PROGAMMING IN JAVA(5BCA2) PART-1

INTRODUCTION

JAVA is an object – Oriented programming language . It is general purpose object- oriented programming language. JAVA was developed in 1991. It is the first technology of fully integrate. A programming language and services or threads, sockets, GUI components etc. So, that we can built programs that an run on a wide range of different types of hardware architecture and many types of operating system.

HISTORY OF JAVA

JAVA was developed y team of programming at “ SUN MICROSYSTEM” in 1991. The team consist of fie members these are------------------- James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan. JAVA creator James Gosling did not design JAVA or the internet. His objective was to create a common development environment for consumer of electronic device such as Microwave oven and remote control which was easily portable from one device to another . Te JAVA was initially called “OAK” but was renamed as JAVA in 1995.

IMPORTANT FEATURES OF JAVA

The inventors of JAVA wanted to design a language which could offers solution to some of the language to not be only reliable, portable and distributed but also simple compact and interactive.

Some important features of JAVA programming are----

1. SIMPLE-- Java is a simple programming language that helps to do complex program easily.
2. PLATFORM INDEPENDENT-- Java is a platform independent programming language. Platform independent means program written on one platform can be run easily on another platform. Due to this platform independent feature, we can say that a Java can be “ write once run anywhere”.

For Ex. :- If we write a program on Windows O.S. It can be easily run on another platform like- Unix, Linux. Mac etc. Java is a platform independent programming language due to Byte code. [Byte code is an intermediate language which can be generated after compilation of Java code and this byte code an be executed on any another platform which has JVM(Java Virtual Machine).]

3. OBJECT ORIENTED LANGUAGE :- Object oriented is an approach to program organization and development that attempts to eliminate some of the drawbacks of conventional programming method by incorporating the basis of structured programming features with several powerful new concept. The following features of object oriented programming are--------------------

(a.) Class :-- A class is a user-defined datatype used to represent a template (format) or several similar type of object. The principle of object oriented programming is to combine both data and the associated functions into a single unit called class.

Class follows hiding principle so, unauthorized user cannot view our document.

(b.) Object :-- An object is a basic building block or object oriented programming. Object corresponds to real life implement.

An object is a unit of software consist of three characteristics :-

i.) State or Attributes

ii.) Behavior or Function

iii.) Identity

________ The " State or attributes" refers to the built-in-characteristics of an object. Whereas " Behavior or operations" of an object refers to its action.

For Example—for duster:- name, weight, length, and color are its state/attribute whether erasing is its behavior.

(c.) Data Abstraction :-- The internal details of the objects are hidden which makes the user abstract and this technique o hiding internal details in an object is called data abstraction.

(d.) Encapsulation :-- Grouping of data and function together through specific mechanism refers to encapsulation. We can use the encapsulated object through he designated interface only. Encapsulation can hide data from classes and function in other classes.

(e.) Inheritance:-- If a derived class inherits all the attributes and behavior of the base class that type of process is known as inheritance. The original or parent class is known as the base class and the child class is known as the derived class.

Base class → Parent class
|
|
Derived class → Child class

(f.) Polymorphism:-- The term polymorphism is made up of two term " Poly" means many and " Morphism" means form. It provides a common interface to carry out similar tasks. In simple, We can say that one name, many duties. Polymorphism is a useful concept in OOPs language and various high level languages.

JVM(Java virtual machine) is an environment for . It is a software which can be use to execute a byte code.

(g.) Robust :-- Java is a Robust(strong) language due to object oriented features. We cannot implement or execute or program outside the class.

(h.) Secure:-- Java is a secured object oriented programming language because each and every code should be written within a class.

<u>SOURCE</u>

Program → Java compiler → Byte code → Executor code
^
Machine code/ JVM

OBJECT ORIENTED CONCEPT

<u>Simple Java Program</u>

```
Class   test
{
        Public  Static void main(string args [])
         {
               System.out.print("Suraj");
          }
    }
```

Class:-- Class is a keyword used to declare a class, every java program must contain at least one class specification.

Main():-- All the java application begins execution at main method(). It accept one argument called args[]. i.e. an array of string object. The interpreter starts execution from the main().

Public:-- Public is access specifier indicates that the method (i.e. main method) associated with class test and only one existence of that class is created.

Void:-- Void is a keyword used to specifies the return type of function. If we specify the void keyword, It indicates that this method doesn't return any method.

Static:-- It is a keyword indicates that this method (i.e. main method) associated with class test and only one existence of that class is created.

.out.print:-- It is the output statement which prints some specific message on screen where system is predefined object of system class and print is predefined method of system class.

Difference between C++ and Java :--

(a.) Java doesn't support operator overloading.
(b.) Java doesn't have a template classes as in C++.
(c.) Java doesn't support the concept of multiple inheritance this can be done using a new feature called interface.

Interface is like a class which contains several classes and methods.

(d.) Java doesn't support global variables every variables and method is declared within the class.
(e.) Java doesn't use pointer.
(f.) Java has replace the destructor function with the help of finalize().
(g.) There is no header file in Java.

JAVA CLASS LIBRARY :-

Java provides a rich set of class libraries that provides a large number of classes and methods that any java program can use it. These libraries are used for Input/output, mathematical, networking, events and many other capabilities. These class library summarise the primary function. These are grouped into various package. A packages is a collection of classes. The various java class libraries/packages are given below:-

(1.) Java.io-> This package supports input/output operations.
(2.) Java.applet -> This package support to built applet program.
(3.) Java.net -> This package supports to enable networking concept in java programming.
(4.) Java.util -> This package supports the java utilities. It is default package for java programming.
(5.) Java.lang -> This package supports java functionalities.
(6.) Java.beans -> This package supports to java software components.
(7.) Java.awt → This supports to design abstract windows toolkit(awt) to construct GUI environment.
<a.> Java.awt.event :- This package supports to handle events from awt component.
<b.> Java.awt.image :- This package supports to perform image processing.

STRUCTURE OF JAVA PROGRAM :-

Java program is the construction of classes and class contains data members and methods that operates on following structure shown in the given below-----------

1. Documentation section
2. Package section
3. Import section
4. Interface section
5. Class definition section
6. Main method class

```
{
   Main method definition
   {
        ------------------
        ------------------
   }
}
```

e.g. :- / ** A program to display a message */

```
Import java.awt

Class test

{

    Public static void main( String args[])

    {

          System.out.print(" first java program");

     }

}
```

(1.) Documentation section :- This section usually contains the comment lines used to specify the name of the program and other details. This section is optional.

a. /* --------------------- */(Multiline comment)
b. // -------------------------(Single line comment)
c. /** ---------------------------- */

It is used for document file, which store statements of this line in another file. Document file is useful for user.

(2.) Package section :- This section is used to declare the package name that which is used to inform the computer that compiler defined here belongs to this package. This section is optional

(3.) Import Section :- This section is used to import the specified class or package like as we include a file in C++ using include statement. This section is optional

(4.) Interface Section :- This section is used to implement the interface concept and includes a group of method declaration. This section is optional. Interface is just like a class.

(5.) Class Declaration :- The class is the primary and essential element of a java program and the java program contains multiple class definition. These classes are used to the objects. This is optional

(6.) Main method Definition class :- The main method specifies the storing point of a java program. This is essential part of a java program. This is used to create objects of various classes and also used to establish communication between them.

Open a text editor (notepad) to create a file that contains the program named as first . java . The file name must matched the class that we declare Java is case (statement) sensitive so be careful wile entering both the name of file and its contents.

COMPILING AND RUNNING A JAVA PROGRAM :-

Compilation :- Before executing a Java program, The program must be compile using the javac statement as shown below----

Step1 – Save the file in specific directory

e.g. – D:\java program\first.java

Step2 – Then open the DOS command prompt.

Step3- Then go to my computer → C drive → program files → java → JDK(java development kit) → Bin → Ten select the path and copy the path.

Step4- In DOS command prompt type or paste the copied path as the statement given below----

```
C:\Priya\>path=”C:\program files\Java\JDK\Bin   <----
```

Step5.- Then press enter key then compile our program by using javac compiler.

Step6 – It will create the byte code which is platform independent.

Step7 – Run our program with the help of Java interpreter by providing file name along with Java.

Java source code	Java compiler	Java virtual machine

JVM is a environment (application software) which convert Byte code into executable code.

VARIABLES :--

The variable is the basic unit of storage in a java program. Its value varies at the time of execution. A variable is defined by the combination of an identifier and an optional initializer.

Declaring a variable :-

Variables must be declared before use in java program. The syntax of variable declaration is ----------------

Syntax :- Datatype var1, var2, ------- varn;

Where,

Datatype is the type of data var1,var2,------------- var n are the list of variable.

e.g. string std name;

in mark1, mark2 , mark3;

float arg;

Rules for naming a variable :-

1. It must begin with alphabet.
2. It is case sensitive i.e. the variable name is different from name.
3. A keyword is not used as a variable.

Datatype:-

1. Predefined
   a. Numeric
   b. Non numeric

2. User defined
   a. Class
   b. Array
   c.Interface

Numeric

1. Integer
   |
   a. Byte- 1 byte
   b. Short- 2 byte
   c. Long- 8 byte
   d. Int- 4 byte

2. Float
   |
   a. Float- 4byte
   b. Double- 8byte

Non- Numeric

1. Char- 2byte
2. Boolean- 1byte

KEYWORDS USED IN JAVA :--

Abstract	Float	Catch
Through	Public	Switch
Constant(cons)	Interface	Import

Boolean	Return	Class
Finally	Long	This
Continue	Go to	Throws
Int	Double	Short
Native	Static	Super
If	New	Package
Else	Implement	extends
Volatile	Char	For
Private	Synchronized	Do
Final	Instance of	Void
While	Default	Until
Break	Byte	Protected
Tangent	Try	

Some Important Points :--

- JVM(java virtual machine) is an environment (application software) which convert byte code into executable code.
- Every term in Java program which starts from upper case i.e. Capital letter is predefined class and that term which starts from small letter is either method or package.
- / Character constant is used to formatting output statement. It is the combination of two character as like------------- /n,/t,/a and so on but give single character.
- Dot operator (.) is known as member selection operator.

- Instance is created in class.
- Instance of () is the special type operator which returns either an object belongs to the given class or not.

```
e.g. – Class    test

           {

              --------
              ---------
            }
             Test t1,t2,t3;
        T1.instance of (test)
```

- Finally is used at the place of destructor.
- If we want to use another program code to our java program this process is done with the help of native keyword.
- Import is used to use package in a program.
- Volatile is a keyword used to create a variable which stores data temporarily.
- Super keyword is used to call constructor of base class.
- Static create an existence of a variable present anywhere.
- Implement keyword is used with interface in case of inheritance.
- Boolean is used to hold true or false.

❖ CONSTANTS :- Constants is a fixed value which doesn't vary during the program execution. It can be divided into two types------

(i) Numeric constant
(ii) Non- numeric constant

❖ OPERATOR :- An operator is a symbol that tells the computer to perform certain mathematical and logical manipulation. It is used in program to manipulate data and variables.

In java different types of operator is used which are----------

(i) Arithmetic operator
(ii) Logical operator
(iii) Relational operator
(iv) Bitwise operator

1) Arithmetic operator—Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra. The basic arithmetic operators supported by java are-----

+ (Addition), -(Subtraction), *(Multiplication), /(Division), %(Modulus), ++(Increment), --(Decrement) and =(Assignment).

2) Logical operator-- & (and), |(or), and !(Not) are logical operators used in java.
3) Relational operator— The relational operator determines the relationship between two operands. The outcome o these operations is a Boolean value. ==(equal to), !=(not equal to), >(greater than), <(less than), >=(greater than equal to), <=(less than equal to) are relational operators.
4) Bitwise operator:-- The bitwise operator is used for the bitwise logical decision making. Bitwise operator can be applied to the integer types, long, int, short, char and byte. The following bitwise operators used in java are------

(a.)	<<	(Shift left)
(b.)	>>	(Shift right)

(c.) >>> (Shift right zero fill)
(d.) <<< (Shift left zero fill)
(e.) ~ (Unary complement)
(f.) | (Bitwise inclusive OR)
(g.) ^ (Bitwise exclusive OR)

Except above operators java facilitates Comma operator, Ternary operator (also called Conditional operator) and Instance of (type of comparison operator)

- ❖ Conditional Statement :--

Conditional statement control the flow of execution to advance and branch based on changes to the state of program. Java language uses following Conditional statement-----

(i) Selection statement
(ii) Iteration statement
(iii) Break statement
(iv) Continue statement
(v) Goto statement

Selection statement include Simple if, If-else, Nested if-else, Switch statement etc. Iteration statement is also called Looping statement which include While, Do-while, For loop etc. Break statement is used to terminate a statement. A continue statement causes control to be transferred directly to the conditional expression that controls the loop. Goto statement is used to ump on a specific statement.

Q. W.A.P. to print name and address.

Solu. Class test

```
   {
      Public static void main (String args[])
       {
           System.out.println("Teaswini");
            System.out.prinln("K.g. road");
       }
   }
```

Q. W.A.P. to add two numbers.

Solu.   Class   add

```
   {
        Public      static void main(String args[])
        {
              Int a, b, Sum;
                A=5;
                B=8;
                 Sum=a+b;
               System.out.println(" sum of two number=" +sum);
         }
   }
```

Take input through keyboard :--

```
    Import java.util.Scanner;
    Public class testinput
{
        Public static void main(String args[])
     {
           Scanner sc=new Scanner (System.in)
                 Int a,b, sum;
             System.out.println(" Enter first no.=");
                 A=sc.nextInt();
              System.out.println(" Enter second no.=");
                  B=sc.nextInt();
                 Sum=a+b;
               System.out.prinln("Sum of two number="+sum);
      }
}
```

Q. W.A.P. to add digits of a number.

Solu.

```
Import java.util.Scanner;
 Class      Digit
{
       Public static void main (String args[])
```

```
        {
             Scanner sc=new scanner(System.in);
                  Int a, Sum;
                 Sum=0;
             System.out.println("enter a no.=");
              A=sc.nextInt();
               While (a!=0)
              {
                   Sum= suum+a%10;
                    A= a/10;
               }
               System.out.println ("\n Sum of digits="+sum);
        }
   }
```

Q. W.A.P. to return reverse of a number.

Solu.

```
Import java.util.Scanner;
  Class        reverse
 {
     Public static void main (String args[])
     {
```

```
                    Scanner sc=new scanner (System.in);
                         Int a, Sum,Rev;
                          Sum=0,Rev=0;
                      System.out.println("Enter a number=");
                           A=sc.nextInt();
                           While(a!=0)
                      {
                            Sum=a%10;
                               A=a/10;
                             Rev=rev*10+sum;
                       }
               System.out.println("\ Reverse of a number="+rev);
             }
       }
```

CLASS:--

A class is a user-defined datatype like structure in C. In the class the members (datatype) and methods are defined. Once the class is defined we can create the object which are called Instances of class.

```
               Class        Class_name
          {
               Variable_declaration;
```

```
                    Variable_ definition;
             }
```

Note: [There is no semi-colon (;) at the end of closing braces.]

CREATING OBJECT :--

Creating an object is the process of allocating block of memory that contains spaces. To store all the variables declare within a class (Instance variable).

The new keyword can be used to create object for specified class and return references of that object.

```
Syntax:--     Class_name      object_name
              Object_name = new  class_name();
                          OR
              Class_name     Object_name = new class_name();

e.g.--              Class       Student
             {
                  Int roll;
                  Int age;
                    Void input();
                    {
                      -----------
                      -----------
                    }
                      Void    display()
                   {
                        ------------
                        ------------
                   }
```

1) Student s= new student();

OR

2) Student   s;

```
        S= new Student();
```

ACCESSING CLASS MEMBER :-

To access the class member we must assign some values before accessing it in the program. The following statement shows how the object are assigned and class members are accessed.

```
e.g.--          Class           Student
             {
                   Int    roll;
                   Int    age;
                     Void input()
                 {
                      --------------
                      ----------------
```

```
            }
                  Void display()
              {
                    -------------
                    --------------
               }
                 Student    s=   new student  ();
                          s.roll=50;
                          s.age=20;
                         s.input();
                         s.display();
```

CONSTRUCTOR:-

Constructor is a method that initialize each object when it is created. Constructor function have the same name of the class.

e.g. :-

```
                    Class          add
        {
                 Int a,b;
                  Add(int x, int y)
              {
                      A=x;
                      B=y;
                    }
                  Int addition()
                {
                      Return(a+b);
                }
    }
               Class     printdata
          {
                   Public static void main(String args[])
               {
                      Add ab = new add(100,25);
                       Int s = ab.addition();
                     System.out.println("sum= "+5);
               }
          }
```

Input Data Through Keyboard :--

```
              Import java.util.Scanner;
              Class     add
   {
            Int a,b;
            Add(int  , int y)
         {
              A= x;
              B= y;
         }
           Int    addition()
        {
              Return  (a+b);
        }
}
  Class   printdata
{
       Public static void main(string args[])
    {
         Scanner  sc= new  Scanner (System.in);
                  Int a,b;
        System.out.println("enter first number = ");
            A=  sc.nextInt();
        System.out.println("enter second number = ");
             B= sc.nextInt();
  Add   ab = new add();
   Int   s= ab.addition();
 System.out.println("Sum of number = "+s);
}
}
```

METHOD OVERLOADING :--

1. Method overloading is also called function overloading.
2. It is the process o polymorphism.
3. It means using the same function name for different purpose the appropriate function is executed depending upon the argument passed to it.

Q. WAP. Or demonstrating function (method) overloading.

Solu.

```
Import java.util.Scanner;
   Class      add
{
     Int  a, b;
     Void    addition(int x , int y)
   {
        A= x;
        B= y;
   }
     Int addition ()
```

```
          {
               Return(a+b);
           }
       }
                    Class          printdata
         {
                    Public static   void main( string args[])
             {
                   Scanner sc = new Scanner(System.in);
                   System.out.println("Enter first number");
                        Int  a1= sc.nextInt();
                   System.out.println("Ente second number= ");
                    Int a2 = sc.nextInt();
                    Add ab = new add();
                    Ab.addition(a1,a2);
                  System.out.println("sum = " +s);
             }
         }
```

STATIC MEMBERS :--

If we want to access a variable or method in all the classes declared in the program then declare that variable or method as a static. In the class the ordinary variable and methods are called instance variable and instance methods. The static variable and methods in the class are called class variables and class methods. In java method class has number of static methods we can directly call it like method square(2):

```
e.g.:-        Class        mathfunction
           {
              Static   int  square(int a)
             {
                  Return(a*a);
              }
                 Static   int   doubles(int b)
             {
                   Return(b*2);
              }
          }
                  Class      Application
            {
                Public static void main(String args[])
               {
                     Int a = Mathfunction.square(4);
                     Int b = Mathfunction. Double(2);
                   System.out.println("a = " +a);
                  System.out.println("b = " +b);
               }
          }
```

RESTRICTION OF USING STATIC VARIABLES OR METHODS :-

(1) The static methods can only use static data.
(2) The static method can only call another static method.
(3) They cannot use a super keyword.

NOTE : [In class the ordinary variables and methods are called instance method. If we want to call the static variable or method no need for the object, we can call it directly with the class name. Static variables and methods are called class variables and methods.]

e.g. ->

```
             class   X
        {
              Int   a;              Instance
              Int  b;                  Variable
                Void show ()
               {
                  ----------            Instance
                  ----------               Method
                }
        }
```

```
⇨          Class        X
     {
            Static    Int a,b     --- Class variable
                  Static    void Show ()     -----    Class methods
                 {
                    --------
                    --------
                  }
     }
```

#   NESTING OF METHODS   :-

We can call a method by another method of the same class is called nesting methods.

```
e.g.  --                    Class           addition
                {
                       Int x,y;
                         Addition(int a , int b)
                     {
                          X=a;
                          Y=b;
                      }
                            Int    Add()
                          {
                               Return(x+y);
                           }
```

```
                    Void      display()
                {
                        Int sum = add();
                System.out.println("Sum of value=+sum");
                 }
            }
                Class          mainin
    {
            Public static void main(String args[])
        {
                Addition   a =    new addition (10,20);
                    a.display();
         }
    }
```

INHERITANCE :--

Inheritance is the process of deriving a new class (sub class) from an existing class (base class). The new class has the properties of the existing class with its properties of the existing class with its own details.

The inheritance properly make the reusability of the existing class. There are many types of inheritance……….

(i) Single inheritance

(ii) Multiple inheritance (Interface)
(iii) Multilevel Inheritance
(iv) Hierarchical Inheritance

(I) Single Inheritance :-- It is the process of deriving a sub class from only one super class (base class) is called single inheritance.
(Single inheritance from single super class)

$$\begin{tikzcd} A \arrow[r] & B \\ \text{(Super class)} & \text{(Sub class)} \end{tikzcd}$$

(II) Multiple Inheritance :-- It is the process of deriving a sub Class from many super classes called multiple inheritance.
(Single sub class from many super classes)

$$\begin{tikzcd} & A \arrow[dr] & \\ \text{(Super class)} & B \arrow[r] & \text{D(sub class)} \\ & C \arrow[ur] & \end{tikzcd}$$

(III) Hierarchical Inheritance :-- The super class has many sub class is called hierarchical inheritance.
(Many sub classes from single super class)

$$\begin{tikzcd} & B \arrow[dl] & \\ A & C \arrow[l] & \text{(Sub class)} \\ \text{(Super class)} & D \arrow[ul] & \end{tikzcd}$$

(IV) Multilevel Inheritance :-- It is the process of deriving a sub class from another sub class is called multilevel inheritance.
(Sub class from another sub class)

A	(Super class)
B	(Intermediate super class)
C	(Sub class)

In the above eg. The sub class is derived from another class. Here, derived class 'B' act as a super class for the sub class 'C'.

Sub Class—

The key word extends inherits the property of super class to the sub class. Sub class has also its own variables and methods. Now sub class will contain the property of super class and its own property.

Syntax :-- Class subclass name extends

```
                    Super class_name
            {
                  Variable declaration;
                   Methods declaration;
            }
```

e.g. :-- Class Add

```
      {
                 Int x,y;
                      Add( int a,int b)
            {
                      X=a;
                      Y=b;
             }
               Int  addition ()
           {
                 Return ( x+y);
           }
}
            Class ex add extends Add
   {
            Int z;
            Ex add (int a,int b,int c)
        {
                Super (a,b);
                  Z=c;
         }
```

```
                        Int ex addition ()
                    {
                            Return (x+y+z);
                     }
              }
                        Class main in
{
                Public static void main ( String agrs[])
      {
              Ex add a1 = new ex add (10,20,30);
                  Int sum1 = a1.add ();
                  Int sum2 = a1.ex add ();
                System.out.println (" Sum 1 = " +Sum1);
                System.out.println (" Sum 2 = " +Sum2);
      }
}
```

The keyword <u>Super</u> is used to call the constructor method of the super class. Super keyword must appear as the <u>first statement</u> of the sub class. The parameter that passed from the Super () must match the order and type of the instance variable declared in the super class.

METHOD OVERRIDING:-

The method defined in super class is inherited by inherited by its sub class and can be used through the object of sub class. This type of method inheritance enables us to define and invoke repeatedly in sub classes without defining it again in sub class.

If, there may be a situation, when an object want to respond some method with different behaviors by calling the method this is the process of overriding the method defined in super class.

This overriding is possible by defining a method in subclass similarly defined in super class. By invoking the method sub class is executed instead of super class method.

E.g. :-

```
                    Class       A
  {
                         Int x,y;
                    A(int a, int b)
    {
                    X=a;
                    Y=b;
    }
```

```
                    Void show()
            {
```

In the above example when show() method is invoked the super class show() method is overridden by subclass show() so it only display the z value. If we want to display the super class show() method.

Note :- By default all the variables & methods are overridden by sub class.

Final variables and method :--

By default all the variables & methods are overridden by sub class. To prevent the overriding of the member of a super class the keyword "Final" is used. If we declare the variable & method as final then it will never modified anywhere.

```
e.g.:-  Final int x= 10;

        Final void show();
```

Final Class:--

We declare final class for security resource. If we declare class as a final class then it is not possible to derive sub class from that final class. If we attends this the compiler will give some error message.

```
e.g. :--        Final     Class A

          {
                    -----------

                    -----------
```

}

Finalizer Method :--

The finalizer method is similar to destructor in C++, Java automatically free the memory used by object using garbage collecting system.

Object may has no resources such as “file description” or “Windows system fonts”. The garbage collector system cannot free these resources.

To free this finalizer method is used. When finalizer methods end of class reaches it will free memory resource which is occupied by a non - object resources such as file descriptions or windows system fonts etc.

Abstract Method & Class :--

Abstract method does opposite job of final. When we declare a class as a abstract class. The method in the class must be redefine in the subclass. Thus making overriding process.

Visibility Mode Control :--

Java provides five types of visibility control or access specifier. These are given below:--

(i) Public :-- If we specify the variable or method as public then that variables or method can be accessed everywhere in the program in all package.

(ii) Friendly access specifier :-- The default access specifier for Java classes are friendly. The friendly access makes the field visible only in the same package but in the case of public access specifier fields are visible in all the package. Package is he group of related class.

(iii) Protected access specifier :-- The protected access specifier makes the fields visible in all the classes and sub classes in the same package but also to sub classes in other package.

(iv) Private access specifier :-- The private access specifier specifies that we cannot inherit the class. They are accessible only within own class.

(v) Private protected access specifier :-- The access specifier private protected specifies the visibility control in between the private & protected access specifier. This makes the fields visible in all sub classes.

ACCESS SPECIFIER

Access Specifier	Public	Private	Friendly	Private Protected	Protected
Same class	Yes	Yes	Yes	Yes	Yes
Sub class in same package	Yes	Yes	Yes	Yes	No
Other classes in same package	Yes	Yes	Yes	No	No
Sub class in Other package	Yes	Yes	No	Yes	No

Non sub class on other classes	Yes	No	No	No	No

ARRAY :--

Array is a group of related data items stored in continuous memory location and in same name.

Declaration of Array :--

Like ordinary variable an array can be declared as follows :--

Syntax :

Datatype Variable_name [ ];

Or,

Datatype [ ] Variable_name ;

Eg :- int a [ ] ;

In the above declaration the array don't mention the size & memory space is created.

Creating memory space :--

After declaring the array new keyword is used to create memory space for that array.

Syntax :-

Variable_name = new datatype [size];

Eg :- a= new int [3];

The above example can be combined by the following statement. We can say that

Syntax :-

```
Datatype Variable_name [ ] = new datatype [size];

Datatype [ ] Variable_name = new datatype [size];
```

Initialization of Array :-

The final step is to put values into the array created. This process is known as initialization. This is done using the array subscripts.

```
Array_name [ ] = value;
```

Eg :-

```
num  [0] = 35;
Num [1] = 40;
----------------------
----------------------
 Num [n] = 19;
```

Note :-- Java creates array starting with the subscript 0 and ends with a values one less than the size specified.

Unlike C, Java protects array from overruns and underruns. Trying to access an array bound its boundaries will generate an error message.

We can also initialize array automatically in the same way as the ordinary variables when they are declared.

Datatype array_name [ ] = { list of values };

The array initializer is a list of values separated by commas and surrounded by curly braces.

Eg :- int num [ ] = {35,40,20,57,19};

ARRAY LENGTH :--

In java all arrays store the allocated size in a variable_name length. We can obtain the length of array using array_name.length variable.

The array _name.length returns or gives the allocated size of array.

Eg :- int a size = a.length

STRING:--

String manipulation is the most common part of many java program string represents a sequence of characters. The easiest way to represents a sequence of characters. The easiest way to represent a sequence of characters. The easiest way to represent a sequence of character in java by using character array.

Eg:- char array_name [ ] = new char [10];

Although character array have the advantage of being able to query their length, they themselves are not good enough to supports the range of operations. We may like to perform an string.

Java offers two class to manipulate strings :--

(i) String class
(ii) String buffer

1. STRING CLASS :--

String class is useful for fixed length string.

Syntax :- String string_name;

String_name = new string ("string value");

Or,

String string name = new string ("string value");

2. STRING LENGTH :--

The length of string can be fixed with the length variable as the statement given below --------

Int len = String name.length;

3. STRING ARRAY :--

We can also create and use array that contain strings. The statement for creating string array is given below --------

String str array [ ] = new string [10];

Q. Write a program to print string array.

Solu. Class test string

{

```
                    Public static void main (string args[ ])
                    {
                         String s[ ] = {"xyz","pqr","mno"};
                          For(int i=0; i<s.length;i++)
                           {
                                  System.out.println("String value=" +s[i]);
                           }
                    }
            }
```

2nd method

<u>Command line argument (Input)</u>

```
             Class test string
{
        Public static void main (string args ["xyz","pqr"])
    {
           For (int I =0;i<args.length;i++)
          {
              System.out.println("string value=" +args[i]);
          }
    }
```

}

STRING METHODS :--

The string class defines a number of methods that allow us to accomplish or complete a variety of string manipulation task.

(a.) tolowercase() :- This method converts a string to lowercase.
Eg.:- String str = new string ("Priya");
S1 = str.tolowercase();
The above example converts the value of string str to lowercase and store into string s1.

```
                 Class  lower
  {
            Public static void main (string args [ ])
       {
           String s = new str ("Priya");
           System.out.println ("Name= " +s.toLowercase());
       }
  }
```

(b.) touppercase () :- This method converts a string to uppercase.
Eg:- String str = new string ("Priya");
S1 = str.touppercase ();

```
               Class  upper
        {
            Public static void main ( string args [ ])
          {
```

```
                    String s[ ]= new s ("Suraj");
                    String s2 = s.touppercase();
                    System.out.println ("Upper case = " +S2);
                }
            }
```

(c.) replace () :-- The replace () replaces a specific character to string by another character.

Syntax :-

```
   Replace ( 'replaced character', 'replacing character');
       E.g. :- String str = new String ("Priya");
                   Str.replace ('y','s');
```

(d.) trim () :-- The trim() function removes white spaces at the beginning and end of the string.

```
e.g. :- String s1 = "----------- Priya ------------"
                          string s2;
                  s2= s1.trim ();
```

(e.) equals () :-- This function returns true if one string is equal to another string otherwise false.

```
    e.g. :- str1.equals (str2);
```

(f.) equal IgnoreCase () :-- This function returns true if first string is equal to second string ignoring the case of character.