Design Pattern Lab Manual

Name: Jaivik Jariwala

Roll No.: 21BCP004

Division: 1

Group: 1

# Behavioral Design Pattern

|  |  |
| --- | --- |
| Sr. No | Name |
| 1 | Iterator |
| 2 | Observer |
| 3 | Mediator |
| 4 | State |
| 5 | Memento |

## Iterator Behavioral Design Pattern

Example: 1 Menu repo

Container.java

public interface *Container* {  
 public *Iterator* getIterator();  
}

Iterator.java

public interface *Iterator* {  
 public boolean hasNext();  
 public *Object* next();  
}

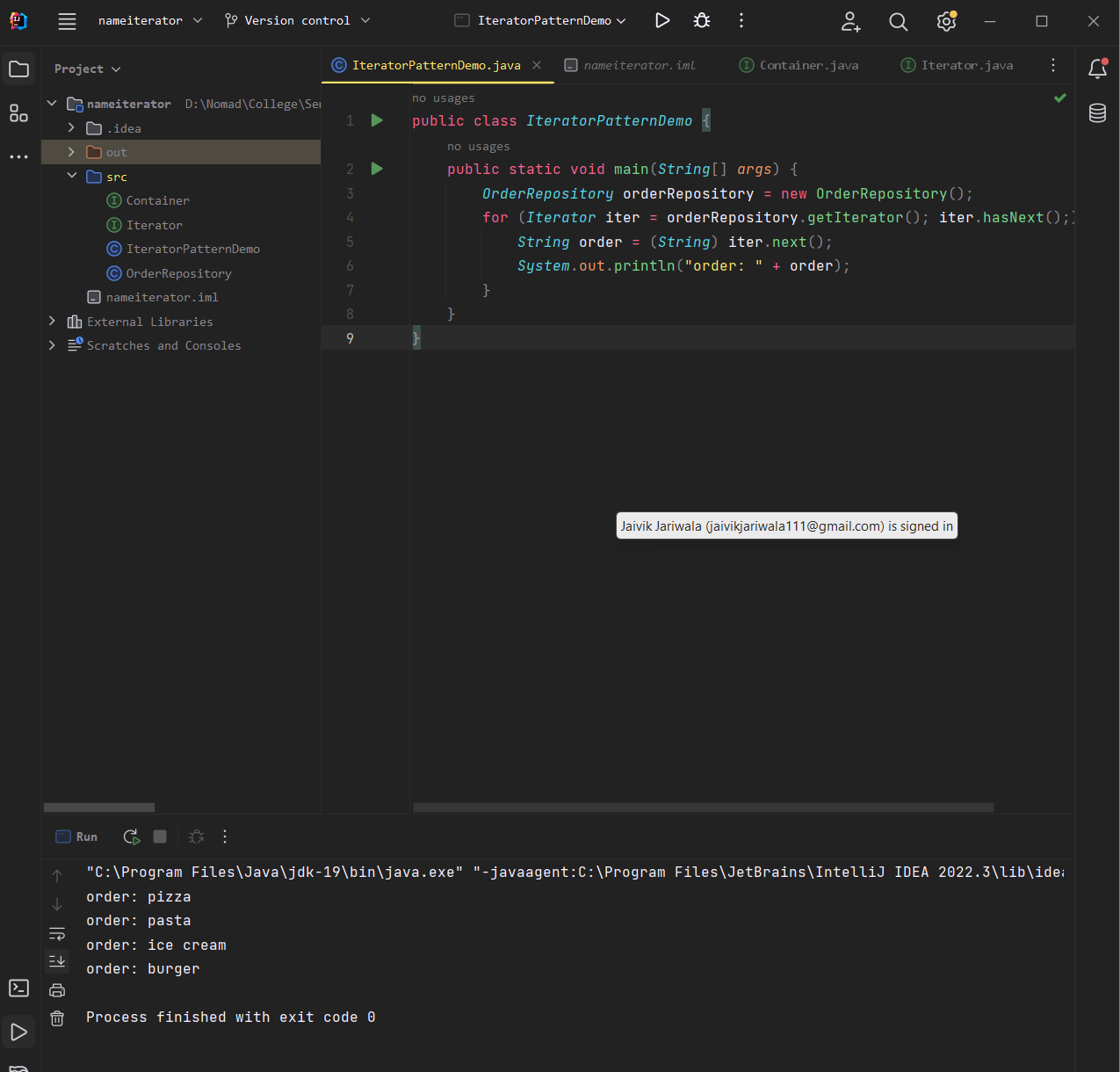
OrderRepository.java

public class *OrderRepository* implements *Container* {  
 public *String* orders[] = {"pizza", "pasta", "ice cream", "burger"};  
  
 public *Iterator* getIterator() {  
 return new OrderIterator();  
 }  
  
 private class *OrderIterator* implements *Iterator* {  
 int index;  
  
 public boolean hasNext() {  
 return index < orders.length;  
 }  
  
 public *Object* next() {  
 if (this.hasNext()) {  
 return orders[index++];  
 }  
 return null;  
 }  
 }  
}

IteratorPatternDemo.java

public class *IteratorPatternDemo* {  
 public static void main(*String*[] *args*) {  
 *OrderRepository* orderRepository = new OrderRepository();  
 for (*Iterator* iter = orderRepository.getIterator(); iter.hasNext();) {  
 *String* order = (*String*) iter.next();  
 *System*.out.println("order: " + order);  
 }  
 }  
}

Output:



Example:2 Shape

Shape.java

public class *Shape* {  
 private int id;  
 private String name;  
  
 public Shape(int *id* , String *name*){  
 this.id = *id*;  
 this.name = *name*;  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int *id*) {  
 this.id = *id*;;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String *name*) {  
 this.name = *name*;  
 }  
  
 @Override  
 public String toString() {  
 return " ID number : " +id+ " Shape is : " +name;  
 }  
}

ShapeIterator.java

import java.util.Iterator;  
  
public class *ShapeIterator* implements Iterator<*Shape*> {  
  
 private *Shape*[] shapes;  
 int pos;  
  
 public ShapeIterator(*Shape* []*shapes*){  
 this.shapes = *shapes*;  
 }  
  
 @Override  
 public boolean hasNext(){  
 if(pos >= shapes.length || shapes[pos] == null)  
 return false;  
 return true;  
 }  
  
 @Override  
 public *Shape* next(){  
 return shapes[pos++];  
 }  
  
 @Override  
 public void remove(){  
 if(pos <= 0)  
 throw new IllegalStateException("wrong place buddy");  
 if(shapes[pos-1] != null){  
 for(int i=pos-1; i <(shapes.length-1);i++){  
 shapes[i] = shapes[i+1];  
 }  
 shapes[shapes.length-1] = null;  
 }  
 }  
}

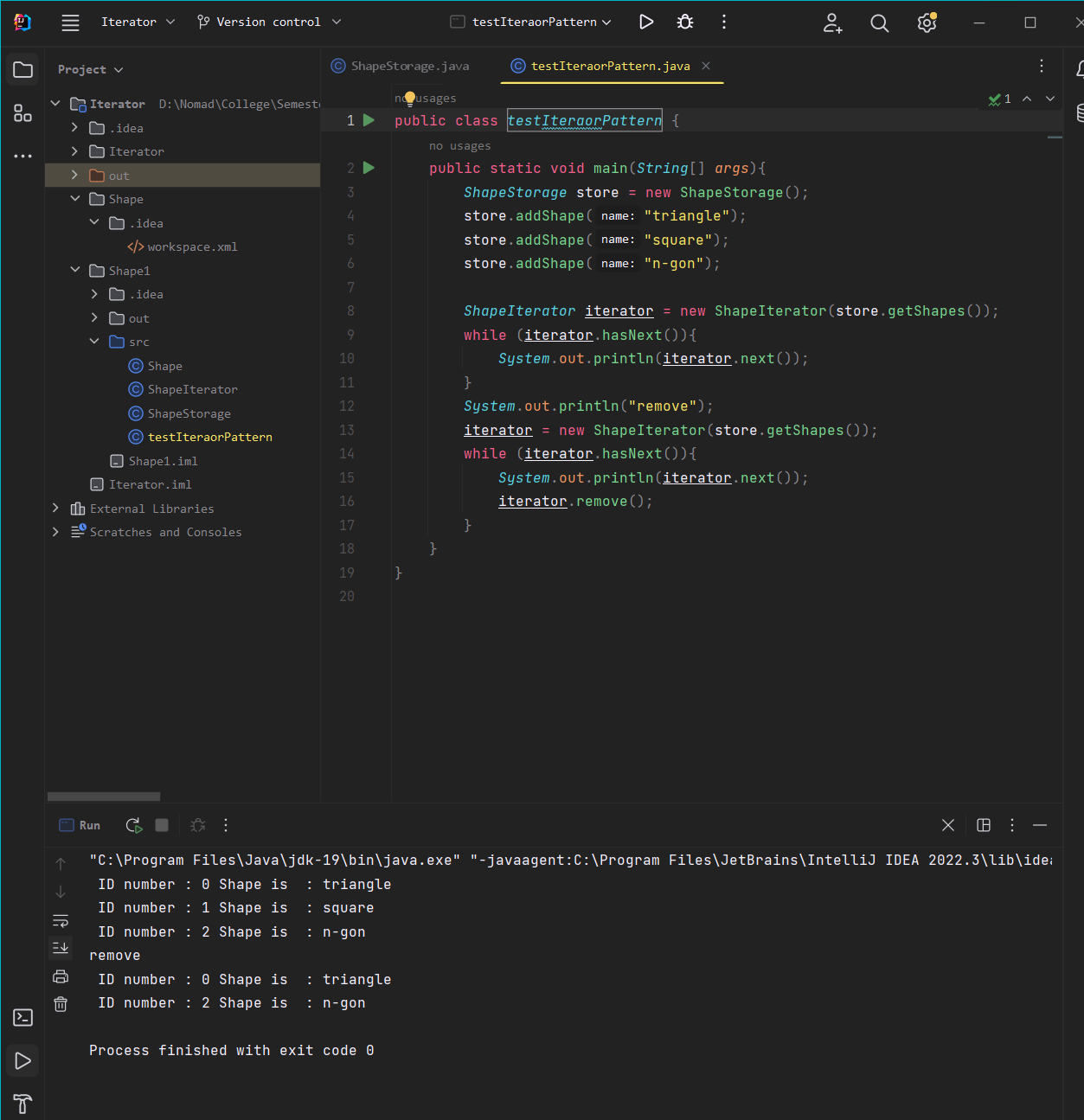
ShapeStorage.java

public class *ShapeStorage* {  
  
 private *Shape*[]shapes = new Shape[3];  
 private int index;  
  
 public void addShape(String *name*){  
 int i = index++;  
 shapes[i]= new Shape(i ,*name*);  
 }  
  
 public *Shape*[] getShapes(){  
 return shapes;  
 }  
}

testIteratorPattern.java

public class *testIteraorPattern* {  
 public static void main(String[] *args*){  
 *ShapeStorage* store = new ShapeStorage();  
 store.addShape("triangle");  
 store.addShape("square");  
 store.addShape("n-gon");  
  
 *ShapeIterator* iterator = new ShapeIterator(store.getShapes());  
 while (iterator.hasNext()){  
 System.out.println(iterator.next());  
 }  
 System.out.println("remove");  
 iterator = new ShapeIterator(store.getShapes());  
 while (iterator.hasNext()){  
 System.out.println(iterator.next());  
 iterator.remove();  
 }  
 }  
}

Output



## Memento Behavioral Design Pattern

Example: 1

CareTaker.java

import java.util.ArrayList;  
import java.util.List;  
public class *CareTaker* {  
 private List<*Memento*> mementoList = new ArrayList<*Memento*>();  
  
 public void add(*Memento state*){  
 mementoList.add(*state*);  
 }  
  
 public *Memento* get(int *index*){  
 return mementoList.get(*index*);  
 }  
}

Memento.java

public class *Memento* {  
 private *String* state;  
 public Memento(*String state*){  
 this.state = *state*;  
 }  
  
 public *String* getState(){  
 return state;  
 }  
}

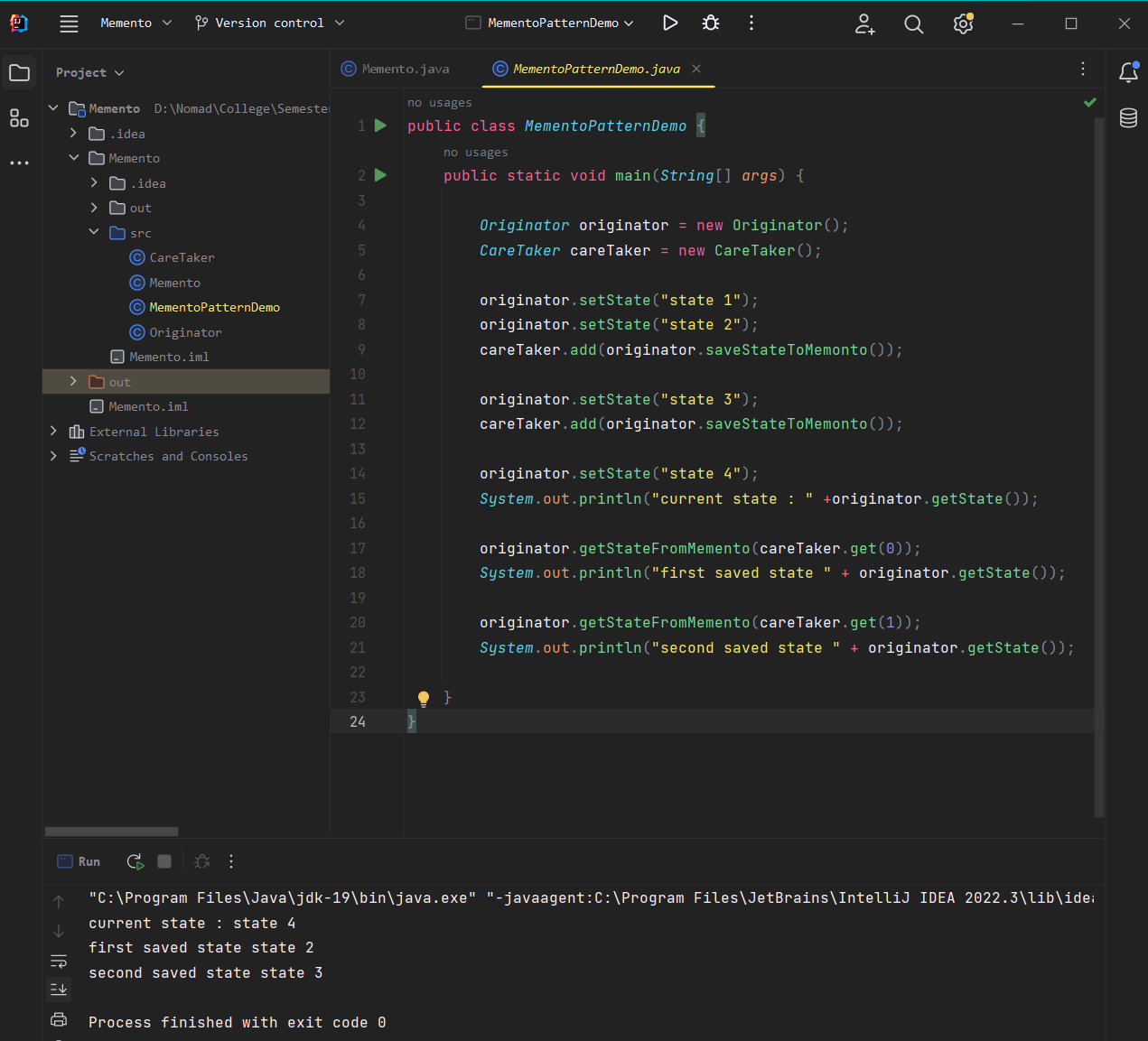
Originator.java

public class *Originator* {  
 private *String* state;  
 public void setState(*String state*){  
 this.state = *state*;  
 }  
  
 public *String* getState(){  
 return state;  
 }  
  
 public *Memento* saveStateToMemonto(){  
 return new Memento(state);  
 }  
  
 public void getStateFromMemento(*Memento memento*){  
 state = *memento*.getState();  
 }  
}

MementoPatternDemo.java

public class *MementoPatternDemo* {  
 public static void main(*String*[] *args*) {  
  
 *Originator* originator = new Originator();  
 *CareTaker* careTaker = new CareTaker();  
  
 originator.setState("state 1");  
 originator.setState("state 2");  
 careTaker.add(originator.saveStateToMemonto());  
  
 originator.setState("state 3");  
 careTaker.add(originator.saveStateToMemonto());  
  
 originator.setState("state 4");  
 *System*.out.println("current state : " +originator.getState());  
  
 originator.getStateFromMemento(careTaker.get(0));  
 *System*.out.println("first saved state " + originator.getState());  
  
 originator.getStateFromMemento(careTaker.get(1));  
 *System*.out.println("second saved state " + originator.getState());  
  
 }  
}

Output



Example:2

Document.java

public class *Document* {  
 private *String* content;  
  
 public void setContent(*String content*) {  
 this.content = *content*;  
 *System*.out.println("Document content updated to: " + *content*);  
 }  
  
 public *Memento* createMemento() {  
 *System*.out.println("Creating memento...");  
 return new Memento(content);  
 }  
  
 public void restoreFromMemento(*Memento memento*) {  
 content = *memento*.getContent();  
 *System*.out.println("Restoring document content from memento: " + content);  
 }  
  
 public *String* getContent() {  
 return content;  
 }  
}

Memento.java

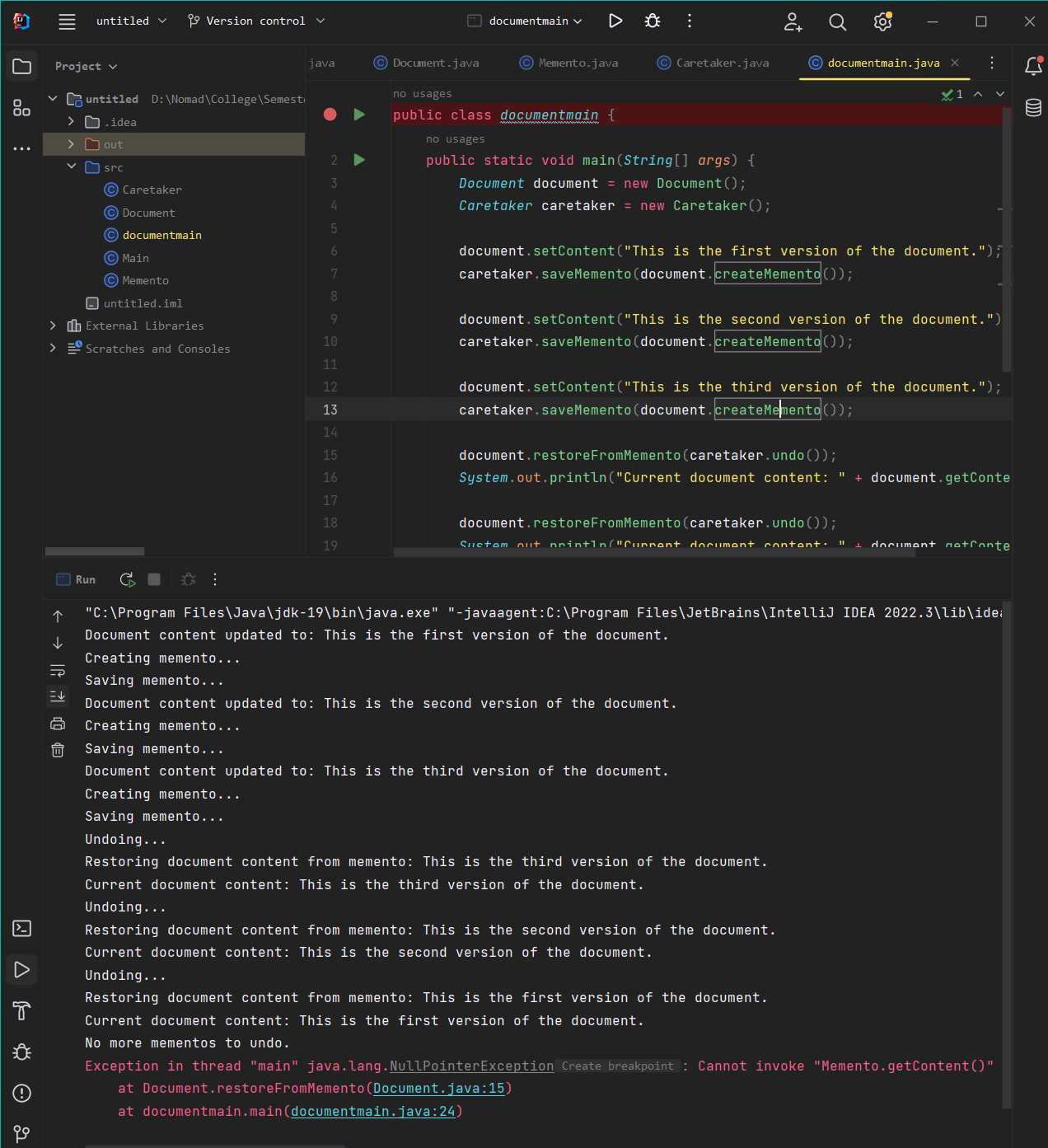
public class *Memento* {  
 private *String* content;  
  
 public Memento(*String content*) {  
 this.content = *content*;  
 }  
  
 public *String* getContent() {  
 return content;  
 }  
}

Caretaker.java

import *java.util.Stack*;  
  
public class *Caretaker* {  
 private *Stack*<*Memento*> mementos = new Stack<>();  
  
 public void saveMemento(*Memento memento*) {  
 *System*.out.println("Saving memento...");  
 mementos.push(*memento*);  
 }  
  
 public *Memento* undo() {  
 if (!mementos.isEmpty()) {  
 *System*.out.println("Undoing...");  
 return mementos.pop();  
 } else {  
 *System*.out.println("No more mementos to undo.");  
 return null;  
 }  
 }  
}

Documentmain.java

public class *documentmain* {  
 public static void main(*String*[] *args*) {  
 *Document* document = new Document();  
 *Caretaker* caretaker = new Caretaker();  
  
 document.setContent("This is the first version of the document.");  
 caretaker.saveMemento(document.createMemento());  
  
 document.setContent("This is the second version of the document.");  
 caretaker.saveMemento(document.createMemento());  
  
 document.setContent("This is the third version of the document.");  
 caretaker.saveMemento(document.createMemento());  
  
 document.restoreFromMemento(caretaker.undo());  
 *System*.out.println("Current document content: " + document.getContent());  
  
 document.restoreFromMemento(caretaker.undo());  
 *System*.out.println("Current document content: " + document.getContent());  
  
 document.restoreFromMemento(caretaker.undo());  
 *System*.out.println("Current document content: " + document.getContent());  
  
 document.restoreFromMemento(caretaker.undo());  
 *System*.out.println("Current document content: " + document.getContent());  
 }  
}



## Observer Behavioral Design Pattern

Example: 1

BinaryObserver.java

public class *BinaryObserver* extends Observer{  
 public BinaryObserver(*Subject subject*){  
 this.subject = *subject*;  
 this.subject.attach(this);  
 }  
  
 *@Override* public void update(){  
 *System*.out.println("Binary String" + *Integer*.*toBinaryString*(subject.getState()));  
 }  
}

HexaObserver.java

public class *HexaObserver* extends Observer{  
 public HexaObserver(*Subject subject*){  
 this.subject = *subject*;  
 this.subject.attach(this);  
 }  
  
 *@Override* public void update(){  
 *System*.out.println("hexa string" + *Integer*.*toHexString*(subject.getState()).toUpperCase());  
 }  
}

OctalObserver.java

public class *OctalObserver* extends Observer{  
 public OctalObserver(*Subject subject*){  
 this.subject = *subject*;  
 this.subject.attach(this);  
 }  
  
 *@Override* public void update(){  
 *System*.out.println("Octal String" + *Integer*.*toOctalString*(subject.getState()));  
 }  
}

Subject.java

import *java.util.ArrayList*;  
import *java.util.List*;  
public class *Subject* {  
 private *List*<Observer> observers = new ArrayList<Observer>();  
 private int state;  
  
 public int getState(){  
 return state;  
 }  
  
 public void setState(int *state*){  
 this.state = *state*;  
 notifyAllObservers();  
 }  
  
 public void attach(Observer *observer*){  
 observers.add(*observer*);  
 }  
  
 public void notifyAllObservers(){  
 for(Observer observer : observers){  
 observer.update();  
 }  
 }  
}

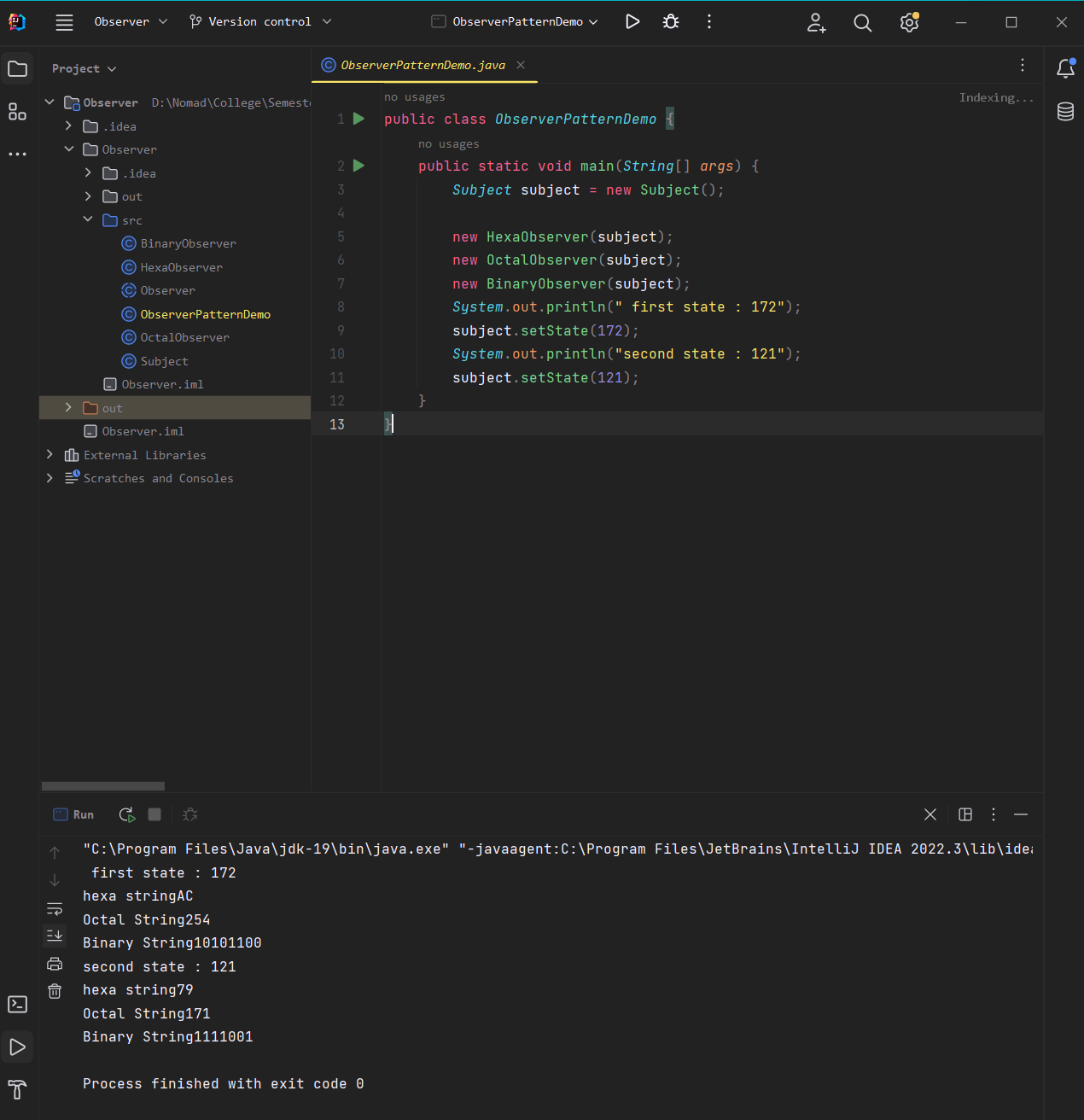
Observer.java

public abstract class Observer {  
 protected *Subject* subject;  
 public abstract void update();  
}

ObserverPatternDemo.java

public class *ObserverPatternDemo* {  
 public static void main(*String*[] *args*) {  
 *Subject* subject = new Subject();  
  
 new HexaObserver(subject);  
 new OctalObserver(subject);  
 new BinaryObserver(subject);  
 *System*.out.println(" first state : 172");  
 subject.setState(172);  
 *System*.out.println("second state : 121");  
 subject.setState(121);  
 }  
}

Output



Example:2

Observer.java

public interface *Observer* {  
 void update();  
 void gotorestaurant(*restaurant res*);  
}

Subject.java

public interface *Subject* {  
 void order(*customer o1*);  
 void cancelorder(*Observer o1*);  
 void notifycustomer();  
 void upload(*String orderlist*);  
}

Customer.java

public class *customer* implements *Observer* {  
 private *String* name;  
 private *restaurant* res=new restaurant();  
 public customer(*String name*) {  
 this.name=*name*;  
 }  
 public void update() {  
 *System*.out.println("hey"+name+"you are at restaurant:"+res.orderlist);  
 }  
 public void gotorestaurant(*restaurant rest*)  
 {  
 res=*rest*;  
 }  
}

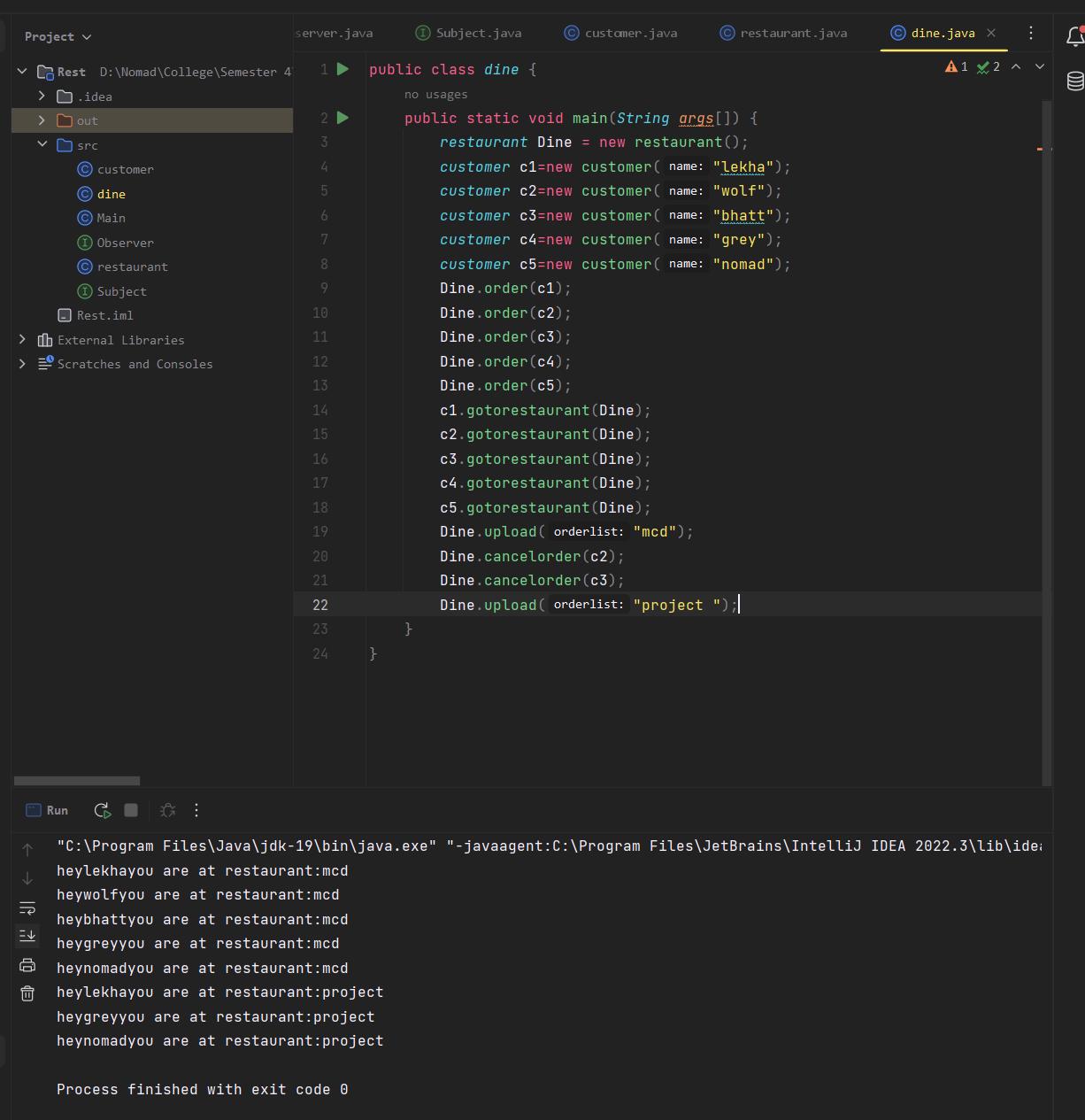
Restaurant.java

import *java.util.ArrayList*;  
import *java.util.List*;  
public class *restaurant* implements *Subject* {  
 private *List*<*customer*> cus=new ArrayList<*customer*>();  
 public *String* orderlist;  
 public void order(*customer o1*) {  
 cus.add(*o1*);  
 }  
 public void cancelorder(*Observer o1*) {  
 cus.remove(*o1*);  
 }  
 public void notifycustomer()  
 {  
 for(*Observer* o1:cus)  
 {  
 o1.update();  
 }  
 }  
 public void upload(*String orderlist*)  
 {  
 this.orderlist=*orderlist*;  
 notifycustomer();  
 }  
}

dine.java

public class *dine* {  
 public static void main(*String args*[]) {  
 *restaurant* Dine = new restaurant();  
 *customer* c1=new customer("lekha");  
 *customer* c2=new customer("wolf");  
 *customer* c3=new customer("bhatt");  
 *customer* c4=new customer("grey");  
 *customer* c5=new customer("nomad");  
 Dine.order(c1);  
 Dine.order(c2);  
 Dine.order(c3);  
 Dine.order(c4);  
 Dine.order(c5);  
 c1.gotorestaurant(Dine);  
 c2.gotorestaurant(Dine);  
 c3.gotorestaurant(Dine);  
 c4.gotorestaurant(Dine);  
 c5.gotorestaurant(Dine);  
 Dine.upload("mcd");  
 Dine.cancelorder(c2);  
 Dine.cancelorder(c3);  
 Dine.upload("project ");  
 }  
}

Output



## Mediator Behavioral Design Pattern

Example: 1

Order.java

public class *Order* {  
 private *String* order;  
 private *OrderMessageMediator* mediator;  
  
 public Order(*OrderMessageMediator mediator*, *String order*) {  
 this.mediator = *mediator*;  
 this.order = *order*;  
 }  
  
 public void sendMessage(*String message*) {  
 mediator.sendMessage(this, *message*);  
 }  
  
 public void receiveMessage(*String message*) {  
 *System*.out.println("Order " + order + " received message: " + *message*);  
 }  
}

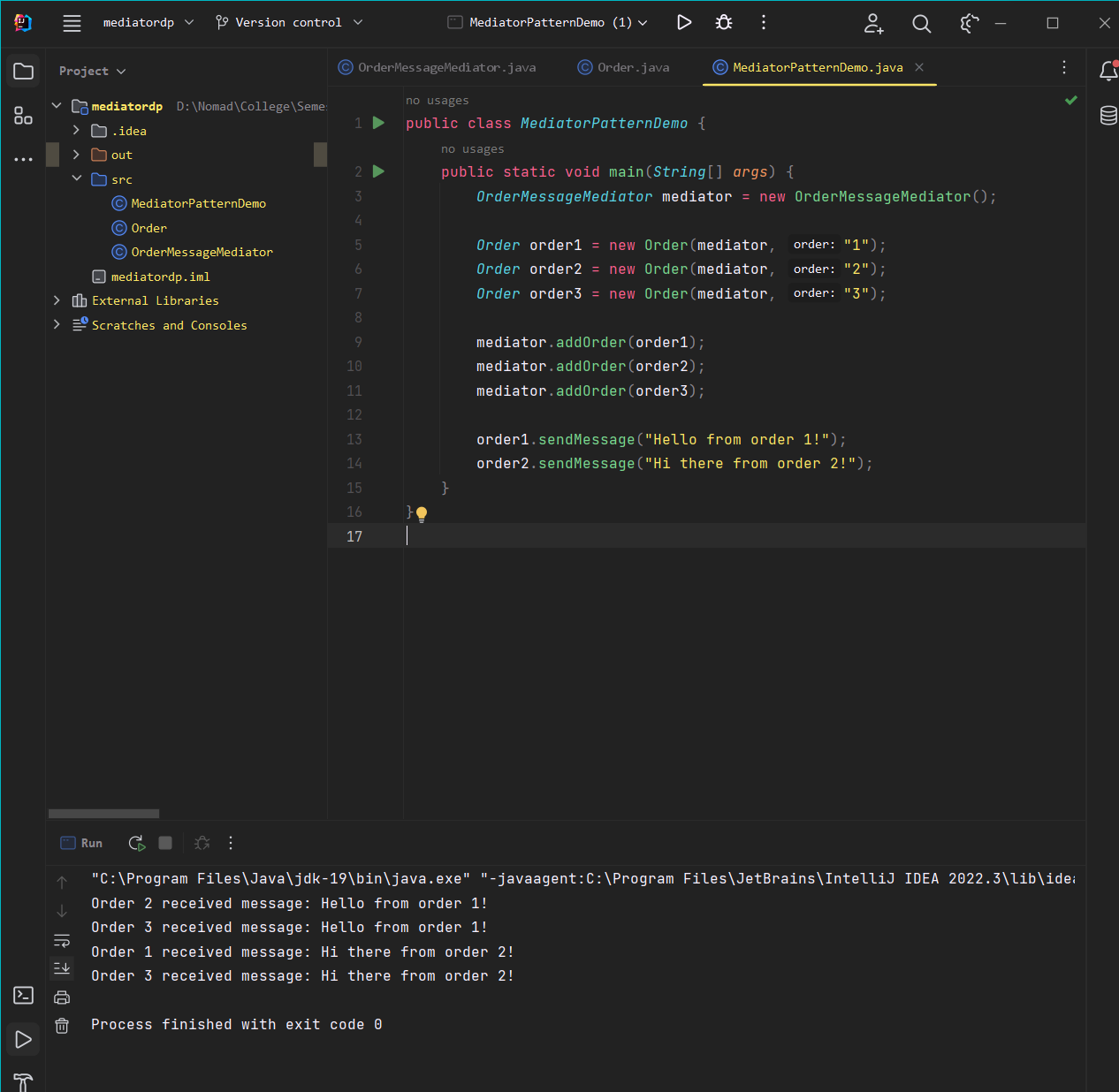
OrderMessageMediator.java

import *java.util.ArrayList*;  
import *java.util.List*;  
  
public class *OrderMessageMediator* {  
 private *List*<*Order*> orders = new ArrayList<>();  
  
 public void addOrder(*Order order*) {  
 this.orders.add(*order*);  
 }  
  
 public void sendMessage(*Order order*, *String message*) {  
 for (*Order* o : orders) {  
 if (o != *order*) {  
 o.receiveMessage(*message*);  
 }  
 }  
 }  
}

MediatorPatternDemo.java

public class *MediatorPatternDemo* {  
 public static void main(*String*[] *args*) {  
 *OrderMessageMediator* mediator = new OrderMessageMediator();  
  
 *Order* order1 = new Order(mediator, "1");  
 *Order* order2 = new Order(mediator, "2");  
 *Order* order3 = new Order(mediator, "3");  
  
 mediator.addOrder(order1);  
 mediator.addOrder(order2);  
 mediator.addOrder(order3);  
  
 order1.sendMessage("Hello from order 1!");  
 order2.sendMessage("Hi there from order 2!");  
 }  
}

Output



Example: 2

Chatroom.java

public interface *Chatroom* {  
 void sendMessage(*User user*, *String message*);  
 void addUser(*User user*);  
}

User.java

public class *User* {  
 private *String* name;  
 private *Chatroom* chatroom;  
  
 public User(*String name*, *Chatroom chatroom*) {  
 this.name = *name*;  
 this.chatroom = *chatroom*;  
 this.chatroom.addUser(this);  
 }  
  
 public void sendMessage(*String message*) {  
 this.chatroom.sendMessage(this, *message*);  
 }  
  
 public void receiveMessage(*String message*) {  
 *System*.out.println(this.name + " received message: " + *message*);  
 }  
  
 public *String* getName() {  
 return name;  
 }  
}

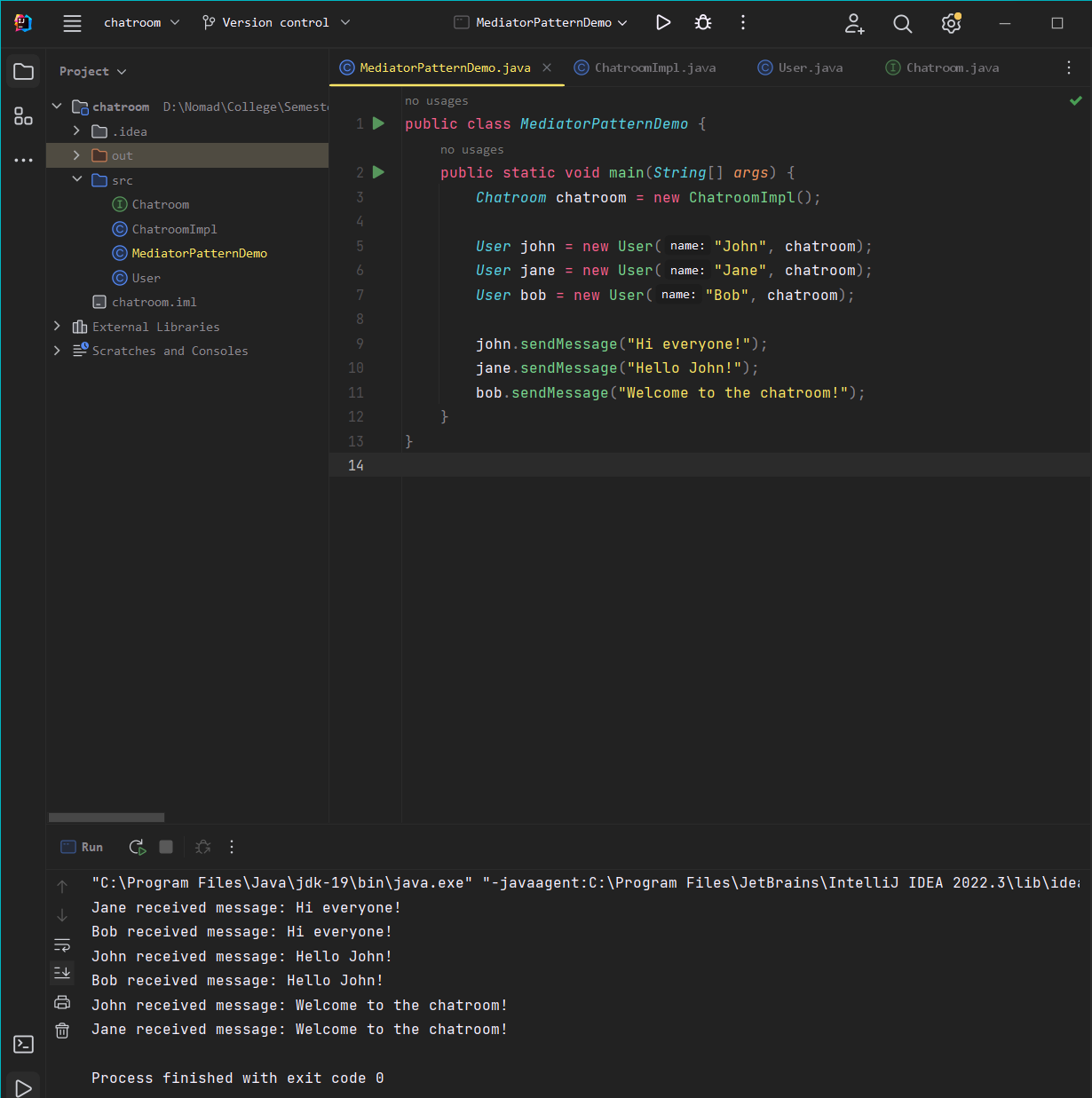
ChatroomImpl.java

import *java.util.ArrayList*;  
import *java.util.List*;  
  
public class *ChatroomImpl* implements *Chatroom* {  
 private *List*<*User*> users;  
  
 public ChatroomImpl() {  
 this.users = new ArrayList<>();  
 }  
  
 *@Override* public void sendMessage(*User user*, *String message*) {  
 for (*User* u : this.users) {  
 if (!u.getName().equals(*user*.getName())) {  
 u.receiveMessage(*message*);  
 }  
 }  
 }  
  
 *@Override* public void addUser(*User user*) {  
 this.users.add(*user*);  
 }  
}

MediatorPatternDemo.java

public class *MediatorPatternDemo* {  
 public static void main(*String*[] *args*) {  
 *Chatroom* chatroom = new ChatroomImpl();  
  
 *User* john = new User("John", chatroom);  
 *User* jane = new User("Jane", chatroom);  
 *User* bob = new User("Bob", chatroom);  
  
 john.sendMessage("Hi everyone!");  
 jane.sendMessage("Hello John!");  
 bob.sendMessage("Welcome to the chatroom!");  
 }  
}

Output



## State Behavioral Design Pattern

Example: 1

State.java

public interface *State* {  
 public void doAction(*Context context*);  
}

Context.java

public class *Context* {  
 private *State* state;  
  
  
 public Context(){  
 state = null;  
 }  
  
 public void setState(*State state*){  
 this.state = *state*;  
 }  
  
 public *State* getState(){  
 return state;  
 }  
}

StartState.java

public class *StartState* implements *State* {  
 public void doAction(*Context context*){  
 *System*.out.println("Game Begins");  
 *context*.setState(this);  
 }  
 public *String* toString(){  
 return "Start State";  
 }  
}

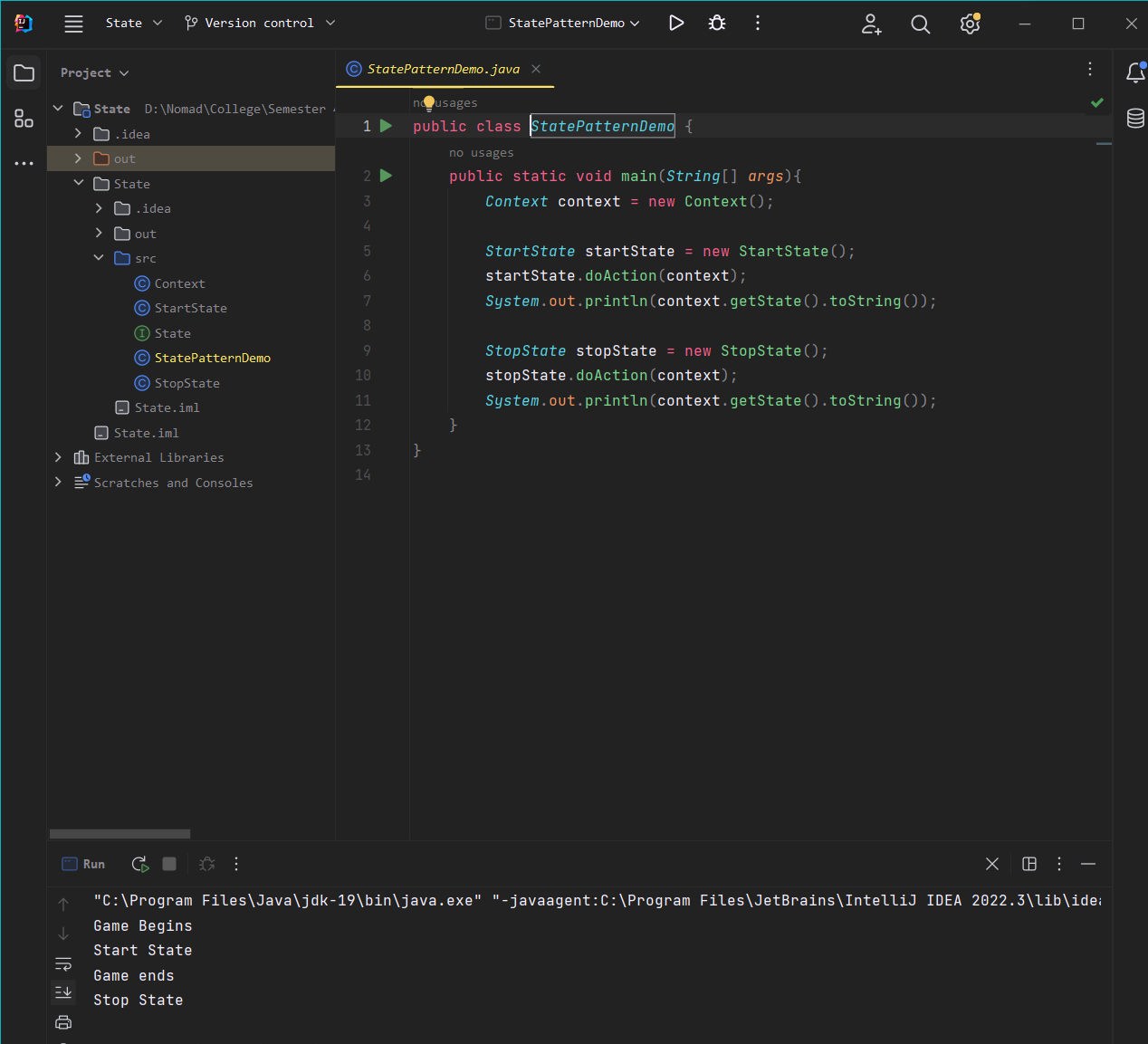
StopState.java

public class *StopState* implements *State* {  
 public void doAction(*Context context*){  
 *System*.out.println("Game ends");  
 *context*.setState(this);  
 }  
 public *String* toString(){  
 return "Stop State";  
 }  
  
}

StatePatternDemo

public class *StatePatternDemo* {  
 public static void main(*String*[] *args*){  
 *Context* context = new Context();  
  
 *StartState* startState = new StartState();  
 startState.doAction(context);  
 *System*.out.println(context.getState().toString());  
  
 *StopState* stopState = new StopState();  
 stopState.doAction(context);  
 *System*.out.println(context.getState().toString());  
 }  
}

Output



Example: 2

MobileAlertState.java

public interface *MobileAlertState* {  
 public void Alert();  
}

MobileContext.java

public class *mobileContext* {  
 private *MobileAlertState* currentState;  
 public mobileContext() {  
 currentState= new Ringing();  
 }  
 public void setState(*MobileAlertState state*) {  
 currentState=*state*;  
 }  
 public void Alert() {  
 currentState.Alert();  
 }  
}

Ringing.java

public class *Ringing* implements *MobileAlertState* {  
 public void Alert()  
 {  
 *System*.out.println("Mobile is Ringing");  
 }  
}

Silent.java

public class *Silent* implements *MobileAlertState* {  
 public void Alert() {  
 *System*.out.println("Mobile is silent");  
 }  
}

Mobile.java

public class *Mobile* {  
 public static void main(*String args*[]) {  
 *mobileContext* mc = new mobileContext();  
 mc.Alert();  
 mc.setState(new Silent());  
 mc.Alert();  
 *System*.out.println("---Set to Ringing Again--");  
 mc.setState(new Ringing());  
 mc.Alert();  
 }  
}

Output

