**FUNDAMENTALS OF OBJECT-ORIENTED PROGRAMMING**

**I. Introduction:**

Java was developed by James Gosling and his team members at Sun Micro systems. It was initially named Oak and later it was renamed as Java. Java is an Object oriented programming (OOP) language which was developed to support both standalone and internet applications.

**II. Object-Oriented Paradigm:**

Object-oriented programming is an approach that provides a way of modularizing programs by creating partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.

Object-Oriented Programming treats data as a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the functions that operate on it and protects it from unintentional modification by other functions.

OOP allows us to decompose a problem into a number of entities called Objects and then build data and functions (known as methods in java) around these entities. The combination of data and methods make up on object.

The data of an object can be accessed only by the methods associated with that object. However methods of one object can access the methods of other objects. Some of the features of object-oriented paradigm are:

* Emphasis is on data rather than procedure.
* Programs are divided into what are known as objects.
* Data structures are designed such that they characterize the objects.
* Methods that operate on the data of an object are tied together in the data structure.
* Data is hidden and cannot be accessed by external functions.
* Objects may communicate with each other through methods.
* New data and methods can be easily added whenever necessary.
* Follows bottom-up approach in program design.

**III.Basic concepts of Object Oriented Programming**

All object-oriented programming languages provide mechanisms that help you implement the object-oriented model. They are

a. Objects and classes

b. Data abstraction and encapsulation

c. Inheritance

d. Polymorphism

e. Dynamic binding

f. Message communication

**a. Object and classes:** Objects are the basic runtime entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program may handle. They may also represent user- defined data types such as vector and lists. Any programming problem is analyzed in terms of object and the nature of communication between them.

The entire se of data and code of an object can be made a user-defined data type using the concept of a class. Once a class has been defined, we can create any number of objects belonging to that class. A class is a collection of objects of similar type.

For example: Mango, apple and orange are members of the class fruits.

**b. Encapsulation and Data Abstraction:** The wrapping up of data and methods into a single unit is known as encapsulation. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world and only those methods, which are wrapped in the class, can access it. These methods provide the interface between the objects data and the program. This protection of the data from direct access by the program is called data hiding (Data abstraction).

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

**c. Inheritance:** Inheritance is the process by where objects of one class acquire the properties of objects of another class. 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.

**d. Polymorphism:** Polymorphism is another important OOP concept. Polymorphism means the ability to take more than one form.

For ex: An operation may exhibit different behavior in different instances. The behavior depends upon the type of data used in the operations. We can have two functions with same name but different number of parameters.

**Ex** : sum(int a,int b)

sum(float a, float b)

**e. Dynamic Binding :** Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding means that the code associated with a given procedure call in not known until the time of the call at runtime .

**f. Message communication:** An object-oriented program consists of a set of objects that communicate with each other. The process of programming in an object-oriented language, therefore, involvers the following basic steps:

1. Creating classes that define objects and their behavior.
2. Creating objects from class definitions.
3. Establishing communication among objects.

**IV. Benefits of OOP:**

OOP offers several benefits to both the program designer and the user. Object orientation contributes to the solution of many problems associated with the development and quality of software products. The principal advantages are:

* Through inheritance we can eliminate redundant code and extend the use of existing classes.
* We can build programs from the standard working modules that communicate with one another rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity
* The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the programs.
* It is possible to have multiple objects to coexist without any interference.
* It is possible to map objects in the problem domain to those objects in the program.
* It is easy to partition the work in a project based on objects.
* The data-centered design approach enables us to capture more details of a model in an implementable form.
* Object-oriented system can be easily upgraded from small to large systems.
* Message passing technique for communication between objects makes the interface description with external system much simpler.
* Software complexity can be easily managed.

**V. Applications of OOP:**

* Real-time systems.
* Simulation and modeling
* Object-oriented databases
* Hypertext hypermedia
* AI and expert systems
* Neural networks and parallel programming
* Decision support and office automation systems
* CIM/CAD system

**VI.Java Features :**

The people who invented java wanted to design a language which can provide solutions to the problems faced in modern programming. The following are the various features of java.

a. Compiled and interpreted

b. Platform independent and portable

c. Object-oriented

d. Robust and secured

e. Distributed

f. Simple, Small and Familiar

g. Multithreaded and interactive

h. High performance

i. Dynamic and extensible

**a. Compiled and interpreted:** Generally a computer language is either compiled or interpreted. Java combines both these approaches thus making Java a two-stage system.

1. First, Java compiler translates source code into byte code instructions. But byte codes instructions are not machine instructions.

2. In the second stage, Java interpreter generates machine code that can be directly executed by the machine that is running the Java program.

In this way Java is not only a compiler but also an interpreter language.

**b. Platform independent and portable:** A java program can execute on any operating system because of its byte code and Java Virtual Machine(JVM). So, Java programs can run on any platform.

Java supports portability. Java programs can be easily moved from one system to another, anywhere and anytime without any modifications to the code.

**c. Object-oriented:** Java is a true object-oriented language. Almost everything in Java is an object. All program code and data reside within objects and classes.The object model in Java is simple and easy to extend.

**d. Robust and secured:** Java program is either free.Java is a robust language. We can prevent compilation errors as well as run time errors.Run time errors can be handled through exception handling.

Security is very important issue for a language that is used for programming on Internet. Java not only verifies all memory access but also ensure that no viruses are communicated with an applet.

**e. Distributed:** Java is a distributed language. It has the ability to share both data and programs. Java applications can open and access remote objects very easily. This enables many programmers residing at different locations and work together on a single project.

**f. Simple, Small and Familiar:** Java is a small and simple language. Many features of C and C+ + are not included in Java because they are redundant.

For example, Java does not use pointers, preprocessor header files, goto statement, preprocessors,header files,operator overloading,multiple inheritance etc.

Java is a familiar language. It is model on C and C+ + languages. Java code "looks like a C++” code. In fact, Java is a simplified version of C++.

**g.Multithreaded and interactive:** Multithreaded means handling multiple tasks simultaneously. Java supports multithreaded programs. This means that we need not wait for the application to finish one task before beginning another.

The Java runtime comes with tools that support multiprocessors synchronization and construct interactive systems running smoothly.

**h.High performance:** Because of using intermediate byte code, Java performance is excellent for an interpreter language.Java programs can be executed speedily when compared to C,C++ programs.

**i.Dynamic and extensible:** Java is a dynamic language which is capable of dynamically linking to new class libraries, methods, and objects.

Java also supports extensibility.It supports functions written in other languages such as C and C++. These functions are known as native methods, which can be linked dynamically at runtime.

**Note: Some of additional features of java are**

|  |
| --- |
| • Ease of Development |
| • Scalability and Performance |
| • Monitoring and Manageability |
| • Desktop Client |
| • Core XML Support |
| • Supplementary character support |
| • JDBC RowSet |

**Overview of java language**

**Introduction:**

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

a.Stand-alone applications

b.Web applets

**a.Stand-alone applications:**

Stand-alone applications are programs written in Java to carry out certain tasks on a stand-alone local computer. Java can be used to develop programs for all kinds of applications which were developed using languages like C and C++. Executing a stand-alone Java program involves two steps:

1. Compiling source code into byte code using javac compiler

2. Executing the byte code program using java interpreter.

**b.Web applets:**

Applets are small Java programs developed for Internet applications. An applet located on a distant computer (Server) can be downloaded via Internet and executed on a local computer (Client) using a Java-capable browser. We can develop applets for doing everything from simple animated graphics to complex games and utilities. Since applets are embedded in an HTML (Hypertext Markup Language) document and run inside a Web page, creating and running applets arc more complex than creating an application.

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

**Simple Java Program Structure:**

Java program consists of one or more classes.Only only one of this classes defines a main method. A class contain data members and methods that operate on the data members of the class. Methods may contain data type declarations and executable statements. To write a java program we first define classes and then put them together.A java program may contain one or more sections.The general syntax of a java program is shown below

.

Documentation Section

Package Statement

Import Statement

Interface Statement

Class Definitions

main method class

{

main ( ) definition

}

Suggested

Optional

Optional

Optional

Optional

Essential

**Documentation Section**

The documentation section comprises a set of comment lines giving the name of the program, the author and other details.The contents of the Comment lines will be ignored by the compiler.Java supports 3 styles of comments.They are

i.The first one multiline comment which may contain several lines.This type of comment begins with /\* and ends with \*/.

ii.The other one is single line comments starts with // and ends with end of the line.

iii.Java also uses a third style of comment /\*\*...\*/ known as documentation comment.

This section is important as it helps in understanding the purpose of the program.

**Package Statement**

This is the first statement in every Java program if needed. This statement tells the compiler that the classes defined in the program belongs to this package. The package statement is optional. Example: package student;

**Import Statements**

This is the next statement after a package declaration but should be written before defining a class.But this statement is similar to the #include statement in C.

Example **import java.io.\*:**

Here the import statement instructs the interpreter to load the classes contained in the package.

**Interface Statements**

An interface is like a class but includes a group of method declarations. The general format of interface statement is

Interface interfacename

{

Variable declaration;

Methods declaration;

}

This is also an optional section and is used only when we wish to implement the multiple inheritance feature in the program.

**Class Definitions**

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

**main Method Class**

This is essential section of a Java program. Every java program must have a class definition that defines the main method. This is essential because main() is the starting point for running java stand-alone programs. The main() method creates objects of various classes and establishes communications between them. The program terminates on reaching the end of the main method.

**Simple java Program**

1. Type the java program in any editor.

Ex : Notepad.

class SampleOne

{

public static void main (String args[ ])

{

System.out.println(“Welcome to Java programming");

}

}

Save the program with file name as that of class name with extension .java.

Ex : Sampleone.java

**About Java Program:**

**\*** Any java program starts with a class because java strictly follows object oriented concepts.Every class will have a class name.

\*Every class definition in java begins with an opening brace “{” and ends with a matching closing brace appearing in the last line in the example.

\* The third line public static void main (String args[ ])

The above line contains a number of keywords, public, static and void.

**public:**The keyword public is an access specifier that declares the main method as unprotected and therefore making it accessible to.all other classes. This is similar to the C++ public modifier.

**static:** Next appears the keyword static, which declares this method as one that belongs to the entire class and not a part of any objects of the class. The main must always be declared as static since the interpreter uses this method before any objects are created.

**void:** The type modifier void states that the main method does not return any value.

**main():**This is similar to the main() function in C/C++. Every Java application program must include the main() method. This is the starting point for the interpreter to begin the execution of the program. A Java application can have any number of classes but only one of them must include a main method to initiate the execution.

Here, String args[ ] declares a parameter named args. This method takes arguments as parameters which are of string type. This is called command line arguments.

**String:** String is a wrapper class which is used to accept string values that is group of characters. The only executable statement in the program is System.out.println(“Welcome to java programming”);

"System" is a wrapper class. "out" represents standard output device object and "println" is a method to display the given message on standard output device.

2.Compile the java program at DOS prompt using javac

Ex : c:\jdk1.6\bin>javac SampleOne.java

If there are any errors in the program then the compiler shoots the error messages, otherwise a class file is created ( Ex : SampleOne.class ) and again the DOS prompt is shown.

3. Execute the java program with java.exe as follows :

Ex : c:\jdk1.6\bin>java SampleOne

Welcome to java programming

c:\jdk1.8\bin>

5. On successful execution the output is obtained and again the DOS prompt.

**Java Character Set**

This is the set of characters like Alphabet, numbers and special characters used to create java tokens that are used to write the program. This characters are defined by the **Unicode** characterset. Developers are trying to support different languages in the world that can be used to write computer programs.

**Java Tokens:**

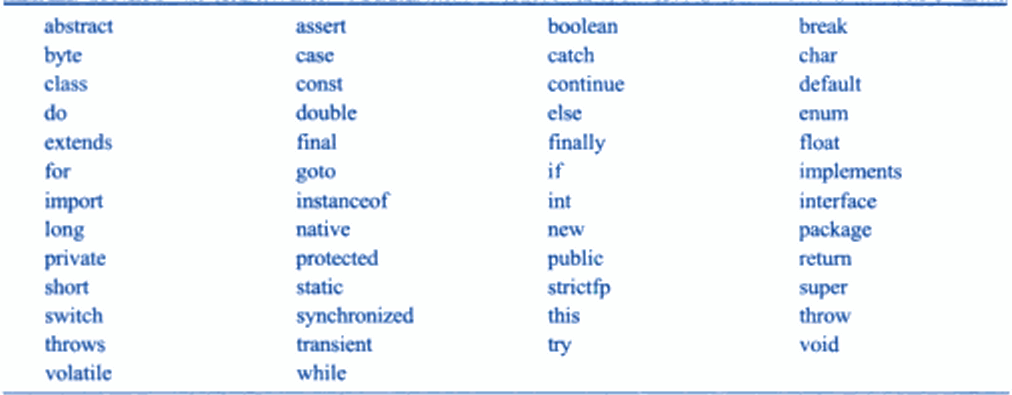
A class is defined by a set of declaration statements and methods containing executable statements. Most statements contain expressions which describe the actions carried out on data. *Smallest individual units in a program are known as tokens.* The compiler recognizes them for building up expressions and statements.

A java program is a collection of tokens, comments and white spaces. Java language includes five types of tokens. They are

* Reserved keywords
* Identifiers
* Literals
* Operators
* Separators

**Keywords**

Keywords are an essential part of a language definition. They implement specific features of the language. Java language has reserved 50 words as keywords. These keywords combined with operators and separators according to a syntax form definition of the java language.



**Identifiers**

Identifiers are programmers-designed tokens. They are used for naming classes, methods variables objects labels packages and interfaces in a program. Java identifiers follow the following rules.

1. They can have alphabets digits ,underscore and dollar sign characters.
2. They must not begin with a digit
3. Blank spaces, special characters are not allowed except dollar sign($) and underscore( \_ ) character.
4. They must not be a keyword.
5. Uppercase and lowercase letters are distinct
6. They can be of any length

**Literals**

Literals in java are a sequence of characters (digits, letters, and other characters) that represent constant values to be stored in variables. Java language specifies five major types of literals. They are:

* Integer literals (ex:10)
* Floating-point literals(ex:25.6)
* Character literals (ex: ‘b’)
* String literals (ex:”java”)
* Boolean literals (ex:true)

**Operators**

An operator is a symbol that takes one or more arguments and operates on them to produce a result. Operator specify an evaluation or calculation to be performed on data or objects.

**Separators**

Separators are symbols used to indicate where groups of code are divided and arranged. They basically define the shape and function of our code.

* **parentheses ( ):**Used for defining precedence in expressions and Used to contain the values of automatically initialized arrays.
* **braces { }:** To define a block of code for classes, methods and used to contain the values
* **brackets [ ]:** Used to declare array types
* **semicolon ; :** Used to separate statements
* **comma , :** Used to separate consecutive identifiers in a variable declaration
* **period . :** Used to separate package names from sub-packages and classes.

**Java Statements**

The statements in java are like sentences in natural languages. A statement is an executables combination of tokens ending with a semicolon (;) mark. Statements are usually executed in sequence in the order in which they appear. However it is possible to control the flow of execution if necessary using special statements. Java implements several types of statements.

* Empty Statement
* Labeled Statement
* Expression Statement
* Selection Statement
* Iteration statement
* Jump statement
* Synchronization statement
* Guarding statement

**Implementing a java program**

Implementation of a java application program involves a series of steps.

* Creating the program
* Compiling the program
* Running the program

In the java program before we begin creating the program, the *Java Development Kit (*JDK) must be properly installed on our system.

**Creating the program**

We can create a program using any text editor. Assume that we have entered the following program:

**Example:**

class test

{

public static void main (String args [ ])

{

System.out.println(“Hello”);

System.out.println(Welcome to the world of java”);

System.out.println(“Let us learn java”);

}

}

We must save this program in a file called test.java ensuring that the filename contains the class name properly. This file is called the source file. All java source files will have the extension java. If a program contains multiple classes the file name must be the classname of the class containing the main method.

**Compiling the program**

To compile the program we must run the java compiler.

**Syntax: javac filename.java**

If no errors in the source program, the java compiler creates a class file that contains the byte code of the program. The byte code file contains the file extension ‘.class’.

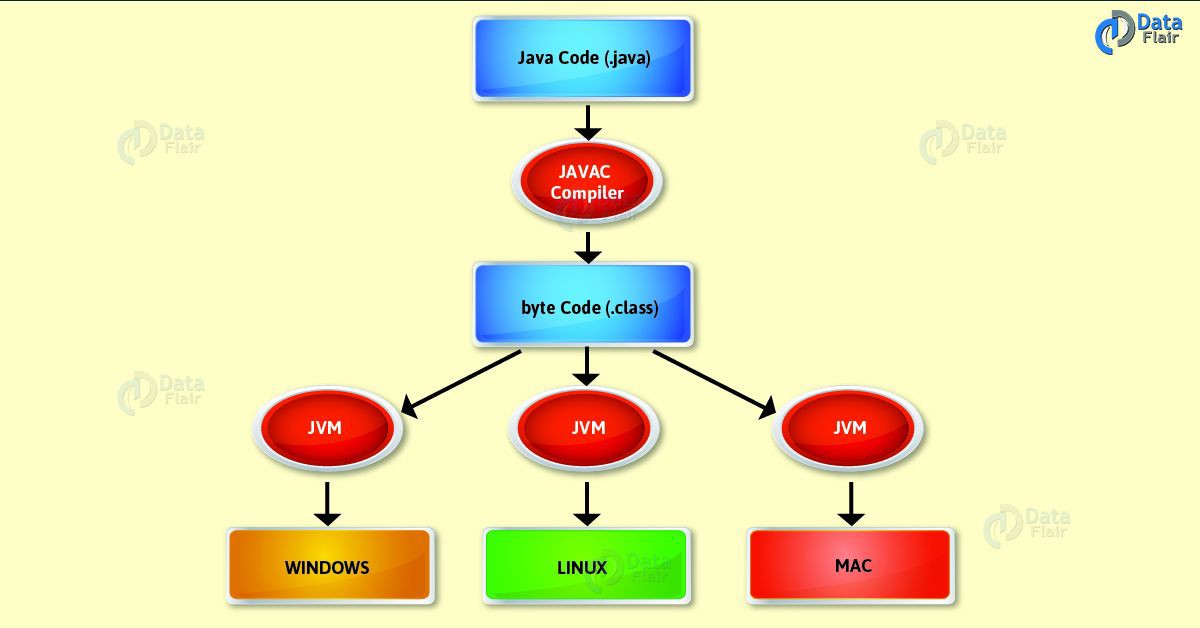
**Running the program**

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

**Syntax: java classname**

Now the interpreter looks for the main method in the program and begins execution from there. When executed our program displays the output.

The following figure explain the implementation of java program

****

**Java Virtual Machine**

Java is both compiled and interpreted language. Firstthe java compiler translates the source code into byte code instructions. In the next stage the java interpreter converts the byte code instructions to machine code that can be directly executed by the machine running the java program. The intermediate code namely the bytecode produced by the java compiler is for a machine. This machine or simulated computer within the computer is called the *Java Virtual Machine* (JVM) and it exists only inside the computer memory. The following figure illustrates the process of compiling a java program into bytecode which is also referred to as virtual machine code.



The virtual machine code is not machine specific. The machine specific code (know as machine code) is generated by the java interpreter by acting as an intermediary between the virtual machine and the real machine as shown in the figure



**Command line arguments**

Command line arguments are parameters that are supplied to the application program at the time of invoking it for execution.

We can write java programs that can receive and use the arguments provided in the command line.In the main() method

public static void main (String args[])

As pointed out earlier ‘args’ is declared as an array of strings (known as string objects). Any arguments provided in the command line (at the time of execution) are passed to the array args as its elements. We can simply access the array elements and use them in the program as we wish.

Example: java test BASIC FORTRAN C++ java

This command line contains four arguments. These are assigned to the array args as follows

BASIC args[0]

FORTRAN args[1]

C++ args[2]

JAVA args[3]

The individual elements of an array are accessed by using an index or subscript like args[i]. The value of I denotes the position of the elements inside the array.

Example:commandline arguments demonstration

class Commanddemo

{

public static void main(String args[])

{

int a,b,c,d;

a=Integer.parseInt(args[0]);

b=Integer.parseInt(args[1]);

c=Integer.parseInt(args[2]);

d=a+b+c;

System.out.println("a value is"+a);

System.out.println("b value is"+b);

System.out.println("c value is"+c);

System.out.println("d value is"+d);

}

}

To compile: javac Commanddemo.java

To run: java Commanddemo 1 2 3

**Constants, variables and data types**

**Introduction:**

A programming language is designed to process certain kinds of data consisting of numbers, characters and strings and to provide useful output known as information. The task of processing data is accomplished by executing a sequence of instructions constituting a program. These instructions are formed using certain symbols and words according to some rigid rules known as syntax rules (or grammar). Every program instruction must conform precisely to the syntax rules of the language.

Like any other language, Java has its own vocabulary and grammar.

**Constants**

Constants in java refer to fixed values that do not change during the execution of a program. Java supports several types of constants has shown in the following figure.

**JAVA CONSTANTS**

NUMERIC CONSTANTS CHARACTER CONSTANTS

CHARACTER STRING

INTEGER REAL CONSTANTS CONSTANTS

CONSTANTS CONSTANTS

**Integer constants**

An integer constant refers to a sequence of digits. There are three types of integers, namely,

1. decimal integer

ii. octal integer

iii. hexadecimal integer.

**i.decimal integer:**Decimal integers consist of a set of digits, 0 through 9 preceded by an optional minus sign. Valid examples of decimal integer constants are

123 -321 0 654321

**ii.octal integer:**Octal integer constant consists of any combination of digits from the set(0 to 7) with a leading 0.

Ex: 037, 045,… etc.

**iii.hexadecimal integer:** A sequence of digits precided by 0x, 0X is considered as hexadecimal integers.They may also include Alphabets ‘A’ through ‘F’ or ‘a’ through ‘f’.The letters ‘A’ through ‘F’ represents the numbers ’10 through 15’.

Ex: 0X5, 0x6f,…. Etc

**Real constants**

Integer numbers are inadequate to represent quantities that vary continuously such distances heights temperatures prices and so on. These quantities are represented by numbers containing fractional parts like 17.548. such numbers are called real (or floating point) constants.

**Character constants:**

**Single character constants:**A single character constant(or simply character constant) contains a single character enclosed within a pair of single quote marks. Examples of character constants are

Ex: ‘5’ ‘X’ ‘;’ ‘ ‘

**String constants:** A string constant is a sequence of characters enclosed within a pair of double quotes. The characters may be alphabets, digits, special characters and blank spaces.

Ex: “hello java” “1997” “well done” “?....!” “5+3” “X”

**Backslash character constants**

Java supports some special backslash character constants that are used in output methods. A list of such backslash character constants is given below

**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

**Variables**

A variable is an identifier which represents a storage location used to store a data value. 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. Some examples of variable names are

* Average
* Height
* Total\_height
* Classstrength

variable names may consist of alphabets, digits, the underscore(\_) and dollar characters subject to the following conditions

1. they must not begin with a digit
2. uppercase and lowercase are distinct. This means that the variable total is not the same as Total or total.
3. it should not be a keyword
4. white space is not allowed
5. variable names can be of any length

**Example**

//EXAMPLE ON WORKING WITH VARIABLES

class Variables1

{

public static void main(String arg[])

{

int a,b,c;

a=3;

b=3;

c=a+b;

System.out.println("Sum is "+c);

}

}

**Data types**

Java is a strongly typed language.Every variable in java has a data type. Data types specify the size and type of values that can be stored.Java language is rich in its data types. There are 8 simple data types defined by Java which can be classified into 4 groups.They are as follows



**Integer types**

Integer types can hold whole numbers such as 123, -96 and 5639. The size of the values that can be stored depends on the integer data type we choose. Java supports four tyes of integer.They are

i.byte

ii.short

iii.int

iv.long

Integer

long

byte

int

short

Java does not support the concept of unsigned types and therefore all java values are signed meaning they can be positive or negative. The memory size and range of all the four integer data types

**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 types**

Integer types can hold only whole numbers and therefore we use another type known as floating point type to when we want to hold numbers containing fractional parts we use floating point types numbers. They are two types

i. float

ii.double

**Size and range of floating point types**

**Type size minimum value maximum value**

float 4bytes 3.4e-38 3.4e+038

double 8bytes 1.7e-308 1.7e+308

**Example**

class FloatEx

{

public static void main(String[] args)

{

float a = 3.14f;

System.out.println("\n a = "+a);

}

}

**Character type:** In order to store character constants in memory java provides a character data type called **char.** The char type assigns a size of 2 bytes but basically it can hold only a single character.

**Boolean type:** Boolean type is used when we want to test a particular condition during the execution of the program.There are only two values that a Boolean type can take.They are

i. true

ii.false.

Boolean type is denoted by the keyword boolean and **uses only one bit of storage**.

**Declaration of variables**

In java, variables are the names of storage locations. After designing suitable variable names we must declare them to the compiler. Declaration does three things

1. It tells the compiler what the variable name is.
2. It specifies what type of data the variable will hold
3. The place of declaration (in the program) decides the scope of the variable
4. A variable must be declared before it is used in the program.

A variable can be used to store a value of any data type. The declaration statement defines the type of variable. The general form of declaration of a variable is as follows

**Syntax:** data type variable1,variable2……variableN;

Variables are separated by commas. A declaration statement must end with a semicolon.

**Example:** int count;

float x,y;

double pi;

byte b;

char c1,c2,c3;

**Giving values to variables**

A variable must be given a value after it has been declared but before it is used in an expression. This can be achieved in two ways

i.by using an assignment statement

ii.by using a read statement

i.**Assignment statement:**A simple method of giving value to a variable is through the assignment statement as follows

**Syntax:** Variable name =value;

**Example:** Initialvalue=0;

Finalvalue=100;

Yes=’x’;

It is also possible to assign a value to a variable at the time of its declaration.This takes the following form

**Syntax:** type variablename=value;

**Example:**

int finalvalue=100;

char yes =’x’;

double total = 75.36;

The process of giving initial values to variables is known as the initialization.

**ii.Read statement:**We may also give values to variables interactively through the keyboard using readLine() method.

**Example:**

import java.io.\*;

class reading

{

Public static void main(String args[])

{

DataInputStream in=new DataInputStream(System.in);

int a,b;

System.out.println(“enter a value”);

a=Integer.parseInt(in.readLine());

System.out.println(“enter b value”);

b=Integer.parseInt(in.readLine());

int c=a+b;

System.out.println(“result is”+c);

}

}

**Scope of variables**

Java variables are actually classified into three kinds. They are

* instance variables
* class variables
* local variables

Both Instance and class variables are declared inside a class. Instance variables are called with objects only.Each object has different values for this variables.

Class variables acts as a global variables of a class and each object of a class holds the same values of the class variables. Class variables are declared with the keyword static.Each class variable has only one memory locations.Instance variables and class variables are available to whole class.

Local variables are defined inside a methods.They are local to the method only in which they are declared.The scope of this variables is local with in that method.It means their scope starts from beginning of methods(i.e ‘{‘ brace) and ends at the ending of the method(i.e ‘}’ brace).Scope is defined as an area of the program where a variable is accessible.

**Symbolic constants**

These constants may appear repeatedly in a number of places in the program. One example of such a constants is 3.142, representing the value of the mathematical constant “pi”. A constant is declared as follows

**Syntax:**  final type symbolic\_name=value;

**Type casting**

The process of converting one type to other is called casting. Casting is offers necessary when a method returns a type different than the one we required. If the two types are compatible then java will perform the conversion automatically.

**Automatic conversion:**

When one type of data is assigned to other type of variable automatically. Automatic type conversion will take place if the following two condition are met

* The two types are compactable.
* The destination type is larger than the source type.

The int type is always large enough to hold all valid byte values.So, no explicit cast statement is required.This type of conversion is called automatic conversion or widening conversion. Casting into a smaller type may result in a loss of data.Casting a floating point value to an integer will result in a loss of fractional part.

**Casting compatible types or narrowing conversions:**

Automatic type conversions will not fulfill all needs.

For example: If we want to assign an int value to a byte variable, This variable will not be performed automatically because a byte is smaller than an int.This kind of conversion is sometimes called a narrowing conversions.

To create a conversion between two impactable types, you must use a cast. A cast is simple explicit type conversion. The general form of a type conversion is

**Syntax:** type variable1=(type) varialble2;

Example: int a;

byte b;

b=(byte)a;

Here if the int value is larger than the range of a byte. It will be reduced modulo(The remainder of an integer division by)the byte range. If a floating point value is assigned to an integer type. The fractional component is lost. The following are some of the casts that results in no loss of information.

**Casts that results in no loss of information**

Byte short, char, int, long, float, double

Short int, long, float, double

Char int, long, float, double

Int long, float, double

Long float, double

Float double

**Operator and expressions**

An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variable. Operators can be classified into a number of related categories. They are

i. Arithmetic operators

ii .Relational operators

iii. Logical operators

iv. Assignment operators

v. Increment and decrement operators

vi. Conditional operator

vii. Bitwise operators

viii. Special operators

**i.Arithmetic operators:** Java provides all the basic arithmetic operators. The operators +,-,\* and /.All work the same way as they do in other languages. These can operate on any built-in numeric data type of java.

In java the modulus operator ‘%’ can be applied to floating point types as well as integer types. The basic arithmetic operators are listed below

**Operator meaning**

+ addition or unary plus

- subtraction or unary minus

\* multiplication

/ division

% modulo division(remainder)

**ii.Relational operators:**The relational operators determine the relationship that one operand has to the other.The relational operators are most frequently used in the expressions that control the ‘if’ statement into the various loop statements.Java supports six relational operators that are listed below.

**Operator meaning**

< is less than

<= is less than or equal to

> is greater than

>= is greater than or equal to

== is equal to

!= is not equal to

**iii.Logical operators:** Java has three logical operators .They are

**Operator meaning**

&& logical AND

| | logical OR

! logical NOT

The logical operators && and | | are used when we want to form compound conditions by combining two or more relations.

**iv.Assignment operators:**Assignment operators are used to assign the value of an expression to a variable.The usual assignment operator is ‘=’. In addition java has a set of shorthand assignment operators which are used in the following form

V op=exp;

Where ‘V’ is a variable name, ‘exp’ is an expression and ‘op’ is a java binary operator. The operator op= is known as the shorthand assignment operator.

The assignment statement

V op=exp;

Is equivalent to

V=V op(exp);

With V accessed only once.

X+=Y+1;

This is same as the statement

X=x+(y+1);

The shorthand operator += means add y+1 to x or increment x by y+1

**v.Increment and decrement operators:**Java has two very useful operators not generally found in many other languages. These are the increment and decrement operators

++ and- -

The operator ++ adds 1 to the operand.The operator -- decreases one to the operand.

Example: ++m or m++

--m or m—

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

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

**vi.Conditional operator:** Java includes a special ternary operator(? :) that can include replace certain types of it then else statements.This operator is used to construct conditional expressions.The general form is

Exp1? Exp2 :Exp3

Where Exp1 Exp2 and Exp3 are expressions.Here Exp1 can be any expression that evaluates to booleanvalue.If ‘Exp1’ is true then ‘Exp2’ is evaluated.Otherwise ‘Exp3’ is evaluated.Both Exp2 and Exp3 are required to return the same type which cannot be void.

**vii.Bitwise operators:** Java has a distinction of supporting special operators known as bitwise operators for manipulation of data at values of bit level. These operators are used for testing the bits or shifting them to the right or left. Bitwise operators may not be applied to float or double.Java supports the following bitwise operators

**Operator meaning**

& bitwise AND

! bitwise OR

^ bitwise exclusive OR

~ one’s complement

<< shift left

>> shift right

>>> shift right with zero fill

**viii.Special operators:** Java supports some special operators of interest such as instanceof operator and member selection operator(.).

**Instanceof operator:** The instanceof is an 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. This operator allows us to determine whether the object belongs to a particular class or not.

Example: Person instanceof student

Is true if the object person belongs to the class student: otherwise it is false.

**Dot operator:**The dot operator (.) is used to access the instance variables and methods of class objects

**Example:** Person1.age // reference to the variable age

Person1.salary( ) // reference to the method salary( )

It is also used to access classes and sub packages from a package.

**Arithmetic expressions:**

An arithmetic expression is a combinations of variables constants are operator as per the syntax of the language. We have used a number of simple expressions in the examples. Java can handle any complex mathematical expression.

**Algebraic expression java expression**

a b-c a\*b-c

(m+n)(x+y) (m+n)\*(x+y)

ab a\*b/c

3x2+2x+1 3\*x\*x+2\*x+1

x x/y+c

y

**i.Evaluation of expressions:** Expressions are evaluated using an assignment statement of the form

Variable= expression

Variable is any valid java variable name. when the statement is encountered the expression is evaluated first and the result then replaces the previous value of the variable on the left-hand side. All variables used in the expression must be assigned values before evaluation is attempted.

Example:x=a+b-c;

y=b/c\*a;

z=a-b/c+d;

**ii.Precedence of arithmetic operator:**An arithmetic expression without any parentheses will be evaluated from left to right using the rules of precedence of operators. There are two distinct priority levels of arithmetic operators in java

High priority \* / %

Low priority + -

The basic evaluation procedure includes two left-to-right passes through the expression. During the first pass the high priority operators(if any) are applied as they are encountered.During the second pass the low priority operators (if any) are applied as they are encountered.Consider the following statement

x=a-b/3+c\*2-1

When a=9, b=12 and c=3 the statement becomes

x=9-12/3+3\*2-1

x=9-12/3+3\*2-1

and is evaluated as follows

first pass

step1: x=9-4+3\*2-1 (12/3 evaluated)

step2:x=9=4+6-1 (3\*2 evaluated)

second pass

step3: x=5+6-1 (9-4 evaluated)

step4: x=11-1 (5+6 evaluated)

step5: x=10 (11-1 evaluated)