CLASSES, OBJECTS & METHODS

Class:

A class is a non-primitive data type that contains member variables (data) and member functions (methods) that operates on the variable. A class is declared by the use of the class keyword. The general form of class definition is as follows.

Syntax:
```
class class-name
 {
  datatype instance-variable _1;
  datatype instance-variable _2;
  ……..
  ……..
  datatype instance-variable_n;

  datatype methodname1(parameter-list)
  {
        method body
  }
  ……..
  ……..
  datatype methodname-n(parameter-list)
  {
        method body;
  }
 }
```

The data or variables defined with in a class are called instance variables. The code is present with in methods. Collection of methods and variables defined with in a class are called members of class.

Variable declaration:

Data is encapsulated in a class by placing data fields inside the body of the class definition. These variables are called instance variables because they are created whenever an object of the class is instantiated. we can declare the instance variables exactly the same way as we declare local variables.

Example:

```
class Rectangle
{
      int length;
      int width;
}
```

Methods declaration:

Methods are declared inside the body of the class but immediately after the declaration of the instance variables. The general form of the method decleration as follows

```
returntype  methodname(parameterlist)
    {
        method-body;
    }
```

Method declaration have four basic parts:

- The name of the method
- The type of value the method return
- A list of parameters
- The body of the method

Example:

```
class sample
{
    double width,height,depth;
    double volume()
    {
        return width*height*depth;
    }
}
```

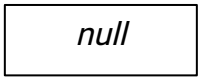
Creating Objects:

An object in Java, is a block of memory that contain space to store all the instance variables. Creating an object is a two-step process.

1. Declare a variable of the class type. At this stage , the variable refers to an object only.
2. Assigning an actual , physical copy of the object and assign it to that variable by means of new operator. The new operator dynamically allocates memory for an object and returns a reference to that object.

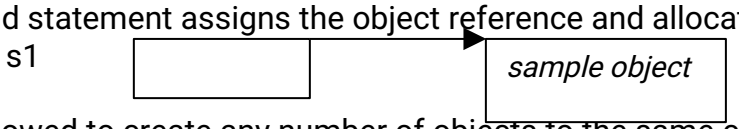
```
sample s1;    //reference
s1 = new sample ( ) //instantiate
            ( or )
sample s1 = new sample ( )
```

The first statement declares variable to hold the object reference



s1

The second statement assigns the object reference and allocates memory.



We are allowed to create any number of objects to the same class

Example:

```
sample s1 = new sample ( );
```

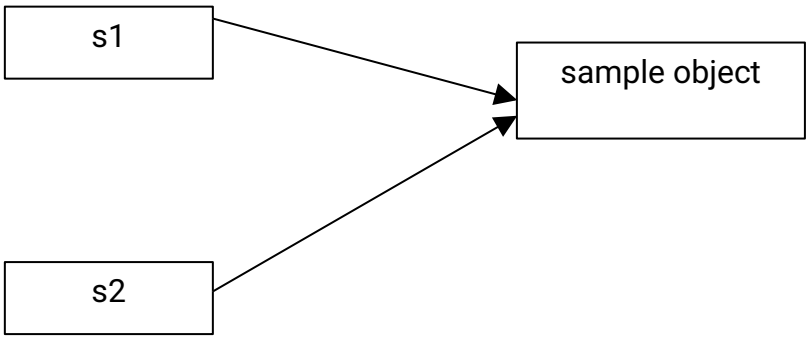
```
sample s2 = new sample( );
```

Each object contains its own instance variables of its class. This means that any changes to the variable of one object have no effect on the variables of another. But the function name is same for both objects.

It is also possible to create two or more references to the same object.

Example:

```
sample s1 = new sample ( );
sample s2 = s1;
```



Accessing class members:

All variables must be assigned values before they are used in our program. All class members are accessed by means of “dot “operator.

The general format of accessing instance variables of the class by dot operator is as follows

Syntax : objectname.variablename=value;

where objectname is name of the object , variable name is the name of the instance variable.

Example:
```
s1.length = 10;
s2 .width = 15;
```

The general format of accessing instance methods of the class by dot operator is as follows

Syntax : objectname . methodname(parameter- list);

Where object is name of the object, method name is the name of the instance method that calls with the object and parameter – list is a list of actual values separated by commas.

Example:
```
void getdata(int x, int y)
{
    length=x;
    width=y;
}
```

```
        s1.getdata(10,20);
```

Example: program to calculate area of a rectangle

```
class Rectangle
{
        int length,width;
        void getdata(int x ,int y)
        {
                length = x;
                width = y;
        }
        int rectArea( )
        {
                int area = length * width;
                return(area);
        }
}
class RectArea
{
        public static void main(String args[ ])
        {
                int area1,area2;
                Rectangle rect1 = new Recatangle ( );
                Rectangle rect2 = new Rectangle ( );
                rect1.length=15;
                rect1.width = 10;
                area1=rect1.length * rect1.wifdth;
                rect2.getdata(10,20);
                area2=rect2.rectArea( );
                System.out.println("area1 =" +area1);
                System.out.println("area2 =" +area2);
        }
}
```

Method Overloading:

In java it is possible to define two or more methods within the same class that share the same name as long as these methods have different set of parameters (Based on the number of parameters, the type of the parameters and the order of the parameters). Then the methods are said to be overloaded, and the process is referred as Method Overloading.

When the overload method is called the Java compiler selects the proper method by examine the number, type and order of the arguments in the cell. Method Overloading is commonly used to create several methods with same name that perform fast accessing.

Note: Two methods differ in only by return type will result in a syntax error.

Advantage of Method :

- Code Reusability
- Code Optimization

Example:

```
class sample
{
        int large(int x, int y)
        {
          if(x>y)
        return x;
        else
        return y;
        }

     int large(int x, int y, int z)
        {
          if(x>y&&x>z)
        return x;
        else if(y>z)
        return y;
        else
        eturn z;
        }
}

class check
{
        public static void main(String args[ ])
        {
        sample s=new sample( );
                System.out.println(" largest of two numbers is:"+s.large(10,20));
          System.out.println(" largest of three numbers is:"+s.large(11,22,33));
        }
}
```

o/p: largest of two numbers is :20
largest of three numbers is :33

Constructor:

A constructor is a special method of class , which is invoked automatically, whenever an object of a class is created.
A constructor has the following characteristics :

- It doesn't have any return type not even void
- It has the same name as its class
- It can be overloaded
- It is normally used to initiate the members of object.
- Constructors should not be static since constructors will be called each and every

time whenever an object is creating.
- Constructors will not be inherited at all.

Different Types of constructors:

- Default Constructor(constructor with out arguments)
- Parameterized Constructor(constructor with arguments)

1). Default Constructor:

A constructor that accepts no parameters is called the default constructor. If no constructors are defined for a class the java system automatically generates the default constructor.

Example:

```
 class sample
{
        int x;
        sample()
        {
        x=10;
        }
        void show()
        {
        System.out.println("x="+x);
        }
}

 class defsample
{
  public static void main(String args[])
 {
   sample s=new sample();
        s.show();
  }
}
output:  x=10;
```

Parameterized Constructor:

A constructor that takes arguments as parameters is called parameterized constructor. If any constructors are defined by a class with the parameters, java will not create a default constructor for the class.

Example:

```
class sample
{
        int x;
        sample( int k)
        {
```

```
                x=k;
                }
                void show()
                {
                System.out.println("x="+x);
                }
        }

        class parsample
        {
          public static void main(String args[])
         {
           sample s=new sample(25);
                s.show();
         }
}
   output:   x=25
```

Constructor Overloading:

A class can have multiple constructors. This is called constructor overloading. All the constructors have the same name as corresponding to the class name and they are differ only in number of arguments or data types or order.

Example:

```
        class sample
        {
                int x,y;
                sample()
                {
                x=0;
                y=0;
                }
                sample(int a)
                {
                x=a;
                }
                sample(int a,int b)
                {
                x=a;
                y=b;
                }
                void show()
                {
                System.out.println("x="+x+" " +"y="+y);
                }
        }
        class sampleover
        {
```

```
    public static void main(String args[])
   {
     sample s1=new sample();
   sample s2=new sample(10);
   sample s3=new sample(11,22);
           s1.show();
           s2.show();
           s3.show();
    }
}
```

output: x=0 y=0
x=10 y=0
x=11 y=22

Static Members:

Generally each class contains instance variables and instance methods. Every time the class is instantiated, a new copy of each of them is created. They are accessed using the objects with the dot operator.

If we define a member that is common to all the objects and accessed without using a particular object i.e., the member belong to the class as a whole rather than the objects created from the class. Such facility can be possible by using the keyword **"static".**

The members declared with the keyword static are called static members.

The static can be: 1. variable (also known as class variable)
2. method (also known as class method)

1) **Java static variable**: If you declare any variable as static, it is known static variable.

- The static variable can be used to refer the common property of all objects¬ (that is not unique for each object) e.g. company name of employees, college name of students etc.
- The static variable gets memory only once in class area at the time of class loading.

Advantage of static variable: It makes your program memory efficient (i.e it saves memory).

Example:

```
static int a ;
static string college="vignan";
```

2) Java static method:
If you apply static keyword with any method, it is known as static method

- A static method can be invoked without the need for creating an instance of a class.
- static method can access static data member and can change the value

Example:

```
static int show( int x, int y);
```

The important points about static class members are:

- Static members can be accessible without the help of objects.

- A static method can access only static variables
- A static member store data at same memory for all objects
- Static method cannot be referred to 'this' or 'super' in anyway.

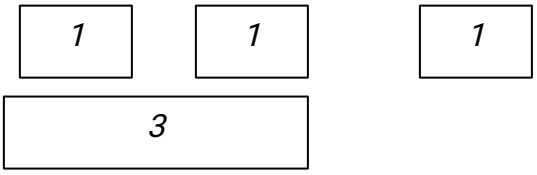
Example1:

```
class sample
{
        static int c=0;
        sample( )
        {
                c++;
        }

        static void display( )
        {
                System.out.println("no of objects="+c);
        }
}
class  Test
{
        public static void main(String args[ ])
        {
                sample s1 = new sample ( );
                sample s2 = new sample ( );
                sample s3 = new sample ( );
                sample.display();
        }
}
        output:         no of objects 3
```

without using static keyword

s1 s2 s3



with static keyword

s1 s2 s3

this keyword:

Java define "this" keyword that can be used inside any method to refer the current object. "this" always a reference to the object on which method was invoked. "this" reference is implicitly used to refer to both the instance variables and methods of current objects.

Example:

```
class Box
{
        int len,dep,hei;
        Box(int len, int dep , int hei)
        {
```

```
                this.len = len;
                this.dep = dep;
                this.hei = hei;
        }

        int volume( )
        {
                return (len * dep * hei);
        }
}
class thisdemo
{
        public static void main(String args[ ])
        {
                Box obj = new Box(1,2,3);
                int x = obj.volume( );
                System.out.println("volume="+x);
        }
}
        output: volume = 6;
```

"this" key word also used explicitly refer to the instance variables.

Usage of java this keyword:

1. The this keyword can be used to refer current class instance variable
2. Using this keyword we can explicitly call the constructors of the same class by passing the corresponding arguments.

Rules for 'this':

1. The statement which explicitly calls the constructor should and must be first statement of the constructor if it is second statement then it leads for a compilation error
2. The statement which explicitly calls a constructor should be present only inside a constructor it can't be present inside function

Nesting of methods:

In java, a method can be called by another method of the same class , this is known as nesting of methods.

Ex:

```
class sample
  {
    int x,y;
  sample()
  {
    x=4;
   y=6;
  }
  int max()
  {
```

```
        if(x>y)
        return x;
        else
        return y;
      }
      void show()
      {
      System.out.println("largest of two numbers="+max());
      }
      }

class nest
{
 public static void main(String args[])
      {
        sample s =new sample();
            s.show();
      }
}
output: largest of two numbers=6.
```

Garbage collector:
In java garbage collector means unreferenced objects.
Garbage collector is process of reclaiming the runtime unused memory automatically. In Other words, it is a way to destroy the unused objects. To do so, we are using free () function in c language and delete () function in c++.but in java it is performed automatically. so, java provides better management.

Advantages of garbage collector:

1. It makes java memory efficient because garbage collector removes unreferenced objects from heap memory.
2. It is automatically done by the garbage collector (a part of JVM) so we don't need to make extra efforts.

Array:

An array is set of similar data items share a common name. The individual values in an array are called elements. The elements of an array can be of any data type. All the elements in an array are of the same type. Array elements are variables. A particular value in indicated by writing a number called index number (or) subscript in brackets after array name. There are two types of arrays.

1. One-dimensional arrays
2. Multi-dimensional arrays

1. One-dimensional arrays:

A list of items can be given one variable name using only one subscript and such a variable is called a single-subscripted variable (or) a one-dimensional array.

Creating an Array:

Like any variables, arrays must be declared and created in the computer memory before they are used. Creation of an array involves three steps:

1. Declaring the array
2. Creating memory locations
3. Putting values into the memory locations

Declaring of Arrays:

Arrays in Java may be declared in two forms

Form1: type arrayname[ ];
Form2: type[ ] arrayname;

Examples:

```
int     number[ ];
float   average[ ];
int[]   counter;
float[ ] marks;
```

Remember , we do not enter the size of the arrays in the declaration.

Creation of Arrays:

After declaring an array , we need to create it in the memory. Java allows us to create arrays using **new** operator only, as shown below

array name = new type[size]

Example:

```
number = new int[5];
average = new float[10];
```

These lines create necessary memory locations for the arrays **number** and **average** and designate them as **int** and **float** respectively. Now the variable **number** refers to an array of 5 integers and **average** refers to an array of 10 floating point values.

It is also possible to combine the two steps declaration and creation into one as shown below

int number[ ]= new int[5];
float average[ ]=new float[10];
Initialization of Arrays:

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

arrayname [subscript] = value;

Example:

```
number[ 0 ]= 35;
number[1] = 40;
……….
……….
number[4] = 50;
```

Note that Java creates arrays starting with the subscript of 0 and ends with a value one less than the **size** specified.

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

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

type arrayname[ ] = { list of values};

The array initializer is a list of values separated by commas and surrounded by curly braces. Note that no size is given in this case. The compiler allocates enough space for all the elements specified in the list.

Example:

int number[ ] = { 35,40,20,50};

Array length:

In Java all arrays store the allocated size in a variable named length. We can obtain the length of the array **a** using **a.length.**

Example:

int size = a.length;

This information will be useful in the manipulation of arrays when their size are not known.

Program to read 5 numbers and print the sum of it.

```
class array1
{
        Public static void main(String args[ ] )
        {
                int sum = 0;
                int no[ ] = new int [10];
                for ( int i = 0;i < 5; i++)
                {
                        no[i] = Integer.parseInt(args[i]);
                        sum = sum+ no[i];
```

```
                }
                System.out.println("Total = "+sum);
        }
}
```

Program for sorting list of numbers:

```
class NumberSorting
{
        Public static void main (String args[ ])
        {
                int number[ ]  = { 55 , 40 , 80 , 65  , 71};
                int n= number.length;
                System.out.println("Given list:");
                for ( int i = 0; i< n ;i+ +)
                {
                        System.out.print(" " + number[i] );
                }
                System.out.println("\n");

                // sorting begins
                for ( int i = 0;i< n;i + +)
                {
                        for(int j= i+1; j< n; j + +)
                        {
                                if ( number[i]>number[j])
                                {
                                        int temp =  number[i];
                                        number[i]= number[j];
                                        number[j] = temp;
                                }
                        }
                }// sorting ends
                System.out.print("Sorted list is:");
                for  ( int I = 0;i < n; i+ + )
                {
                        System.out.print(" " + number[i] );
                }
        }
}
```

Output :

```
Given list    :    55    40    80    65    71
Sorted list   :    80    71    65    55    40
```

2. Two- dimensional Arrays:

A Two-dimensional array is a collection of homogeneous data items that share a

common name using two subscripts. It stores table of data . This representation is very useful for matrix representations as first subscript represents the number of rows and second subscript represents the number of columns.

Syntax : datatype arrayname[ ] [ ]; //declaration

new datatype [size1][size2]; // allocation of memory

(or)

datatype arrayname[ ] [ ] = new datatype[size1][size2];

Where ,

datatype defines the datatype of the variables stored in the array
arrayname defines the name of the array
new operator is used to allocate dynamic memory in the computer for the array
size1 defines the number of rows of memory location of the array
size2 defines the number of columns of memory locations of the array

Example:

```
int number[ ] [ ] = ;
number = new int[5][5];
```

It can also be initialize arrays automatically when they are declared. The syntax is as follows

Syntax: type arrayname[ ] [ ] = {list of values};

Or

type arrayname[ ] [ ]={ row1 values},{row2 values};

Three Dimensional Arrays :

Set of two dimensional arrays called three dimensional array. A three dimensional array is group of variable specified by three scripts. Three dimensional array can be declared as shown below.

data type array name[ ][ ][ ]=new data type [size1][size2][size3];

Where data type is the elements data type in array. size1, size2, size3 indicates page, row, column sizes respectively.

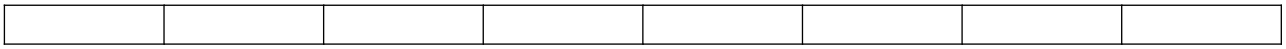
Three dimensional array can be viewed as set of two-dimensional arrays.

Ex: int a [] [] []=new int [2][2][2];

This declares a array which holds 8-integer elements. The elements will be referred as a[0][0][0], a[0][0][1], a[0][1][0], a[0][1][1], a[1][0][0], a[1][0][1], a[1][1][0], a[1][1][1]

The memory representation of the above 3-dimensional array is as shown in fig. below.

a



000 001 010 011 100 101 110 111

Accessing elements of 3-dimensional array:

Elements of 3-dimensional array can be accessed by using 3-indexes as shown below.

```
int a[][][]=new int [3][3][3];
int i,j,k,l=1;
for(i=0; i<3; i++)   ——▶ page index
        {
                for(j=0; j<3; j++)  ——▶ row index
                        {
                                for(k=0; k<3; k++)  ——▶Column index
                                        a[i][j][k]=l;
                                        l++;
                        }
        }
```

Program to display elements of 3-dimensional array

```
class 3darray
 {
 public static void main(String args[])
        {
                int a[][][]=new int [2][2][2];
                int i,j,k,l=1;
                for(i=0; i<2; i++)
                        {
                                for(j=0; j<2; j++)
                                        {
                                                for(k=0; k<2; k++)
                                                a[i][j][k]=l;
                                                l++;                                                    }
                        }

                for(i=0;i<2; i++)
                        {
                System.out.println("page "+ i);
                        for(j=0; j<2; j++)
                                {
                                        for(k=0; k<2; k++)
                                        System.out.print(a[i][j][k]);
                                        System.out.println;
                                }
                        }
                        }
                        }
                        }
```

Command Line Arguments:

Java has the facility of command line arguments. Java main method consists array of string objects .When we run the program, the array will fill with the values of any

arguments it was given in the command line.

Example: program for command line arguments.

```
Class Commandline
{
        public static void main(String args[])
        {
                System.out.println(args[0]);
                System.out.println(args[0]);
        }
}
c:> javac Commandline.java
c:>java Commandline  10 20
10
20
```