Experiment 3: Write a Java program to demonstrate the use of different types of constructors.

(a) Default Constructor

Aim:
To write a Java program to demonstrate the use of a default constructor.

Theory:
A default constructor is automatically invoked when an object is created. It initializes data members with default values.

Algorithm:

1. Create a class.
2. Define a default constructor.
3. Create an object of the class.
4. Display initialized values.

Program Code:

```
class DefaultConstructorDemo {

   int id;

   String name;


   DefaultConstructorDemo() {

      id = 1;

      name = "Student";

   }
```

```
    void display() {
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
    }

    public static void main(String[] args) {
        DefaultConstructorDemo obj = new DefaultConstructorDemo();
        obj.display();
    }
}
```

Result: The default constructor is invoked and initializes the object successfully.

(b) Parameterized Constructor

Aim:
To write a Java program to demonstrate the use of a parameterized constructor.

Theory:
A parameterized constructor accepts arguments to initialize object data members.

Algorithm:

1. Create a class.
2. Define a parameterized constructor.
3. Pass values while creating an object.
4. Display object data.

Program Code:

```
class ParameterizedConstructorDemo {
   int id;
   String name;

   ParameterizedConstructorDemo(int i, String n) {
      id = i;
      name = n;
   }

   void display() {
      System.out.println("ID: " + id);
      System.out.println("Name: " + name);
   }

   public static void main(String[] args) {
      ParameterizedConstructorDemo obj =
            new ParameterizedConstructorDemo(101, "Rahul");
        obj.display();
    }
}
```

Result: The parameterized constructor initializes the object using provided values.

(c) Copy Constructor

Aim:
To write a Java program to demonstrate the use of a copy constructor.

Theory:
A copy constructor initializes a new object using another object of the same class. Java does not provide a built-in copy constructor, but it can be user-defined.

Algorithm:

1. Create a class.
2. Define a parameterized constructor.
3. Define a copy constructor.
4. Create a new object using the existing object.

Program Code:

```
class CopyConstructorDemo {
   int id;
   String name;

   CopyConstructorDemo(int i, String n) {
      id = i;
      name = n;
   }

   CopyConstructorDemo(CopyConstructorDemo obj) {
      id = obj.id;
      name = obj.name;
   }
```

```
    void display() {
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
    }

    public static void main(String[] args) {
        CopyConstructorDemo obj1 =
            new CopyConstructorDemo(201, "Amit");
        CopyConstructorDemo obj2 =
            new CopyConstructorDemo(obj1);

        obj1.display();
        obj2.display();
    }
}
```

Result: The copy constructor successfully creates a new object using an existing object.