Experiment No.- 04

AIM: Edit, compile , execute and test a Java program to implement constructors.

Theory : Constructors in Java

Constructors are used to initialize the object's state. Like methods, a constructor also contains **collection of statements(i.e. instructions)** that are executed at time of Object creation.

Need of Constructor

Think of a Box. If we talk about a box class then it will have some class variables (say length, breadth, and height). But when it comes to creating its object(i.e Box will now exist in computer's memory), then can a box be there with no value defined for its dimensions. The answer is no.

So constructors are used to assign values to the class variables at the time of object creation, either explicitly done by the programmer or by Java itself (default constructor).

When is a Constructor called ?

Each time an object is created using **new()** keyword at least one constructor (it could be default constructor) is invoked to assign initial values to the **data members** of the same class.

A constructor is invoked at the time of object or instance creation. For Example:

```
class Constructor
{
  .......

  // A Constructor
new Constructor () {}

  .......
}

// We can create an object of the above class
// using the below statement. This statement
// calls above constructor.
IT obj = new IT();
```

Rules for writing Constructor:

- Constructor(s) of a class must has same name as the class name in which it resides.
- A constructor in Java can not be abstract, final, static and Synchronized.
- Access modifiers can be used in constructor declaration to control its access i.e which other class can call the constructor.

Types of constructor

There are two type of constructor in Java:

1. **No-argument constructor:** A constructor that has no parameter is known as default constructor. If we don't define a constructor in a class, then compiler creates **default constructor(with no arguments)** for the class. And if we write a constructor with arguments or no-arguments then the compiler does not create a default constructor. Default constructor provides the default values to the object like 0, null, etc. depending on the type.
2. **Parameterized Constructor:** A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with your own values, then use a parameterized constructor.
3. **Does constructor return any value?**

There are no "return value" statements in constructor, but constructor returns current class instance. We can write 'return' inside a constructor.

4. **Constructor Overloading**

Like methods, we can overload constructors for creating objects in different ways. Compiler differentiates constructors on the basis of numbers of parameters, types of the parameters and order of the parameters

CONCLUSION : Thus we have successfully studied about editing, compiling , executing and testing a Java program to implement constructors.

Experiment No.- 04

AIM: Edit, compile , execute and test a Java program to implement constructors.

PROGRAM :

```
      // Java Program to illustrate calling a
   // no-argument constructor
   import java.io.*;

   class Constructor
      {
          int num;
          String name;

          // this would be invoked while an object
         // of that class is created.
          Constructor ()
          {
              System.out.println("Constructor called");
          }
      }

      class GFG
      {
          public static void main (String[] args)
          {
           // this would invoke default constructor.
           Constructor constructor1 = new Constructor();

           // Default constructor provides the default
           // values to the object like 0, null
           System.out.println(constructor1.name);
           System.out.println(constructor1.num);
       }
   }
```

Output :

```
Constructor called

null

0
```

PROGRAM :

```
// Java Program to illustrate calling of
// parameterized constructor.
importjava.io.*;

class Constructor
{
    // data members of the class.
    String name;
    int id;

    // constructor would initialize data members
    // with the values of passed arguments while
    // object of that class created.
    Constructor(String name, int id)
    {
        this.name = name;
        this.id = id;
    }
}

classGFG
{
    public static void main (String[] args)
    {
        // this would invoke the parameterized constructor.
        Constructor constructor1 = newConstructor("adam", 1);
        System.out.println("ConstructorName :"+ constructor1.name +
                           " and ConstructorId :"+ constructor1.id);
    }
}
```

Output:

```
ConstructorName :adam and ConstructorId :1
```

PROGRAM :

```
// Java Program to illustrate constructor overloading
// using same task (addition operation ) for different
// types of arguments.
Import java.io.*;

Class Constructor
{
    // constructor with one argument
   Constructor(String name)
    {
        System.out.println("Constructor with one "+
                     "argument - String : "+ name);
    }

    // constructor with two arguments
   Constructor(String name, int age)
    {

        System.out.println("Constructor with two arguments : "+
                " String and Integer : "+ name + " "+ age);

    }

    // Constructor with one argument but with different
    // type than previous..
    Constructor(long id)
    {
        System.out.println("Constructor with one argument : "+
                                            "Long : "+ id);
    }
}

class GFG
{
    public static void main(String[] args)
    {
        // Creating the objects of the class named 'Constructor'
        // by passing different arguments

        // Invoke the constructor with one argument of
        // type 'String'.
        Constructor constructor2 = new Constructor ("Shikhar");

        // Invoke the constructor with two arguments
        Constructor constructor3 = new Constructor ("Dharmesh", 26);

        // Invoke the constructor with one argument of
        // type 'Long'.
        Constructor constructor4 = new Constructor (325614567);
    }
}
```

Output:

```
Constructor with one argument - String :Shikhar

Constructor with two arguments - String and Integer :Dharmesh 26

Constructor with one argument - Long : 325614567
```