Design Pattern Lab Manual

Name: Jaivik Jariwala

Roll No.: 21BCP004

Division: 1

Group: 1

# Structural Design Pattern

|  |  |
| --- | --- |
| Sr. No | Name |
| 1 | Adapter |
| 2 | Composite |
| 3 | Façade |
| 4 | Decorator |
| 5 | Flyweight |

## Adapter Structural Design Pattern

### Example 1: AdvancedMedia Player

MediaPlayer.java

public interface *MediaPlayer* {  
 public void play(*String audioType*, *String fileName*);  
}

AdvancedMediaPlayer.java

public interface *AdvancedMediaPlayer* {  
 public void playVlc(*String fileName*);  
 public void playMp4(*String fileName*);  
}

VlcPlayer.java

public class *VlcPlayer* implements *AdvancedMediaPlayer*{  
 *@Override* public void playVlc(*String fileName*) {  
 *System*.out.println("Playing vlc file. Name: "+ *fileName*);  
 }  
  
 *@Override* public void playMp4(*String fileName*) {  
 *//do nothing* }  
}

Mp4Player.java

public class *Mp4Player* implements *AdvancedMediaPlayer*{  
  
 *@Override* public void playVlc(*String fileName*) {  
 *//do nothing* }  
  
 *@Override* public void playMp4(*String fileName*) {  
 *System*.out.println("Playing mp4 file. Name: "+ *fileName*);  
 }  
}

MediaAdapter.java

public class *MediaAdapter* implements *MediaPlayer* {  
  
 *AdvancedMediaPlayer* advancedMusicPlayer;  
  
 public MediaAdapter(*String audioType*){  
  
 if(*audioType*.equalsIgnoreCase("vlc") ){  
 advancedMusicPlayer = new VlcPlayer();  
  
 }else if (*audioType*.equalsIgnoreCase("mp4")){  
 advancedMusicPlayer = new Mp4Player();  
 }  
 }  
  
 *@Override* public void play(*String audioType*, *String fileName*) {  
  
 if(*audioType*.equalsIgnoreCase("vlc")){  
 advancedMusicPlayer.playVlc(*fileName*);  
 }  
 else if(*audioType*.equalsIgnoreCase("mp4")){  
 advancedMusicPlayer.playMp4(*fileName*);  
 }  
 }  
}

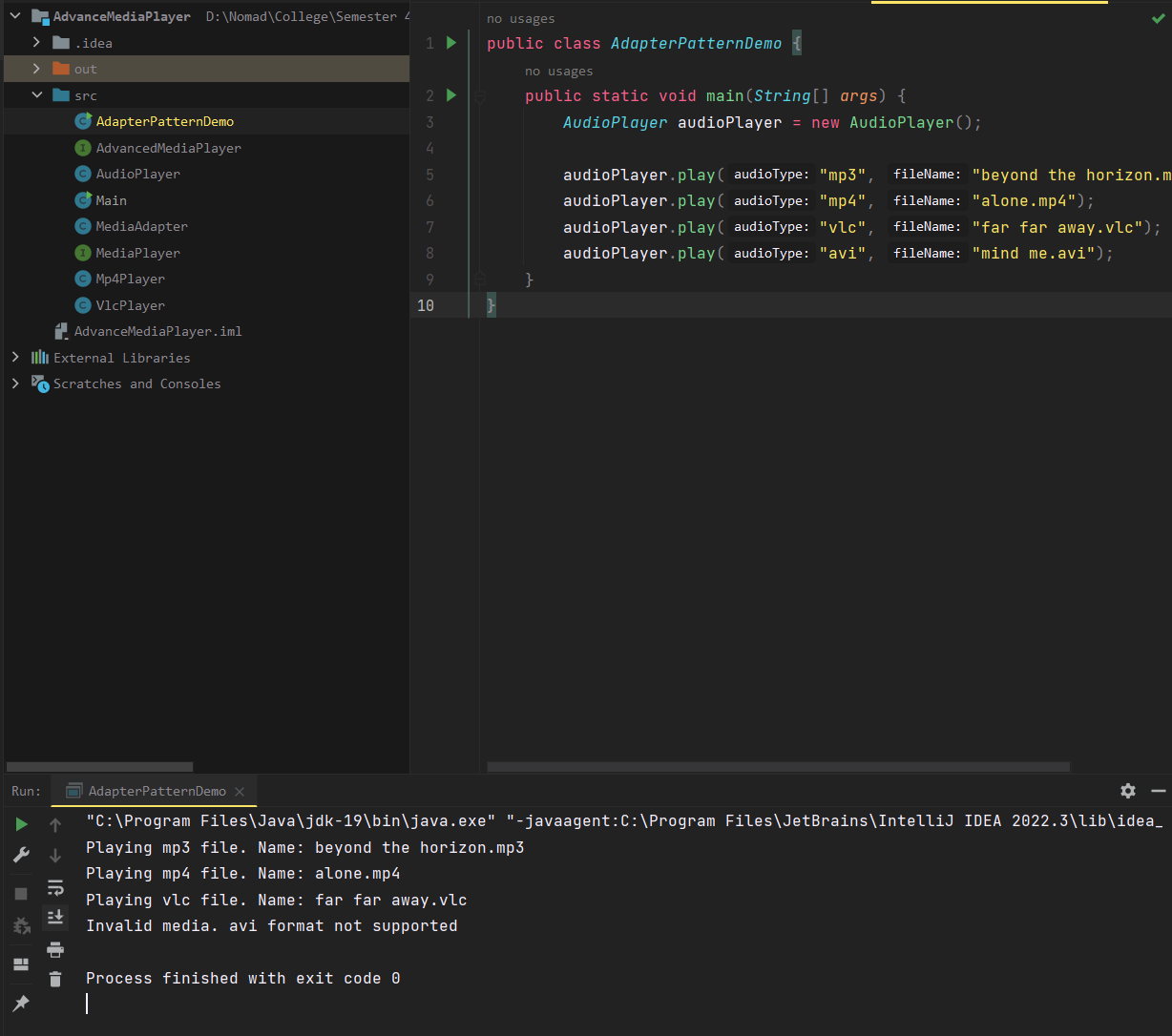
AudioPlayer.java

public class *AudioPlayer* implements *MediaPlayer* {  
 *MediaAdapter* mediaAdapter;  
  
 *@Override* public void play(*String audioType*, *String fileName*) {  
  
 *//inbuilt support to play mp3 music files* if(*audioType*.equalsIgnoreCase("mp3")){  
 *System*.out.println("Playing mp3 file. Name: " + *fileName*);  
 }  
  
 *//mediaAdapter is providing support to play other file formats* else if(*audioType*.equalsIgnoreCase("vlc") || *audioType*.equalsIgnoreCase("mp4")){  
 mediaAdapter = new MediaAdapter(*audioType*);  
 mediaAdapter.play(*audioType*, *fileName*);  
 }  
  
 else{  
 *System*.out.println("Invalid media. " + *audioType* + " format not supported");  
 }  
 }  
}

AdapterPatternDemo.java

public class *AdapterPatternDemo* {  
 public static void main(*String*[] *args*) {  
 *AudioPlayer* audioPlayer = new AudioPlayer();  
  
 audioPlayer.play("mp3", "beyond the horizon.mp3");  
 audioPlayer.play("mp4", "alone.mp4");  
 audioPlayer.play("vlc", "far far away.vlc");  
 audioPlayer.play("avi", "mind me.avi");  
 }  
}

Output



### Example 2: Bank-Access

CreditCard.java

public interface *CreditCard* {  
 public void giveBankDetails();  
 public *String* getCreditCard();  
}

BankDetails.java

public class *BankDetails*{  
 private *String* bankName;  
 private *String* accHolderName;  
 private long accNumber;  
  
 public *String* getBankName() {  
 return bankName;  
 }  
 public void setBankName(*String bankName*) {  
 this.bankName = *bankName*;  
 }  
 public *String* getAccHolderName() {  
 return accHolderName;  
 }  
 public void setAccHolderName(*String accHolderName*) {  
 this.accHolderName = *accHolderName*;  
 }  
 public long getAccNumber() {  
 return accNumber;  
 }  
 public void setAccNumber(long *accNumber*) {  
 this.accNumber = *accNumber*;  
 }  
}

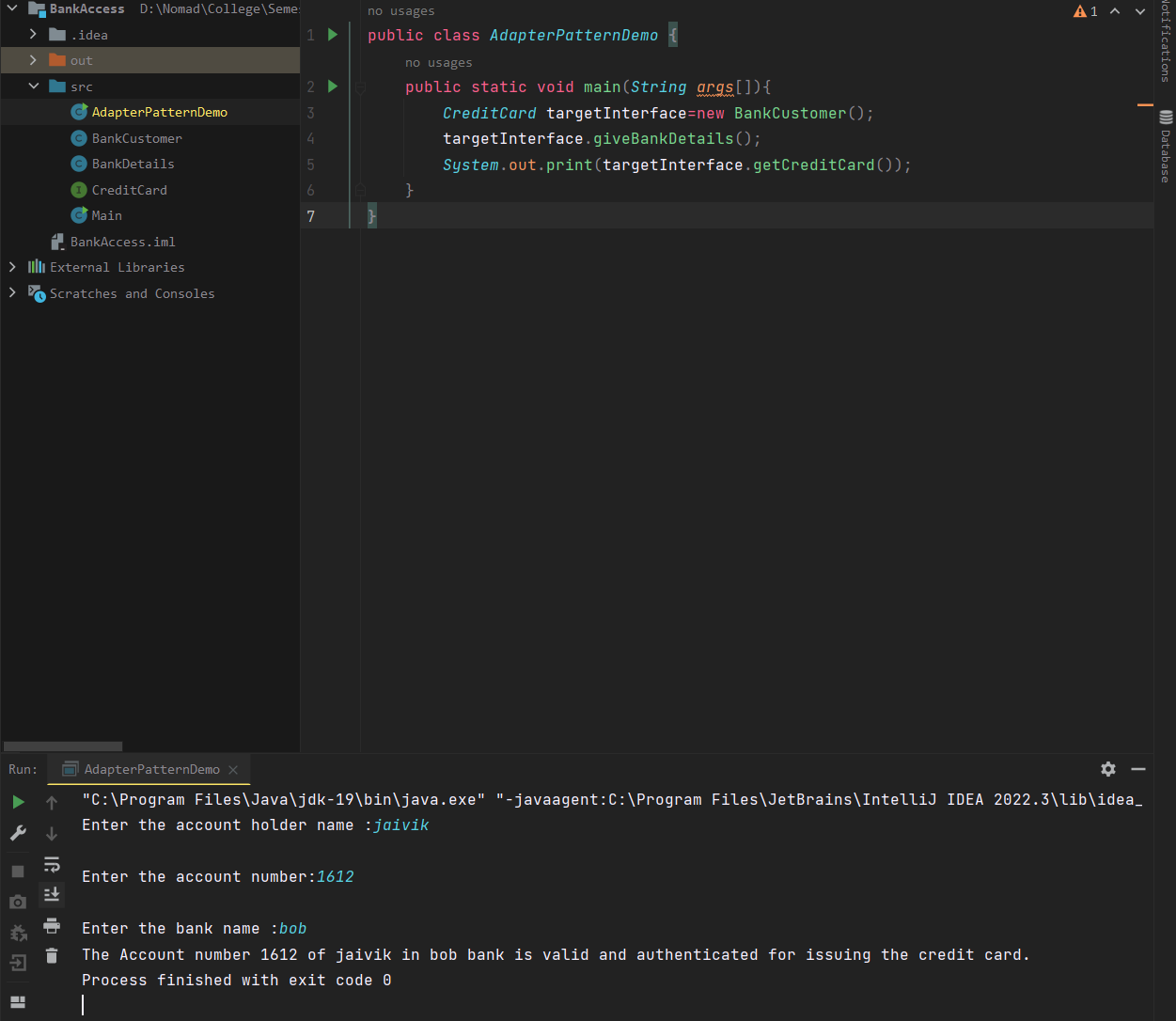
BankCustomer.java

public class *BankDetails*{  
 private *String* bankName;  
 private *String* accHolderName;  
 private long accNumber;  
  
 public *String* getBankName() {  
 return bankName;  
 }  
 public void setBankName(*String bankName*) {  
 this.bankName = *bankName*;  
 }  
 public *String* getAccHolderName() {  
 return accHolderName;  
 }  
 public void setAccHolderName(*String accHolderName*) {  
 this.accHolderName = *accHolderName*;  
 }  
 public long getAccNumber() {  
 return accNumber;  
 }  
 public void setAccNumber(long *accNumber*) {  
 this.accNumber = *accNumber*;  
 }  
}

AdapterPatternDemo.java

public class *AdapterPatternDemo* {  
 public static void main(*String args*[]){  
 *CreditCard* targetInterface=new BankCustomer();  
 targetInterface.giveBankDetails();  
 *System*.out.print(targetInterface.getCreditCard());  
 }  
}

Output



## Composite Structural Design Pattern

### Example 1: EmployeeField data

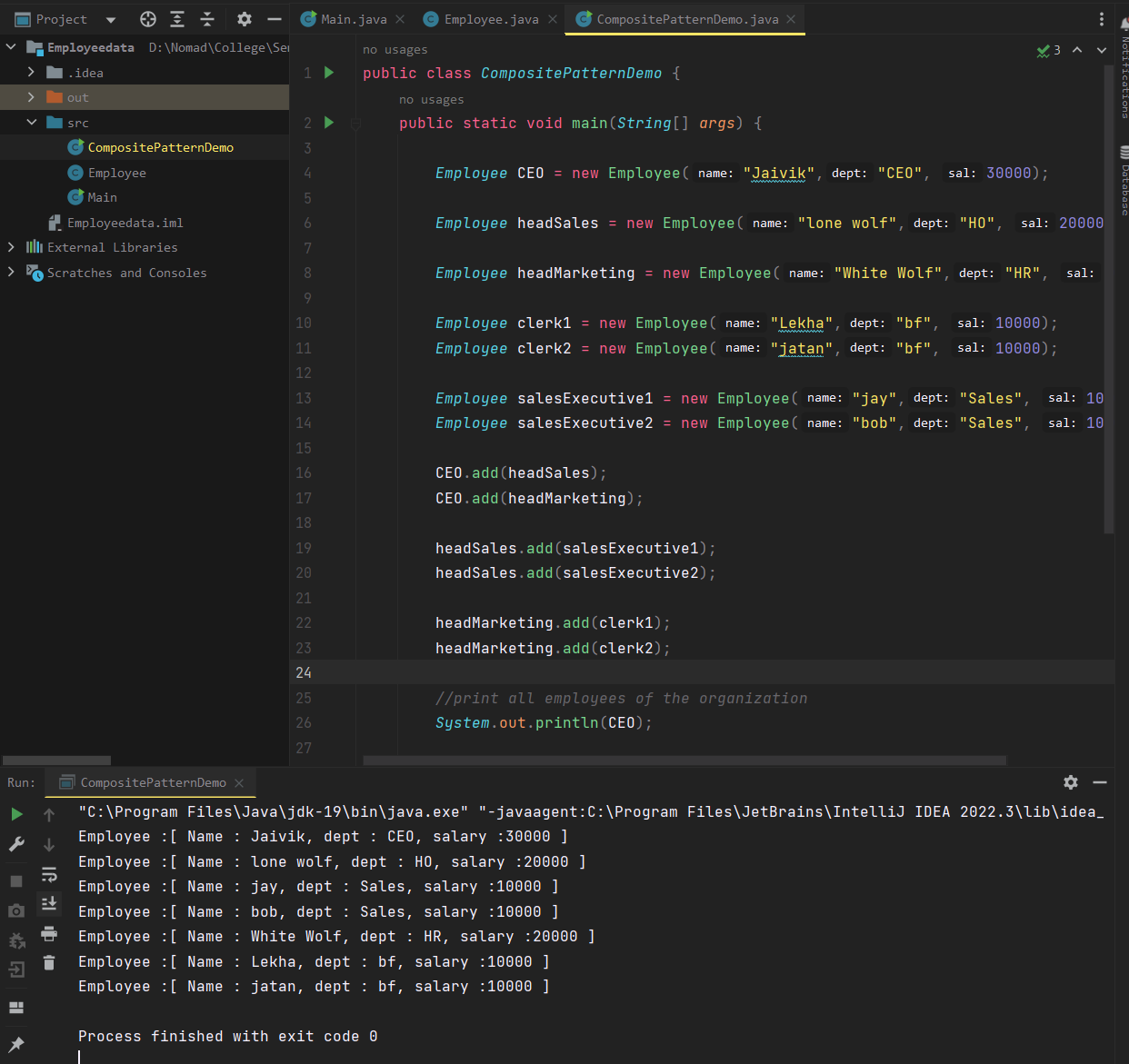
Employee.java

import *java.util.ArrayList*;  
import *java.util.List*;  
  
public class *Employee* {  
 private *String* name;  
 private *String* dept;  
 private int salary;  
 private *List*<*Employee*> subordinates;  
  
 *// constructor* public Employee(*String name*,*String dept*, int *sal*) {  
 this.name = *name*;  
 this.dept = *dept*;  
 this.salary = *sal*;  
 subordinates = new ArrayList<*Employee*>();  
 }  
  
 public void add(*Employee e*) {  
 subordinates.add(*e*);  
 }  
  
 public void remove(*Employee e*) {  
 subordinates.remove(*e*);  
 }  
  
 public *List*<*Employee*> getSubordinates(){  
 return subordinates;  
 }  
  
 public *String* toString(){  
 return ("Employee :[ Name : " + name + ", dept : " + dept + ", salary :" + salary+" ]");  
 }  
}

CompositePatternDemo.java

public class *CompositePatternDemo* {  
 public static void main(*String*[] *args*) {  
  
 *Employee* CEO = new Employee("Jaivik","CEO", 30000);  
  
 *Employee* headSales = new Employee("lone wolf","HO", 20000);  
  
 *Employee* headMarketing = new Employee("White Wolf","HR", 20000);  
  
 *Employee* clerk1 = new Employee("Lekha","bf", 10000);  
 *Employee* clerk2 = new Employee("jatan","bf", 10000);  
  
 *Employee* salesExecutive1 = new Employee("jay","Sales", 10000);  
 *Employee* salesExecutive2 = new Employee("bob","Sales", 10000);  
  
 CEO.add(headSales);  
 CEO.add(headMarketing);  
  
 headSales.add(salesExecutive1);  
 headSales.add(salesExecutive2);  
  
 headMarketing.add(clerk1);  
 headMarketing.add(clerk2);  
  
 *//print all employees of the organization  
 System*.out.println(CEO);  
  
 for (*Employee* headEmployee : CEO.getSubordinates()) {  
 *System*.out.println(headEmployee);  
  
 for (*Employee* employee : headEmployee.getSubordinates()) {  
 *System*.out.println(employee);  
 }  
 }  
 }  
}

Output



### Example : 2 DrawShapeswithColour

Shape.java

public interface *Shape* {  
 public void draw(*String fillColor*);  
}

Circle.java

public class *Circle* implements *Shape* {  
  
 *@Override* public void draw(*String fillColor*) {  
 *System*.out.println("Drawing Circle with color "+*fillColor*);  
 }  
  
}

Triangle.java

public class *Triangle* implements *Shape*{  
 *@Override* public void draw(*String fillColor*) {  
 *System*.out.println("Drawing Triangle with color "+*fillColor*);  
 }  
}

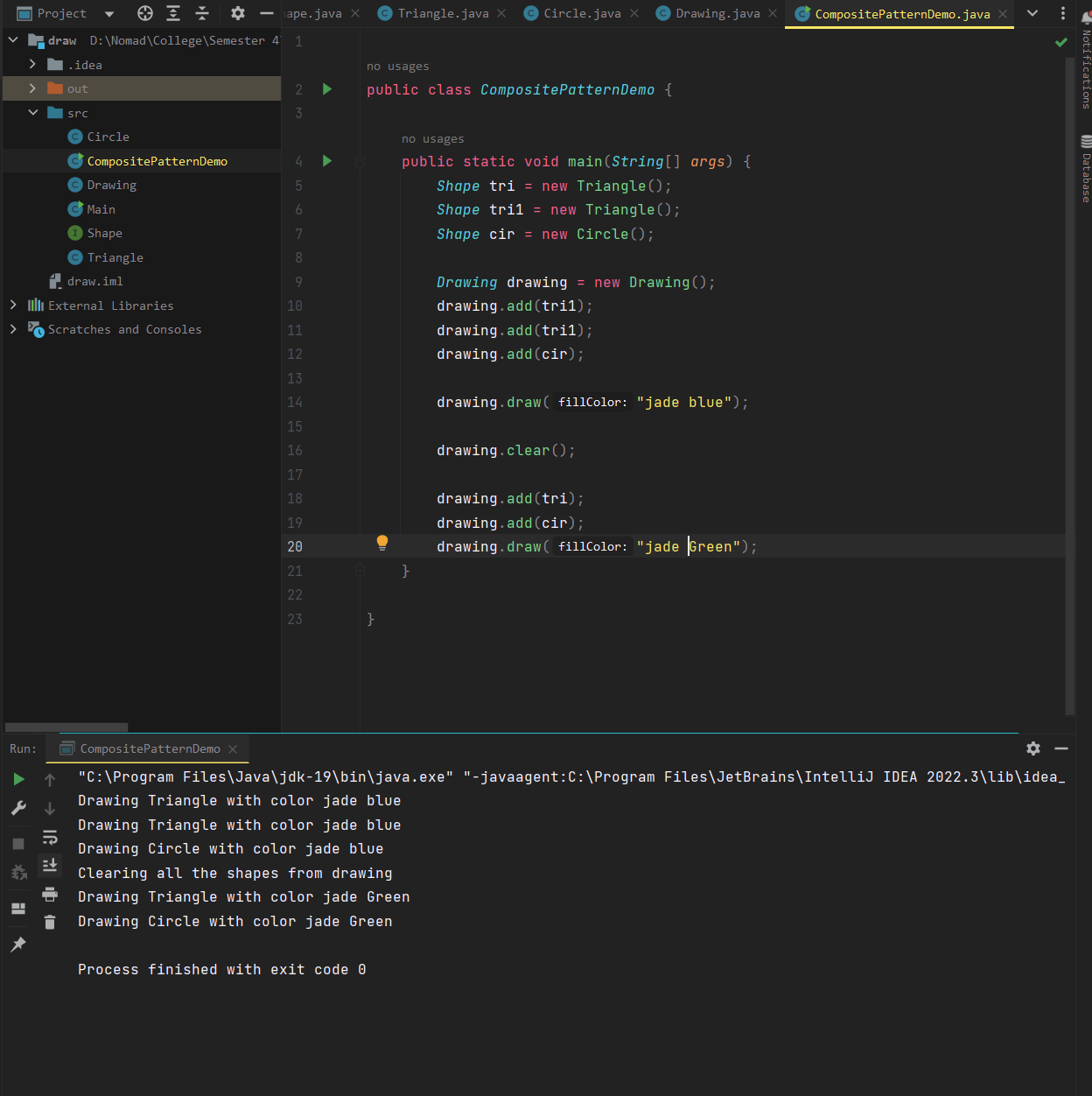
Drawing.java

import *java.util.ArrayList*;  
import *java.util.List*;  
  
public class *Drawing* implements *Shape*{  
  
 *//collection of Shapes* private *List*<*Shape*> shapes = new ArrayList<*Shape*>();  
  
 *@Override* public void draw(*String fillColor*) {  
 for(*Shape* sh : shapes)  
 {  
 sh.draw(*fillColor*);  
 }  
 }  
  
 *//adding shape to drawing* public void add(*Shape s*){  
 this.shapes.add(*s*);  
 }  
  
 *//removing shape from drawing* public void remove(*Shape s*){  
 shapes.remove(*s*);  
 }  
  
 *//removing all the shapes* public void clear(){  
 *System*.out.println("Clearing all the shapes from drawing");  
 this.shapes.clear();  
 }  
}

CompositePatternDemo.java

public class *CompositePatternDemo* {  
  
 public static void main(*String*[] *args*) {  
 *Shape* tri = new Triangle();  
 *Shape* tri1 = new Triangle();  
 *Shape* cir = new Circle();  
  
 *Drawing* drawing = new Drawing();  
 drawing.add(tri1);  
 drawing.add(tri1);  
 drawing.add(cir);  
  
 drawing.draw("jade blue");  
  
 drawing.clear();  
  
 drawing.add(tri);  
 drawing.add(cir);  
 drawing.draw("jade Green");  
 }  
  
}

Output



## Facade Structural Design Pattern

### Example: 1 Draw shapes

Shape.java

public interface *Shape* {  
 void draw();  
}

Rectangle.java

public class *Rectangle* implements *Shape* {  
 *@Override* public void draw() {  
 *System*.out.println("Rectangle::draw()");  
 }  
}

Square.java

public class *Square* implements *Shape* {  
 *@Override* public void draw() {  
 *System*.out.println("Square::draw()");  
 }  
}

Circle.java

public class *Circle* implements *Shape* {  
 *@Override* public void draw() {  
 *System*.out.println("Circle::draw()");  
 }  
}

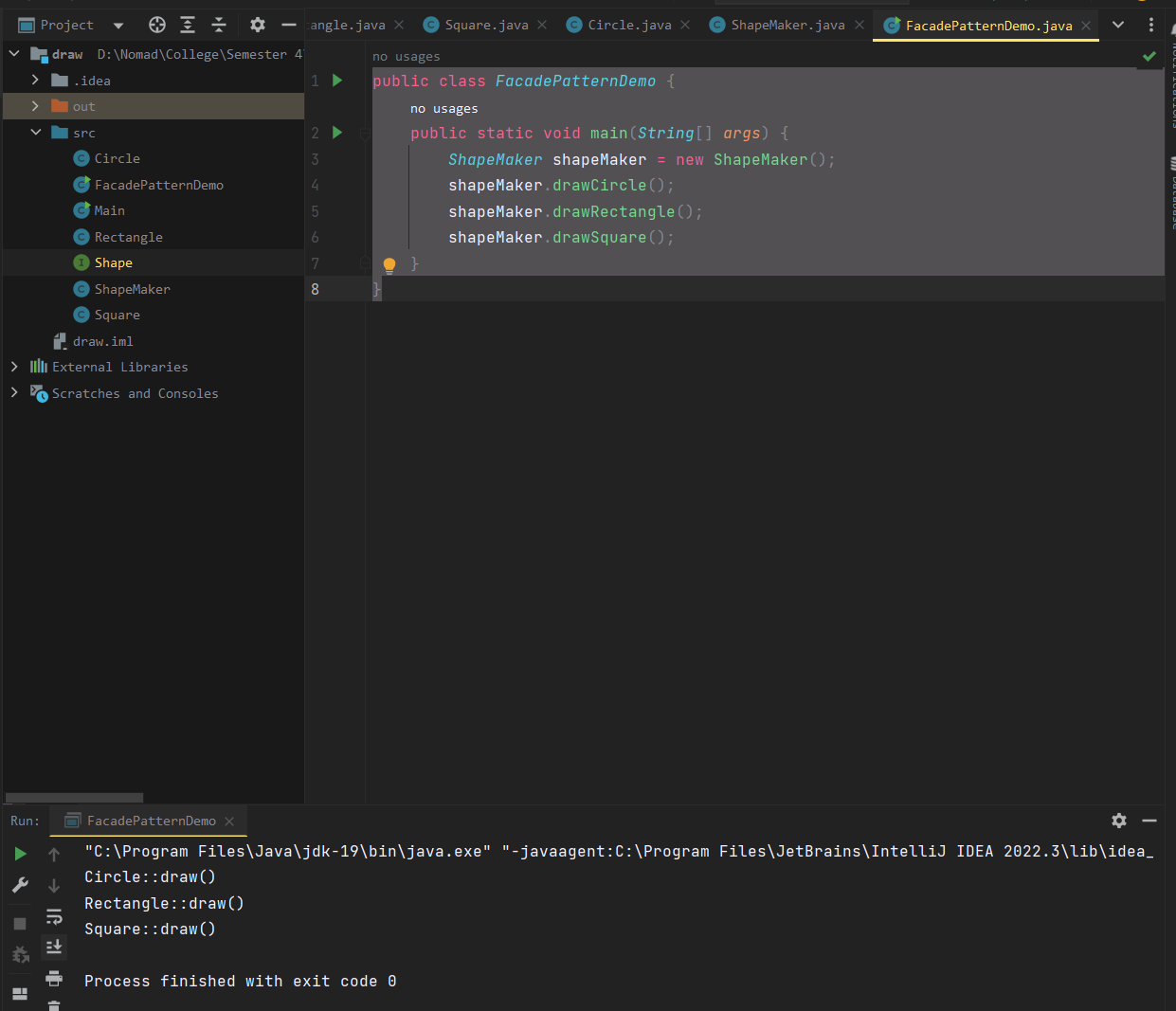
ShapeMaker.java

public class *ShapeMaker* {  
 private *Shape* circle;  
 private *Shape* rectangle;  
 private *Shape* square;  
  
 public ShapeMaker() {  
 circle = new Circle();  
 rectangle = new Rectangle();  
 square = new Square();  
 }  
  
 public void drawCircle(){  
 circle.draw();  
 }  
 public void drawRectangle(){  
 rectangle.draw();  
 }  
 public void drawSquare(){  
 square.draw();  
 }  
}

FacadeDemoPattern.java

public class *FacadePatternDemo* {  
 public static void main(*String*[] *args*) {  
 *ShapeMaker* shapeMaker = new ShapeMaker();  
 shapeMaker.drawCircle();  
 shapeMaker.drawRectangle();  
 shapeMaker.drawSquare();  
 }  
}

Output



### Example: 2 Server-based System

Subsystem1.java

class *Subsystem1* {  
 public void operation1() {  
 *System*.out.println("Subsystem1 operational");  
 }  
}

Subsystem2.java

class *Subsystem2* {  
 public void operation2() {  
 *System*.out.println("Subsystem2 operational");  
 }  
}

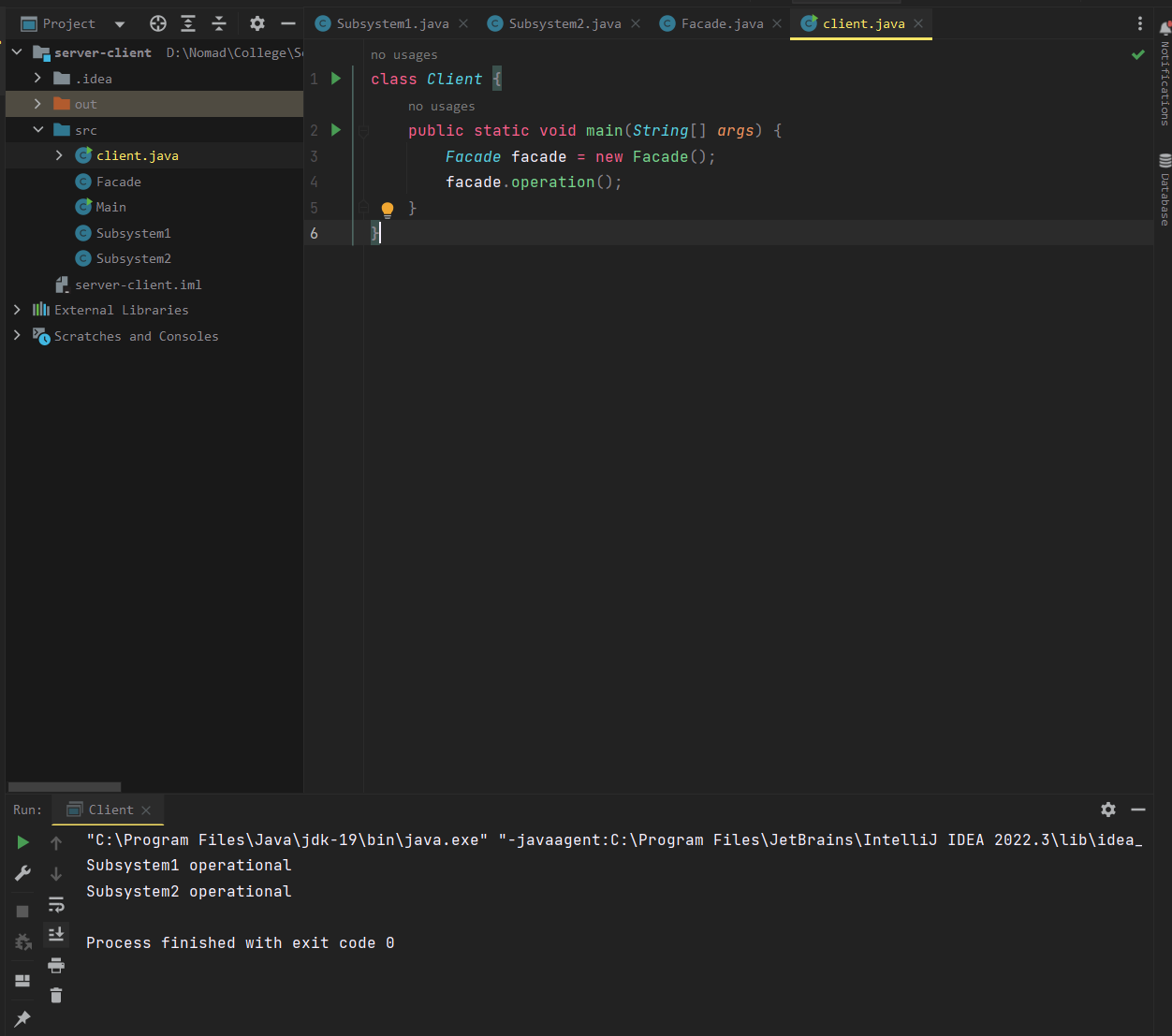
Façade.java

class *Facade* {  
 private *Subsystem1* subsystem1;  
 private *Subsystem2* subsystem2;  
  
 public Facade() {  
 subsystem1 = new Subsystem1();  
 subsystem2 = new Subsystem2();  
 }  
  
 public void operation() {  
 subsystem1.operation1();  
 subsystem2.operation2();  
 }  
}

Client.java

class *Client* {  
 public static void main(*String*[] *args*) {  
 *Facade* facade = new Facade();  
 facade.operation();  
 }  
}

Output



## Decorator Structural Design Pattern

### Example: 1 Draw

Shape.java

public interface *Shape* {  
 void draw();  
}

Rectangle.java

public class *Rectangle* implements *Shape*{  
 *@Override* public void draw(){  
 *System*.out.println("Shape : rectangle");  
 }  
}

Circle.java

public class *Circle* implements *Shape* {  
 *@Override* public void draw() {  
 *System*.out.println("Shape: Circle");  
 }  
}

ShapeDecorator.java

public abstract class ShapeDecorator implements *Shape* {  
 protected *Shape* decoratedShape;  
 public ShapeDecorator(*Shape decoratedShape*){  
 this.decoratedShape = *decoratedShape*;  
 }  
 public void draw(){  
 decoratedShape.draw();  
 }  
}

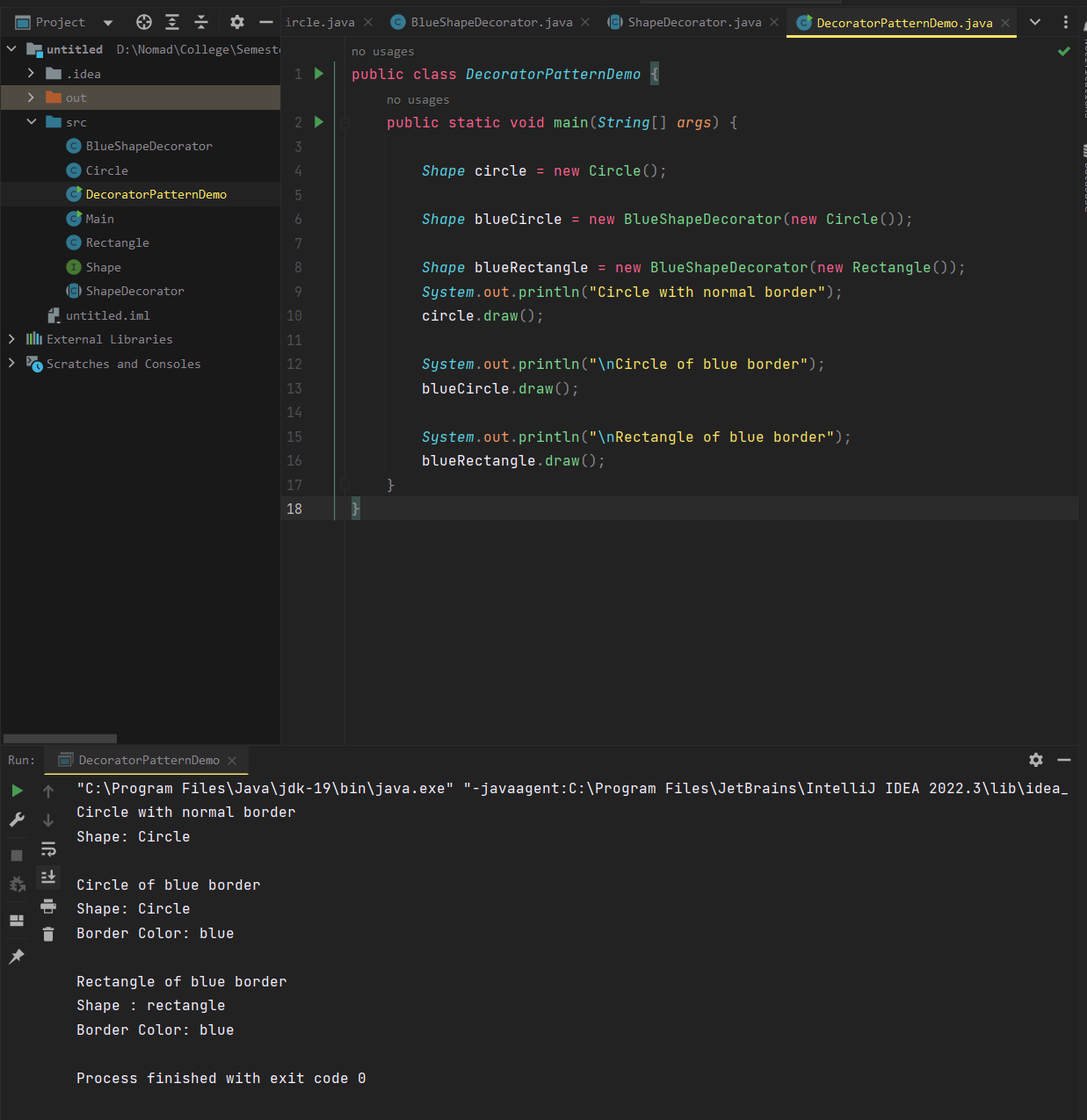
BlueShapeDecorator.java

public class *BlueShapeDecorator* extends ShapeDecorator {  
  
 public BlueShapeDecorator(*Shape decoratedShape*) {  
 super(*decoratedShape*);  
 }  
  
 *@Override* public void draw() {  
 decoratedShape.draw();  
 setBlueBorder(decoratedShape);  
 }  
  
 private void setBlueBorder(*Shape decoratedShape*){  
 *System*.out.println("Border Color: blue");  
 }  
}

DecoratorPatternDemo.java

public class *DecoratorPatternDemo* {  
 public static void main(*String*[] *args*) {  
  
 *Shape* circle = new Circle();  
  
 *Shape* blueCircle = new BlueShapeDecorator(new Circle());  
  
 *Shape* blueRectangle = new BlueShapeDecorator(new Rectangle());  
 *System*.out.println("Circle with normal border");  
 circle.draw();  
  
 *System*.out.println("\nCircle of blue border");  
 blueCircle.draw();  
  
 *System*.out.println("\nRectangle of blue border");  
 blueRectangle.draw();  
 }  
}

Output



### Example: 2 car driving

Car.java

public interface *Car* {  
 void drive();  
}

BasicCar.java

public class *BasicCar* implements *Car*{  
 public void drive(){  
 *System*.out.println("driving car");  
 }  
}

CarDecorator.java

public class *CarDecorator* implements *Car*{  
 protected *Car* car;  
 public CarDecorator(*Car car*){  
 this.car = *car*;  
 }  
 public void drive(){  
 car.drive();  
 }  
}

Bugatti.java

public class *bugatti* extends *CarDecorator*{  
 public bugatti(*Car car*){  
 super(*car*);  
 }  
 public void drive(){  
 car.drive();  
 *System*.out.println("driving bugatti");  
 }  
}

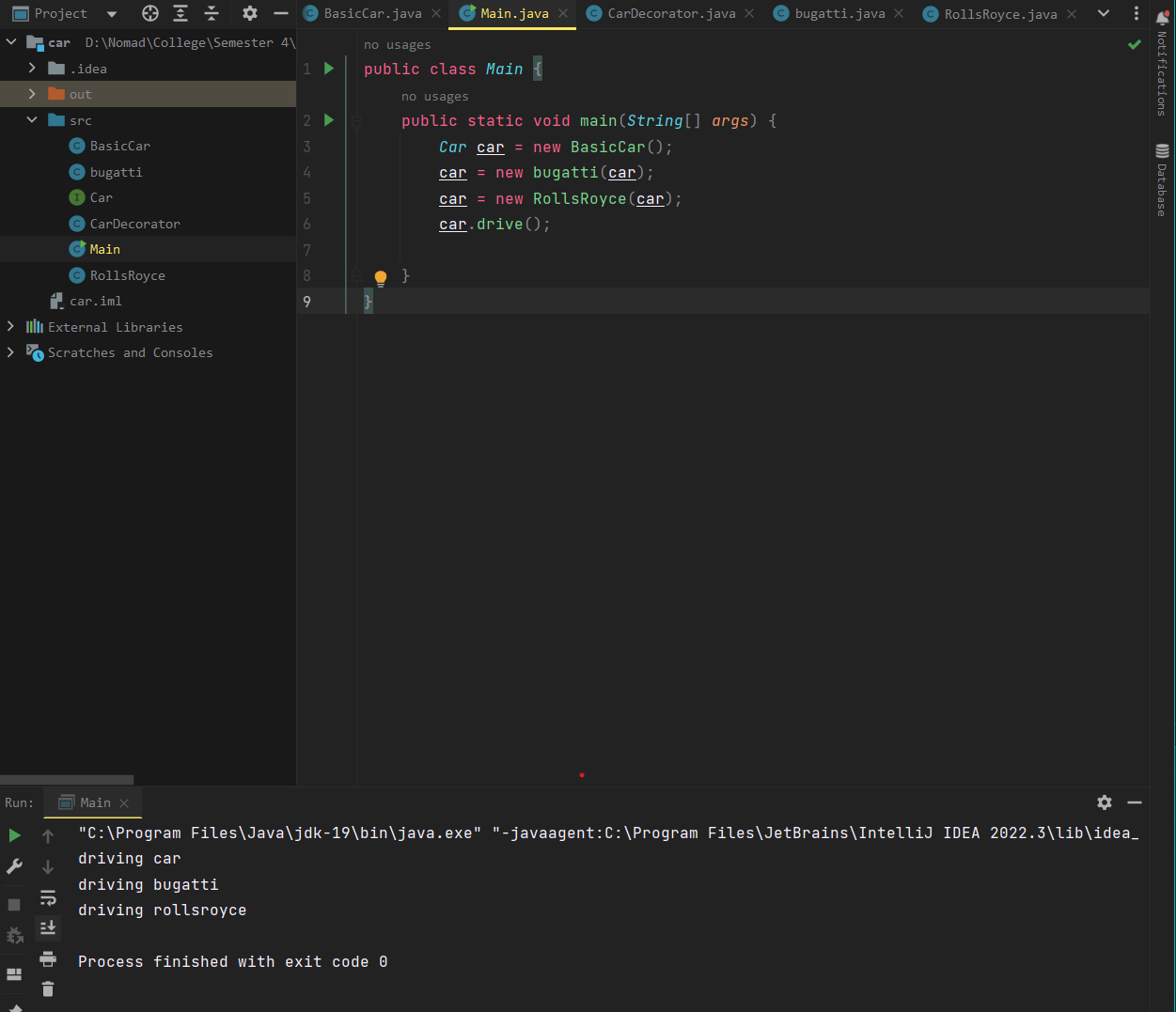
RollsRoyce.java

public class *RollsRoyce* extends *CarDecorator*{  
 public RollsRoyce(*Car car*){  
 super(*car*);  
 }  
 public void drive(){  
 car.drive();  
 *System*.out.println("driving rollsroyce");  
 }  
}

Main.java

public class *Main* {  
 public static void main(*String*[] *args*) {  
 *Car* car = new BasicCar();  
 car = new bugatti(car);  
 car = new RollsRoyce(car);  
 car.drive();  
  
 }  
}

Output



## Flyweight Structural Design Pattern

### Example: 1 Draw

Shape.java

public interface *Shape* {  
 void draw();  
}

Circle.java

public class *Circle* implements *Shape* {  
 private *String* color;  
 private int x;  
 private int y;  
 private int radius;  
  
 public Circle(*String color*){  
 this.color = *color*;  
 }  
  
 public void setX(int *x*) {  
 this.x = *x*;  
 }  
  
 public void setY(int *y*) {  
 this.y = *y*;  
 }  
  
 public void setRadius(int *radius*) {  
 this.radius = *radius*;  
 }  
  
 *@Override* public void draw() {  
 *System*.out.println("Circle: Draw() [Color : " + color + ", x : " + x + ", y :" + y + ", radius :" + radius);  
 }  
}

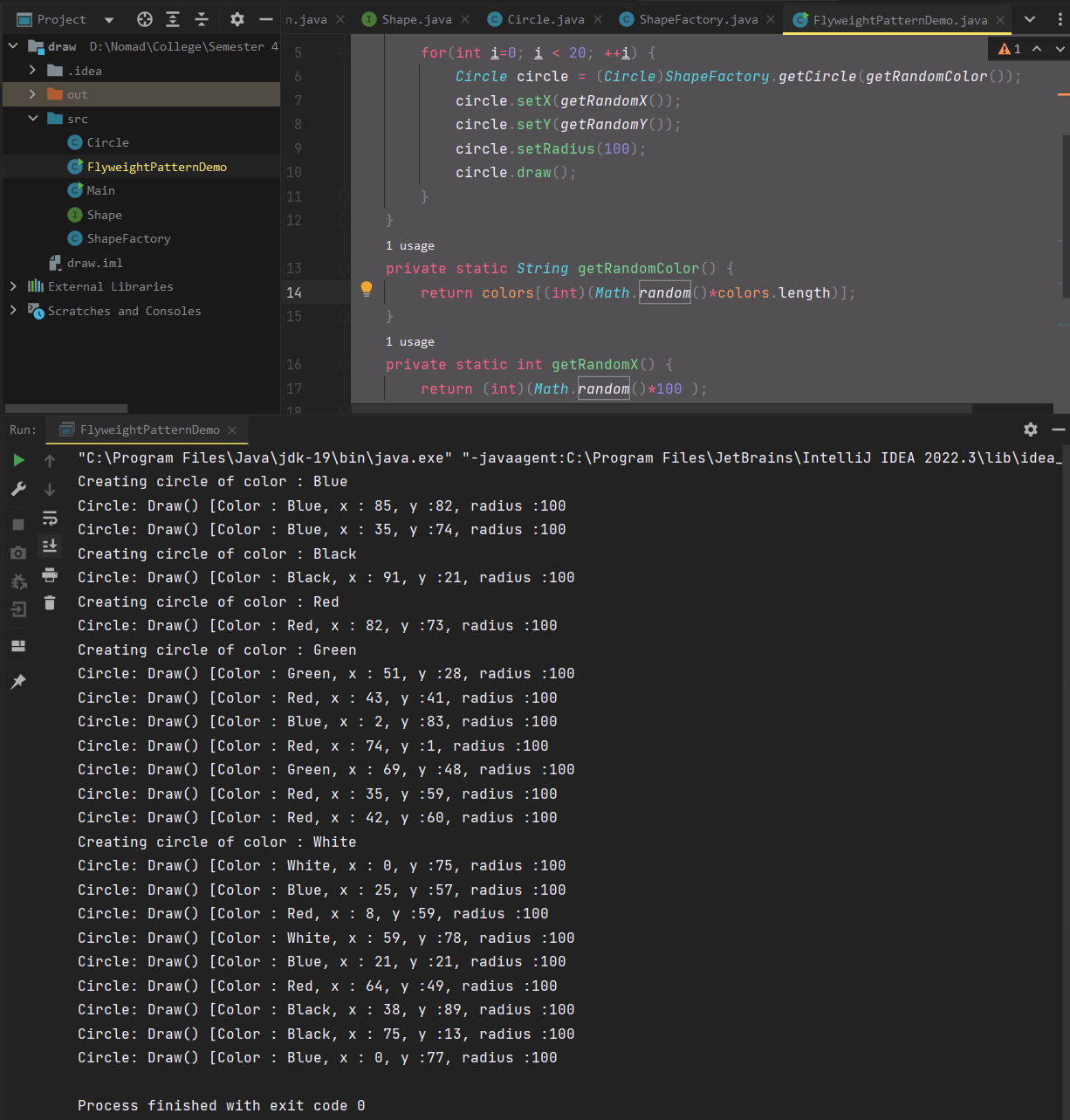
ShapeFactory.java

import *java.util.HashMap*;  
  
public class *ShapeFactory* {  
  
 private static final *HashMap* circleMap = new HashMap();  
  
 public static *Shape* getCircle(*String color*) {  
 *Circle* circle = (*Circle*)circleMap.get(*color*);  
  
 if(circle == null) {  
 circle = new Circle(*color*);  
 circleMap.put(*color*, circle);  
 *System*.out.println("Creating circle of color : " + *color*);  
 }  
 return circle;  
 }  
}

FlyweightPatternDemo

public class *FlyweightPatternDemo* {  
 private static final *String* colors[] = { "Red", "Green", "Blue", "White", "Black" };  
 public static void main(*String*[] *args*) {  
  
 for(int i=0; i < 20; ++i) {  
 *Circle* circle = (*Circle*)*ShapeFactory*.*getCircle*(*getRandomColor*());  
 circle.setX(*getRandomX*());  
 circle.setY(*getRandomY*());  
 circle.setRadius(100);  
 circle.draw();  
 }  
 }  
 private static *String* getRandomColor() {  
 return colors[(int)(*Math*.*random*()\*colors.length)];  
 }  
 private static int getRandomX() {  
 return (int)(*Math*.*random*()\*100 );  
 }  
 private static int getRandomY() {  
 return (int)(*Math*.*random*()\*100);  
 }  
}

Output



## Example: 2 learning to code this

Flyweight.java

public interface *Flyweight* {  
 public void operation();  
}

FlyweightFactory.java

import *java.util.HashMap*;  
  
public class *FlyweightFactory* {  
  
 private static final *HashMap*<*String*, *Flyweight*> flyweights = new HashMap<*String*, *Flyweight*>();  
  
 public static *Flyweight* getFlyweight(*String key*) {  
 *Flyweight* flyweight = flyweights.get(*key*);  
  
 if(flyweight == null) {  
 flyweight = new ConcreteFlyweight();  
 flyweights.put(*key*, flyweight);  
 }  
  
 return flyweight;  
 }  
}

ConcreteFlyweight.java

public class *ConcreteFlyweight* implements *Flyweight* {  
  
 public void operation() {  
 *System*.out.println("ConcreteFlyweight operation");  
 }  
}

Client.java

public class *Client* {  
  
 public static void main(*String*[] *args*) {  
 *Flyweight* flyweight = *FlyweightFactory*.*getFlyweight*("key");  
 flyweight.operation();  
 }  
}

Output

