

1.1 Basic Concepts of Object Oriented Programming

Q. What are the features of (Object of Oriented Programming) OOP ?

W-12

Q. State any four features of Java.

S-13, W-13

The concepts of OOP are as follows :

- | | |
|--------------------|----------------------|
| • Objects | • Classes |
| • Data abstraction | • Data encapsulation |
| • Inheritance | • Polymorphism |
| • Dynamic binding | • Message passing |

1.1.1 Objects

- Objects are basic run time entities in object oriented system. They may represent a place, a bank account, a table of data, or any item that the program must handle. They may represent user defined data such as time, limit.
- In C++, the problem is explored in terms of objects and the type of interaction between them. When program is being executed, the object interacts with each other by passing messages to one another.
- For example, if "customer" and "account" are two objects in a program, then customer object can send a message to the account object requesting for the blank balance.
- Each object holds data and related code to manipulate the related data. Objects can communicate with each other without knowing the details of each other's data or code.
- Only the following information is required :
 - The type of message accepted
 - The type of response returned by the objects.
- Fig. 1.1.1 shows two notations that are popularly used in object oriented analysis and design.

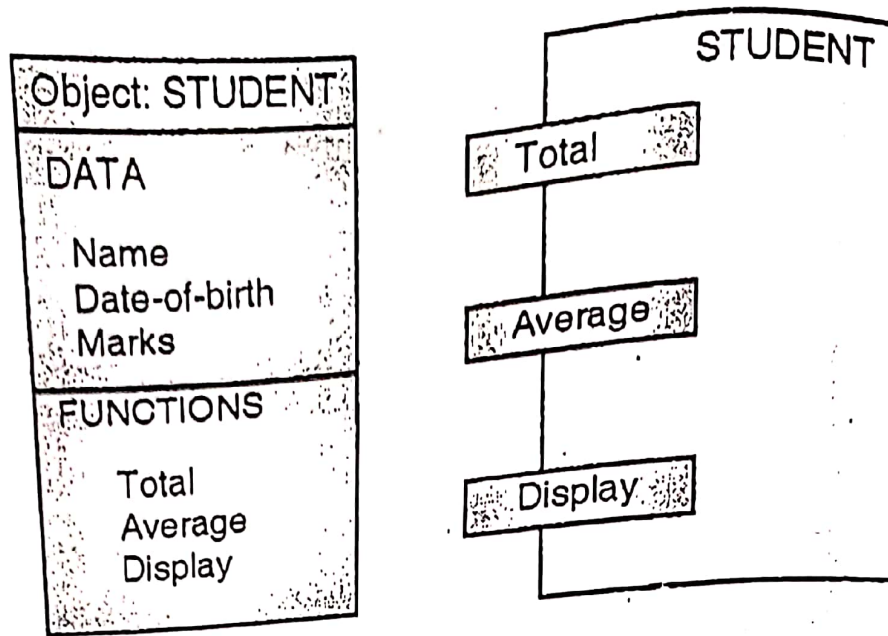


Fig. 1.1.1 : Two ways of representing an object

1.1.2 Classes

- A class is a group of objects with similar properties (attributes), common behaviour (operations), common relationship to other objects and common semantics. The entire set of data and code of an object can be made a user defined data type with the help of a class.
- The objects are variable of type class. A class is a collection of objects of similar type of objects. Classes are user defined data types and behave like the build in types of a programming language.
- Once the user defined data type class has been defined, any number of objects of that class can be created. Every object is always related with the data of type class from which they are created.
- For example, rose, jasmine and lotus are members of the class flower. As the object is a variable of class that is user defined data type, syntax used to create an object is similar with the syntax of an integer objects in C. If flower has been defined as a class, then the statement flower rose will create an object rose belonging to the class flower.
- The objects derive their individuality from the difference in their attribute values and relationships to other objects.

1.1.3 Data Abstraction and Encapsulation

Q. Explain : Data encapsulation features of Java

S-11

- **Data Abstraction** means representing the essential features and excluding the background details or explanations.

Classes are defined with the attributes or characteristics and the functions that operates on it, therefore it use the concept of abstraction. Classes encapsulate all the required and essential properties of the objects. As the classes uses the concepts of data abstraction, they known as Abstract Data Types (ADT).

- **Data encapsulation** is one of the salient features of C++, which means that data and functions are wrapped into a single unit called class. The data which is declared as private cannot directly accessible to the outside of the class and only the functions which are wrapped in the class can access it.
- Therefore, these functions provide the interface between the object's data and the program.
- Such kind of insulation of the data from direct access by the program is called data hiding.

1.1.4 Inheritance

Q. What are the different forms of inheritance ?

S-12

- **Inheritance** is the process by which objects of one class can acquire the properties of objects of another class.

Inheritance means one class of objects inherits the data and behaviour from another class. Inheritance supports the hierarchical classification in which a class inherits from its parents.

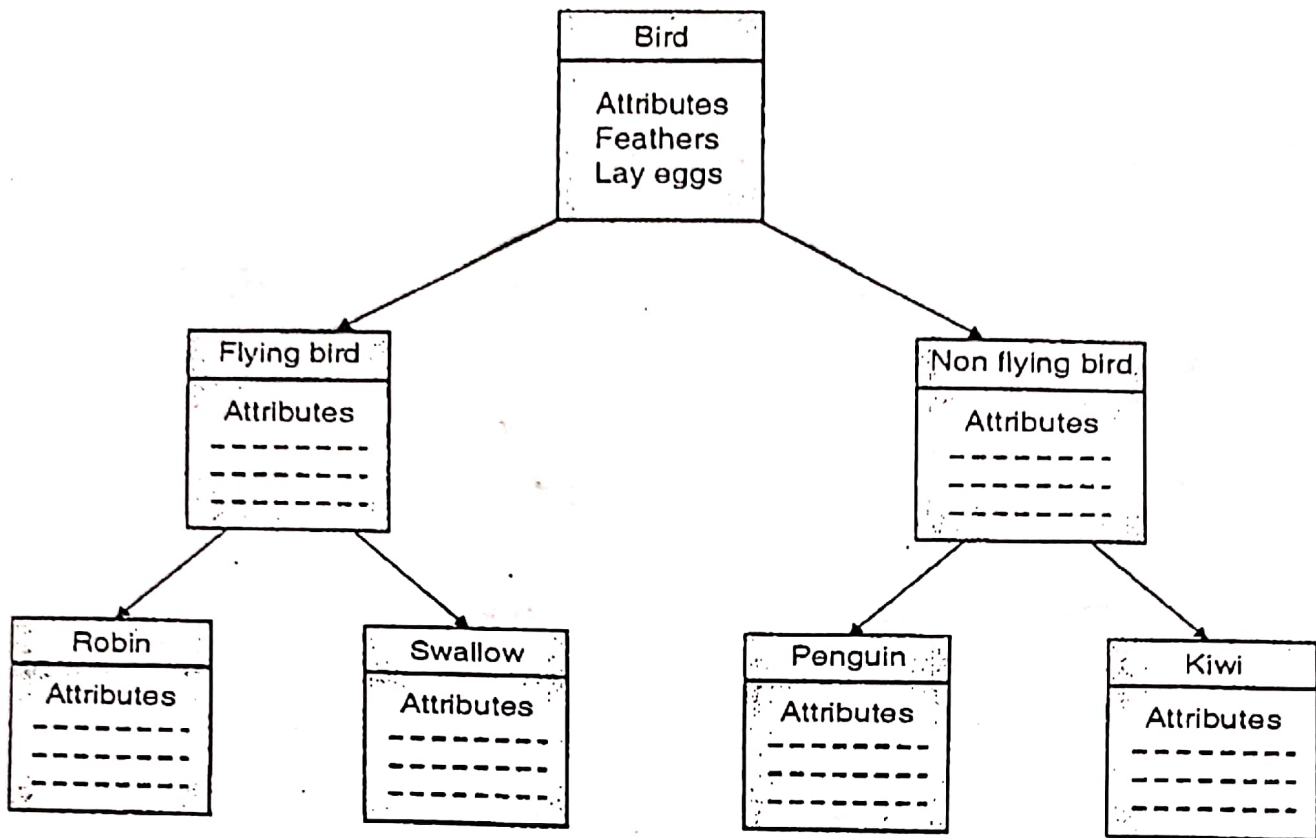


Fig. 1.1.2 : Property inheritance

- Inheritance provides the important feature of OOP that is reusability. This means that we can add additional features to an existing class without modification. This is possible by deriving a new class from existing one.
- The old class is referred as base class and the new one is called the derived class. The new class has combined features of both the classes.
- For example, the bird robin is a part of the class flying bird which is again part of the class bird. As shown in Fig. 1.1.2, the principle behind this sort of division is that each derived class shares common characteristics with the class from which it is derived.

1.1.5 Polymorphism

- Polymorphism is one of the significant OOP concepts. Polymorphism refers to the capability to take more than one form.

- An operation execution may be different in different instances. The execution depends upon the types of data used in the operation.
- For example, in addition operation if two numbers are used then sum will be produced. If the string operands are used, then the operation will be string concatenation.
- Fig. 1.1.3 shows that a single function name shape() is used to handle different types of arguments like box, triangle and circle.
- If radius and starting coordinates are the arguments of draw() function then it will draw circle. If arguments are the four coordinates to draw() function then it will draw box. If arguments are the three coordinates to draw() function then it will draw triangle.

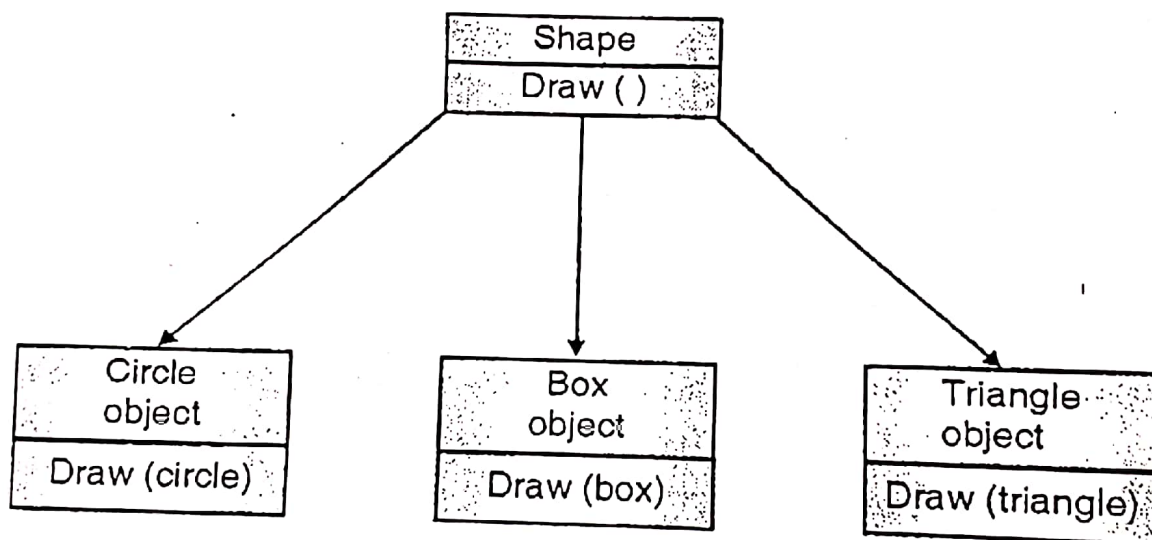


Fig. 1.1.3 : Polymorphism

- Polymorphism plays an important role in allowing objects having different internal structures to share the same internal interface.
- This means that a general class of operations may be accessed in the same manner even though specific actions associated with each operation may differ. Polymorphism is extensively used in implementing inheritance.

1.1.6 Dynamic Binding

- Binding is act of linking of a procedure or function call to the code to be executed in response to the call.

- In dynamic or late binding, procedure call is unaware of the related code to be executed till execution or run time.
- Dynamic binding is extensively used in polymorphism and inheritance. In polymorphism, a function call related with a polymorphic type of reference depends on the dynamic type of that reference.
- Now consider the function "draw" in Fig. 1.1.3. Due to inheritance, every object will have this procedure. The algorithm of draw for circle, box and triangle is unique to each object. So the draw procedure is redefined in each class that defines the object.
- At run-time, the code of function is matched with object under current reference will be called.

1.1.7 Benefits of OOP

- OOP offers several advantages listed below :
 1. **Reusability** : Through inheritance redundant code can be eliminated and existing classes can be reused.
 2. Development time can be saved and productivity can be increased due to inheritance, instead of starting from the scratch existing code can be reused.
 3. Security can be achieved through data hiding.
 4. It is Possible to have multiple objects to coexist without any interference.
 5. It is possible to map objects from problem domain to objects in the program.
 6. Easy to partition the work in project based on objects.
 7. It supports data centered approach.
 8. Possible to upgrade small to large one.
 9. Interface descriptions with external system can be made simpler through message passing techniques for communication.
 10. Software complexity can be managed.

1.1.8 Java is pure 100% Object Oriented Language

Q. Why Java is not 100% object oriented language.

W-11

In Java, everything is not considered as a Object, primitive data types are also available. For example : int data type is used for numbers which is not a object type. (Only Integer is object type).

All features of OOP language is not fully supported by Java. for example : Multiple Inheritance, Operator Overloading, etc.

Pure object oriented language should satisfy following 6 features :

1. Encapsulation/Information Hiding.
2. Inheritance.
3. Polymorphisms/Dynamic Binding.
4. All pre-defined types should be Objects.
5. All operations performed by sending messages to Objects.
6. All user-defined types are Objects But in java, features 4 and 5 are absent.

Therefore java is not 100% pure object oriented language.

1.2 Java Features

Q. State any four features of Java.

W-08, W-10, S-12

Q. Explain concept of JVM with respect to portability feature of Java.

W-09

Q. Explain features of Java.

W-11, W-12

- Java is a general-purpose, object-oriented programming language developed by Sun Microsystems of USA in 1991, which was originally called as 'Oak' by James Gosling. The important feature of the language is that it is a platform-neutral language. Java is the first programming language, which is not tied to any particular hardware or operating system. Programs developed in Java can be executed anywhere on any system.

- **Features of java are listed below**

- (1) Compiled and Interpreted
- (2) Platform-Independent and Portable
- (3) Object-Oriented
- (4) Robust and Secure
- (5) Distributed
- (6) Familiar, Simple and Small
- (7) Multithreaded and Interactive
- (8) High Performance
- (9) Dynamic and Extensible

1. **Compiled and Interpreted**

Basically a computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system.

- (i) First, Java compiler converts or translates source code into byte code instructions.
- (ii) Then, Java interpreter generates machine code for byte code instructions which is directly executed by the machine on which Java program is running.

2. **Platform-Independent and Portable**

Q. Explain Platform Independence

S-11

- Java supports the feature portability. Due to portability moving Java programs from one PC to another is quite easy.
- Upgradation of operating systems, processors and system resources now never force the changes in Java programs. Therefore, Java is suitable as well as popular language for programming on Internet where interconnection of different kinds of systems is required.

- Java provides this type of portability in following two ways.
 - (i) First, Java compiler converts or translates source code into byte code instructions. Then, Java interpreter generates machine code for byte code instructions which is directly executed by the machine on which Java program is running
 - (ii) The size of the primitive data types is machine independent that is not depends upon the memory addressing of operating system.

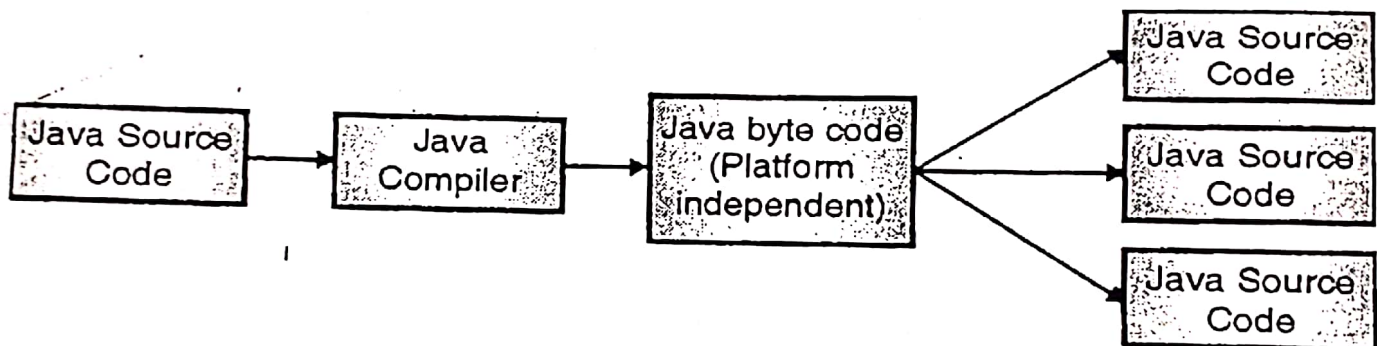


Fig. 1.2.1 : Java is platform independent language

3. Object-oriented

Q. Why Java is called as truly object oriented? Explain.

S-09, W-09, S-12

Q. Why Java is not 100% object oriented language?

W-11

Q. What is garbage collection in Java? Explain.

W-13

- Java is a truly object-oriented language. Almost everything in Java is an *object*.
- All program code and data reside within objects and classes.
- Java comes with an extensive set of classes, arranged in packages that can be used in programs by inheritance. The object model in Java is simple and easy to extend.

4. Robust and secure

- **Robust** : Java is a more robust language than others. As it has two time execution strategy that is compilation and interpretation, therefore provides reliability. Garbage-collection is one of the features of Java language, which reduces programmer's task of memory management. Java provides exception handling, which catches the severe errors and reduces risk of crashing the system.
- **Secure** : The programming languages which are used for programming on internet should provide the security as the hazard of viruses and exploitation of resources are everywhere.

Java programming system verifies all memory access and ensures that viruses cannot communicate with an Applet. The absence of pointers in Java leads to proper memory access.

5. Distributed

- Java is a very popular and used as distributed language for creating applications on networks because of its ability to share both data and programs.
- Java applications are more robust hence can open and access remote objects on Internet as easily as on local system. This feature of java enables multiple programmers at various remote locations to work together on a single project

Familiar, simple and small

- Java is a very simple language which has many features of C and C++ that are redundant or sources of unreliable code are not part of Java. For example, it does not use pointers, pre-processor header files, goto statement and many others. Java language also eliminates operator overloading and multiple inheritance.

- Familiarity is another prominent feature of Java. It is modelled on C and C++ languages to make the language look familiar to the existing programmers and it uses many of the constructs of C and C++.

Multithreaded and interactive

- Multithreaded means handling multiple tasks concurrently or simultaneously by supporting multithreaded programs. This means that there is no need to wait for the application to complete one task before beginning another. Therefore improving the interactive performance in graphics application.
- The Java runtime or JDK comes with tools that support multi-process synchronization and construct smoothly running interactive systems.

High performance

- Java performance is very remarkable for an interpreted language, mainly due to the use of intermediate code called byte code.
- The inclusion of multithreading enhances the execution speed of Java programs.

Dynamic and extensible

- Java is also a dynamic language with the capacity of dynamically linking in new class libraries, methods, and objects. Java can also determine the type of class with the help of query, making it possible to either dynamically link or abort the program, depending on the response.
- Native Methods : Functions of other languages (C C++) are supported by the Java programs. These methods are known as native methods. So that programmer can make use of efficient functions available in these languages. These methods can be linked dynamically at runtime.

1.2.1 Difference between Java and C

Q. State different features of Java. Those make it differ from 'C' language. **S-10**

Sr. No.	Java	C
1.	Java is an object-oriented language and has mechanism to define classes and objects.	C is procedure oriented language has mechanism for creating procedures.
2.	Java does not include the C unique statement keywords sizeof, and typedef.	C has unique keywords sizeof for memory management and typedef.
3.	Java does not contain the data type's struct and union.	C supports the user defined data types struct and union.
4.	Java does not define the type modifiers keywords auto, extern, register, signed, and unsigned.	C supports the type modifiers keywords auto, extern, register, signed, and unsigned.
5.	Java does not support explicit pointer type statements.	C supports an explicit pointer type.
6.	Java does not have a preprocessor and therefore we cannot use # define, # include, and # ifdef statements.	C supports the preprocessor and therefore we can use # define, # include, and # ifdef statements.
7.	Java requires that the functions with no arguments must be declared with empty parenthesis.	In C the function with no parameters must be declared with the void keyword or empty parenthesis.

Sr. No.	Java	C
8.	Java adds new operators such as instanceof and >>.	C does not support instanceof
9.	Java adds labelled break and continue statements.	C supports only simple break and continue statement.
10.	Java adds many features required for object-oriented programming.	C never supports object oriented features because it is procedure oriented.

1.2.2 Difference between Java and C++

Sr. No	Java	C++
1.	It is a true object-oriented language	It is basically C with object-oriented extension
2.	It does not support operator overloading.	It support operator overloading.
3.	It does not have template classes.	It has template classes.
4.	Multiple inheritance of classes is not supported in Java. Interface is used to implement it.	It supports multiple inheritance of classes.
5.	It does not support global variables. Every variable and method is declared within a class and forms part of that class	It supports global variables.
6.	It does not use pointers.	Pointers are used in C++.
7.	It has replaced the destructor function with a finalize() function	It has the destructor function.
8.	There are no header files in Java	It includes header files in program.

1.2.3 Java and Internet

Q: Why Java is popular on internet. S-09

- The evolution of internet helped to make a java as leading programming language.
- Over the network there are two categories of objects that are transmitted between the server and client computers :
 - Active Objects**
A self executing program is dynamic data. This data is called as active agent on the client computer.
 - Passive objects**
These objects are in the form of information and active programs. When an email is viewed it is viewing the passive data. A downloaded program, until it is executed is called as passive data.
These dynamic networked programs create problems in terms of security and portability. Java effectively handles these problems. Introducing a new form of program known as Applet does this.

Java Applets

- Java communicates with Web page through a special tag called <APPLET>. An applet as a special program that is transmitted over the network or Internet. It automatically gets executed by Java - compatible Web Browser. An applet is downloaded as per user requirement.
- An applet is said to an "Intelligent Program" because the applet tends to react to the user input and changes its status dynamically.
- Working with applet is very interesting. Java takes care of major issues related with security and portability of any program code over Internet.

Fig. 1.2.2 shows the Java's interaction with Web.

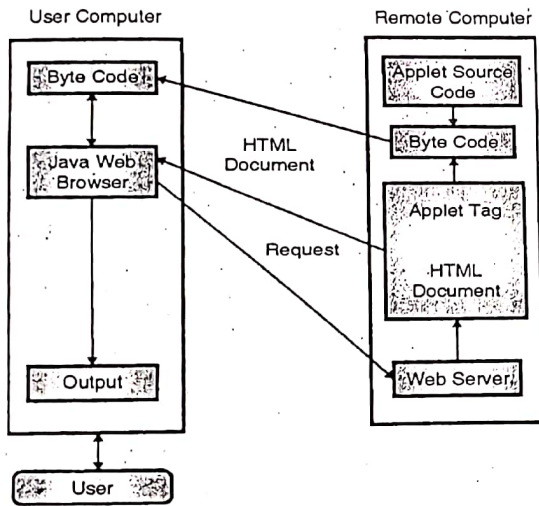


Fig. 1.2.2 : Java programs interaction with the Web

Security and portability

Security becomes an important issue for a language which is used for programming on Internet. Threat of viruses and abuse of resources are everywhere. Java system not only verifies 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 without proper authorization.

Java supports the feature portability. Due to portability moving Java programs from one PC to another is quite easy. Upgradation of operating systems, processors and system resources now never force the changes in Java programs. Therefore, Java is suitable as well as popular language for programming on Internet where interconnection of different kinds of systems is required.

- Java provides this type of portability in following two ways.
 - First, Java compiler converts or translates source code into byte code instructions. Then, Java interpreter generates machine code for byte code instructions which is directly executed by the machine on which Java program is running
 - The size of the primitive data types is machine - independent that is not depends upon the memory addressing of operating system.

1.3 Java Components

1.3.1 Java Byte code and Java Virtual Machine

Q.	What is byte code? Explain any two tools available in JDK.	S-09 S-12
Q.	What is Byte code explain JVM.	W-11, W-13
Q.	What is JVM? What is byte code.?	S-13

- Compiler translates source code into machine code for specific computer (platform dependent). But the Java compiler produces an intermediate code known as byte code, for a machine that does not exist. This machine is called as 'Java Virtual Machine' (JVM) and it only exist inside the computer memory. It is simulated computer within the computer and performs all major functions of real computer. Fig. 1.3.1 shows the process of compilation.

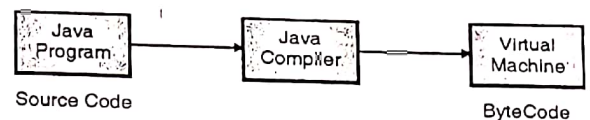


Fig. 1.3.1 : Process of Compilation

- The output produced by the Java compiler is not executable code. It is byte code, highly optimized set of instructions executed by java run time system or JVM. JVM is an interpreter for the byte code.

The virtual machine code generated by Java Interpreter is not machine specific (portable), but it acts as an intermediary between the virtual machine and real machine as shown in following Fig. 1.3.2.



Fig. 1.3.2 : Process of Converting byte code into machine code

Following Fig. 1.3.3 shows how Java works on typical computer. The Java object framework (Java API) acts as an intermediary between the user programs and virtual machine which in turn acts as the intermediary between the operating system and Java object framework.

Concept of Java programs is being interpreted and not compiled to solve major problems related with downloading programs from Internet.

By translating a Java program into byte code makes it easier to run the program on a variety of environment thus JVM provides the most convenient way of making portable program. The byte code requires the JVM for execution it makes the secure. This is because the JVM can control and prevent the programs from generating side effects of outside system.

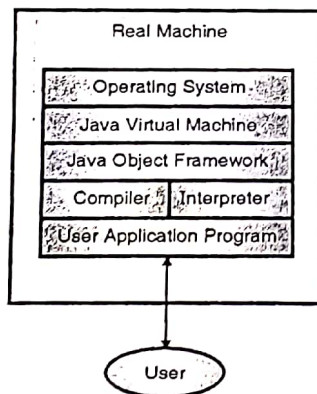


Fig. 1.3.3 : Layers of interaction for Java programs

When program is compiled and then interpreted, the processing time is reduced because the byte code is highly optimized set of instructions and JVM executes them much faster.

1.3.2 Java Environment

- Java development environment includes a number of development tools, classes and methods. It is a part of the system known as Java Development Kit (JDK). The classes and methods are part of the Java Standard Library known as JSL, and it is also known as the Application Programming Interface (API).

- Java Development Kit (JDK) :** The Java Development Kit is a collection of tools which are used for developing, designing, debugging, executing and running Java programs. They include :

(i) **Appletviewer (for viewing Java applets) :**

A pre-created applet can be viewed using applet viewer enables us to run Java applets (without actually using a Java-compatible browser). It has syntax:

```
c:\> applet viewer <URL of the .html file>
```

(ii) **javac (Java compiler) :**

A java program can be created by using any text editor and save file with extension java. Java compiler, which translates Java source code to byte code files that the interpreter can understand. Or it converts the "java" file to a ".class" file. It has following syntax:

```
c:\> javac filename.java
```

(iii) **java (Java interpreter) :**

The Java interpreter is used to execute a compiled Java application (.class). Java interpreter, which runs applets and applications by reading and interpreting byte code files. The byte code created as a result of compilation is interpreted. The syntax is (file extension is needed);

```
c:\> java filename
```

(iv) **javap (Java disassembler) :**

Java disassembler, it used to convert byte code files into a program, description.

(v) **javah (for C header files) :**

It is used for producing header files for use with native methods of C and C++.

(vi) **javadoc (for creating HTML documents) :**

It is used to create HTML formatted documentation from Java source code files.

(vii) **jdb (Java debugger) :**

Java debugger, which helps us to find errors in our programs

These tools are applied to build and run application programs are illustrated in Fig. 1.3.4. The steps are listed below

- (i) To create a Java program, first create a source code file using a text editor.
- (ii) The source code is then compiled using the Java compiler javac.
- (iii) Then it is executed using the Java interpreter java.
- (iv) The Java debugger jdb is used to find errors, if any, in the source code.
- (v) Java dis-assembler javap can be used to convert compiled Java program into a source code.

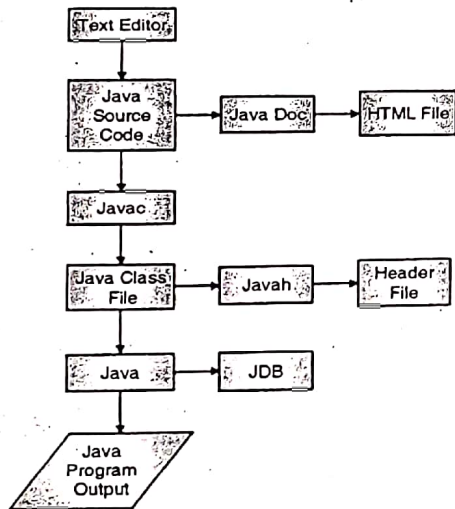


Fig. 1.3.4 : Process of building and running Java application programs

1.4 Simple Java Program

```
class Sampled
{
    public static void main(String args[ ])
    {
        System.out.println("Welcome to the world of JAVA");
    }
}
```

Above is very simple program displaying the statement "Welcome to the world of JAVA" but it follows some features of language listed below

(i) **Class declaration**

- The first line - class Sample - declares the class, which is an object oriented construct. As the Java is object-oriented language everything must be placed within the class. Class is the keyword and declares the new class definition. Sample is the class name or Java identifier.
- The file must be saved with the name of the class in which the main() function resides with the extension java. So above program is saved with the name Sample.java.

(ii) **Opening brace**

Every class definition begins with the opening bracket '{' and ends with the closing bracket '}'.

(iii) **The Main Line**

- public static void main(String args[]) statement defines a main method main(). Each and Every eJava application begins with main() function, and a starting point for the interpreter to begin the execution of the program.
- A Java program can have any number of classes but only one class can have main() method to initiate the execution. The Java applet does not use the main() method.

This line includes following keyword :

public : the keyword public is an access specifier or access modifier that declares the main method as unprotected and therefore it is accessible to other classes. It indicates that the class member can be accessed from anywhere in the program. As the main is declared as public it can be accessed by JVM.

static : static keyword allows the main to be called without creating instance of the class. But the copy of main is available in memory even if the instance of it is created. Declares this method as it belongs to the entire class and not a part of any object of the class. The main is always declared as static since interpreter uses this method before any objects are created. JVM first invokes the main method to start execution of program. Hence, the main should be static.

void : This is a type modifier which states that the main method does not return any value.

String args[] : it is a parameter passed to main method from command prompt. All parameters to a method are declared inside a pair of parentheses. The `String args[]` declares a parameter named args, which contains an array of objects of class type String. Arguments passed from command prompt are stored in array 'args' in string format.

the output line

The only one executable statement in above program is :

`System.out.println("Welcome to the world of JAVA");`
 The `println` method is a member of the `out` object, which is the static member of `System` class. This line prints the statement in double quotes. Every statement in Java must end with semicolon.

Java Program Structure

Documentation Section	- Suggested
Package Statement	- Optional
Import Statements	- Optional
Interface Statements	- Optional

Class Definitions	- Optional
Main Method class	- Essential
{	
Main Method Definition	
}	

1.4.2 Implementing Java Program

- Implementing Java programs includes following steps :

- | |
|--------------------------|
| (i) Creating the program |
| (ii) Compiling a program |
| (iii) Running a program |

- Creating the program

Java program can be created by using any text editor. Save this file by using the extension `.java`. The filename should be same as class name. This file is called as source file. If a program contains multiple classes, then the file name must be the class name of the class containing the main method.

- Compiling a program

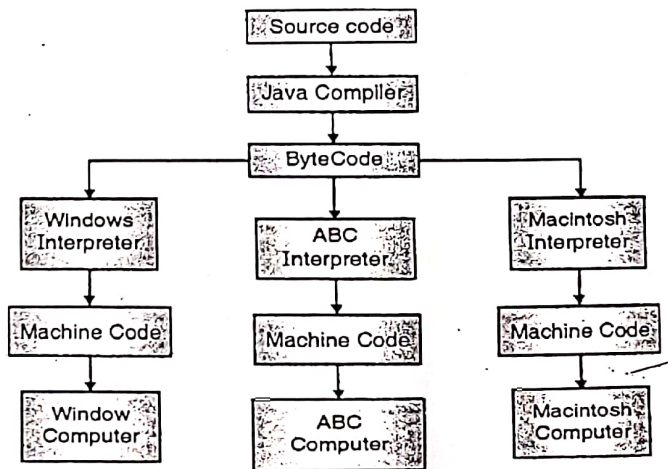


Fig. 1.4.1 : Implementation of Java Program

To compile the program, the java compiler javac with the name of the source file on the command line as shown below:

```
javac file-name.java
```

If file contains no errors then the javac compiler creates a file called filename.class, it contains the byte codes of the program. The compiler automatically names the byte code file as filename.class.

Running a program

Java interpreter is used to run the program, at the command prompt type java filename.

1.4.3 Command Line Argument

- Command line arguments are parameters that are supplied to the application program at the time of invoking its execution. They must be supplied at the time of its execution after the file name.
- In the main () method, the args is declared as an array of string known as string objects. Any arguments provided in the command line at the time of program execution, are passed to the array args as its elements. Using index or subscripted notation can access the individual elements of an array. The number of elements in the array args can be getting with the length parameter.

Like cnt = args.length;

- > **Program 1.4.1 :** Write a program to Find Factorial of Given number using command line arguments

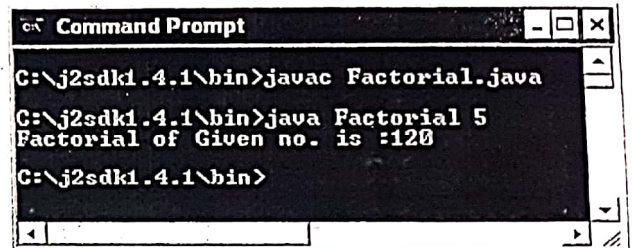
Q. Write a program to accept a number and print its factorial.

W-09 W-13

```
/*Write a program to Find Factorial of Given no. */
```

```
class Factorial
{
    public static void main(String args[])
    {
```

```
int num = Integer.parseInt(args[0]); //take argument as
command line
int result = 1;
while(num>0){
    result = result * num;
    num--;
}
System.out.println("Factorial of Given no. is :"+result);
}
```



- > **Program 1.4.2 :** Write a program to add two nos.using command line arguments

Q. Write a program to accept two numbers as command line arguments and print the addition o those numbers.

S-13

```
class Addition
{
    public static void main(String args[])
    {
        int num1 = Integer.parseInt(args[0]);
        int num2 = Integer.parseInt(args[1]);

        //take argument as command line
        int result = 0;
        result = num1 + num2 ;
    }
}
```

```

}
System.out.println("Addition of two nos.: " + result);
}
}

```

Output

```

C:\j2sdk1.4.1\bin> javac Addiotn.java
C:\j2sdk1.4.1\bin> java Addiotn 5 6
C:\j2sdk1.4.1\bin> Addition of two nos.: 11

```

1.5 Constants, Variables and Data Types

- A programming language consists of data in the form of numbers, characters and strings, which provides output known as information.
- The program is sequence of instructions performs the task of processing data.
- The instructions are formed using certain symbols and words according to some rules known as syntax rules (or grammar).
- Every program instruction must conform precisely to the syntax rules of the language.

1.5.1 Constants

- In Java constants, refer to fixed values that do not change during the execution of a program. Java supports several types of constants listed below and shown in Fig. 1.5.1.

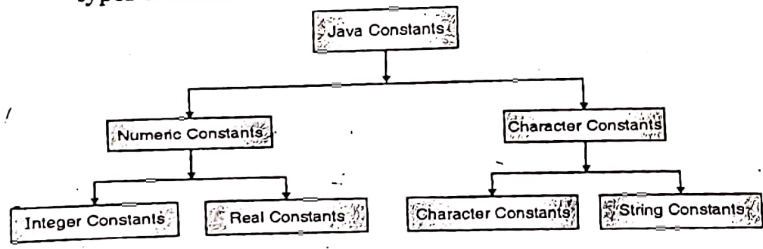


Fig. 1.5.1 : Java Constants

1. Integer Constants

An integer constant means the sequence of digits. There are three types of integers,

- (i) **Decimal Integer** : Decimal integers consist of a set of digits, 0 through 9, preceded by an optional minus sign. Examples are : 325, -978, 45673
Embedded spaces, commas, and non-digit characters are not allowed between digits. For example : 15 78, \$657-, # 47 is illegal number.
- (ii) **Octal Integer** : An octal integer constant consists of any combination of digits from the set 0 through 7, with a leading 0. Examples are : 027, 0, 0231, 0341
- (iii) **Hexadecimal integer** : A sequence of digits preceded by Ox or ox is considered as hexadecimal integer (hex integer). They may also include alphabets A through F or a through f. A letter A through F represents the numbers 1 through 15.

Examples : Ox4 Ox1F Oxabcd Ox

2. Real Constant

Integer numbers are insufficient to represent quantities that vary continuously, such as distance heights, temperatures, prices, and so on. Numbers having fractional parts like 17.548 represents these quantities. Such numbers are called real (or floating point) constants.

Examples :
1.9783, 0 0047

The real number constant can comprise of four parts

- A whole number
- A fractional part
- A decimal point
- An exponent part

These numbers are shown in decimal notation, having a whole number followed by a decimal point and the fractional part, which is an integer.

Example : 0.035, 111.98

A real number may also be expressed in exponential (or scientific) notation. For example, the value 215.65 may be

written as 2.1565e2 in exponential notation. e2 means multiply by 10². The general form is : mantissa e exponent
Examples : 0.65e4, 15e-5, 2.6e+6, 56.E4

3. Single character constant

A single character constant or character constant contains a single character enclosed within a pair of single quote marks. Examples are : '7', 'A', '*', ''

The character constant '7' is not the same as the number 7. The last constant is a blank space.

4. String constant

A string constant is a sequence of characters and always enclosed between double quotes. It can be characters, alphabets, digits, special characters and blank spaces.

Examples :

"WELCOME", "@june,", "Abhishek", "NIDHI!"

5. Backslash Constant

The backslash character constants are used with output statements for getting more effects like tab, new line, single quote etc. Java supports some special backslash character constants. A list of such backslash character constants is given in following Table 1.5.1. Each one of them represents one character, although they consist of two characters. These characters combinations are known as *escape sequences*.

Table 1.5.1 : Backslash Character Constants

Constant	Meaning
'\b'	Back space
'\f'	Form feed
'\n'	New line
'\r'	Carriage return
'\t'	Horizontal tab
'\"'	Single quote
'\"'	Double quote
'\\'	Backslash

1.5.2 Variables

- A variable is an identifier that denotes a storage location used to store a data value or a variable is a basic unit of storage. Unlike constants that remain unchanged during the execution of a program. A variable may take different values at different times during the execution of the program.
- A variable name can be chosen by the programmer in a meaningful way so as to reflect data stored in it. For example : radius, squ, total_height, classStrength. A variable names may consist of alphabets, digits, the underscore (_) and dollar (\$) characters.
- **Following are rules for naming variables :**
 - (i) They must not begin with a digit.
 - (ii) Uppercase and lowercase are distinct. This means that the variable Abc is not the same as abc or ABC
 - (iii) White space is not allowed
 - (iv) It should not be a keyword.
 - (v) Variable names can be of any length
- Variable must be declared before it is used in the program. Variables must be declared to the compiler. Declaration does following three things:
 - (i) It tells variable name to the compiler.
 - (ii) It specifies the data type of the data hold by variable.
 - (iii) The place of declaration in the program decided the scope of the variable.
- The variable can be declared as :

Type variable1, variable2,, variable_n ;
- Variables are separated by commas. A declaration statement must end with semicolon.

Example :

```
int a ;
float ax ;
double fl ;
byte f ;
char k3 ;
```

Initialization of variable

The process of giving initial values to the variables is known as the initialization. The variables are automatically initialized to zero if not assigned to any other value. A variable must be given a value after it has been declared but before it is used in an expression. It can be done in following two ways

(i) By using assignment statement

A value can be assigned to the variable by using assignment operator as follows

```
Variable_name = value;
```

```
For example : b = 789.
```

A variable value can be given at the time of its declaration also as follows:

```
Type Variable_name = value;
```

```
For example : int a = 79;
```

(ii) By using read statement

The readLine() method which is invoked using an object of the class DataInputStream, which is then converted to the corresponding data type using the data type wrapper classes.

> Program 1.5.1 : Reading data from keyboard

```
//Reading data from keyboard.
import java.io.DataInputStream
class Reading
{
    public static void main(String args[])
    {
        DataInputStream in = new DataInputStream(System.in);
        int ni = 0;
        float nf = 0.0f;
        try
        {
            System.out.println("Enter an integer:");
```

```
ni = Integer.parseInt(in.readLine());
System.out.println("Enter float number:");
nf = Float.valueOf(in.readLine()).floatValue();

catch (Expression e)

System.out.println("intNumber = " + ni);
System.out.println("floatNumber = " + nf)
```

1.5.3 Data Types

- Q Write all primitive data type available in Java with their storage sizes in bytes. **W-09, S-12**
- Q List different data types in java. **S-10**
- Q Give all primitive data types available in Java with their storage sizes in bytes. **S-13**

In Java every variable has a data type. Data type specifies the size and type of data that can be stored.

1. Primitive Data Type (Intrinsic)

- (i) Numeric
 - (a) Integer
 - (b) Floating Point
- (ii) Non-Numeric
 - (a) Character
 - (b) Boolean

2. Non-Primitive (Derived)

- (a) Classes
- (b) Arrays
- (c) Interfaces

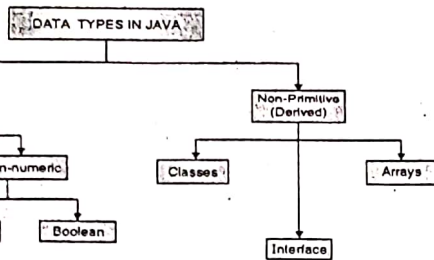


Fig. 1.5.2

Integer types

Integer types can hold whole numbers like 34546, 568, - 9347. Java does not support the concept of unsigned data types therefore all values in Java are signed (positive or negative). Java supports four types of integer data type shown in Table 1.5.2 with their size :

Table 1.5.2 : Size and Range of Integer Data Type

Type	Size	Minimum Value	Maximum Value
byte	1 Byte	- 128	127
short	2 Byte	- 31, 768	32, 767
int	4 Byte	- 2, 147, 483, 648	2, 147, 483, 647
long	8 Byte	- 9, 223, 372, 036, 854, 775, 808	9, 223, 372, 036, 854, 775, 807

Floating point types

Floating point type can hold numbers containing fractional part such as 67.8, - 9.02. There two types of floating point numbers in Java.

(a) Single precision value

The float types are single precision number. Floating point numbers are treated as double precision quantities. To force them for single precision mode, just append f or F to the numbers. For example: 3.45f, 34.87F.

(b) Double precision value

Double precision types are used when greater precision in storage is required. All mathematical functions, such as sin, cos and sqrt returns double type value. Floating point data type support a special value known as Not-a-number (NaN). NaN is used to represent the result of operations such as dividing zero by zero, where an actual number is not produced. Most operations that have NaN as an operand will produce NaN as a result.

Table 1.5.3 shows the floating point numbers with their size

Table 1.5.3 : Size and Range of Float Data Type

Type	Size	Minimum Value	Maximum Value
Float	Four Byte	- 3.4e - 038	3.4e + 038
Double	Eight Byte	1.7e - 308	1.7e + 308

1. Character types

Java provides a character data type called char to store character constants in memory. The char type requires size of 2 bytes but, it can hold only a single character.

2. Boolean types

Boolean type is used when to test a condition during the execution of the program, which has two values either true or false. Boolean only one bit of storage.

1.5.4 Scope of Variables

- Scope of the variables is nothing but the life time of a variable. It depends upon the where in the program that variable is declared. The area of the program where the variable is accessible is called its scope.
- Java variables are actually classified into three categories and their scope is defined according their category. The types of java variables are listed below:

(i) Instance variable

Instance is declared inside the class. Instance variables are created when the object are instantiated

or created and those are associated with the objects. They take different values for each object.

(ii) **Class variables**

Class variables are declared inside the class. They are global to the class. It belongs to all object created by class. Only one memory location is created for each class variable.

(iii) **Local variables**

Local variables declared and used inside the methods. They are not available outside of the method definition. Local variables can be declared inside the program blocks that are defined between the opening ({) and closing braces (}). These variables are visible to the program only. When the program control leaves block, all the variables in the block will cease to exist.

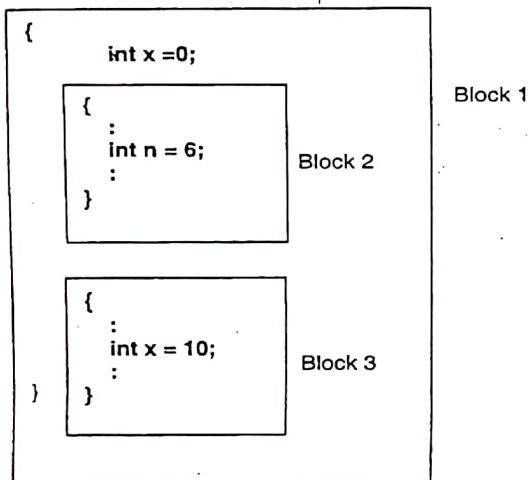


Fig. 1.5.3 : Nested Program Blocks

1.5.5 Symbolic Constants

- Symbolic constants refer to the variable name having value which can not be modified. For example pi= 3.14 now the pi can be used instead of 3.14.
- For the convention symbolic names are written in CAPITALS to distinguish from other variable names.
- After declaration of symbolic constants, they should not be reassigned any other value.
- Symbolic constants are declared for types.
- Symbolic constants cannot be declared inside a method. They should be used only once in class data members in the beginning of the class.

1.5.6 Type Conversion and Type Casting

Q. Explain 'type casting' with suitable example. **S-11**
 Q. What do you mean by type casting? When is it needed? **W-12**

- The process of converting one data type to another is called casting. Type casting occurs when there is a need to store a value of one data type into a variable of another type. Casting is necessary when a method returns a type different than one require.

Type Casting refers to changing an entity of one data type into another. This is important for the type conversion in developing any application. If you will store an int value into a byte variable directly, this will be illegal operation. For storing your calculated int value in a byte variable you will have to change the type of resultant data which has to be stored.

- It is possible to cast a value to be stored by preceding it with the type name in parenthesis. The syntax is

Type variable1 = (type) variable2;

- **Examples are :**

```

int x = 60;
byte n = (byte) x;
long cnt = (long) x;
  
```


The data types like float, double and four integer types can be cast to any other type but not in Boolean. Casting of large data type into smaller type may result in a loss of data. Casting of a large floating type value of into smaller integer will result in a loss of the fractional part. Table 1.5.4 shows the casts that result in no loss of information.

Table 1.5.4 : Casts that Results in No Loss of Information

From	To
byte	Short, char, int, float, double
short	int, long, float, double
char	int, long, float, double
int	long, float, double
long	Float, double
float	Double

Automatic Type Conversion : For assigning one type of data value to another type of data variables, casting is required. But Java provides this type of assignment without casting known as automatic type conversion. It is possible only if the destination type has enough precision to store the source value, like int data type is large enough to hold byte data type value. Therefore following statements are valid.

```
byte a = 70; int b = a;
```

Widening or Promotion : Widening or promotion are nothing but the process of assigning smaller data type to a larger one.

Narrowing : The procedure of assigning a larger type to a smaller type is known as narrowing, which results in loss of data.

1.5.7 Getting Values of Variable

Java supports two output methods that can be used to send the results to the screen :

(i) print() method : Prints and waits

This method sends information into a buffer, which is not flushed until a new line or end of line character is sent. The print() method prints a line until a new line character is encountered.

(ii) println() method : Print a line and move to next line

This method displays the provided information on a line followed by a line feed (carriage-return).

1.5.8 Standard Default Values

Every variable has a default value in Java. If variable is not initialized when it is first created, Java provides default value to that variable type automatically as shown in Table 1.5.5

Table 1.5.5 : Default values for Various Types

Type of Variable	Default Value
Byte	Zero (byte 0)
Short	Zero (short 0)
Int	Zero : 0
Long	Zero : 0L
Float	0.0f
Double	0.0d
Char	Null character
Boolean	False
Reference	Null

1.5.9 A Simple Java Program (Input / Output Statement)

> Program 1.5.2 : Simple Java Program

```
class GoodDay
{
    public static void main (String args[]) {
        System.out.println("Good Day!");
    }
}
```

Output

```
C:\j2sdk1.4.1\bin>javac GoodDay.java
C:\j2sdk1.4.1\bin>java GoodDay
GoodDay
C:\j2sdk1.4.1\bin>
```

> **Program 1.5.3 :** Write a program to convert given number of days into months and day.

Example : Input - 69 Output - 69 days = 2 Month and 9 days */

```
/* (Assume that each month is of 30 days)
class DayMonthDemo{
    public static void main(String args[]){
        int num = Integer.parseInt(args[0]);
        int days = num%30;
        int month = num/30;
        System.out.println(num+" days = "+month+" Month and
        "+days+" days");
    }
}
```

Output

```
C:\j2sdk1.4.1\bin>javac DayMonthDemo.java
C:\j2sdk1.4.1\bin>java DayMonthDemo 56
56 days = 1 Month and 26 days
C:\j2sdk1.4.1\bin>
```

> **Program 1.5.4 :** Write a program to Swap the values

```
/* Write a program to Swap the values */
class Swap{
    public static void main(String args[]){
        int num1 = Integer.parseInt(args[0]);
        int num2 = Integer.parseInt(args[1]);
        System.out.println("\n*** Before Swapping ***");
        System.out.println("Number 1 : "+num1);
        System.out.println("Number 2 : "+num2);
        //Swap logic
    }
}
```

```
num1 = num1 + num2;
num2 = num1 - num2;
num1 = num1 - num2;
System.out.println("\n*** After Swapping ***");
System.out.println("Number 1 : "+num1);
System.out.println("Number 2 : "+num2);
}
```

Output

```
C:\j2sdk1.4.1\bin>javac Swap.java
C:\j2sdk1.4.1\bin>java Swap 56 22
*** Before Swapping ***
Number 1 : 56
Number 2 : 22
*** After Swapping ***
Number 1 : 22
Number 2 : 56
C:\j2sdk1.4.1\bin>
```

1.6 Operator and Expression

- Java supports a wide range of operators. The basic arithmetic operators are =, +, - and *. An operator is a symbol that indicates operations to be performed may be certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables. They usually form a part of mathematical or logical expressions.
- Java operators can be classified into following categories :
 1. Arithmetic operators
 2. Relational operators
 3. Logical operators
 4. Assignment operators
 5. Increment and decrement operators
 6. Conditional Operator

1.6.1 ✓ Arithmetic Operators

Q. Describe arithmetic operators with suitable example. **W-08, S-13**

Arithmetic operators are used to build mathematical expressions and the computation as same in algebra. Java provides the basic arithmetic operators.

These can operate on built in data types of Java. They are listed in Table 1.6.1.

Table 1.6.1

Operator	Meaning
+	Addition or unary plus
-	Subtraction or unary minus
*	Multiplication
/	Division
%	Modulo division (Remainder)

There are few types of arithmetic operation :

(i) Integer Arithmetic

When an expression contains both the integer operands, then the expression is called Integer Expression. The operations performed are called as Integer Arithmetic and produces integer result. For example $A + B$ where A and B are integers.

(ii) Real Arithmetic

When both the operands in the expression are real, then the expression is called Real Expression and the operation is called as Real Arithmetic. A real operand may assume values either in decimal or exponential notation. Since floating point values are rounded to the number of significant digits permissible, the final value is an approximation of the correct result.

(iii) Mixed Mode Arithmetic

When both the operands are of the different type (example real and integer), then expression is called a *mixed-mode arithmetic* expression.

If either operand is of the real type, then the other operand is converted to real and the real arithmetic is performed. The result will be a real.

✓ 1.6.2 Relational Operators

Q. Explain relational operators in java. **S-10, W-11**

- When comparison of two quantities is performed depending on their relation, certain decisions are made. For example, we may compare the weight of two persons, or the price of two items, and so on. These comparisons can be performed with the help of relational operators. When the expression contains relational operators then it is termed as a relational expression.
- The value of relational expression is either true or false. For example, if $a = 10$, then $a < 20$ is true while $20 < a$ is False.
- Java supports six relational operators in all. These operators and their meanings are shown in Table 1.6.2.

Table 1.6.2

Operator	Meaning
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
!=	Not equal to

Some examples with their values are listed in Table 1.6.3.

Table 1.6.3

Expression	Result
45 < 10	False
20 > 40	True
20 <= 30	True
4.6 >= 0.89	True
10 <= 7+3	True

1.6.3 Logical Operators

Q. Explain logical operators in java. **S-10, W-11, W-13**

- The logical operators are the operators which are used to form multiple conditions by combining two or more relations. An example is `a != b && c == 50`. Java has three logical operators shown in Table 1.6.4.

Table 1.6.4

Operator	Meaning
&&	Logical AND
	Logical OR
!	Logical NOT

- Logical expression is an expression which combines two or more relational expressions or a compound relational expression. The logical expression gives true or false value, according to the truth table shown in Table 1.6.5.
- An expression `a != b && c == 50` is true if both the conditions are true. It gives false if either (or both) of them are false.

Table 1.6.5

op-1	op-2	op-1 && op-2	op-1 op-2
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

1.6.4 Assignment Operators

Assignment operators are used to assign the value of an expression to a variable. Assignment operator, '=' is used in the program for assignment. In addition, Java has a set of 'shorthand' assignment operators which are used in the form

$$\text{Variable_name op} = \text{exp}$$

Where,

Variable_name is a variable, exp is an expression

op is a Java binary operator.

The operator `op =` is known as the shorthand assignment operator.

The assignment statement `x op= exp;` is equivalent to `x = x op(exp);` with x accessed only once.

Consider an example `x += y+ 1`. This is same as the statement `x = x+(y+1);`

Commonly used shorthand assignment operators examples are listed in Table 1.6.6.

Table 1.6.6

Simple Assignment operators	Statement with shorthand operators
<code>A = A + 1</code>	<code>A += 1</code>
<code>A = A - 1</code>	<code>A -= 1</code>
<code>A = A * (n + 1)</code>	<code>A *= (n+1)</code>
<code>A = A / (n + 1)</code>	<code>A /= (n+1)</code>
<code>A = A % B</code>	<code>A %= B</code>

Using shorthand assignment operators leads to following three advantages :

1. Repetition is avoided because what appears on the left-hand side need not be repeated and therefore it becomes easier to write.
2. The statement becomes more simple, short and easier to read.
3. Use of shorthand operator results in a more useful code.

1.6.5 Increment and Decrement Operators

Java has increment (++) which adds one and decrement (--) operators which subtracts one. These operators has following forms :

- (i) Postfix Increment (x++)
- (ii) Prefix Increment (++x)
- (iii) Postfix Decrement(x--)
- (iv) Prefix Decrement(--x)

++x; is equivalent to x = x + 1; (or x += 1;)

--x; is equivalent to x = x - 1; (or x -= 1;)

> **Program 1.6.1 :** Write a output of following expression. If a = 1, b = 2, c = 3 also give the contents of a, b, c at the end of every expression.

(i) a=d++ + --b

(ii) b=b+c-- * --a

```
class bitwise {
    public static void main(String args[]) {
        int a=1, b=2, c=3;
        a = (b++) + (--b);
        b = b + (c--) * (--a);
        System.out.println("a = " + a);
        System.out.println("b = " + b);
        System.out.println("c = " + c);
    }
}
```

Output

```
a = 3
b = 11
c = 2
```

1.6.6 Conditional Operators

The character pair ? : is a ternary operator of Java, which is used to construct conditional expressions of the following form:

expr1 ? expr2 : expr3

where expr1, expr2, and expr3 are expressions.

The ternary operator ? : works in three steps :

1. First expr1 is evaluated for result.
2. If it is true or nonzero, then the expression expr2 is evaluated for result. Now it will be the value of the conditional expression.
3. If expr1 is false, expr3 is evaluated and its value will be the value of the conditional expression.

Therefore, only one of the expressions (either expr2 or expr3) is evaluated.

For example, consider the following statements:

```
x = 10 ;
y = 15 ;
a = (x > y) ? x : y;
a = (10 > 15) ? 10 : 15;
a = 15
```

> **Program 1.6.2 :** Program demonstrating Conditional Operators by command line arguments

```
//Find Minimum of 2 nos. using conditional operator
class Minof2 {
    public static void main(String args[]) {
        //taking value as command line argument
        //Converting String format to Integer value
        int i = Integer.parseInt(args[0]);
        int j = Integer.parseInt(args[1]);
        int result = (i < j) ? i : j;
        System.out.println(result + " is a minimum value");
    }
}
```

Output

```
C:\j2sdk1.4.1\bin>javac Minof2.java
C:\j2sdk1.4.1\bin>java Minof2 4 7
4 is a minimum value
C:\j2sdk1.4.1\bin>
```

> Program 1.6.3 : Find Minimum of 2 numbers using conditional operator

```
class Minof22{
    public static void main(String args[]){
        int i = 10;
        int j = 9;
        int result = (i<j)?i:j;
        System.out.println(result+" is a minimum value");
    }
}
```

Output

```
C:\j2sdk1.4.1\bin>javac Minof22.java
C:\j2sdk1.4.1\bin>java Minof22
9 is a minimum value
C:\j2sdk1.4.1\bin>
```

1.6.7 BitWise Operators

Q. Describe various Bitwise operators with examples. W-08, W-13

The bit-wise operator acts upon single bit of their operands.

Table 1.6.7 shows bit wise operator :

Table 1.6.7

Operator	Meaning
&	Bitwise AND
	Bitwise OR
^	bitwise exclusive OR

Operator	Meaning
<<	Left Shift
>>	Right Shift
~	One's Complement
&=	Bitwise AND assignment
=	Bitwise OR assignment

Bitwise logical operators

Following table shows the bitwise logical operators and result when applied on single bit.

Table 1.6.8

A	B	A B	A and B	A ^ B	~A
0	0	0	0	0	1
0	1	1	0	1	1
1	0	1	0	1	0
1	1	1	1	0	0

The Left shift operator :

The left shift operator shifts all the bits in a value to the left, a specified number of times. It is denoted by "<<". It has following general form:

$$\text{Value} \ll \text{num};$$

Here, num specifies the number of positions to the left shift.

The result of left shift is that the first operand is multiplied by 2 raised to the number of times specified by the second operand.

$$128 \gg 1 = 128 * 2^1 = 256$$

$$256 \gg 4 = 256 * 2^4 = 64$$

Example : the << operator

This example shows the effect of using the << operator.

Note : the >> operator preserves the leftmost bits. The leftmost bits are filled with the previous content. This is to do with sign extension. In this case there is a 1 at the left and it is preserved. If you do not want to keep the 1 to the left, use the >>> operator which shifts 0's into the leftmost bits

Use : Shift right divides by 2;

On some older computers it was faster to use shift instead of multiply or divide.

```
y = x << 3;           // Assigns 8/x to y.
y = (x << 2) + x;     // Assigns 5/x to y.
```

➤ **Program 1.6.4 :** Given that a=64, I = a<<2. Write a program to find the output of I

```
class bitwise
{
    public static void main(String args[])
    {
        int a=64, I;
        I = a << 2;
        System.out.println("a = "+a);
        System.out.println("I = "+I);
    }
}
```

Output

```
a = 64
I = 256
```

1.6.8 Special Operators

Java supports following special operators:

1. Instanceof operator

The instanceof is object reference operator and returns true if the object on the left hand side is an instance of the class given on the right hand side. For example: Nilima instanceof Student

The instanceof keyword can be used to test if an object is of a specified type.

if (objectReference instanceof type)

The following if statement returns true.

```
public class MainClass {
    public static void main(String[] a) {
        String s = "Hello";
        if (s instanceof java.lang.String) {
            System.out.println("is a String");
        }
    }
}
```

Output

```
is a string
```

However, applying instanceof on a null reference variable returns false. For example, the following if statement returns false.

```
public class MainClass {
    public static void main(String[] a) {
        String s = null;
        if (s instanceof java.lang.String) {
            System.out.println("true");
        } else {
            System.out.println("false");
        }
    }
}
```

Output

```
false
```

2. Member of operator (dot operator)

The dot operator is used to access the instance variables and methods of class objects. For example : person.name – will reference to the variable name.

1.6.9 Operator Precedence and Associativity in Java

- Java has well-defined rules for specifying the order in which the operators in an expression are evaluated when the expression has several operators. For example, multiplication and division have a higher precedence than addition and subtraction.
- Precedence rules can be overridden by explicit parentheses.
- When several operations occur in an expression, each part is evaluated and resolved in a predetermined order called operator precedence.
- Parentheses can be used to override the order of precedence and force some parts of an expression to be evaluated before other parts.
- Operations within parentheses are always performed before those outside. Within parentheses, however, normal operator precedence is maintained.
- When expressions contain operators from more than one category, arithmetic operators are evaluated first, comparison operators are evaluated next, and logical operators are evaluated last.
- Comparison operators all have equal precedence; that is, they are evaluated in the left-to-right order in which they appear. Arithmetic and logical operators are evaluated in the following order of precedence:
- When multiplication and division operator are present in an expression, operation is evaluated as it occurrences from left to right. Similarly, when addition and subtraction operator both present in an expression then, they are evaluated in order of occurrences from left to right.

Following table shows the Java Operators from highest to lowest precedence, their associativity.

Table 1.6.9

Precedence	Operator	Description	Associativity
1	[] . () ++ --	access array element access object member invoke a method post-increment post-decrement	left to right

Precedence	Operator	Description	Associativity
2	++ -- + - ! ~	pre-increment pre-decrement unary plus unary minus logical NOT bitwise NOT	right to left
2	() new	cast object creation	right to left
3	* / %	multiplicative	left to right
4	+ - +	additive string concatenation	left to right
5	<<>> >>>	shift	left to right
6	< <= > >= instanceof	relational type comparison	left to right
7	== !=	equality	left to right
8	&	bitwise AND	left to right
9	^	bitwise XOR	left to right
10		bitwise OR	left to right
11	&&	conditional AND	left to right
12		conditional OR	left to right
13	?:	conditional	right to left
14	= += -= *= /= %= &= ^= = <<= >>= >>>=	assignment	right to left

Precedence order

When two operators share an operand the operator with the higher *precedence* goes first. For example, $1 + 2 * 3$ is treated as $1 + (2 * 3)$, whereas $1 * 2 + 3$ is treated as $(1 * 2) + 3$ since multiplication has a higher precedence than addition.

Associativity

When two operators with the same precedence the expression is evaluated according to its *associativity*. For example $x = y = z = 17$ is treated as $x = (y = (z = 17))$, leaving all three variables with the value 17, since the $=$ operator has right-to-left associativity (and an assignment statement evaluates to the value on the right hand side). On the other hand, $72 / 2 / 3$ is treated as $(72 / 2) / 3$ since the $/$ operator has left-to-right associativity.

Expression Evaluation

Rules for expression evaluation

- First the expressions given in parentheses are evaluated.
- When the expression containing nested parentheses, the evaluation order is from the innermost to the outermost parentheses.
- Higher operators enlisted in table have higher precedence
- Operators enlisted in the same row in the table have equal precedence.
- To control the order of evaluation parentheses may be used

In Java programming language, the left operand is always evaluated before the right operand. The same is applicable to function arguments

1. Following code fragment shows the pre increment and post decrement

```
int x = 5;
int y = 10;
int z = ++x * y--;
```

Answer : z = 60

Here x = 5 and after pre increment its value is 6.

2. Following code fragment shows the evaluation with respect to parenthesis.

```
System.out.println("3 + 4 =" + 3 + 4);
System.out.println("3 + 4 =" + (3 + 4));
```

Answer: 3 + 4 = 34 and 3 + 4 = 7, respectively.

If operands of the $+$ operator is a string, the other operand is automatically cast to a string. Here, string concatenation and addition have the same precedence. Since they are left-associative, the operators are evaluated left to right.

In the second statement, parentheses ensure that the second $+$ operator perform addition instead of string concatenation.

1.6.10 Mathematical Functions

- For performing the various mathematical operations the `java.lang` package contains a class named as `Math`, which contains most commonly used mathematical functions like `abs`, `min`, `max`, `sqrt`, `pow`, `round` etc.
- Syntax :
`Math.method_name()`
Here `method_name` is replaced by different `math` class methods word

`min()` :

The minimum of two values. To find the minimum between two values the `math` class contains the method which will find out the minimum value. This method is overloaded with four different versions for integers and decimal numbers.

`Math.min(variable1, variable2)`

```
static int min( int value1, int value2);
static long min(long value1, long value2);
static float min(float value1, float value2);
static double min(double value1, double value2);
```