11. CONSTRUCTORS

1.

Default Constructors-

```
Class demo
{
   int x;
   int y;
   float z;
   demo()
   {
      X=1;
      Y=2;
      Z=3;
   }
   void display()
   {
   System.out.println("values of x, y and z are:"+x+" "+y" "+" "+z);
   }
}
Class Main
{
   public static void main(String args[])
   {
      demo d1=new demo();
      d1.display();
   }

}
```

Parameterized Constructor-

```
Class demo
{
   int x;
   int y;
   float z;
   demo(int x1,int y1,int z1)
   {
      x=x1;
      y=y1;
      z=z1;
   }
   void display()
   {
      System.out.println(—values of x, y and z are:‖+x+‖ —+y‖+‖ —+z);
   }
}
Class Main
{
   public static void main(String args[])
   {
      demo d1=new demo(1,2,3);
      d1.display();
   }
}
```

2.

```
class Super {
   String s;

   public Super(){
         System.out.println("Super");
   }
}

public class Sub extends Super {

   public Sub(){
         System.out.println("Sub");
   }

   public static void main(String[] args){
         Sub s = new Sub();
   }
}
```

Output-

```
Super
Sub
```

3.

Private-

```
class A{
         private int data = 10;
         private void displayData(){
                  System.out.println("Data:=" + data);      //can be accessed within class
         }
}

public class Demo{
         public static void main(String args[]){
                  A obj = new A();
                  System.out.println(obj.data);     //Compile Time Error
                  obj.display();     //Compile Time Error
         }
}
```

Default-

```
public class A {
   int data = 10;
}
```

```
public class B {
   A obj = new A();

   public int getData{
      return obj.data;
   }
}
```

Protected-

```
public class A {
   protected int data = 10;
}

public class B {
   A obj = new A();

   public int getData{
      return obj.data;
   }
}
```

Public-

```
public class A {
   public int data = 10;
}

public class B {
   A obj = new A();

   public int getData{
      return obj.data;
   }
}
```

4.
Constructors in Java do not have any return type

5.
Constructor is called automatically when we create an object using new keyword. It is called only once for an object at the time of object creation and hence, we cannot invoke the constructor again for an object after it is created.