**Name**: Ravi Jamanbhai Makwana

**Roll No**: 21BCP418

Div-6, G-12

**Assignment-4 (Abstract Factory Pattern)**

**Aim:**

To implement Abstract Factory Design Pattern using Accommodation management system.

**Code:**

**RoomType.java**

package AbstractFactoryPattern;

public interface RoomType{

    public void allotRoom();

}

**SingleRoom.java**

package AbstractFactoryPattern;

public class SingleRoom implements RoomType{

    @Override

    public void allotRoom(){

        System.out.println("Single Room alloted");

    }

}

**DoubleRoom.java**

package AbstractFactoryPattern;

public class DoubleRoom implements RoomType{

    @Override

    public void allotRoom(){

        System.out.println("Double Room alloted");

    }

}

**QuadRoom.java**

package AbstractFactoryPattern;

public class QuadRoom implements RoomType{

    @Override

    public void allotRoom(){

        System.out.println("Quad Room alloted");

    }

}

**FlooringType.java**

package AbstractFactoryPattern;

interface FlooringType{

    void applyFlooring();

}

**MarbleFlooring.java**

package AbstractFactoryPattern;

public class MarbleFlooring implements FlooringType{

    @Override

    public void applyFlooring(){

        System.out.println("Marble Flooring applied");

    }

}

**WoodenFlooring.java**

package AbstractFactoryPattern;

public class WoodenFlooring implements FlooringType{

    @Override

    public void applyFlooring(){

        System.out.println("Wooden Flooring applied");

    }

}

**RoomFactory.java**

package AbstractFactoryPattern;

public class RoomFactory extends AbstractFactory{

    //getRoomType method returns object of input type RoomType

    @Override

    public RoomType getRoomType(String roomtype){

        if (roomtype == null){

            return null;

        }

        if (roomtype.equalsIgnoreCase("Single")){

            return new SingleRoom();

        }

        else if (roomtype.equalsIgnoreCase("Double")){

            return new DoubleRoom();

        }

        else if (roomtype.equalsIgnoreCase("Quad")){

            return new QuadRoom();

        }

        return null;

    }

    @Override

    public FlooringType getFlooringType(String flooring){

        return null;

    }

}

**FloorFactory.java**

package AbstractFactoryPattern;

public class FloorFactory extends AbstractFactory{

    //getFlooringType method returns object of input type FlooringType

    @Override

    FlooringType getFlooringType(String flooring){

        if (flooring == null){

            return null;

        }

        if (flooring.equalsIgnoreCase("Marble")){

            return new MarbleFlooring();

        }

        else if (flooring.equalsIgnoreCase("Wooden")){

            return new WoodenFlooring();

        }

        return null;

    }

    @Override

    RoomType getRoomType(String room){

        return null;

    }

}

**AbstractFactory.java**

package AbstractFactoryPattern;

public abstract class AbstractFactory {

    abstract RoomType getRoomType(String room);

    abstract FlooringType getFlooringType(String flooring);

}

**FactoryProducer.java**

package AbstractFactoryPattern;

public class FactoryProducer{

    //this is factory of factories, so we use this factory (super) to select other factories, to further make objects

    //AbstractFactory method returns object of input type AbstractFactory

    public static AbstractFactory getFactory (String choice){

        if (choice.equalsIgnoreCase("Room")){

            return new RoomFactory();

        }

        else if (choice.equalsIgnoreCase("Flooring")){

            return new FloorFactory();

        }

        return null;

    }

}

**Test.java**

package AbstractFactoryPattern;

public class Test {

    public static void main(String args[]){

        //get RoomFactory

        AbstractFactory rf = FactoryProducer.getFactory("Room");

        //get type of room

        RoomType a1 = rf.getRoomType("Single");

        //call method allotRoom for this room object

        a1.allotRoom();

        //get FlooringFactory

        AbstractFactory ff = FactoryProducer.getFactory("Flooring");

        //get flooring type

        FlooringType a2 = ff.getFlooringType("Wooden");

        //call method applyFlooring for this flooring object

        a2.applyFlooring();

    }

}

**Output:**

