

Savitribai Phule Pune University

S.Y.M.Sc(Comp. Sci.) Practical Examination

Lab Course on CS-601-MJ (Software Architecture & Design Patten)

SlipNo.1 & 20: Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.

```
package sadp20;  
import java.io.*;  
  
class LowerCaseReader extends FilterReader {  
    protected LowerCaseReader(Reader in) {  
        super(in);  
    }  
    public int read() throws IOException {  
        int c = super.read();  
        return (c == -1 ? c : Character.toLowerCase((char)c));  
    }  
}  
  
public class Main {  
    public static void main(String[] args) throws Exception {  
        Reader r = new LowerCaseReader(new FileReader("input.txt"));  
        int ch;  
        while ((ch = r.read()) != -1) {  
            System.out.print((char) ch);  
        }  
        r.close();  
    }  
}
```

SlipNo.4 & 19: Write a Java Program to implement Factory method for Pizza Store with **createPizza()**, **orederPizza()**, **prepare()**, **Bake()**, **cut()**, **box()**. Use this to create variety of pizza's like **NyStyleCheesePizza**, **ChicagoStyleCheesePizza** etc.

Pizza.java

```
package sadp18;  
public abstract class Pizza {  
    String name;  
    public void prepare() {  
        System.out.println("Preparing " + name);  
    }  
    public void bake() {  
        System.out.println("Baking " + name);  
    }  
    public void cut() {  
        System.out.println("Cutting " + name);  
    }  
    public void box() {  
        System.out.println("Boxing " + name);  
    }  
    public String getName() {  
        return name;  
    }  
}
```

NYStyleCheesePizza.java

```
package sadp18;

public class NYStyleCheesePizza extends Pizza {

    public NYStyleCheesePizza() {
        name = "NY Style Cheese Pizza";
    }

}
```

ChicagoStyleCheesePizza.java

```
package sadp18;

public class ChicagoStyleCheesePizza extends Pizza {

    public ChicagoStyleCheesePizza() {
        name = "Chicago Style Cheese Pizza";
    }

    @Override
    public void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}
```

PizzaStore.java

```
package sadp18;

public abstract class PizzaStore {

    public Pizza orderPizza(String type) {
        Pizza pizza = createPizza(type);
        pizza.prepare();
        pizza.bake();
    }
}
```

```
    pizza.cut();
    pizza.box();
    return pizza;
}

// Factory Method

protected abstract Pizza createPizza(String type);

}
```

NYPizzaStore.java

```
package sadp18;

public class NYPizzaStore extends PizzaStore {

    @Override

    protected Pizza createPizza(String type) {

        if (type.equalsIgnoreCase("cheese")) {

            return new NYStyleCheesePizza();

        }

        return null;

    }

}
```

ChicagoPizzaStore.java

```
package sadp18;

public class ChicagoPizzaStore extends PizzaStore {

    @Override

    protected Pizza createPizza(String type) {

        if (type.equalsIgnoreCase("cheese")) {

            return new ChicagoStyleCheesePizza();

        }

    }

}
```

```
    return null;  
}  
}
```

Main.java

```
package sadp18;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        PizzaStore nyStore = new NYPizzaStore();  
        PizzaStore chicagoStore = new ChicagoPizzaStore();  
  
        Pizza pizza1 = nyStore.orderPizza("cheese");  
        System.out.println("Ordered a " + pizza1.getName() + "\n");  
  
        Pizza pizza2 = chicagoStore.orderPizza("cheese");  
        System.out.println("Ordered a " + pizza2.getName() + "\n");  
    }  
}
```

SlipNo.6:Write a Java Program to implement command pattern to test Remote Control.

Command.java(Interface)

```
package sadp6;  
public interface Command {  
    void execute();  
}
```

Light.java

```
package sadp6;  
public class Light {  
    public void on() {  
        System.out.println("Light is ON");  
    }  
    public void off() {  
        System.out.println("Light is OFF");  
    }  
}
```

LightOnCommand.java

```
package sadp6;  
public class LightOnCommand implements Command {  
    private Light light;  
    public LightOnCommand(Light light) {  
        this.light = light;  
    }  
    public void execute() {  
        light.on();  
    }  
}
```

```
}
```

```
}
```

LightOffCommand.java

```
package sadp6;

public class LightOffCommand implements Command {
    private Light light;

    public LightOffCommand(Light light) {
        this.light = light;
    }

    public void execute() {
        light.off();
    }
}
```

RemoteControl.java

```
package sadp6;

public class RemoteControl {
    private Command command;

    public void setCommand(Command command) {
        this.command = command;
    }

    public void pressButton() {
        command.execute();
    }
}
```

Main.java

```
package sadp6;

public class Main {

    public static void main(String[] args) {

        Light livingRoomLight = new Light();

        Command lightOn = new LightOnCommand(livingRoomLight);

        Command lightOff = new LightOffCommand(livingRoomLight);

        RemoteControl remote = new RemoteControl();

        // Turn ON

        remote.setCommand(lightOn);

        remote.pressButton();

        // Turn OFF

        remote.setCommand(lightOff);

        remote.pressButton();

    }

}
```

Slip.No.7:Write a Java Program to implement undo command to test Ceiling fan.

CeilingFan.java

```
package sadp7;

public class CeilingFan {

    private String speed;

    public void high() {
        speed = "High";
        System.out.println("Ceiling fan is on High");
    }

    public void medium() {
        speed = "Medium";
        System.out.println("Ceiling fan is on Medium");
    }

    public void off() {
        System.out.println("Ceiling fan is Off");
    }

    public String getSpeed() {
        return speed;
    }
}
```

CeilingFanHighCommand.java

```
package sadp7;

public class CeilingFanHighCommand implements Command {

    private CeilingFan fan;

    private String prevSpeed;

    public CeilingFanHighCommand(CeilingFan fan) {

        this.fan = fan;
    }

    public void execute() {

        prevSpeed = fan.getSpeed();

        fan.high();
    }

    public void undo() {

        if(prevSpeed != null) {

            switch (prevSpeed) {

                case "Medium":

                    fan.medium();

                    break;

                case "High":

                    fan.high();

                    break;

                default:

                    fan.off();

                    break;
            }
        }else {
            fan.off();
        }
    }
}
```

Command.java(interface)

```
package sadp7;

public interface Command {  
    void execute();  
    void undo();  
}
```

RemoteControl.java

```
package sadp7;

public class RemoteControl {  
    private Command command;  
  
    public void setCommand(Command command) {  
        this.command = command;  
    }  
  
    public void pressButton() {  
        command.execute();  
    }  
  
    public void pressUndo() {  
        command.undo();  
    }  
}
```

Main.java

```
package sadp7;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        CeilingFan fan = new CeilingFan();
        Command fanHigh = new CeilingFanHighCommand(fan);
        RemoteControl remote = new RemoteControl();
        remote.setCommand(fanHigh);
        remote.pressButton();
        remote.pressUndo();
    }
}
```

SlipNo.12: Write a Java Program to implement Decorator Pattern for interface Car to define the assemble() method and then decorate it to Sports car and Luxury Car.

BasicCar.java

```
package sadp12;  
public class BasicCar implements Car {  
    public String assemble() {  
        return "Basic Car";  
    }  
}
```

Car.java(interface)

```
package sadp12;  
public interface Car {  
    String assemble();  
}
```

CarDecorator.java

```
package sadp12;  
public abstract class CarDecorator implements Car {  
    protected Car car;  
    public CarDecorator(Car car) {  
        this.car = car;  
    }  
    public String assemble() {  
        return car.assemble();  
    }  
}
```

LuxuryCar.java

```
package sadp12;

public class LuxuryCar extends CarDecorator {

    public LuxuryCar(Car car) {
        super(car);
    }

    public String assemble() {
        return super.assemble() + ",Adding features of Luxury Car";
    }
}
```

SportsCar.java

```
package sadp12;

public class SportsCar extends CarDecorator {

    public SportsCar(Car car) {
        super(car);
    }

    public String assemble() {
        return super.assemble() + ", Adding features of Sports Car";
    }
}
```

Main.java

```
package sadp12;

public class Main {

    public static void main(String[] args) {
        Car basicCar = new BasicCar();
        System.out.println(basicCar.assemble());

        Car sportsCar = new SportsCar(basicCar);
        System.out.println(sportsCar.assemble());

        Car luxuryCar = new LuxuryCar(basicCar);
        System.out.println(luxuryCar.assemble());

        Car sportsLuxuryCar = new SportsCar(new LuxuryCar(basicCar));
        System.out.println(sportsLuxuryCar.assemble());
    }
}
```

SlipNo.17: Write a Java Program to implement Abstract Factory Pattern for Shape interface.

Shape.java(interface)

```
package sadp17;  
public interface Shape {  
    void draw();  
}
```

Circle.java

```
package sadp17;  
public class Circle implements Shape {  
    public void draw() {  
        System.out.println("Drawing a Circle");  
    }  
}
```

Square.java

```
package sadp17;  
public class Square implements Shape {  
    @Override  
    public void draw() {  
        System.out.println("Drawing a Square");  
    } }
```

ShapeFactory.java

```
package sadp17;  
public abstract class ShapeFactory {  
    public abstract Shape createShape(String shapeType);  
}
```

ColorShapeFactory.java

```
package sadp17;

public class ColorShapeFactory extends ShapeFactory {

    @Override
    public Shape createShape(String shapeType) {
        if(shapeType.equalsIgnoreCase("CIRCLE")){
            return new Circle();
        }else if(shapeType.equalsIgnoreCase("SQUARE")){
            return new Square();
        }
        return null;
    }
}
```

Main.java

```
package sadp17;

public class Main {
    public static void main(String []args) {
        ShapeFactory factory = new ColorShapeFactory();

        Shape circle = factory.createShape("CIRCLE");
        circle.draw();

        Shape square = factory.createShape("SQUARE");
        square.draw();
    }
}
```

