**Name**: Ravi Jamanbhai Makwana

**Roll No**: 21BCP418

Div-6, G-12

**Assignment-9 (Flyweight Pattern)**

**Aim:**

To implement Flyweight Design Pattern using Accommodation management system.

**Code:**

**Room.java**

/\*

 \* Flyweight DP is used when we need to create lots of objects of same class

 \* which will lead to lot of memory usage.

 \* So, instead we can create the object only once and then share the attributes

 \* of that object instead of creating separate objects eveytime.

 \*/

/\*

 \* Shared properties(attributes) are called Intrinsic state

 \* Properties(attributes) NOT shared are called Extrinsic state.

 \*/

/\*

 \* All Rooms share prop. like number of lights, fans, chairs, tables (Intrinsic states)

 \* but they do not share some prop. like availability of AC, flooring type

 \*/

public interface Room{

    public void setFlooring(String Flooring);

    public void setACavailability(String ac);

    public void showProperties();

}

**GeneralRoom.java**

public class GeneralRoom implements Room{

    //common attributes (Intrinsic state)

    private final int numberOfLights;

    private final int numberOfFans;

    private final int numberOfChairs;

    private final int numberOfTables;

    //NON-common attributes (Extrinsic state)

    private String flooring;

    private String ac;

    //put those things (initilize those things) in constructor,

    //which will be COMMON attributes for all objects (Instrinsic state).

    public GeneralRoom(){

        this.numberOfLights = 4;

        this.numberOfFans = 2;

        this.numberOfChairs = 2;

        this.numberOfTables = 3;

    }

    //below methods are for initializing Extrinsic state (for attributes that are NOT common).

    @Override

    public void setFlooring(String flooring) {

        this.flooring = flooring;

    }

    @Override

    public void setACavailability(String ac) {

        this.ac = ac;

    }

    @Override

    public void showProperties() {

        System.out.println("Properties of room are:");

        System.out.println("Number of Lights: "+numberOfLights);

        System.out.println("Number of Fans: "+numberOfFans);

        System.out.println("Number of Chairs: "+numberOfChairs);

        System.out.println("Number of Tables: "+numberOfTables);

        System.out.println("\*\*Flooring Type: "+flooring);

        System.out.println("\*\*AC Available: "+ac);

        System.out.println("----------------------------------------");

    }

}

**RoomFactory.java**

import java.util.\*;

public class RoomFactory {

    public static HashMap<String,Room> m = new HashMap<String,Room>();

    public static Room getRoom (String st){

        Room r = null;

        //if object of that type exists in the map..

        if (m.get(st) != null){

            r = m.get(st);

        }

        //if object of that type does NOT exist in the map..

        //then CREATE the object and

        //then add it to the map

        else{

            if (st.equalsIgnoreCase("GeneralRoom")){

                System.out.println("ROOM OBJECT CREATED");

                //new object created

                r = new GeneralRoom();

            }

            else{

                System.out.println("NO SUCH ROOM AVAILABLE");

            }

            //add the object created to the map

            m.put(st,r);

        }

        return r;

    }

}

**Test.java**

public class Test {

    public static void main(String args[]){

        Room r1 = RoomFactory.getRoom("GeneralRoom");

        r1.setFlooring("Wooden");

        r1.setACavailability("Yes");

        r1.showProperties();

        Room r2 = RoomFactory.getRoom("GeneralRoom");

        r2.setFlooring("Marble");

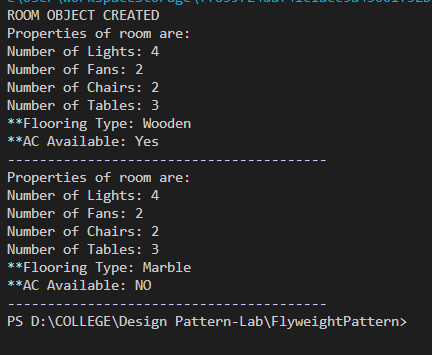
        r2.setACavailability("NO");

        r2.showProperties();

    }

}

**Output:**



The first line of output (in capital letters) shows that object was created just once. So, the attributes common to all objects thereafter created will not use much space as some of these attributes are shared between all objects.

Although a new object is created, but they share attributes, which eventually helps in saving space.

**Class Diagram:**

