**INTRODUCTION:**

Java is a general-purpose, object-oriented programming language. We can develop two types of programs in Java:

* + Standalone applications
  + Web applets
  + Java Servlets

They are implemented as shown in Fig. 3.1.

1. Standalone applications are programs written in Java to carry out certain tasks on a standalone local computer. In fact, Java can be used to develop programs for all kinds of applications, which earlier, were developed using languages like C and C++. As pointed out earlier, HotJava itself is a Java application program. Executing a standalone Java program involves two steps:

* Compiling source code into bytecode using javac compiler.
* Executing the bytecode program using java interpreter.

1. Applets are small Java programs developed for Internet applications. An applet located on a distant computer (Server) can be downloaded via Internet and executed on a local computer (Client) using a Java-capable browser. We can develop applets for doing everything from simple animated graphics to complex games and utilities.

Applets are embedded in an HTML (Hypertext Markup Language) document and run inside a Web page.

1. Java Servlet programs run in computers that provide web services. They are also often called server side programs or servlets.

Standalone programs can read and write files and perform certain operations that applets cannot do. An applet can only run within a Web browser.

**Simple Java Program:**

We begin with a very simple program that prints a line of text as output.

class SampleOne

{

public static void main(String args[])

{

System.out.println(“Java is better than C++. “);

}

}

**Class Declaration:**

The first Line ***class SampleOne*** declares a class. As stated earlier, Java is a true object-oriented language and therefore, everything must be placed inside a class.class is a keyword and declares that a new class definition. SampleOne is a Java identifier that specifies the name of the class to be defined

**Opening Brace:**

Every class definition in Java begins with an opening brace "{" and ends with a matching closing brace "}", appearing in the last line in the example. This is similar to C++ class construct. (Note that a class definition in C++ ends with a semicolon.)

**The Main Line:**

public static void main (String args[])

defines a method named main. Conceptually, this is similar to the main( ) function in C/C++. Every 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 number of classes but only one of them must include a main method to initiate the execution. (Note that Java applets will not use the main method at all)

Public: The keyword public is an access specifier that declares the main method as unprotected and therefore making it accessible to all other classes.

Since, main has to be called by a code outside the class, from the JVM, therefore it must be declared public. The opposite of public is private. A Member declared private can only be accesses by another member of the same class.

Static: Next appears the keyword static, which 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 states that the main method does not return any value back to the caller, i.e JVM (but simply prints some text to the screen.)

All parameters to a method are declared inside a pair of parentheses. Here, String args[] declares a parameter named args, which contains an array of objects of the class type String. You can use any other name in place of args, provided it is not a keyword of java.

**The Output Line:**

The only executable statement in the program is

System.out.println ("Java is better than C++.");

Since Java is a true object oriented language, every method must be part of an object. The println method is a member of the out object, which is a static data member of System class. This line prints the string

Java is better than C++.

to the screen. The method println always appends a newline character to the end of the string. This means that any subsequent output will start on a new line. Note the semicolon at the end of the statement. Every Java statement must end with a semicolon.

The input and output in Java are controlled by streams. The class System of the package java.lang provides access to input and output streams. Therefore System.out refers to the object of PrintStream Class which has two methods- print and println().

More of Java

Assume that we would like to compute and print the square root of the number.

|  |  |
| --- | --- |
|  | A Java program with multiple ts |
| Program 3.2 |
|  |

import java.lang.Math;

class SquareRoot

{

public static void main (String arqs[ ])

{  
double x = 5;

double y;

y = Math.sqrt (x);

System.out.println("y= " + y);

}

}

The structure of the program is similar to the previous one except that it has more number of statements. The statement

double x = 5;

declares a variable x and initializes it to the value 5 and the statement

double y;

merely declares a variable y. Note that both of them have been declared as double type variables.

The statement

y = Math. sqrt (x) ;

invokes the method sqrt of the Math class, computes square root of x and then assigns the result to the variable y. The output statement

System.out.println("y= “+ y) ;

displays the result on the screen as

y = 2.23607

Note the use of + symbol. Here, the + acts as the concatenation operator of two strings. The value of y is converted into a string representation before concatenation)

**Use of Math Functions:**

Import java.lang.Math;

Instructs the interpreter to load the Math class from the package lang.Math class contains the sqrt method.

**JAVA PROGRAM STRUCTURE**

A Java program may contain many classes of which only one class defines a main method. Classes contain data members and methods that operate on the data members of the class. Methods may contain data type declarations and executable statements. To write a Java program, we first define classes and then put them together. A Java program may contain one or more sections as shown in Fig. 3.2.

**Documentation Section**

The documentation section comprises a set of comment lines giving the name of the program, the author and other details, which the programmer would like to refer to at a later stage. This would greatly help in maintaining the program. In addition to the two styles of comments discussed earlier, Java also uses a third style of comment /\*\*....\*/ known as documentation comment. This form of comment is used for generating documentation automatically.

**Package Statement**

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

Example:

package student;

The package statement is optional. That.is, our classes do not have to a part of package.

**Import Statements**

The next thing after a package statement but before any class definitions) may he a number of import statements.

import student.test;

This statement instructs the interpreter to load the test class contained in the package student.

**Interface Statements**

An interface is like a class but includes a group of method declarations. This is also an optional section and is used only when we wish to implement the multiple inheritance features in the program.

**Class Definitions**

A Java program may contain multiple class definitions. Classes are the primary and essential elements of a Java program. These classes are used to map the objects of real-world problems. The number of classes used depends on the complexity of the problem.

**Main Method Class**

Since every Java standalone program requires a main method as its starting point, this class is the essential part of a Java program. A simple Java-program may contain only this part. The main method creates objects of various classes and establishes communications between them.