EXPERIMENT 9

// Introduction to Constructors and Constructor Overloading.

Prerequisite Knowledge -

Constructor - A constructor is a special member function of a class, whose purpose is to initialise the data members of the object of the class at the time of object creation.

- ➔ A constructor is the first function called by an object once it is created.

Also, any other tasks that are required to be performed by the object at the time of object creation can be mentioned in the body of the constructor.

- ➔ The name of a constructor is the same as the name of the class in which it is created.
- ➔ There is no return type of a constructor because by default its return type is the class itself in which it is created.

Types of Constructors -

There are mainly two types of constructors in Java:

- ➔ **Default Constructor -** This constructor is present by default in Java i.e., we don't need to define it explicitly, and it is called automatically when an object is created.
  - ◆ A default constructor does not accept any arguments for initialising an object. By default, it initialises the data members of a class with 0.

  Syntax:

  ```
  class_name obj_name = new class_name();
  ```

  Example:

  ```
  Sample obj1 = new Sample();
  ```

- ➔ **Parameterised Constructor -** This constructor needs to be defined explicitly inside a class, and it works only when it is called with some parameters by an object. So, it initialises an object with the parameters passed to it.

  Syntax:

  ```
  class_name(list of formal parameters)
  {
      data members being assigned passed parameters
  }
  ```

  Example:

  ```
  class Sample
  {
      int a,b;
  ```

```
        Sample(int x,int y)
        {
            a=x; b=y;
        }
}
```

By manipulating these two types of constructors, we can also create two more types of constructors in Java -

- ➔ **No Argument Default Constructor -** We create this by defining the default constructor explicitly as per our choice of initialisation values but without passing any arguments to it.

  Example:

```
class Sample
{
    int a,b;

    Sample()
    {
        a=1; b=2;
    }
}
```

- ➔ **Copy Constructor -** It is created for initialisation of an object with the initial values of some other object of the same class. For this the parameterized constructor is defined with objects as formal parameters, and while calling, the objects are passed to it.

  Example:

```
class Sample
{
    int a,b;

    Sample(Sample obj)
    {
        a=obj.a;
        b=obj.b;
    }
}
```

  - ★ Here, the formal parameter object ‘obj’ becomes another reference variable of the passed object and helps in copying the data member values of the passed object into the calling object.

Constructor Overloading -

When two or more types of constructors are defined in the same class, the phenomenon is known as constructor overloading.

Just like method overloading, the name of the constructors remains the same but they can initialise different objects differently depending upon the number and type of arguments passed to them while calling.

PROGRAM 9

// To demonstrate the mechanism of constructor overloading.

Source Code -

```
class Overload_Constructor
{
   int a,b;

   Overload_Constructor()
   {
      a=b=1;
   }

   Overload_Constructor(int x,int y)
   {
      a=x; b=y;
   }

   Overload_Constructor(Overload_Constructor obj)
   {
      a = obj.a;
      b = obj.b;
   }

   void ShowValues()
   {
      System.out.println("Data Members are: " + a + " and " + b);
   }
}
class MainClass
{
```

```
    public static void main(String args[])
    {
        Overload_Constructor obj1,obj2,obj3;
        obj1 = new Overload_Constructor();
        obj2 = new Overload_Constructor(5,10);
        obj3 = new Overload_Constructor(obj2);

        obj1.ShowValues();
        obj2.ShowValues();
        obj3.ShowValues();
    }
}
```

Output -