implementation of a Java program has three steps .They are given as follows.

- Creating a program
- Compiling a program
- Running a program

Creating a program:

We are going to type the Java program in the notepad and save the program with the file name as “classname.java”.

```
class Sample
{
	public static void main(String args[ ])
	{
		System.out.println(“Hello world”);
	}
}
```

The above program should be saved in the name of "simple.java" because simple is the classname.Java program should be saved in the name of classname.java. This file is called as source file.File name should be class name containing main method.

Program Explanation:

Class declaration:

The first line class Sample declares a class. Java is object oriented language and everything (including main function) should be placed inside the class.

Opening brace:

Every class definition in Java begins with an open brace"{" and ends with a matching closing brace " } ".

Main function:

Main function is the third line in the program.

(i.e) public static void main(String args[ ])

Defines the main method. Java applications can have any number of classes but we have only one main function inside a program. The main function line has many keywords .They are explained as follows

- public → The keyword public is access specifier that declares the main method. It is accessible to all the classes
- static → Statement declares this method as one that belongs to the entire class and not a part of any objects of the class. The main must always be declared as static since the interpreter uses this method before any objects are created
- void → the type modifier void stands for main method does not return any value
- String args[ ] → here string args[ ] declares a parameter named args which contains an array of objects of class type string

Output Line:

The only executable line in the program is

System.out.println("hai");

This is similar to the printf statement in C,The output displayed is

Hai

Executable statement in Java ends with a semicolon.

Compiling a Java program:

To compile the program we must run the Java complier javac, with the name of the source file on the command line as shown below

Syntax javac filename.java

Example javac simple.java

Here simple is the class name.If every statement is correct the compiler creates a file with extension of "filename.class" . ".class file" has byte code of the program.

Running a Java program:

We need a Java interpreter to run a standalone program. In the command prompt with the syntax "java classname " .Now the interpreter looks for the main method in the program and starts executing from there.Finally after successful interpretation it displays output

---