**DESIGN PATTERN PRACTICE (6428317 Mukesh P)**

**1.SingletonPatternExample**

**Logger.java**

**package** spexample;

**public** **class** Logger {

**private** **static** Logger *instance*;

**private** Logger() {

System.***out***.println("Logger initialized");

}

**public** **static** Logger getInstance() {

**if**(*instance*==**null**) {

*instance*=**new** Logger();

}

**return** *instance*;

}

**public** **void** log(String message) {

System.***out***.println("Log: "+message);

}

}

**TestLogger.java:**

**package** spexample;

**public** **class** TestLogger {

**public** **static** **void** main(String[]args) {

Logger log1=Logger.*getInstance*();

Logger log2=Logger.*getInstance*();

log1.log("HI! THIS IS LOG MESSAGE");

**if**(log1==log2) {

System.***out***.println("Already this instance used");

}

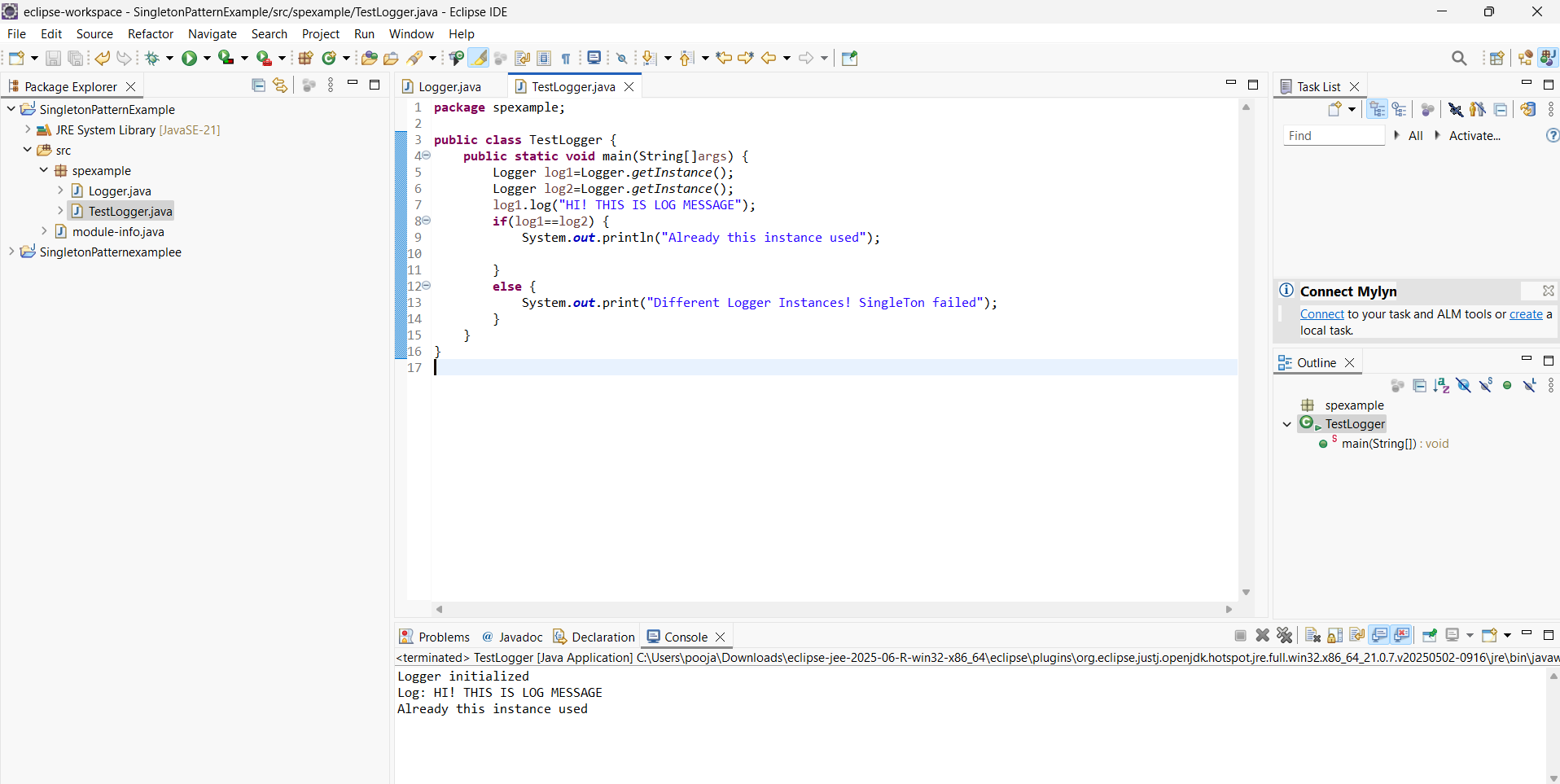
**else** {

System.***out***.print("Different Logger Instances! SingleTon failed");

}

}

}



Output 1

**2.FactoryMethodPattern Example**

**Docment.java:**

**package** documents;

**public** **interface** Document {

**void** open();

}

**Pdf.java:**

**package** documents;

**public** **class** pdf **implements** Document{

**public** **void** open() {

System.***out***.println("Opening Pdf");

}

}

**Word.java:**

**package** documents;

**public** **class** word **implements** Document{

**public** **void** open() {

System.***out***.println("Opening Word Document");

}

}

**Excel.java**

**package** documents;

**public** **class** excel **implements** Document{

**public** **void** open() {

System.***out***.println("Opening Excel Sheet");

}

}

**DocumentFactory.java**

**package** documents;

**public** **abstract** **class** DocumentFactory {

**public** **abstract** Document createDocument();

}

**PdfFactory.java**

**package** documents;

**public** **class** PdfFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** pdf();

}

}

**ExcelFactory.java:**

**package** documents;

**public** **class** ExcelFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** excel();

}

}

**wordFactory.java**

**package** documents;

**public** **class** WordFactory **extends** DocumentFactory {

**public** Document createDocument() {

**return** **new** word();

}

}

**TestFactory.java**

**package** documents;

**public** **class** TestFactory {

**public** **static** **void** main(String[]args) {

DocumentFactory pdfFactory=**new** PdfFactory();

Document pdf=pdfFactory.createDocument();

pdf.open();

DocumentFactory wordFactory=**new** WordFactory();

Document word=wordFactory.createDocument();

word.open();

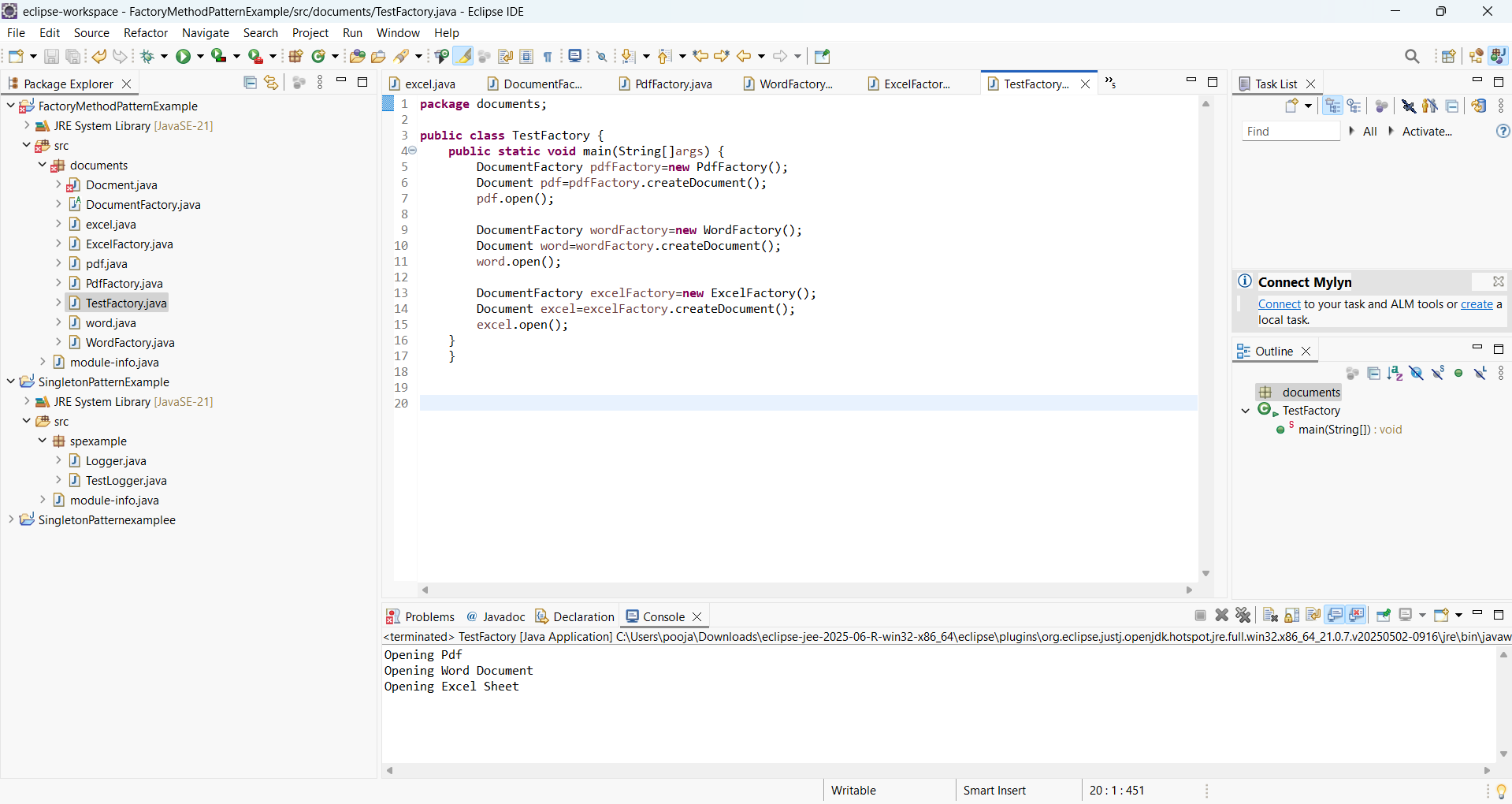
DocumentFactory excelFactory=**new** ExcelFactory();

Document excel=excelFactory.createDocument();

excel.open();

}

}



Output 2

**3.BuilderPatternExample**

**Computer.java**

**package** computer;

**public** **class** Computer {

**private** String CPU;

**private** String RAM;

**private** String storage;

**private** String graphicsCard;

**private** Computer(Builder builder) {

**this**.CPU = builder.CPU;

**this**.RAM = builder.RAM;

**this**.storage = builder.storage;

**this**.graphicsCard = builder.graphicsCard;

}

**public** **static** **class** Builder {

**private** String CPU;

**private** String RAM;

**private** String storage;

**private** String graphicsCard;

**public** Builder setCPU(String CPU) {

**this**.CPU=CPU;

**return** **this**;

}

**public** Builder setRAM(String RAM) {

**this**.RAM = RAM;

**return** **this**;

}

**public** Builder setStorage(String storage) {

**this**.storage = storage;

**return** **this**;

}

**public** Builder setGraphicsCard(String graphicsCard) {

**this**.graphicsCard = graphicsCard;

**return** **this**;

}

**public** Computer build() {

**return** **new** Computer(**this**);

}

}

@Override

**public** String toString() {

**return** "Computer [CPU=" +CPU+ ", RAM=" +RAM+ " Storage=" + storage+ ", GraphicsCard="+graphicsCard+"]";

}

}

**TestBuilder.java**

**package** computer;

**public** **class** TestBuilder {

**public** **static** **void** main(String[]args) {

Computer gamingPC = **new** Computer.Builder()

.setCPU("Intel i9")

.setRAM("32GB")

.setStorage("2TB SSD")

.setGraphicsCard("NVIDIA RTX 4090")

.build();

System.***out***.println("Gaming PC: " + gamingPC);

Computer officePC = **new** Computer.Builder()

.setCPU("Intel i5")

.setRAM("16GB")

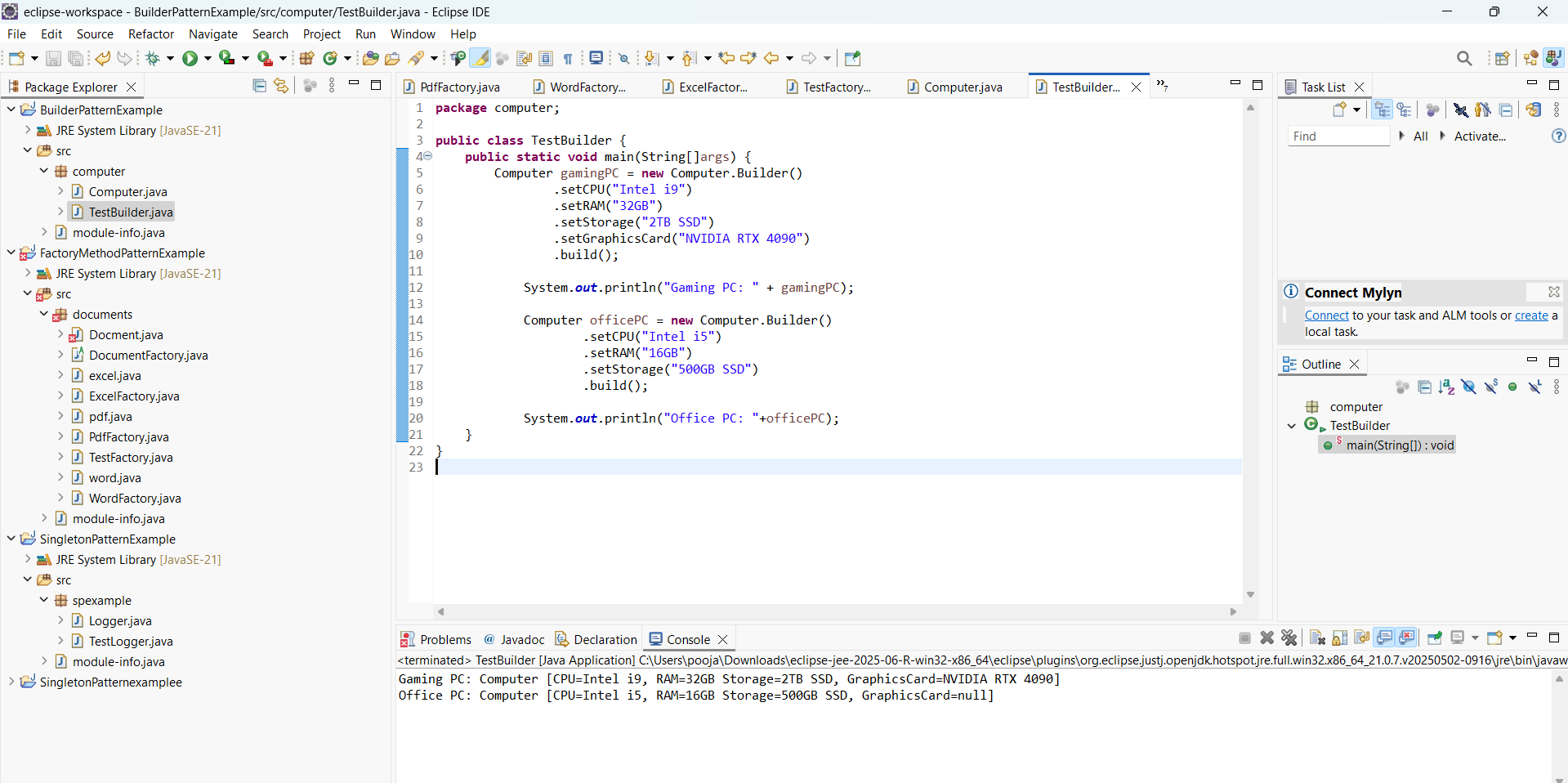
.setStorage("500GB SSD")

.build();

System.***out***.println("Office PC: "+officePC);

}

}



Output 3

**4.AdapterPattermExample**

**PaymentProcessor.java**

**package** payment;

**public** **interface** PaymentProcessor {

**void** processPayment(**double** amount);

}

**PaytmGateway.java:**

**package** payment;

**public** **class** PaytmGateway {

**public** **void** makePaytmPayment(**double** amount) {

System.***out***.println("Payment processed via Paytm: ₹" + amount);

}

}

**PaytmAdapter.java**  
  
**package** payment;

**public** **class** PaytmAdapter **implements** PaymentProcessor {

**private** PaytmGateway paytmGateway;

**public** PaytmAdapter() {

paytmGateway=**new** PaytmGateway();

}

@Override

**public** **void** processPayment(**double** amount) {

paytmGateway.makePaytmPayment(amount);

}

}

**TestAdapter.java**

**package** payment;

**public** **class** TestAdapter {

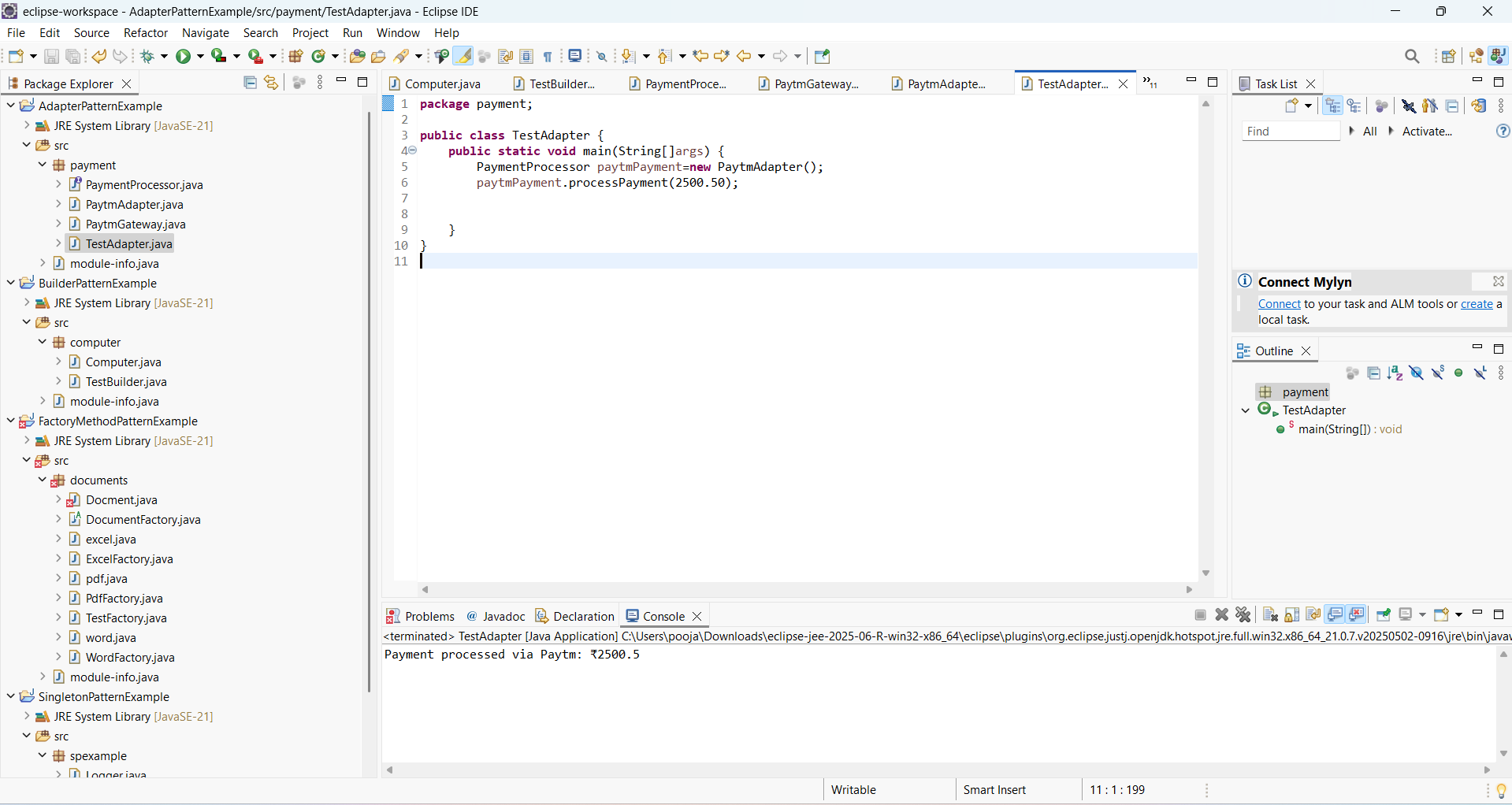
**public** **static** **void** main(String[]args) {

PaymentProcessor paytmPayment=**new** PaytmAdapter();

paytmPayment.processPayment(2500.50);

}

}



Output 4

**5.DecoratorPatternExample**

**Notifier.java:**

**package** notifier;

**public** **interface** Notifier {

**void** send(String message);

}

**NotifierDecorator.java:**

**package** notifier;

**public** **abstract** **class** NotifierDecorator **implements** Notifier {

**protected** Notifier notifier;

**public** NotifierDecorator(Notifier notifier) {

**this**.notifier = notifier;

}

@Override

**public** **void** send(String message) {

notifier.send(message);

}

}

**EmailNotifier.java:**

**package** notifier;

**public** **class** EmailNotifier **implements** Notifier{

@Override

**public** **void** send(String message) {

System.***out***.println("Email sent: " + message);

}

}

**SlackNotifier.java:**

**package** notifier;

**public** **class** SlackNotifierDecorator **extends** NotifierDecorator{

**public** SlackNotifierDecorator(Notifier notifier) {

**super**(notifier);

}

@Override

**public** **void** send(String message) {

**super**.send(message);

System.***out***.println("Slack message sent: " + message);

}

}

**SMSNotfier.java:**

**package** notifier;

**public** **class** SMSNotfierDecorator **extends** NotifierDecorator{

**public** SMSNotfierDecorator(Notifier notifier) {

**super**(notifier);

}

@Override

**public** **void** send(String message) {

**super**.send(message);

System.***out***.println("SMS SENT: "+message);

}

}

**TestDecorator.java:**

**package** notifier;

**public** **class** TestDecorator {

**public** **static** **void** main(String[] args) {

Notifier emailNotifier = **new** EmailNotifier();

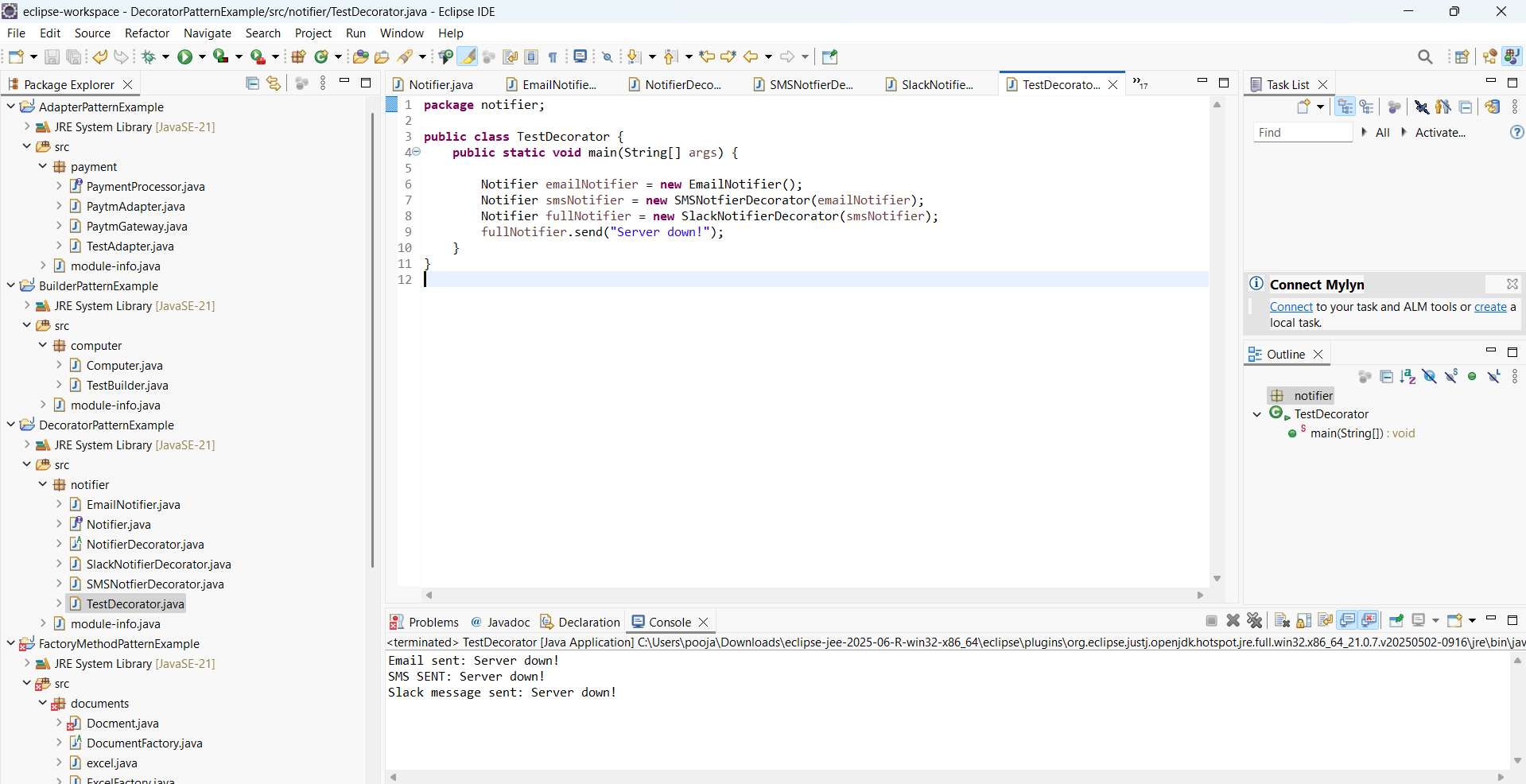
Notifier smsNotifier = **new** SMSNotfierDecorator(emailNotifier);

Notifier fullNotifier = **new** SlackNotifierDecorator(smsNotifier);

fullNotifier.send("Server down!");

}

}



Output 5

**6.PROXY PATTERN**

**Image.java**

**package** imageviewer;

**public** **interface** Image {

**void** display();

}

**ProxyImage.java**

**package** imageviewer;

**public** **class** ProxyImage **implements** Image {

**private** RealImage realImage;

**private** String filename;

**public** ProxyImage(String filename) {

**this**.filename = filename;

}

@Override

**public** **void** display() {

**if** (realImage == **null**) {

realImage = **new** RealImage(filename);

}

realImage.display();

}

}

**RealImage.java:**

**package** imageviewer;

**public** **class** RealImage **implements** Image{

**private** String filename;

**public** RealImage(String filename) {

**this**.filename=filename;

loadFromDisk();

}

**private** **void** loadFromDisk() {

System.***out***.println("Loading "+filename+" from disk..");

}

@Override

**public** **void** display() {

System.***out***.println("Displaying"+filename);

}

}

**TestImage.java**

**package** imageviewer;

**public** **class** TestProxy {

**public** **static** **void** main(String[]args) {

Image image1 = **new** ProxyImage("TRAINIM.jpg");

Image image2 = **new** ProxyImage("TRAINPIC.jpg");

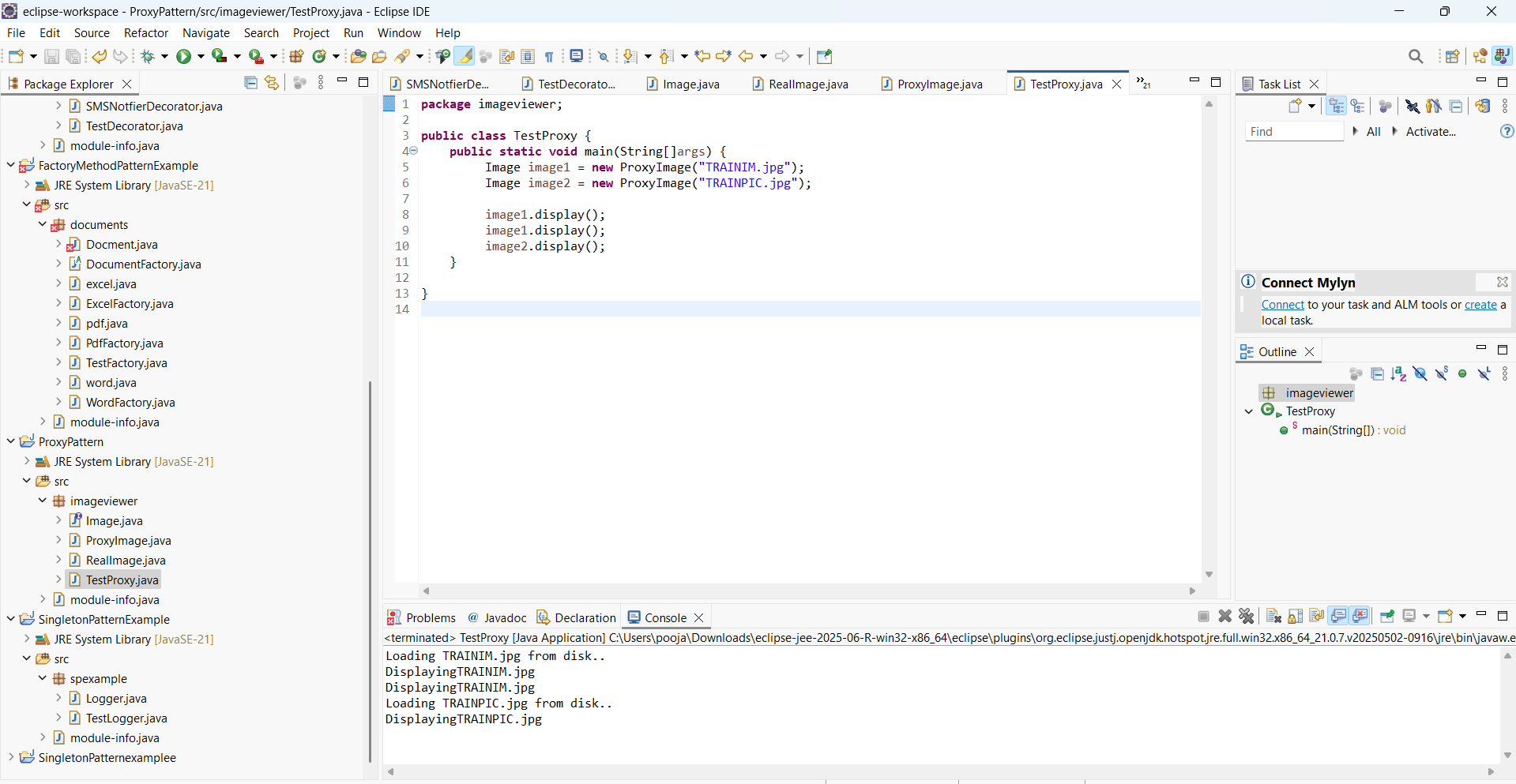
image1.display();

image1.display();

image2.display();

}

}



Output 6

**7.ObserverPattern**

**Observer.java**

**package** stockmarket;

**public** **interface** Observer {

**void** update(String stockName,**double** price);

}

**Stock.java**

**package** stockmarket;

**public** **interface** Stock {

**void** registerObserver(Observer observer);

**void** removeObserver(Observer observer);

**void** notifyObservers();

}

**Stockmarket.java**

**package** stockmarket;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** StockMarket **implements** Stock {

**private** List<Observer> observers = **new** ArrayList<>();

**private** String stockName;

**private** **double** price;

**public** **void** setStock(String stockName, **double** price) {

**this**.stockName = stockName;

**this**.price = price;

notifyObservers();

}

@Override

**public** **void** registerObserver(Observer observer) {

observers.add(observer);

}

@Override

**public** **void** removeObserver(Observer observer) {

observers.remove(observer);

}

@Override

**public** **void** notifyObservers() {

**for** (Observer o : observers) {

o.update(stockName, price);

}

}

}

**MobileApp.java**

**package** stockmarket;

**public** **class** MobileApp **implements** Observer {

**private** String user;

**public** MobileApp(String user) {

**this**.user = user;

}

@Override

**public** **void** update(String stockName, **double** price) {

System.***out***.println("MobileApp [" + user + "] - " + stockName + " updated to ₹" + price);

}

}

**WebApp.java**

**package** stockmarket;

**public** **class** WebApp **implements** Observer {

**private** String clientName;

**public** WebApp(String clientName) {

**this**.clientName = clientName;

}

@Override

**public** **void** update(String stockName, **double** price) {

System.***out***.println("WebApp [" + clientName + "] - " + stockName + " updated to ₹" + price);

}

}

**TestObserver.java**

**package** stockmarket;

**public** **class** TestObserver {

**public** **static** **void** main(String[] args) {

StockMarket stockMarket = **new** StockMarket();

Observer mobileUser1 = **new** MobileApp("Mukesh");

Observer mobileUser2 = **new** MobileApp("Demo");

Observer webUser1 = **new** WebApp("TradeX");

stockMarket.registerObserver(mobileUser1);

stockMarket.registerObserver(mobileUser2);

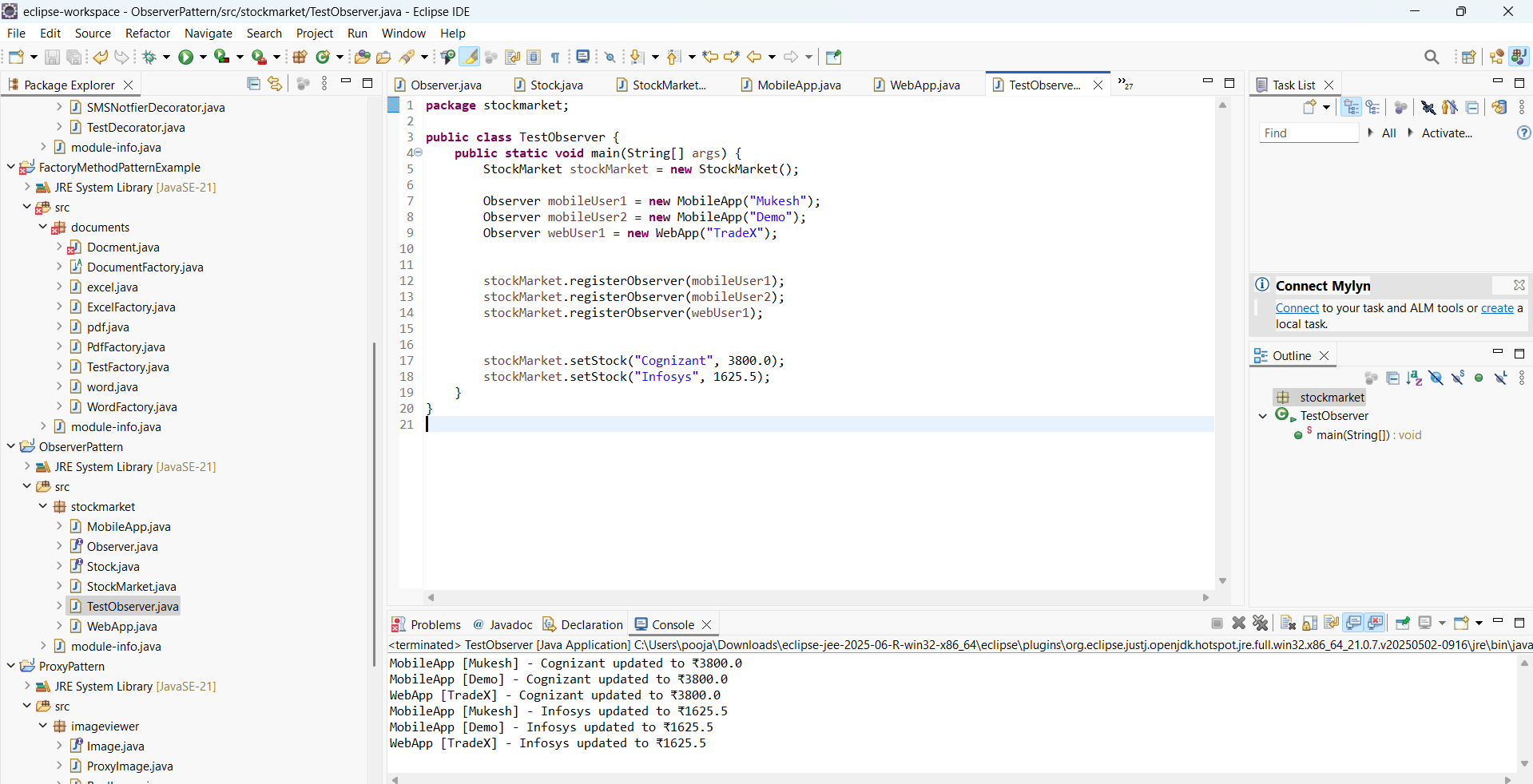
stockMarket.registerObserver(webUser1);

stockMarket.setStock("Cognizant", 3800.0);

stockMarket.setStock("Infosys", 1625.5);

}

}



Output 7

**8.StrategyPayment**

**PaymentStrategy.java:**

**package** payment;

**public** **interface** PaymentStrategy {

**void** pay(**double** amount);

}

**PaymentContext.java:**

**package** payment;

**public** **class** PaymentContext {

**private** PaymentStrategy strategy;

**public** **void** setPaymentStrategy(PaymentStrategy strategy) {

**this**.strategy = strategy;

}

**public** **void** payAmount(**double** amount) {

**if** (strategy == **null**) {

System.***out***.println("Payment strategy not set.");

} **else** {

strategy.pay(amount);

}

}

}

**CreditCardpayment.java:**

**package** payment;

**public** **class** CreditCardPayment **implements** PaymentStrategy{

**private** String cardNumber;

**public** CreditCardPayment(String cardNumber) {

**this**.cardNumber = cardNumber;

}

@Override

**public** **void** pay(**double** amount) {

System.***out***.println("Paid ₹" + amount + " using Credit Card [" + cardNumber + "]");

}

}

**PaytmPayment.java**

**package** payment;

**public** **class** PaytmPayment **implements** PaymentStrategy {

**private** String email;

**public** PaytmPayment(String email) {

**this**.email = email;

}

@Override

**public** **void** pay(**double** amount) {

System.***out***.println("Paid ₹" + amount + " using PayPal [" + email + "]");

}

}

**TestStrategy.java**

**package** payment;

**public** **class** TestStrategy {

**public** **static** **void** main(String[] args) {

PaymentContext context = **new** PaymentContext();

context.setPaymentStrategy(**new** CreditCardPayment("1034-0000-9476-7562"));

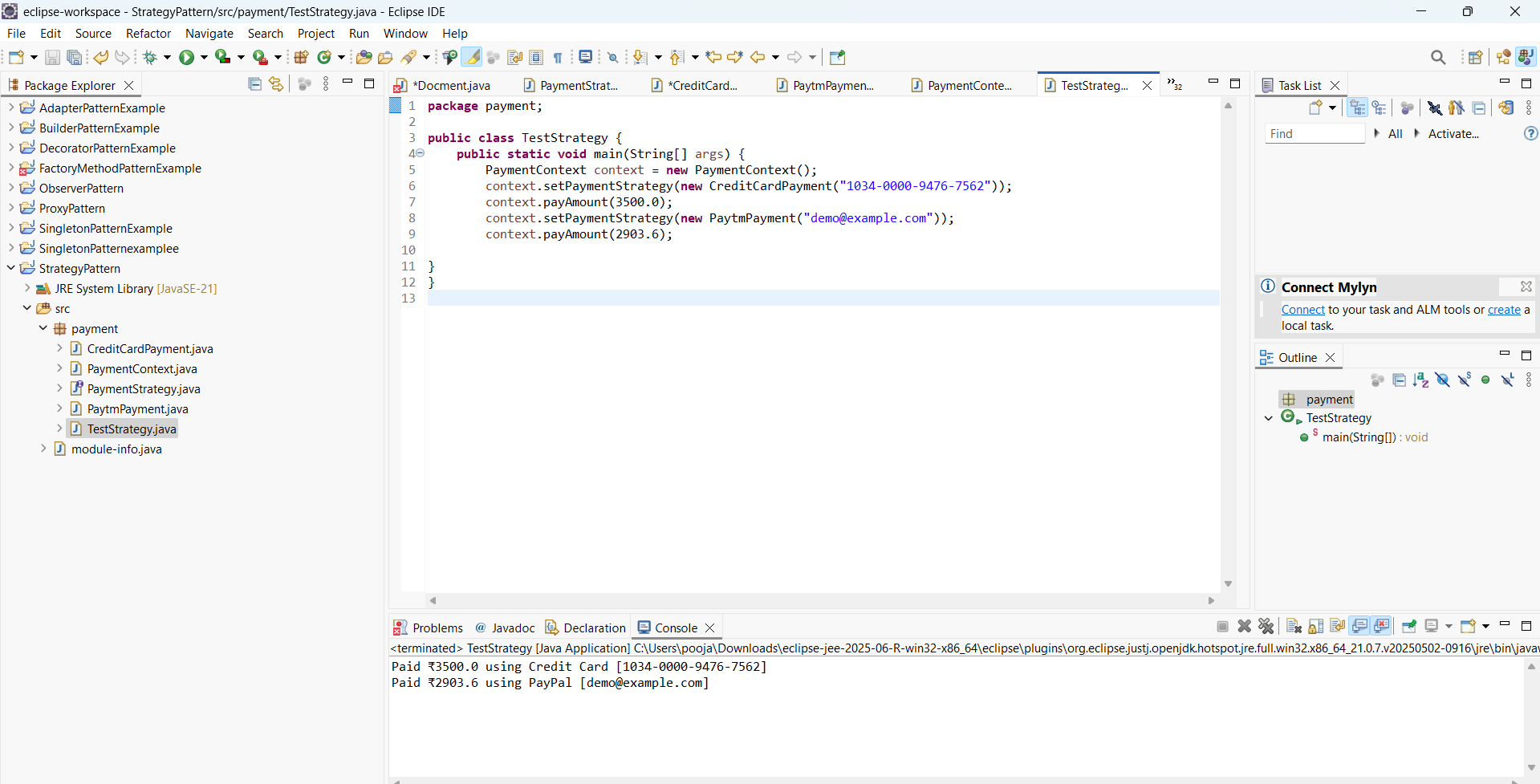
context.payAmount(3500.0);

context.setPaymentStrategy(**new** PaytmPayment("demo@example.com"));

context.payAmount(2903.6);

}

}



Output 8

**9.CommandPattern**

**Command.java**

**package** remote;

**public** **interface** Command {

**void** execute();

}

**Light.java**

**package** remote;

**public** **class** Light {

**public** **void** turnOn() {

System.***out***.println("Light is ON");

}

**public** **void** turnOff() {

System.***out***.println("Light is OFF");

}

}

**RemoteControl.java**

**package** remote;

**public** **class** RemoteControl {

**private** Command command;

**public** **void** setCommand(Command command) {

**this**.command = command;

}

**public** **void** pressButton() {

**if** (command != **null**) {

command.execute();

} **else** {

System.***out***.println("No command set");

}

}

}

**LightOnCommand.java:**

**package** remote;

**public** **class** LightOnCommand **implements** Command{

**private** Light light;

**public** LightOnCommand(Light light) {

**this**.light = light;

}

@Override

**public** **void** execute() {

light.turnOn();

}

}

**LightOffCommand.java:**

**package** remote;

**public** **class** LightOffCommand **implements** Command {

**private** Light light;

**public** LightOffCommand(Light light) {

**this**.light = light;

}

@Override

**public** **void** execute() {

light.turnOff();

}

}

**TestCommand.java:**

**package** remote;

**public** **class** TestCommand {

**public** **static** **void** main(String[] args) {

Light livingRoomLight = **new** Light();

Command lightOn = **new** LightOnCommand(livingRoomLight);

Command lightOff = **new** LightOffCommand(livingRoomLight);

RemoteControl remote = **new** RemoteControl();

remote.setCommand(lightOn);

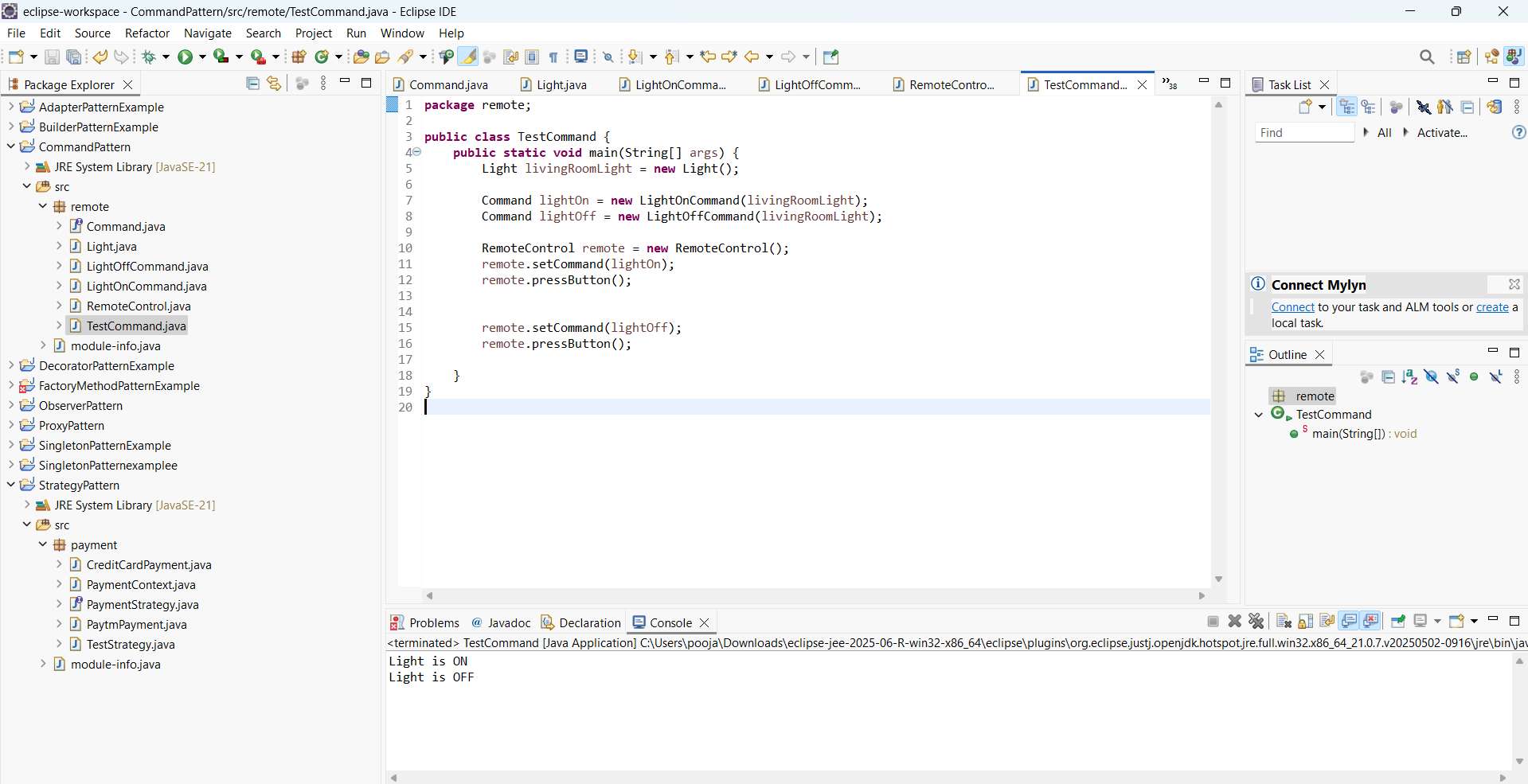
remote.pressButton();

remote.setCommand(lightOff);

remote.pressButton();

}

}



Output 9

**10.MVCPatternExample**

**Student.java**

**package** studentmvc;

**public** **class** Student {

**private** String name;

**private** String id;

**private** String grade;

**public** Student(String name, String id, String grade) {

**this**.name = name;

**this**.id = id;

**this**.grade = grade;

}

**public** String getName() { **return** name; }

**public** String getId() { **return** id; }

**public** String getGrade() { **return** grade; }

**public** **void** setName(String name) { **this**.name = name; }

**public** **void** setId(String id) { **this**.id = id; }

**public** **void** setGrade(String grade) { **this**.grade = grade; }

}

StudentController.java

**package** studentmvc;

**public** **class** StudentController {

**private** Student model;

**private** StudentView view;

**public** StudentController(Student model, StudentView view) {

**this**.model = model;

**this**.view = view;

}

**public** **void** setStudentName(String name) { model.setName(name); }

**public** **void** setStudentId(String id) { model.setId(id); }

**public** **void** setStudentGrade(String grade) { model.setGrade(grade); }

**public** **void** updateView() {

view.displayStudentDetails(model.getName(), model.getId(), model.getGrade());

}

}

**StudentView.java**

**package** studentmvc;

**public** **class** StudentView {

**public** **void** displayStudentDetails(String name, String id, String grade) {

System.***out***.println("=== Student Details ===");

System.***out***.println("Name : " + name);

System.***out***.println("ID : " + id);

System.***out***.println("Grade : " + grade);

System.***out***.println();

}

}

**TestMVC.java**

**package** studentmvc;

**public** **class** TestMVC {

**public** **static** **void** main(String[] args) {

Student student = **new** Student("Jack", "S303", "A");

StudentView view = **new** StudentView();

StudentController controller = **new** StudentController(student, view);

controller.updateView();

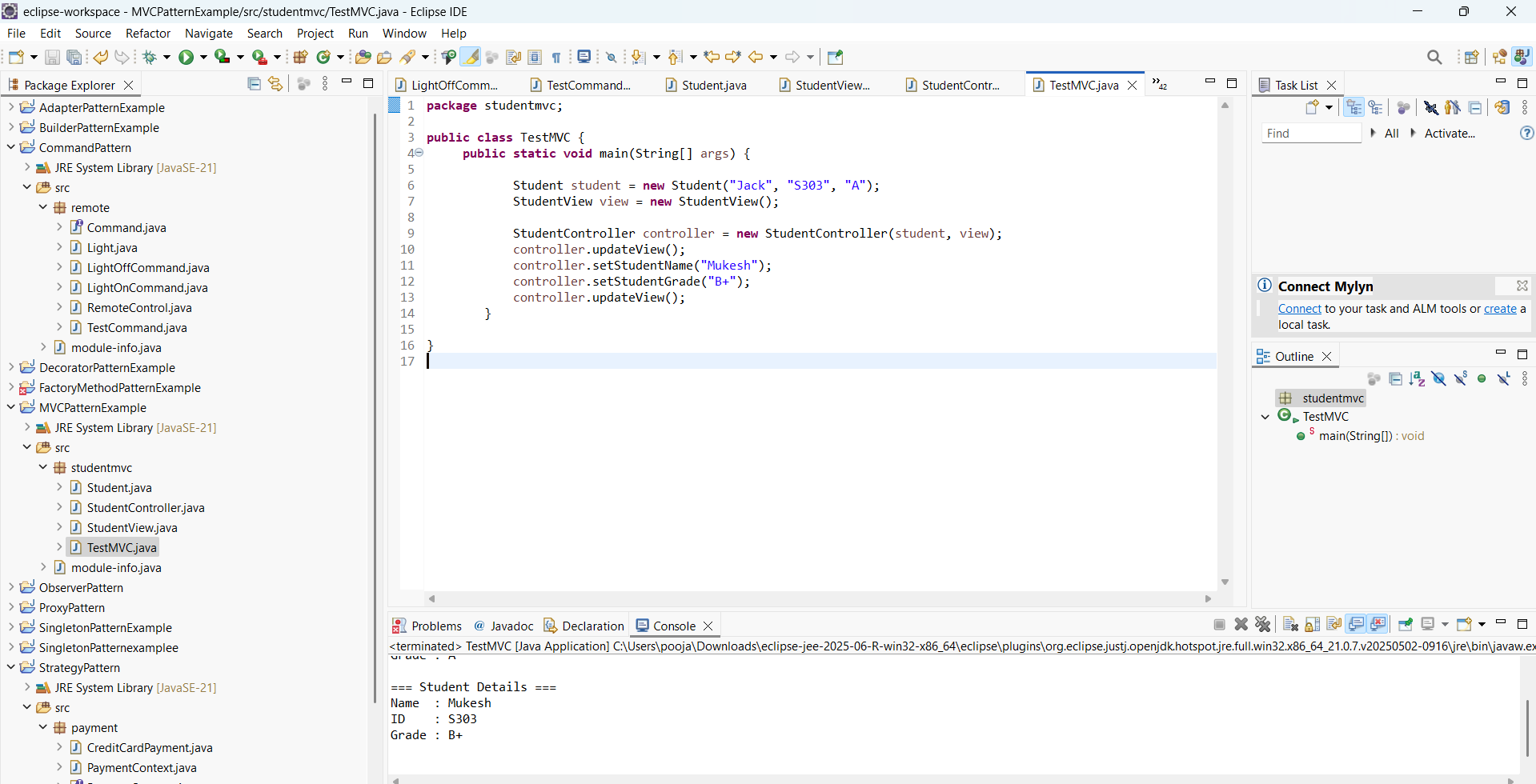
controller.setStudentName("Mukesh");

controller.setStudentGrade("B+");

controller.updateView();

}

}



Output 10