Objects and Classes in Java

```
class BookDetail

{

   int id;

    String name;

      public static void main(String args[]){

BookDetail b1=new BookDetail ();

    System.out.println(b1.id);

    System.out.println(b1.name);

 }

}
```

```
                             O/P :    0

                                         NULL
```

```
// main method is in another class
class Book2
{       int id;
        String name;
}
class Book2Demo
{
      public static void main(String args[])
{
        Book2 b1=new Book2();
          System.out.println("id=="+b1.id);
          System.out.println("Name=="+b1.name);
 }
}
```

```
                                          O/P :   0

                                                   NULL
```

Object Initialization

There are 3 ways to initialize object in Java.

- using variable
- using method
- using constructor

Using reference variable

```
class Book1{
 int id;
 String name;
void display()
{ System.out.println(id+" "+name);}
}
class BookDemo
{
 public static void main(String args[]){
  Book1 b1=new Book1();
  Book1 b2=new Book1();
  b1.id=511;
  b1.name=" JAVA";
  b1.display();
  b2.id=512;
  b2.name=" Advance JAVA";
  b2.display();
 }
}
```

O/P:

```
511     JAVA
512 Advance JAVA
```

Using Method

```
class BookMethod
{
 int srollno;
 String sname;
void getData( int rollno , String name)
       {
        srollno=rollno;
        sname=name;
     }
void display()
    {      System.out.println(srollno+" "+sname);}
}
class BookMethodDemo
       {
        public static void main (String args[])
        {
         BookMethod b1=new BookMethod();
         BookMethod b2=new BookMethod();
         b1.getData(123, "ABC" );
         b1.display();
         b2.getData(222, "XYZ");
        b2.display();
 }
}                  O/P:     123 ABC
                            222 XYZ
```

```
class CustomerAccount{
        int cacc_no;
        String cname;
        double camount;
        void getData(int acc_no,String name,double amount)
                 {
                 cacc_no=acc_no;
                cname=name;
                camount=amount;
        }
        void deposit(double amt)
{
    camount=camount+amt;
    System.out.println(amt+" deposited");
}

void withdraw(double amt)
{
if(camount<amt)
     {
       System.out.println("Insufficient Balance");
     }
else
{
camount=camount-amt;
System.out.println(amt+" withdrawn");
}
}
void checkBalance(){System.out.println("Balance is: "+camount);}
void display(){System.out.println(cacc_no+" "+cname+" "+camount);}
}
class CustomerAccountDemo
{
public static void main(String[] args){
CustomerAccount   obj=new CustomerAccount();
obj.getData(2022001,"ABC",1000);
obj.display();
obj.checkBalance();
obj.deposit(5000);
obj.checkBalance();
obj.withdraw(20000);
obj.checkBalance();
}}
          /*  O/P:
                         2022001 ABC 1000.0
                         Balance is: 1000.0
                         5000.0 deposited
                         Balance is: 6000.0
                         Insufficient Balance
                         Balance is: 6000.0  */
```

Using Constructor

```
class BookConstructor
  {    int srollno;
        String sname;
       BookConstructor( int rollno , String name)
             {
               srollno=rollno;
               sname=name;
             }
       void display()     {      System.out.println(srollno+" "+sname);}
}
class BookConstructorDemo
     {
       public static void main (String args[])
       {
        BookConstructor b1=new BookConstructor(123, "ABC");
        BookConstructor b2=new BookConstructor(222, "XYZ");
               b1.display();
              b2.display();
 }
}                  O/P:    123 ABC
                           222 XYZ
```

Constructor Overloading: In addition to overloading methods, we can also overload constructors in java. Overloaded constructor is called based upon the parameters specified when new is executed.

```
// Constructor Overloading
class BoxVolume
{
  double width, height, depth;

  BoxVolume(double w, double h, double d)
  {
    width = w;
```

```
        height = h;
        depth = d;
    }

       BoxVolume(double len)
    {
        width = height = depth = len;
    }
         double volume()
    {
        return width * height * depth;
    }
}
  public class BoxVolumeDemo
   {
    public static void main(String args[])
    {
        BoxVolume mybox1 = new BoxVolume(10, 20, 30);

      BoxVolume mycube = new BoxVolume(4);
          double vol;
                          vol = mybox1.volume();
        System.out.println(" Volume of mybox1 is " + vol);
                         vol = mycube.volume();
        System.out.println(" Volume of cube is " + vol);
    }
}
                  O/P: Volume of mybox1 is  6000
                           Volume of  cube is  64
```

//Copy Constructor

```
class BookConstructorDemo2 {
    int bid;
    String bname;
    BookConstructorDemo2(int i,String n){
    bid = i;
    bname = n;
    }
    BookConstructorDemo2(){}
    void display()
```

```
        {System.out.println(bid+" "+bname);}

   public static void main(String args[])
   {  BookConstructorDemo2 b1 = new  BookConstructorDemo2(2022,"JAVA");
      BookConstructorDemo2 b2 = new BookConstructorDemo2();
   b2.bid=b1.bid;
   b2.bname=b1.bname;
   b1.display();
   b2.display();
  }
}                    O/P:  2022    JAVA
                           2022   JAVA
```

```
// Copy Constructor in JAVA
class Employee
  {
   int eid;
   String ename;
     Employee (int id,String name)
           {
              eid = id;
              ename =name;
            }
     Employee (Employee obj)
           {
             eid = obj.eid;
             ename =obj.ename;
           }
   void display(){System.out.println(eid+" "+ename);}

   public static void main(String args[]){
   Employee e1 = new Employee(101,"ABC");
   Employee e2 = new Employee(e1);
   e1.display();
   e2.display();
  }
}
```

Difference between Constructor and Method

There are many differences between constructors and methods. They are given below.

Java Constructor	Java Method
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.
The constructor’s name must be same as the class name.	The method name may or may not be same as the class name.

Static variables in Java

- Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.
- There would only be one copy of each class variable per class, regardless of how many objects are created from it.
- Static variables are created when the program starts and destroyed when the program stops.
- Visibility is similar to instance variables. However, most static variables are declared public since they must be available for users of the class.
- Default values are same as instance variables. For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null. Values can be assigned during the declaration or within the constructor. Additionally, values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name ClassName.VariableName.

- When declaring class variables as public static final, then variable names (constants) are all in upper case. If the static variables are not public and final, the naming syntax is the same as instance and local variables.

//Java Program to demonstrate the use of static variable

```
     class EmployeeData
     {  int eid;                    //instance variable
        String ename;
        static String org="NERIST";      //static variable
         EmployeeData(int id, String name)
             {   eid = id;
                 ename = name;
             }
           void display (){System.out.println(eid+" "+ename+" "+org);}
     }
     public class EmployeeDataStatic
     {
      public static void main(String args[]){
      EmployeeData s1 = new EmployeeData(101, "ABC");
     EmployeeData s2 = new EmployeeData(102, "XYZ");
      //we can change the college of all objects by the single line of code
      //EmployeeData.org="NERIST  NIRJULI";
      s1.display();
      s2.display();
      }
  }                               O/P:   101 ABC NERIST
                                         102 XYZ NERIST
     class WebsiteCounter
      {
     int count=0;//will get memory each time when the instance is created

     WebsiteCounter()
         {
             count++;
             System.out.println(count);
     }

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

```
//Creating objects
WebsiteCounter c1=new WebsiteCounter();
WebsiteCounter c2=new WebsiteCounter();
WebsiteCounter c3=new WebsiteCounter();
WebsiteCounter c4=new WebsiteCounter();
}
}                    O/P:  1
                            1
                            1
                            1
class WebsiteCounter
 {
static int count=0;

WebsiteCounter()
    {
        count++;
        System.out.println(count);
}

public static void main(String args[]){
//Creating objects
WebsiteCounter c1=new WebsiteCounter();
WebsiteCounter c2=new WebsiteCounter();
WebsiteCounter c3=new WebsiteCounter();
WebsiteCounter c4=new WebsiteCounter();
}
}                    O/P:  1
                           2
                           3
                           4                              4
```

Java static method: If you apply static keyword before the method name, it is known as static method.

- A static method belongs to the class rather than object of a class.
- 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 of it.

```
class SquareStatic
{
  static int square(int x)
           {
          return x*x;
            }
    }
class  SquareStaticDemo
{
  public static void main(String args[])
{
  int s=SquareStaticDemo.square(16);
  System.out.println(s);
  }
}
```