UNIT-1

INTRODUCTINO TO JAVA

Fundamentals of Object Oriented Programming

Object Oriented Programming is an approach to program development which eliminates disadvantages of programming methods by incorporating new concepts.

Ex: c++,Java, c#.

Object Oriented Paradigm

1. OOPs treats data as a critical element which is closely bind to the functions and protected from modifications by other program functions.
2. OOPs decompose problem into a number of entities called Objects and then build data and functions around these entities.
3. The data of the object can be accessed only by the methods associated with that object.

Fig. 1.1 *Object = Data + Methods*

Some of the features of object oriented paradigm are:

1. Emphasis is on data rather than procedure.
2. Programs are divided into what are known as objects.
3. Data structures characterize the objects.
4. Data is hidden cannot be accessed by external functions.

5. Objects communicates with each other through methods.
6. New data & methods can be added whenever necessary.

Basics concepts of Object Oriented Programming

General concepts of OOP are:

Object and Classes

1. Objects are the basic runtime entities in an object oriented system.
   Ex: a person, place, a bank account.
2. When a program is executed, the objects interact by sending message to one another.
   Ex: customs and account are two objects of banking program. Customer object may send a message to the account object requesting for the balance. The figure below shows the notation to represent object.

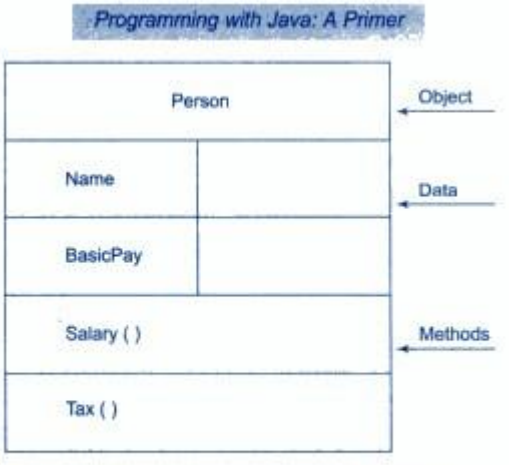


Fig. 1.2 *Representation of an object*

3. Class can be thought of as a ‘datatype ‘ and an object as a ‘variable’ of that datatype.
4. Once a class has been defined we can create any number of objects belonging to that class.

5. Class is a collection of objects of similar type.

   Ex: mango, apple and orange are members of class Fruit.

6. If Fruit has been defined as a class, than the statement.

   Fruit mango;

   Will create an object mango belonging to the class fruit.

Data Abstraction and Encapsulation.

1. The wrapping up of data and methods into a single unit called class is known as encapsulation.
2. Data is accessible to the methods which are wrapped in the class.
3. These methods provide the interface between the data and the program.
4. This insulation of the data from direct access by the program is called data hiding.

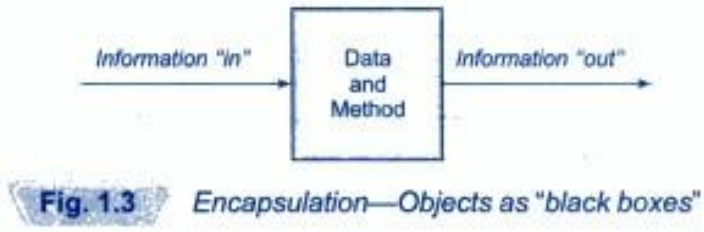


Fig. 1.3 *Encapsulation—Objects as "black boxes"*

5. Abstraction refers to the act of representing essential features without including the background details.

   Ex: Mobile phone is a complex electronic device, which is used for text and voice communication.

   How the internal circuitry of the mobile phone functions to enable the end user to talk to a remote person is irrelevant from his view point.

Inheritance

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

Ex: the bird robin is a part of class flying bird, which is again a part of the class bird as illustrated in fig

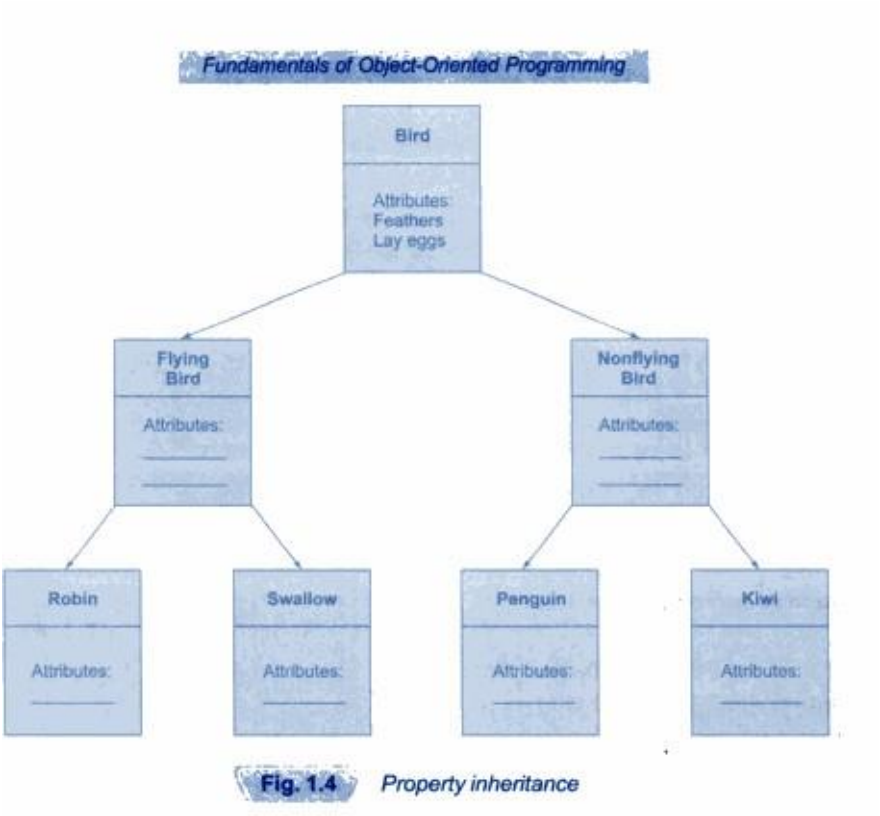


Fig. 1.4 Property inheritance

2. In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it.

Polymorphism

1. Polymorphism means ability to take more than one form.
2. The operation may exhibit different behaviour in different instances depending upon the types of data used in the operation.

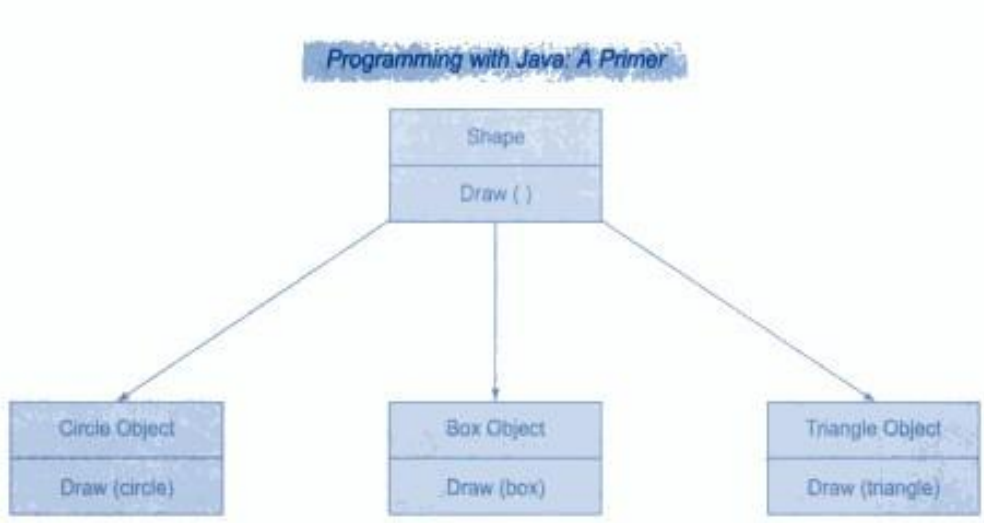


Fig. 1.5 Polymorphism

3. The below fig 5 illustrates that a single function name can be used to handle different number and different types of arguments.

Compile time and Runtime Mechanisms

1. Compile time may refers to any of the following:
    - Operations performed by compiler such as syntactic & Symantic analysis.
    - Pre-runtime assessment of the programs behaviour.
    - Implementation of inheritance (child class inherits the public data members and methods of the base class.
2. Runtime is the time period when a syntactically correct program is actually executed in the computer system.

Dynamic Binding

1. Binding refers to the linking of a procedure call to the code to be executed in response to the call.
2. Dynamic Binding means to that the code associated with a given procedure call is not known until the time of the call at runtime.

Message communication

1. An object-oriented program consists of a set of objects that communicate with each other.
2. The process of programming in an object-oriented language, involves the following steps:
    - Creating classes that defines objects and their behaviour.
    - Creating objects from class definitions.
    - Establishing communication among objects.
3. Objects communicate with one another by sending & receiving information as shown in fig.

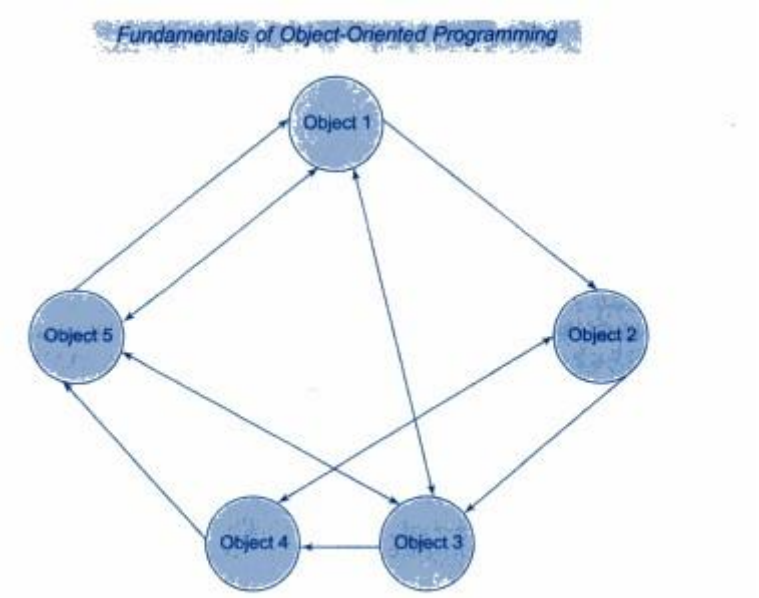


Fig. 1.6 Network of objects communicating between them

4. A message for an object is a request for execution of a procedure & therefore will invoke a method in the receiving object that generates the desired result as. Shown in fig:

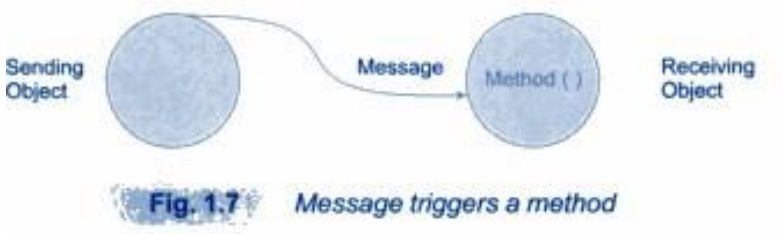


Fig. 1.7 *Message triggers a method*

5. Message passing involves specifying the name of the object, the name of the method (message) and information to be sent.

   Ex. Employee.salary(name);

   Here Employee is the object, salary is the message and name is the parameter that contains information.

Benefits of OOP

1. Through inheritance, we can eliminate redundant code & extend the use of existing classes.
2. Can save development time and higher productivity by building programs from the standard working modules rather than writing code from the scratch.
3. The principle of data hiding helps the programmer to build secure programs.
4. It is possible to have multiple objects to coexist without any interference.
5. It is easy to partition the work in a project based on objects.
6. Object-oriented systems can be easily upgraded from small to large systems.

7. Message passing techniques for communication between objects make the interface descriptions with external system much simpler.
8. Software complexity can be easily managed.

Applications of OOP

1. Real-time systems.
2. Simulations & Modelling.
3. Object-oriented databases
4. Hyper-text, hypermedia.
5. AI & Expert systems.
6. Neural Networks & Parallel Programming.
7. Office automation systems.
8. CIM/ CAD/ CAD Systems.

JAVA EVOLUTION

JAVA HISTORY: - JAVA is a general purpose object oriented programming language developed by Sun Microsystem at USA in 1990's. Originally it was termed as OAK by JAMES GASLING the first publically available version at Java (Java 1.0) was released in 1995. The current version of java is (Java 1.7) which is also known as Java 7.

JAVA FEATURES: - The most striking features at java languages is platform independent it is also called as platform neutral language. It is first programming language which is not bound to any particular hardware operating System.

Programs developed in Java can be executed anywhere on any systems. The features is:

- Compiled and interpreted.

- Platform independent and portable.
- Object oriented.
- Robust and secure.
- Distributed.
- Familiar, simple and small.
- Multi thread and interactive.
- Dynamic and Extensible.

1. Compiled and interpreted:

   Other computer languages will be either compiled or interpreted but java combines both this approach known as two stage system.

   i. First Java compiler translated source code into byte code instruction.

   ii. Byte code in the second stage will be interpreted as machine code that can be directly executed by the machine. This work will be done by Java Interpreter.

   iii. Therefore Java is both compiled and interpreted language.

2. Platform Independent and Portable:

   i. Significate feature of JAVA is its portability.

   ii. Java programs can be easily moved from one computer System to another computer system anywhere and anytime.

   iii. Changes such as upgrade OS processors and system resources will not force any changes in Java programs so Java is used in internet.

3. Object oriented:

   i. Java is true object oriented language since it uses all object oriented programming concepts(oops).

   ii. All programs code and data resides within object and classes.

iii. The object model in Java is simple and Easy to Extend.

4. Robust and Secure:
   i. Java is Robust language. It provides reliable code. It has a strict and compiled time and run time checking for datatypes.
   ii. Java is designed as a garbage collected language, helping the programmers in memory management problems.
   iii. Java also incorporate the concept of the exception handling which captures errors and Element any rise at crashing the system.
   iv. Java language become secure because of following properties.
      - No pointers.
      - Byte code verifies.
      - Class loader.
      - Security manager.

5. Distributed:
   i. Java is designed for distributed Environment of Internet because it handles TCP-IP protocols.
   ii. Distributed means that the program written can be run on any other System also.
   iii. Java also supports remote method invocation. This feature unable a program to invoke a method across a network.

6. Dynamic and Extensions:
   i. Java is dynamic language as it is capable of dynamically linking with new class library and function.
   ii. Java program supports functions written in other language such as c and c++. This functions are known as native method.

iii. Native methods are linked dynamically during runtime with the help of Java native interface(JNI tools).

7. Multithreaded and interactive:
   i. Multithread means handling multiple tasks simultaneously.
   ii. so the user need not to wait for application to finish one task before beginning another.
   iii. This feature includes the interactive the performance of graphical application.

8. High performance:
   i. Java performance is impressive due to the use of intermediate byte code.
   ii. Java byte codes can be compiled faster than c++ code using just in time compiler.
   iii. Multithreaded Enhances the overall execution speed of java programs.

9. Simple, Small and Familiar:
   i. Java is simple and small language.
   ii. It does not use pointers, pre-processors headerfiles, goto statements, operators overloading and multiple inheritance.
   iii. Familiar another Striking features of java as it is a simplified version of c++.

HOW JAVA DIFFERS FROM C AND C++

DIFFERENCE BETWEEN C AND JAVA: -

C	JAVA
1. C is a procedure oriented language. 2. It includes unique keywords namely size of and typedef. 3. It supports datatypes namely structure and unions. 4. It supports pointer. 5. It includes pre-processor directives. 6. Void must be used for NULL arguments. 7. It support goto statement. 8. It support global variable. 9. It doesn't include operator like instance of and >>>>. 10. It has explicit garbage collection.	1. Java is an object oriented language. 2. Java does not include unique keyword namely size of and typedef. 3. Java doesn't supports datatypes namely structure and unions. 4. It does not support pointer. 5. It does not include pre-processor directives. 6. Void must not used for No arguments. 7. It doesn't support goto statement. 8. It doesn't support global. 9. It adds new operator such as instanceof and >>>>. 10. It has built in/automatic garbage collections.

DIFFERENCE BETWEEN C++ AND JAVA

C++	JAVA
1. It is platform dependent language.	1. Java is independent platform language.
2. It supports operator overloading.	2. Java does not support operator overloading.
3. It support multiple Inheritance.	3. It doesn't support multiple inheritance.
4. It support pointers.	4. It doesn't support pointer.
5. It has distractors function.	5. It has replaced distractor function with the finalize function.
6. It has header file.	6. It has no header file.
7. It has template classes.	7. It does not have template classes.
8. Constant keyword used or declare constants.	8. Final keyword is used to declared constants.
9. It supports global variables.	9. Java doesn't support global variables.
10. It supports goto statements.	10. Java supports label with loop & statement blocks.

JAVA AND WORLD WIDE WEB

1. World Wide Web an open ended information retrievals system design to be used in the internet distributed environment.
2. Java was meant to be used in distributed environment such as internet.
3. Before java was introduced, WWW was limited to display still images and text.

The incorporation of Java into webpages has made it capable of supporting animation graphics, games and wide range of special effects.

4. Java communication with the webpages through a special tag called Applet. The following steps shown below are the communication detail.
   I. The user send the request for HTML document to the remote computers web page (servers). Webserver is a program that accepts the request, processor process it and send the required document.
   II. HTML document returned to the user browser the document contents the Applet Tag which identifies the Applet.
   III. The corresponding applet Byte code is transferred to the user computer. This byte code had been previously created by the Java compiler using the Java source code file for that applet.
   IV. Java enabled browser on the user computer interprets the byte code and Provides the output.

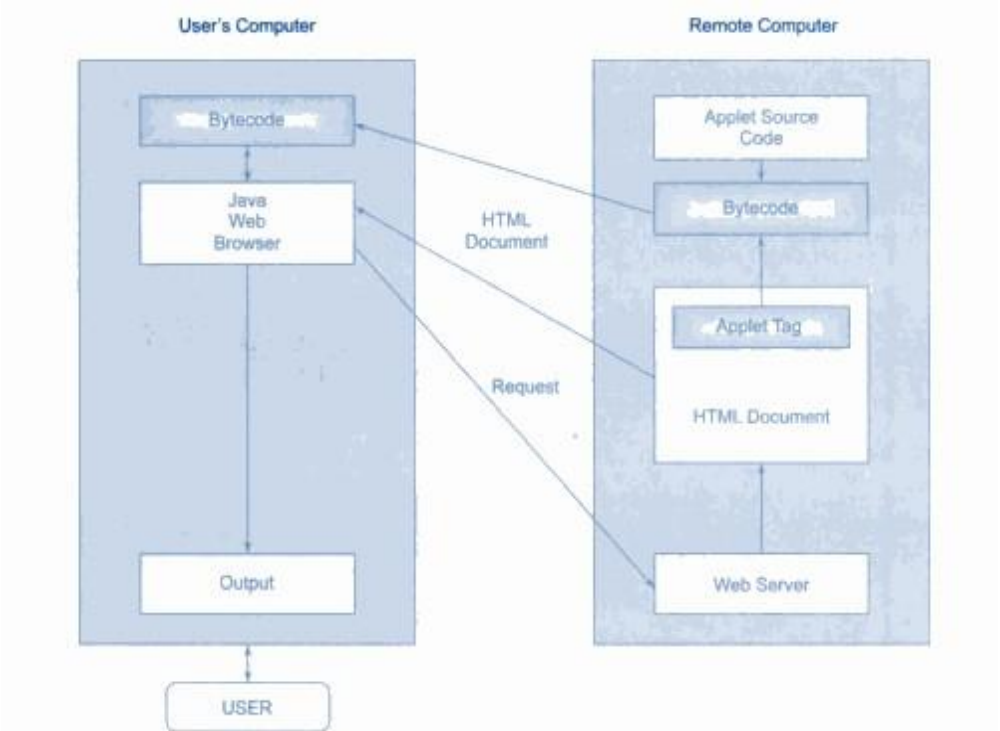


Fig. 2.4 Java's interaction with the web

JAVA ENVIRONMENT

Java Environment includes large no of development tools and hundreds of classes and methods. The development tool are the part of java development tool kit [JDK]. And classes and methods are the part of Java Standard Library [ISL]. This is also known as application programming interface [API].

Java Development Kit: Java Development Kit [JDK] comes with collection of tools that are used for developing and running Java Programmes.

They include: -

- APPLET VIEW: It enables user to runs to Java Applet.
- JAVA C: Java Compiler, which translates Java source code to byte code file that the interpreter can understand.
- JAVA: It is Java interprets which runs applets and application by reading and interpreting the byte code files.
- JAVA P: It is a dis-assembler which enables the user to convert byte code file into program description.
- JAVA H: It produces header files to use native methods.
- JDB: It is Java debugger which helps user to find errors in the programs.
- JAVA DOC: It creates HTML format documentation from Java Source code files.

The Applet view of Java is as follows: -

- Java c
- Java
- Java P
- Java H
- JDB
- Java DOC

APPLICATION PROGRAMMING INTERFACE [API]

Java Standard Library i.e API includes hundreds of classes and methods grouped into several function packages. They are: -

1. Language Support Package: It is a collection of classes and methods that are required for implementing basic features of Java.
2. Utilities Package: It is a collection of classes that provide utility function such as date and time functions.
3. Input/output Package: It is a collection of classes required for input/output manipulations.
4. Networking Package: It is a collection of classes for communicating with other computer via internet.
5. AWT Package: Abstract-window tool kit package contains classes that implement platform independent GUI.
6. Applet Package: This includes a set of classes that allows the user to create the java applets.

JAVA RUNTIME ENVIRONMENT [JRE]

Java runtime environment i.e. JRE facilities the execution of programs developed in java. It comprises of following.

- Java Virtual Machine [JVM].
- Runtime class libraries.
- User Interface toolkit.
- Deployment technologies.

1. **JAVA VIRTUAL MACHINE:** It is a program that interprets Java byte code and generates desired Output.

2. **RUNTIME CLASS LIBRARIES:** There are set of class libraries that are required for Execution of java programs.
3. **USER INTERFACE TOOKIT:** AWT [Abstract Window toolkit] and swing are the examples of user interface toolkit.
4. **DEPLOYMENT TECHNOLOGIES:** It comprises of following.
   - Java plugin: Enables the execution of java applet in the browser.
   - Java web start: Enables remoted deployment at an application.

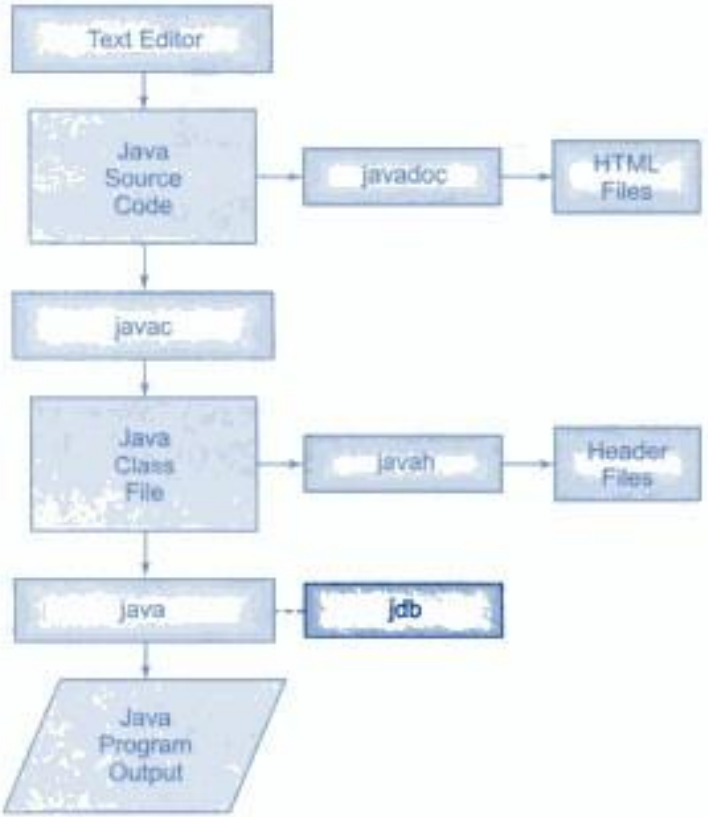


Fig. 2.5 *Process of building and running Java application programs*

SIMPLE JAVA PROGRAM:

The best way to learn a language is to write few simple examples programs and execute them.

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

```
class Sample
{
    public static void main(String args[])
        {
            System.out.println("This is sample Java program");
        }
}
```

CLASS DECLARATION:

The first line class Sample declares a class here **class** is a keyword, sample is a identifier that specifies the name of the class.

OPENING BRACE:

Every class definition in Java begins with an opening brace and Ends with an matching Closing brace.

THE MAIN LINE:

public static void main (String args[]) defines a method name main, this is similar to main function in c or c++

- This is the starting point for the interpreter to begin the execution of the program.
- A java application can have any no. of classes but only one of them must include a main method to initiate the execution.

  This line contains no. of keyword public, static, void:

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

> static: Which declares this method as one that belongs to the Entire class and not a part of any objects of the class.

> void: States that the main method does not return any values.

> String args[]: Declares a parameter name args[] which contains an array of object of the class type string.

THE OUTPUT LINE:

The only executable statement in the program is System.out.println("This is Sample Java program");

This is similar to printf statement of c.

The println method is a member of out object, which is a Static data members of System class. This line prints the String "This is Sample Java program".

The method println always appends a newline character to the end of the String.

Note: Every Statement in java must End with semicolon.

More of java

Write A Java Program that Computes the Squareroot of a number.

```
Import java .lang.Math

class sqaureroot

{

public static void main(String args[])

{

double root,n=25;

root=Math.sqrt(n);

System.out.println("The Square Root is="+root);
```

}

}

- double root,n=25 declares two variables which is at type double and initializes the values of n to 25.
- root=Math.sqrt(n); which invokes the method sqrt of the class Math and computes the squareroot of n and places the root in root.
- System.out.println which displays the result on the screen.

USE OF MATH FUMCTION:

import java.lang.Math

The purpose of this statement is to instruct the interpreter to load the Math class from the package lang.

COMMENTS:

Java Permits both single line and multiple line comments.

- The single line comment is indicated by //.
- Multiple line comment begin with /* and end with the */.

Write a java program to display the current date on console output/screen:

import.java.util.;*

class Datademo

{

public static void main(strings args[])

{

Date d=new Date (); //creating object

```
System.out.println("the system date is"+d);
}
}
```

APPLICATION WITH TWO CLASSES:

- **Write a program to compute the addition of two numbers and print a result using a third variable and using multiple classes.**

```
class Sum
{
int a,b;
void set(int x,int y)
{
a=x;
b=y;
}
void add ()
{
int c;
c=a+b;
System.out.println("The sum is:"+c);
}
class add1
{
```

```
public static void main(Strings args[])
{
Sum s1=new Sum();
s1.set(20,60);
s1.add();
}
}
```

- **Write a program to compute the area of rectangle using multiple classes.**

```
class Area
{
int l,b;
void set(int x,int y)
{
l=x;
b=y;
}
void cal()
{
int a;
a=l*b;
System.out.println("The area is:"+a);
```

```
}

}

class Area1

{

public static void main(String args[])

{

Area a1=new Area();

a1 .set (10,5);

a1. cal();

}

}
```

OR

```
class Rectangle

{

int l,b;

void set(int x,int y)

{

l=x;

b=y;

}

}

class Rect
```

```
{
public static void main(Sring args[])
{
int results;
rectangle;
 Rectangle r1=new Rectangle();
r1.rect(10,20);
result=r1.l*r1.b;
System.out.println("The area is:"+result);
}
}
```

The above program defines two classes Rectangle and Rect. The Rectangle class defines two variables and one method to assign values to these variables.

The class Rect contains the main method that initiates the execution.

The main method declares a local variable result and an object r1 of type Rectangle, then it assigns the values to the data members of class Rectangle by using set method ,and finally it calculates the area and prints result.

JAVA PROGRAM STRUCTURE:

The program structure for the java program is as given in the following figures-

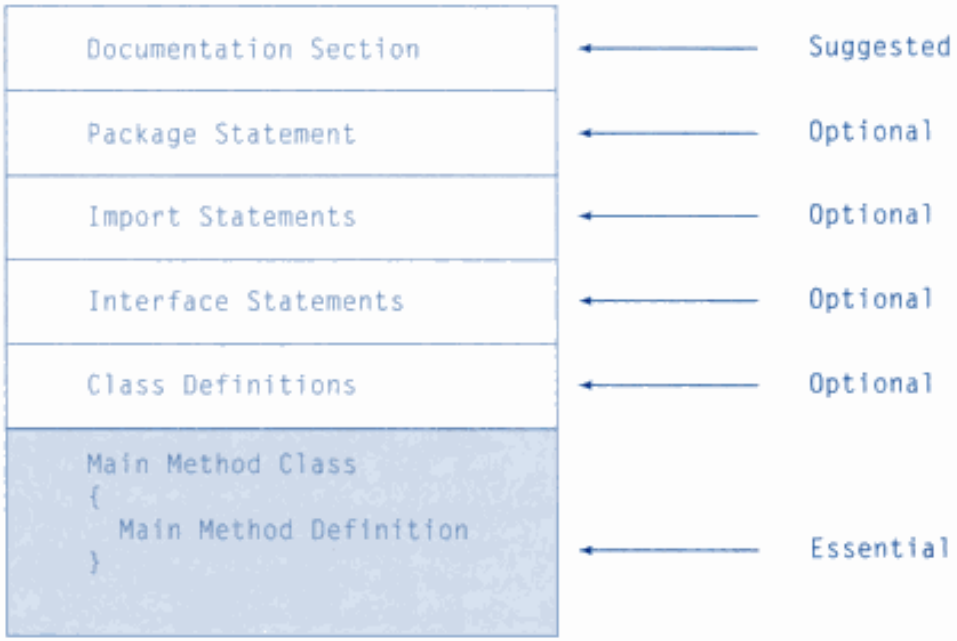


Fig. 3.2 *General structure of a Java program*

DOCUMENT SECTION:

The section provide the information about the source program. This section contains the information which is not compiled by the Java. Everything written in this section is written as comment.

PACKAGE SECTION:

It consists of the name of the package by using the keyword package. When we use the classes from this package in program then it is necessary to write the package statement in the beginning of the program.

IMPORT STATEMENT SECTION:

All the required Java API can be implemented by the import statement. There are some core packages present in the java. These packages include the classes and methods required for java programming. These packages can be imported in the program in order to use the classes and methods of the program.

INTERFACE STATEMENT:

An interface is like an class but includes a group of methods. It also an optional section, and used only when we need to implement the multiple inheritance feature.

CLASS DEFINATION SECTION:

The class defined section contains the definition of the class. This class normally contains the data and the methods manipulating the data.

MAIN METHOD CLASS:

This is called the main method class because it contains the main ( ) function. This class can access the methods defined in outer class.

JAVA TOKENS:

This Smallest individual and logical unit of the java statements is called tokens. In java there are five types of tokens used. These tokens are:-

- Reserved Keywords
- Literals
- Separators
- Identifier
- Operators

RESERVED KEYWORD:

Keywords are the reserved words which are enlisted as below –

Abstract, asserts, Boolean, byte, break, case, char, continues, do ,default ,else ,float, if, go to ,import, int, public, void

Super, switch, null etc.

IDENTIFIERS:

Identifiers are the kind of tokens defined by the programmer. They are used for naming the classes, methods, objects, variables, interfaces and packages in the program**.**

Following are the rules to be followed for the Identifiers:-

1. The identifiers can be written using alphabets, digits, underscore and dollar sign.
2. They should not contain any other special character within them.
3. There should not be a space within them
4. The identifier must not start with a digit, it should always start with alphabet.
5. The identifier are case-sensitive. For example –
   Int a;
   Int A;
   In above code a and A are treated as two different identifiers.
6. The identifier can be of any length.

LITERALS:

Literals are the kind of variable that store the sequence of character for representing the constant values. Five types of literals are –

- Integer literal.
- Floating point literals.
- Boolean literals.
- Character literals.
- String literals.

As the name suggest the corresponding data type constants can be stored within these literals.

OPERATORS:

Operators are the symbols used in the expression for evaluating them.

SEPARATORS:

For dividing the group of code and arranging them systematically certain symbols are used , which are known as separators.

Following tables describes various separators –

Table 3.2 Java Separators

Name	*What it is used for*
parentheses ( )	Used to enclose parameters in method definition and invocation, also used for defining precedence in expressions, containing expressions for flow control, and surrounding cast types.
braces { }	Used to contain the values of automatically initialized arrays and to define a block of code for classes, methods and local scopes
brackets [ ]	Used to declare array types and for dereferencing array values
semicolon ;	Used to separate statements
comma ,	Used to separate consecutive identifiers in a variable declaration, also used to chain statements together inside a 'for' statement
period .	Used to separate package names from sub-packages and classes; also used to separate a variable or method from a reference variable.

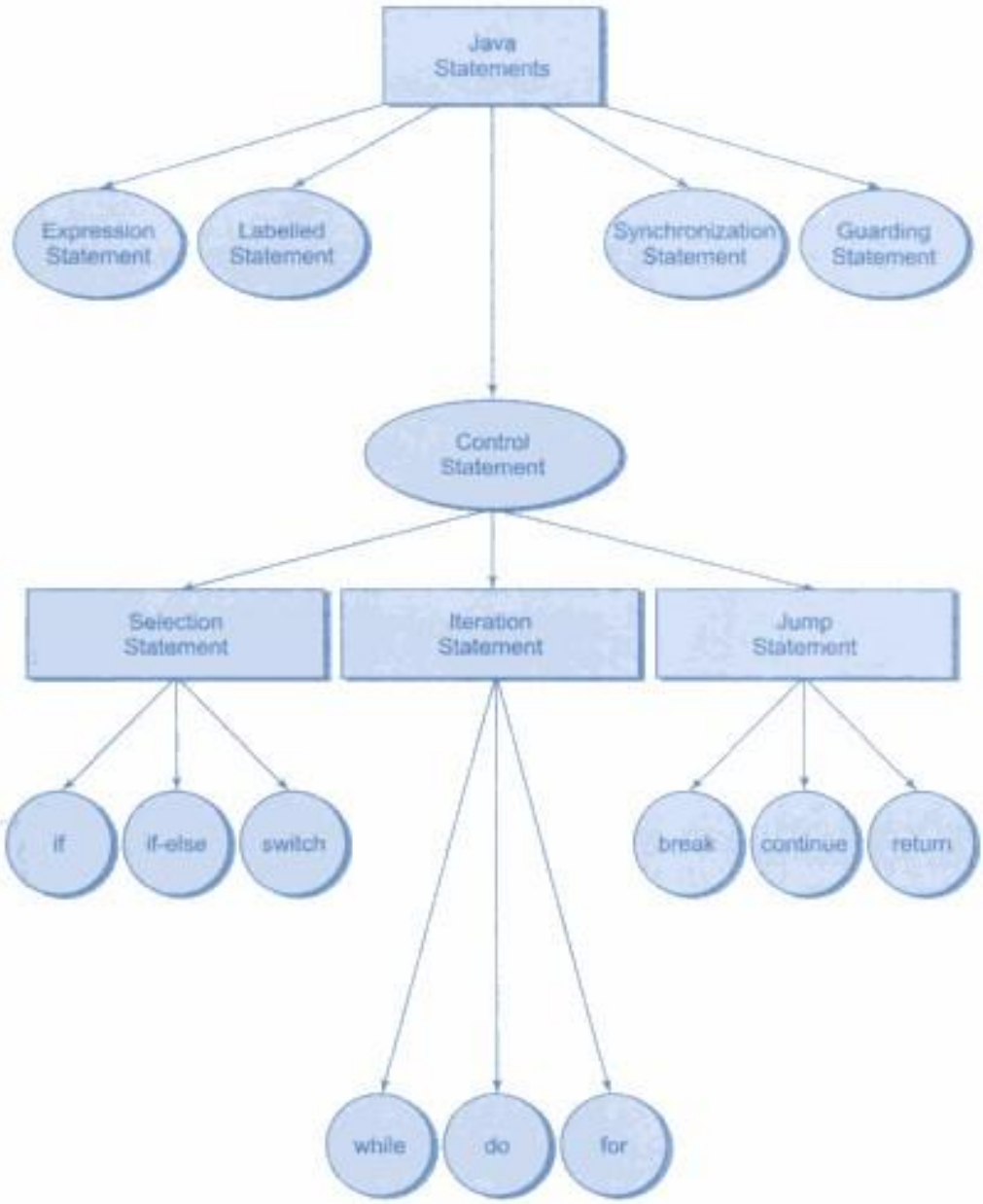


Fig. 3.4 *Classification of Java statements*

JAVA STATEMENT:

Statements are the executable sentences in java that are terminated by semicolon. Various types of statements used in java are –

EMPTY STATEMENT:

These statements do nothing but used simply as a placeholder.

LABELLED STATEMENT:

Labelled statement are the kind of statement which begin with the labels. The labels must not be the reserved keywords or the identifiers that are already defined in the program. These statements are reachable by using jump statements.

EXPRESSION STATEMENT:

Majority statements in java are of expression type. Assignment statements, pre and post increment as well as pre and post decrement statements and memory allocation statement are few examples of expression statements.

SELECTION STATEMENT:

The selection statements are the control structure statement which makes use of the programming constructs such as if, if else and switch.

ITERATION STATEMENT:

For specifying the looping these types of statements are used. The programming constructs such as while, do while, and for are used for iteration statement.

JUMP STATEMENT:

Jump statements are used to transfer the control to any desired location in the program.

SYNCHRONIZATION STATEMENTS:

For handling the multi-threading in java these statements are used.

GUARDING STATEMENT:

The guarding statement are used in order to handle errors or to avoid the program from crashing. The try-catch block statements are called the guarding statements.

IMPLEMENTING A JAVA PROGRAM:-

Implementation of a java application program involves a series steps. They include:

- Creating the program
- Compiling the program
- Running the program

Remember that ,before we begin creating the program, the Java Development Kit (JDK) must properly installed on our systems.

CREATING PROGRAM:-

We can create a program using any text editor.

Assume that we have entered for the following program –

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

We must save this program in a file called test.java ensuring that the file name contains the class name properly. This file is called the source file . Note all the java source file will be have the extension java. Note also that if a program contains multiple classes, the file name must be the classname of the class contained the main method.

COMPILING THE PROGRAM:

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

javac Test.java

If everything is ok , the javac compiler creates a file called test.Class containing the byte codes of the program. Note that the compiler automatically names the byte code file as classname.class

RUNNING THE PROGRAM:

We need to use the java interpreter to run a stand-alone program at the command prompt, type java Test.

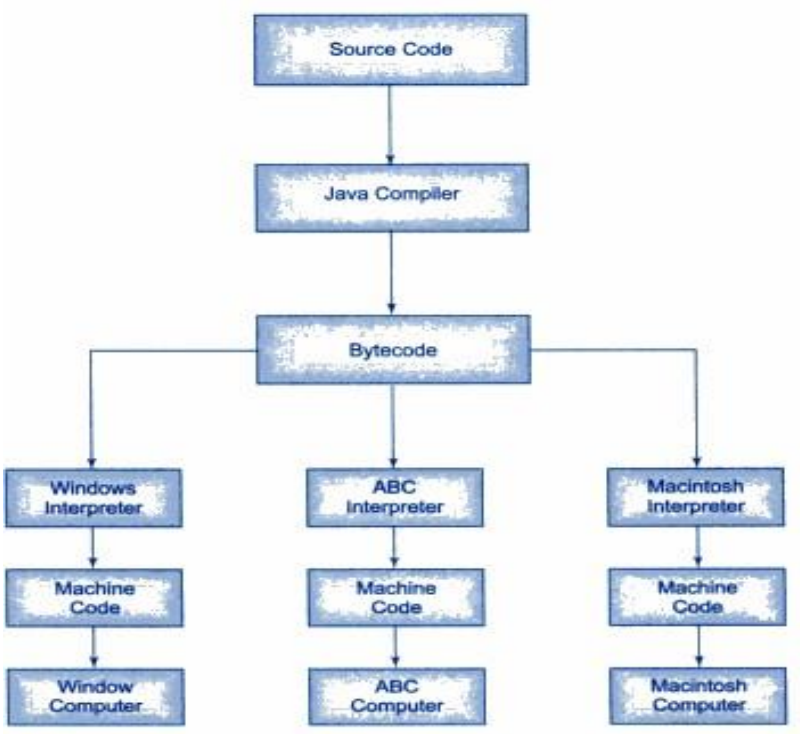


Fig. 3.5 *Implementation of Java programs*

Now , the interpreter looks for the main method in the program and begins execution from there .After execution, our program displays the following:

Hello!

JAVA VIRTUAL MACHINE[JVM]:

* Java compiler process an intermediate code known as byte code. This is also called as virtual machine code. This virtual machine code is not machine understandable code.

* This virtual machine code [Byte code] can be converted as machine code by java interpreter.

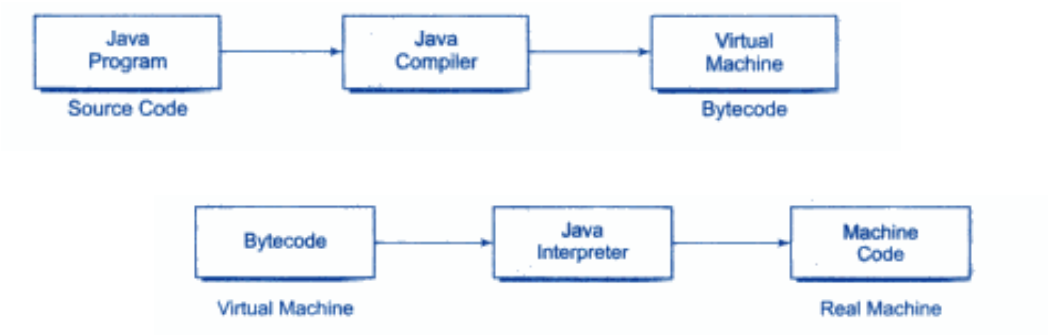


Fig. 3.6 *Process of compilation*

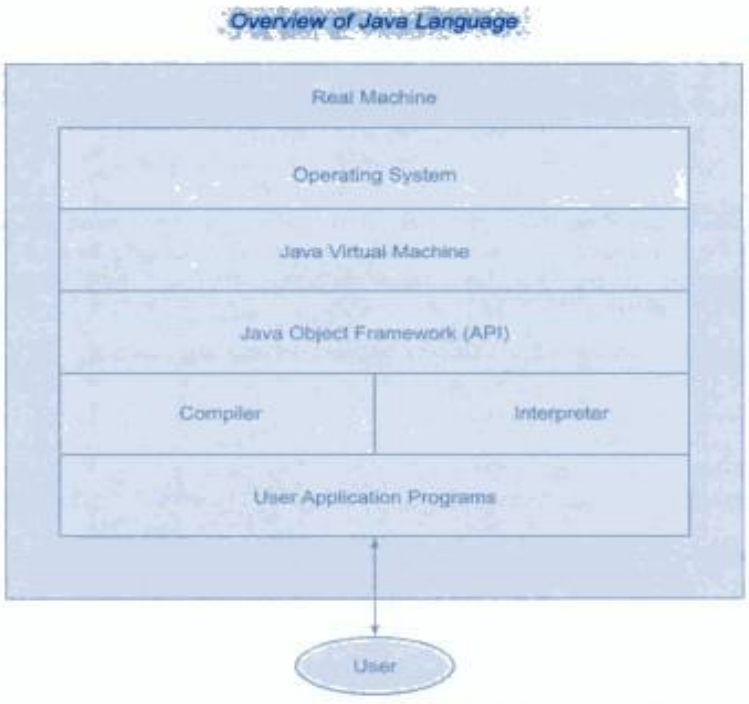


Fig. 3.8 *Layers of interactions for Java programs*

COMMAND LINE ARGUMENT:

These are the parameters that are supplied to the application program at the time of execution.

Public static void main(String args [ ])

Args is declare as an array of strings.

Any argument provided in the command line(at the time of execution) are passed to the array args as its elements.

Consider the following example program: -

```
class CommandLine
{
public static void main(Strings args[ ])
{
int count,i=0;
String string;
count=args.length;
System.out.println("The number of arguments is " +count);
for (i=0;i<n;i++)
{
String=args[i]
System.out.println("Java is " +string);
}
}
```

}

Output:

Compile: Javac CommandLine.java

Run: Java CommandLine Simple Robust Secure

No of argument=3

Java is simple!

Java is Robust!

Java is Secure!

PROGRAMMING STYLE:

Java is a free form language we need not have to indent any lines to make the program work properly. Java system does not cares where on the line we begin typing. But it will be considered as bad programming.

For Ex:- The statement System.out.println("Java is wonderful"); can be written as

System

.

Out

.

Println

(

"Java is

Wonderful!");

CONSTANT:

Constant are fixed value that do not change during the execution of the program.

INTEGER CONSTANT:

An integer is a sequence of digits. These are 3 types of integer namely Decimal, Octal, Hexadecimal.

DECIMAL:

This integer consist of digits from 0 to 9 proceeded by an optional(-) sign.

Ex: 123,-123,0,654321

OCTAL:

This integer consist of digits from 0 to 7 with leading 0.

Ex: 034,0,0435,0551

HEXADECIMAL:

Sequence of digit proceeded by 0x or 0X are called as Hexadecimal digits.

They may also included alphabets A to F for 10 to 15 digits.

REAL CONSTANT:

An integer number with decimal point and fractional part are called as Real Constants.

For Ex:0.083, -0.75, 485.78

A real number can also be expressed in exponential notation.

Example: 215.65 can also be expressed as 2.1565e2.e2 means multiple by 10^2.

SINGLE CHARACTER CONSTANT:

A single character constant contains a single character. Enclosed within a pair of single code marks

(quote marks)

Ex: '4','x','X',';','z'

STRING CONSTANT:

A string constant is a sequence of character enclosed b/w a double quotes. The character may be alphabets,

digits, special character and blank spaces.

EX: "Hello", "1997", "Welcome To Acharya".

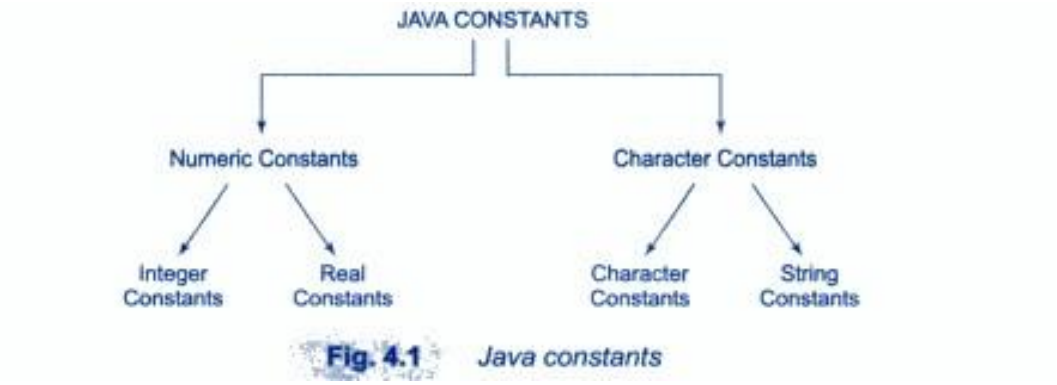


Fig. 4.1 *Java constants*

BACKSLASH CONSTANTS

Java supports some special backslash character constant that are used in output methods. For example, the symbol '\n' stands for newline character. A list of such backslash character constants is given in table 4.1. Note that each one of them represents one character, although they consist of two characters. These character combinations are known as escape sequences

Table 4.1 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

VARIABLE:

A variable is an identifier that denotes a storage location used to store a data values of variable during

Execution of program.

Examples of variable names are :

- Average
- Height, Max
- Total Height

RULES OF NAMING VARIABLE:

- They must not begin with the digit.
- Uppercase and lowercase are distinct.
- It should not be a key word.
- White space is not allowed.
- Variable name can be a any length.

DATA TYPE:

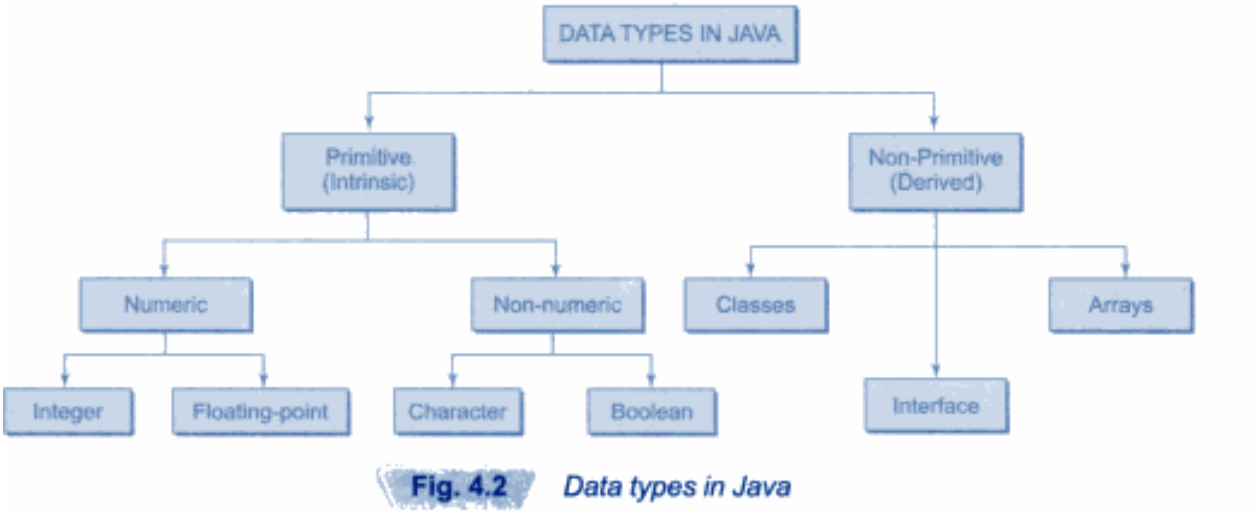


Fig. 4.2 *Data types in Java*

Every variable in java has a datatype. This specifies the size and type of values that can be stored.

INTEGER TYPE:

Integer type can hold whole number such as 56, -3 and 5639

There are 4 types of integer:

Byte, Short, Int, long

Represented in a keyword int.

Table 4.3 Size and Range of Integer Types

Type	*Size*	*Minimum value*	*Maximum value*
byte	One byte	–128	127
short	Two bytes	–32, 768	32, 767
int	Four bytes	–2, 147, 483, 648	2, 147, 483, 647
long	Eight bytes	–9, 223, 372, 036, 854, 775, 808	9, 223, 372, 036, 854, 775, 807

FLOATING POINT TYPE:

This type hold the number containing fractional parts such as 12.59 and -9.8 represented as float.

There are 2 types of floating point

- Float
- Double

SIZE AND RANGE OF Floating TYPE:

Table 4.4 Size and Range of Floating Point Types

Type	*Size*	*Minimum value*	*Maximum value*
float	4 bytes	3.4e-038	3.4e+038
double	8 bytes	1.7e-308	1.7e+308

Fig. 4.4 *Floating point data types*

CHARACTER TYPE:

Character data type stores characters constants in memory. Character type assumes size of two bytes but. It can hold only a single character represented as char.

BOOLEAN TYPE:

Boolean type can take only two values they are true or false represented as Boolean.

SCOPE OF VARIABLE:

The variable are classified into 3 types.

- Class Variables
- Instances Variables
- Local Variables

INSTANCE VARIABLE:

The variable are created when the object are instantiated and therefore they are associated with objects.

LOCAL VARIABLES:

Variables declares and use inside the methods are called local variables. They are not available to use outside the method definition. Local variables are also declares inside program blocks stating with {ending with}. The area of the program where this variable is accessible is called scope.

CLASS VARIABLES:

The variables are declare inside a class and also used inside a class.

SYMBOLIC CONSTANTS:

SYNTAX: **final type Symbolic_name= VALUE**

Valid examples of constant declaration are:

```
final int    STRENGTH = 100;
final int    PASS_MARK = 50;
final float  PI = 3.14159;
```

Guidelines for Naming Symbolic constants:

- Symbolic names takes the same from as variable names. But they are written in capitals to distinguish between two normal variables.
- They should not be assigned any other value within the program once after declaration.

  Ex. Final int STRENGTH=200; is illegal to use inside the program.
- They cannot be declared inside a method. They should only used as class data members in the beginning of class.
- Symbolic constants are declared for types this is not done in C and C++ where symbolic constants are defined using #define statements.

TYPE CASTING:

- Changing the value of one type into a variable of another type is known as Type casting.
- Casting is necessary when a method returns a type different than what we require

  The syntax for Type casting is:

  type variable1 = (type) variable2;

  Example:

  int m=50;

  byte n = (byte)m;

  long count = (long)m;

  1. Four integer type can be cast to any other types except Boolean casting into smaller types may result in loss of data.
  2. Casting a floating point value to an integer will result in loss of fractional part

     Example:-

     float b=3.157;

int a= (int)b;

value of a = 3;

3. The process of assigning smaller type to larger one is known as widening or promotion.
4. Assigning larger type a smaller one is known as narrowing or demotion.

Example:

```
class IntToByteConversion
{
   public static void main (String arg [])
   {
      int a = 35;
      byte b;

      b = (byte) a;

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

   }
}
```

STANDARD DEFAULT VALUES:

In Java Every variable as a default value. If we don't initialize the variables when it is first created, Java provides default values to that variable automatically as shown below:

Table 4.6 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

SPECIAL OPERATOR:

Java supports some special operators such as instance of operators and member selection operator (dot operator.

Instance of Operator:

Instance of an object reference Operator and return True if the object on left hand side is an instance of the class given on right hand side.

Example: Person instance of student the above statement is true if the object person belongs to the class (Student 1) else false.

System.out.println(S instanceof simple);

Example Programming:

```
class Simple1{
 public static void main(String args ]){
 Simple1 s=new Simple1 ();
 System.out.println(s instanceof Simple1); //true
 }
}
```

Dot Operator (member selection operator):

Dot operator is used to access the instance variable and methods of class object.

Example:

Person1.age;

Person1.salary ();

Example Programming:

```
class DotOperatorDemo
 {
    int add(int a, int b)
        {
            return a+b;
        }
  public static void main(String[] args) {

    // create object of the class DotOperatorDemo
    DotOperatorDemo dod = new DotOperatorDemo();

    //using dot(.) operator calling method of that class
    int result = dod.add(5, 10);

    System.out.println("Addition of 5 and 10 is :"+result);
  }
}
```

MATHEMATICAL FUNCTIONS:

We are going to import mathematical function by Math.Function name:

Table 5.12 Math Functions

Functions	*Action*
sin(x)	Returns the sine of the angle x in radians
cos(x)	Returns the cosine of the angle x in radians
tan(x)	Returns the tangent of the angle x in radians
asin(y)	Returns the angle whose sine is y
acos(y)	Returns the angle whose cosine is y
atan(y)	Returns the angle whose tangent is y
atan2(x,y)	Returns the angle whose tangent is x/y
pow(x,y)	Returns x raised to y (x^y)
exp(x)	Returns e raised to x (e^x)
log(x)	Returns the natural logarithm of x
sqrt(x)	Returns the square root of x
ceil(x)	Returns the smallest whole number greater than or equal to x. (Rounding up)
floor(x)	Returns the largest whole number less than or equal to x (Rounded down)
rint(x)	Returns the truncated value of x.
round(x)	Returns the integer closest to the argument
abs(a)	Returns the absolute value of a
max(a,b)	Returns the maximum of a and b
min(a,b)	Returns the minimum of a and b

Note: *x and y are double type parameters. a and b may be ints, longs, floats and doubles.*

Ex: import java.lang.Math.function name;

LABELLED LOOPS:

→Java permits a jump from one statement to the End or beginning of a loop as well as out of loop.

→Java supports three jump statement: break, continue or return. These statements transfer control to another part of your program.

Break Statement:

- In Java, the break statement has three uses.
    i. It terminates the statements sequence in switch statement.

ii. It can be used to exit a loop.

iii. It can be used as a 'civilized' form of goto.

- When a break statement is encountered inside a loop, the loop is terminated and program control resumes at the next statement of following the loop.

  Example:

```
for (int i=0; i<=5 ;i++)
{
System.out.print(i);
if (i==4)
break;
}
```

- By using the break statement, you can break out of one or more blocks at code. These blocks need not to be part of a loop or a switch.

  They can be any block.

  Syntax:

```
break label1;
```

  Example for labelled loops

```
Loop1: for(i=0;i<n;i++)
{
    Loop2:if (x<10)
    {
        break Loop1;
        System.out.println("end2");
    }
    System.out.println("end1");
}
```

Here the label loop1 labels the outer loop and therefore the statement

break loop1;

causes the execution to break out of both the loops

Continue Statement.

→The continue statement skips the current iteration of the loop and continues with the next iteration of the loop.

Example:

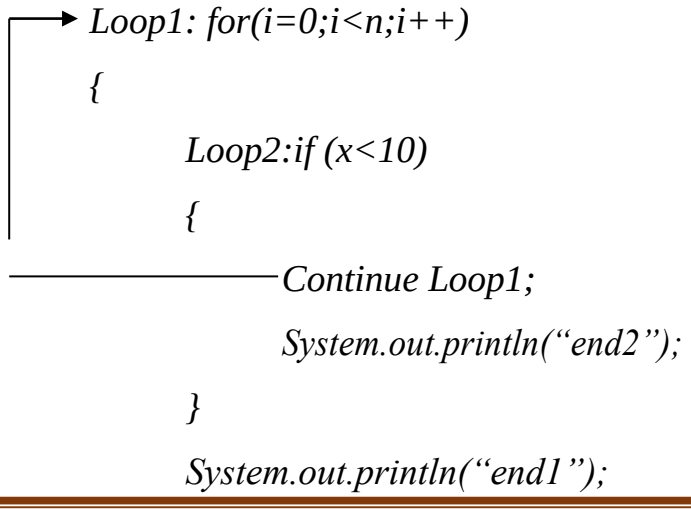
```
for (int i=1 ; i<=10; i++)
{
if (i%2==0)
continue i;
System.out.println(i);
}
```

→The labelled continue Statement can be used inside inner loop and continue with the next iteration of the outermost loop.

Syntax:

continue label;

Example for labelled loops



}

Here, the continue statement terminates the inner loop when x<10 and continues with the next iteration of the Loop1.

Return statement

→The return statement return the result to the calling method. The control flows jumps back to the line where the method was invoked.

Example:

```
res = add();

System.out.println(res);

static int add()

{

res = a+b;

return res;

}
```

The return statement immediately terminates the method in which it is executed.

OPERATORS AND EXPRESSIONS:

Operators:

The operator are special symbols that perform specific operations on one (or) more operands.

The different operation are used in mathematics expression in the same way that they are used in Algebra.

List of arithmetic Operators:

1. +: - Addition (unary plus).
2. - : - Subtraction (unary minus).
3. * : - Multiplication.
4. / : - Division.
5. %: - Modulus.
6. +=: - Addition.
7. -=: - Subtraction.
8. *=: - Multiplication Assignment.
9. /=: - Division Assignment.

The Operator of the arithmetic operators must be of a numeric type.

Increment and Decrement Operators

- The increment Operator (++) increase its operand by one.
- The decrement Operator (--) decrease its operator by one.
- These operator are unique in that they can appear both in postfix form.

  Ex:

  postfix = x++, y--;

  prefix = ++x, --y;

Bitwise Operator

- Java defines several bitwise operator that can be applied to integer types, long int, short, char and type.
- This bitwise operator manipulate the bits within an integer.
- These operator are upon the individual bits of their operand.

  List of Bitwise Operators

  1. n: - Bitwise unary not.
  2. &: - Bitwise and (AND).
  3. ! : - Bitwise OR.

4. ^: - Bitwise exclusive OR.
5. >>: - Shift right.
6. >>>: - Shift right zero fill.
7. <<: - Shift left assignment.

Conditional Operators

The Conditional Operators are used to construct conditional Expression of the form

exp 1 ? exp 2 : exp 3;

It is called as ternary operators.

Relational Operators:

The relational operators determine relationship that one operand has to the other.

Specifically they determine Equality and Ordering.

List of Relational Operator are:

1. == : - Equal to.
2. != : - Not Equal to.
3. > : - Greater than.
4. < : - Lesser than.
5. >= : - Greater than or Equal to.
6. <= : - Less than or Equal to.

Assignment Operator:

- Assignment Operator is "=" it assigns as the new value to variable and expression.
  Ex: - int a = 10;
- The Assignment Operator allows you to create a chain of assignment.

Ex: - x=y=z=10;

- The Assignment Operator are evaluated from right to left.

Logical Operator:

- The Boolean Operators operate only on Boolean operands.
- All the Binary Logical Operator Combines 2 Boolean values to form a resultant between values.

  List of Logical Operator

  1. && : - Logical AND.
  2. || : - Logical OR.
  3. ! : - Logical NOT.

EXPRESSIONS:

Expressions are the constructs made of Variables, Operators, Operands and can also involve method invocations to compute a single value and store in a particular variable.

Syntax: variable = expression;

Example: x = a*b-c;

Uses of Expressions:

1. To compute values.
2. To assign values.
3. To transfers the flow of Execution to particular method.

Types of Expressions.

1. Integer Expressions.

   Example programming:

   class IntExample

   {

```
public static void main (String args [])
{
int a = 1;
   int b = 2;
   int c = 4;
   a = b = a * b / c + 1;
   System.out.println("A="+a);
   System.out.println("B="+b);
}
}
```

2. Real Expressions.

3. Mixed Expressions.

DECISION MAKING:

- Selection Statement allow your program to choose different path of Execution based upon the outcome of Expressions or the state of variable.
- Java supports 2 selection statement: if and switch: - These Statement allow you to control the flow of you programs execution based upon condition known only during runtime.

If Statement:

- The if statement is Java's conditional branch statement. It can be used to route program execution through two different paths.

  Syntax: - if (condition) statement 1;

  Else statement 2;

  Here, Each Statement may be a single statement or a compound statement enclosed in curly braces.

This conditions is any expression that returns a boolean values.

The else clause is optional

Example: -

```
class ifexample
{
public static void main (String args[])
{
int a=10, b=20;
if (a>b)
System.out.println("The number is largest="+a);
else
System.out.println("The number is largest="+b);
}
}
```

- The different if statements are: -
  1. if.
  2. If-else.
  3. Nested-if.
  4. If-else-if ladder.

Nested if statements

→A nested if is an if statement that is the target at another if or else.

→Nested it's are very common in programming syntax:-

Syntax: -

```
if (condition)
{
```

Example: -

```
if (i==10)
{
```

```
if (condition)
{
//. . . . .
}
else
{
//. . . .
}
}
{
//. . . . .
}
```

```
if (j<20)
a=b;
if (k>100)
e=d;
else
a=c;
}
else
a=d;
```

- **The if –else if ladder**
  1. A common programming construct that is based upon a sequence of nested ifs is the if-else if ladder.

     Syntax: -

     if (condition)
     Statement
     else if(condition)
     Statement;
     else if (condition)
     Statement;
     .
     .
     .

else

Statement;

2. The if statements are executed from top down.
3. As soon as one of the conditions controlling the if is true, the Statement associated with that if is executed, and the rest of the ladder is by passed.
4. If none of the condition is true, then the final else statement will be Executed.
5. If there is no final else and all other condition are false, then no action will the place.

→**Example: -**

```
class if_else_ladder
{
public static void main (String args[])
{
        int month = 4;
        String season;
        if (month ==12 || month == 1|| month == 2)
        season= "Winter";
        else if (month ==3 || month == 4|| month==5)
        season="spring";
        else if (month == 9 || month ==10 || month ==11)
        season="Autumn";
        else if (month == 6 || month ==7 || month == 8)
```

season=”Summer”;

else

season=”Bogus month”;

System.out.println(“The season is=”+season);

}

}

- **Switch Statement**

  It is a multiway branching statement which gives better alternatives to series to of if-else if statement.

  Syntax: - switch (Expressions){

  case value 1:

  // Statement sequence

  break;

  case value 2:

  //Statement sequence

  break;

  .

  .

  .

  case value n:

  //Statement sequence

  break;

  default:

  //default Statement sequence }

  - The Expressions must be of type, byte, short, int, char.
  - Each case value must be a unique literal.

- The break statement is used inside the switch to terminates a statement sequence.
- If no case statement matches then the default statement is executed.
- The witch can only test for equality. Whereas if can evaluate any type of boolean expression.

Example Programming:

```
class Test {
public static void main(String args[]) {
 char grade = 'C';
      switch(grade) {
   case 'A' :
     System.out.println("Excellent!");
     break;
   case 'B' :
   case 'C' :
     System.out.println("Well done");
     break;
   case 'D' :
     System.out.println("You passed");
   case 'F' :
     System.out.println("Better try again");
     break;
```

```
        default :
            System.out.println("Invalid grade");
        }
        System.out.println("Your grade is " + grade);
    }
}
```

BRANCHING AND LOOPING:

→These are used to execute a block at statements repeatedly until a condition is specified.

→Java's iteration statement are for, while and do-while. These statements create what we commonly called loops.

- While loop:
  - The while loop is Java's most fundamental loop statement
  - It executes a statement or block of statements until the condition is true.

    Syntax: while (condition){
    //body of Loop
    }
  - The body at the loop will be executed as long as the conditional expression is true. When condition become false, control passes to next line at code.

Example:

```
class while_example
{
```

```
public static void main ( String args[])
{
    int i=1;
    while(i<=10)
    {
    System.out.println("I="+i);
    i++;
    }
    }
}
```

- The while loop evaluate if conditions expressions at top.
- The body at while or any other java's loop can be empty, this is because a null statement is syntactically valid in Java.

- Do-while loop:
  - The do while loop always executes its body at least once because its conditional expressions is at the bottom at the loop.

    Syntax: - do
    {
    //body of loop
    }while (condition);
  - Each iteration at the do-while loop first executes the body of loop and then evaluate the conditional expression.

    If this expressions is true, the loop will repeat-otherwise the loop terminates.

    Example:

```
class dowhile_example
{
public static void main(String args[])
{
int i=1;
do
{
System.out.println("I="+i);
i++;
}while (i<=10);
}
}
```

- The do-while loop is especially useful when you process a menu selection because you will usually want the body at a menu looping to execute at once.

- **For Loop:**
  - For loop provides a compart way to iterate the range at values.
  - These are used at arrays value.

    Syntax:- for(initialization; condition; iteration)

    {

    // body of loop

    }
  - When the loop first starts.
    i. The initialization position at loop is executed.
    ii. Next, the condition is evaluated.
    iii. Next, the iteration portion at loop is executed.
  - This process repeats until the controlling Expressions is false

    Example:

```
class for_example
{
public static void main (String args[])
{
 int i=1;
   for (i=1;i<=10;i++)
        {
        System.out.println ( "I="+i);
        }
}
}
```

- If only one statement is being repeated then there is no need for the curly braces.
- You can intentionally create. An infinite loop i.e a loop that never terminates; if you leave all there parts of the for empty.

```
for(: :)
{
//. . .
}
```