**PROBLEM 1:**

//inside filename Logger.java

package com.singleton;

public class Logger {

// Private static instance

private static Logger instance;

// Private constructor

private Logger() {

System.out.println("Logger Initialized");

}

// Public method to provide access to the instance

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

// Logging method

public void log(String message) {

System.out.println("LOG: " + message);

}

}

//Main.java

package com.singleton;

public class Logger {

// Private static instance

private static Logger instance;

// Private constructor

private Logger() {

System.out.println("Logger Initialized");

}

// Public method to provide access to the instance

public static Logger getInstance() {

if (instance == null) {

instance = new Logger();

}

return instance;

}

// Logging method

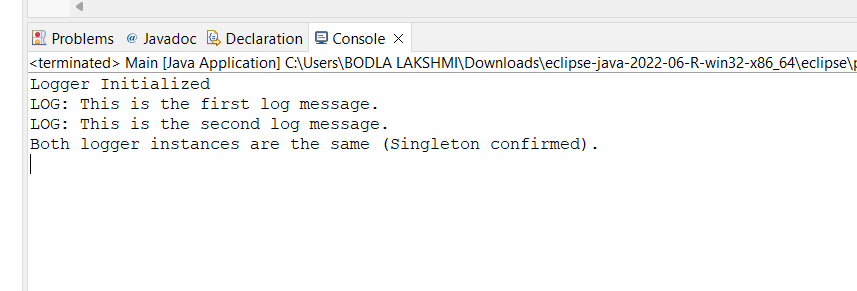
public void log(String message) {

System.out.println("LOG: " + message);

}

}

OUTPUT:



**PROBLEM 2:**

//Document.java

**package** com.factorymethod;

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

**void** open();

}

//PdfDocument.java

**package** com.factorymethod;

**public** **class** PdfDocument **implements** Document {

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

System.***out***.println("Opened PDF Document");

}

}

//WordDocument.java

**package** com.factorymethod;

**public** **class** WordDocument **implements** Document {

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

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

}

}

//Main.java

**package** com.factorymethod;

**public** **class** Main {

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

Document doc1 = DocumentFactory.*createDocument*("word");

doc1.open();

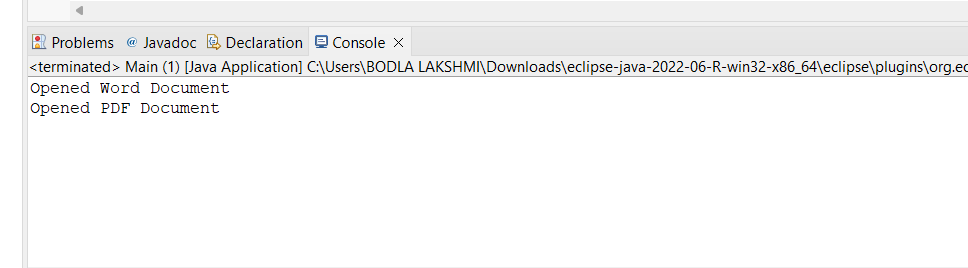
Document doc2 = DocumentFactory.*createDocument*("pdf");

doc2.open();

}

}

OUTPUT:



Exercise :3

//Computer.java

**package** com.builder;

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

// Required and optional attributes

**private** String cpu;

**private** String ram;

**private** String storage;

**private** String graphicsCard;

// Private constructor

**private** Computer(Builder builder) {

**this**.cpu = builder.cpu;

**this**.ram = builder.ram;

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

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

}

// Static nested Builder class

**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**);

}

}

// Display method

**public** **void** displayConfiguration() {

System.***out***.println("CPU: " + cpu);

System.***out***.println("RAM: " + ram);

System.***out***.println("Storage: " + storage);

System.***out***.println("Graphics Card: " + graphicsCard);

}

}

//Main.java

**package** com.builder;

**public** **class** Main {

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

// Basic computer

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

.setCPU("Intel i3")

.setRAM("8GB")

.setStorage("256GB SSD")

.build();

System.***out***.println("Basic Computer:");

basicComputer.displayConfiguration();

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

// High-end gaming computer

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

.setCPU("Intel i9")

.setRAM("32GB")

.setStorage("1TB SSD")

.setGraphicsCard("NVIDIA RTX 4090")

.build();

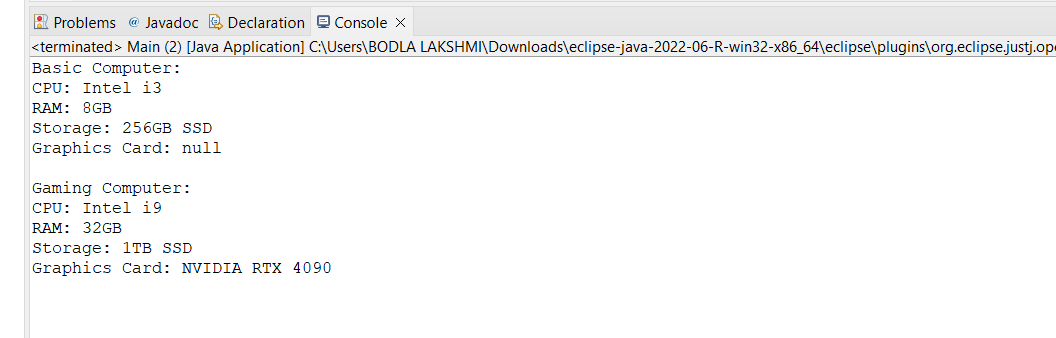
System.***out***.println("Gaming Computer:");

gamingComputer.displayConfiguration();

}

}

OUTPUT:



EXERCISE:4

//PaymentProcessor.java

**package** com.adapter;

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

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

}

//PayPalAdapter.java

**package** com.adapter;

**public** **class** PayPalAdapter **implements** PaymentProcessor {

**private** PayPalGateway paypal;

**public** PayPalAdapter(PayPalGateway paypal) {

**this**.paypal = paypal;

}

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

paypal.sendPayment(amount);

}

}

//PayPalGateway.java

**package** com.adapter;

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

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

}

//StripeAdapter.java

**package** com.adapter;

**public** **class** StripeAdapter **implements** PaymentProcessor {

**private** StripeGateway stripe;

**public** StripeAdapter(StripeGateway stripe) {

**this**.stripe = stripe;

}

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

stripe.makePayment(amount);

}

}

//StripeGateway.java

**package** com.adapter;

**public** **class** StripeGateway {

**public** **void** makePayment(**double** amountInRupees) {

System.***out***.println("Processing payment through Stripe: ₹" + amountInRupees);

}

}

//Main.java

**package** com.adapter;

**public** **class** Main {

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

// Using PayPal via adapter

PaymentProcessor paypalProcessor = **new** PayPalAdapter(**new** PayPalGateway());

paypalProcessor.processPayment(1500.00);

// Using Stripe via adapter

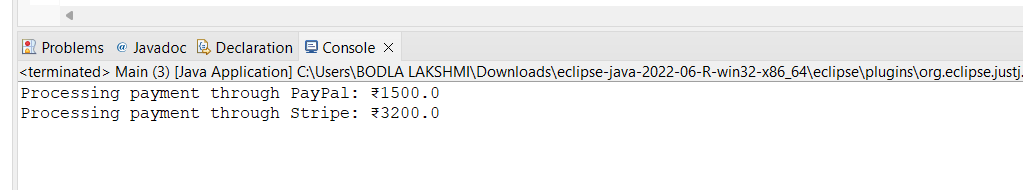
PaymentProcessor stripeProcessor = **new** StripeAdapter(**new** StripeGateway());

stripeProcessor.processPayment(3200.00);

}

}

OUTPUT:



EXERCISE:5

//EmailNotifier.java

**package** com.decorator;

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

**public** **void** send() {

System.***out***.println("Sending Email Notification");

}

}

//Notifier.java

**package** com.decorator;

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

**void** send();

}

//NotifierDecorator.java

**package** com.decorator;

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

**protected** Notifier notifier;

**public** NotifierDecorator(Notifier notifier) {

**this**.notifier = notifier;

}

**public** **void** send() {

notifier.send(); // delegate to the original notifier

}

}

//SlackNotifierDecorator.java

**package** com.decorator;

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

**public** SlackNotifierDecorator(Notifier notifier) {

**super**(notifier);

}

**public** **void** send() {

**super**.send();

System.***out***.println("Sending Slack Notification");

}

}

//SMSNotifierDecorator.java

**package** com.decorator;

**public** **class** SMSNotifierDecorator **extends** NotifierDecorator {

**public** SMSNotifierDecorator(Notifier notifier) {

**super**(notifier);

}

**public** **void** send() {

**super**.send();

System.***out***.println("Sending SMS Notification");

}

}

//Main.java

**package** com.decorator;

**public** **class** Main {

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

// Base notifier: Email only

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

// Add SMS on top of Email

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

// Add Slack on top of Email + SMS

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

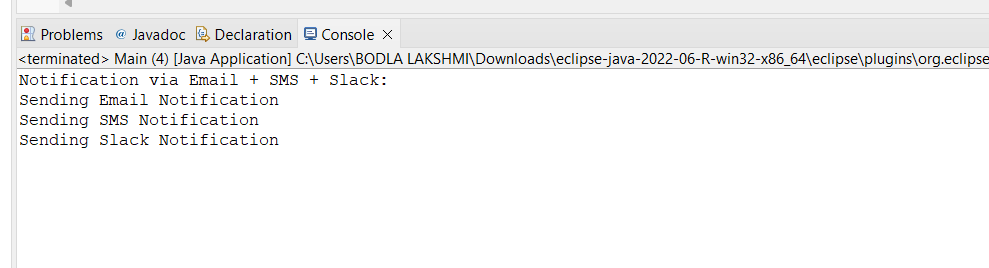
System.***out***.println("Notification via Email + SMS + Slack:");

allNotifier.send();

}

}

OUTPUT:



EXERCISE 6:

//Image.java

**package** com.proxy;

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

**void** display();

}

//ProxyImage.java

**package** com.proxy;

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

**private** RealImage realImage;

**private** String fileName;

**public** ProxyImage(String fileName) {

**this**.fileName = fileName;

}

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

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

realImage = **new** RealImage(fileName); // Lazy initialization

}

realImage.display();

}

}

//RealImage.java

**package** com.proxy;

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

**private** String fileName;

**public** RealImage(String fileName) {

**this**.fileName = fileName;

loadFromRemoteServer();

}

**private** **void** loadFromRemoteServer() {

System.***out***.println("Loading image from remote server: " + fileName);

}

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

System.***out***.println("Displaying image: " + fileName);

}

}

//Main.java

**package** com.proxy;

**public** **class** Main {

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

Image image1 = **new** ProxyImage("photo1.jpg");

Image image2 = **new** ProxyImage("photo2.jpg");

// First time: Loads from remote

image1.display();

// Second time: Uses cached real image

image1.display();

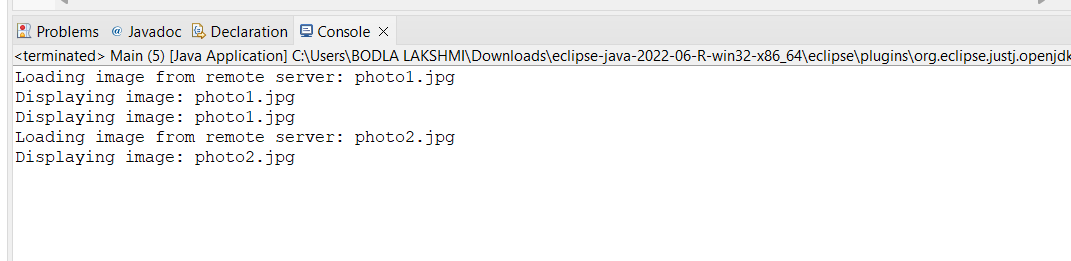
// Loads another image

image2.display();

}

}

OUTPUT:



EXERCISE:7

//Stock.java

package com.observer;

public interface Stock {

void registerObserver(Observer o);

void removeObserver(Observer o);

void notifyObservers();

}

//Observer.java

package com.observer;

public interface Observer {

void update(String stockName, double price);

}

//StockMarket.java

package com.observer;

import java.util.ArrayList;

import java.util.List;

public class StockMarket implements Stock {

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

private String stockName;

private double stockPrice;

public void setStockPrice(String stockName, double stockPrice) {

this.stockName = stockName;

this.stockPrice = stockPrice;

notifyObservers(); // notify whenever stock price changes

}

public void registerObserver(Observer o) {

observers.add(o);

}

public void removeObserver(Observer o) {

observers.remove(o);

}

public void notifyObservers() {

for (Observer o : observers) {

o.update(stockName, stockPrice);

}

}

}

//MobileApp.java

package com.observer;

public class MobileApp implements Observer {

private String name;

public MobileApp(String name) {

this.name = name;

}

public void update(String stockName, double price) {

System.out.println("[" + name + " - Mobile] Stock Update: " + stockName + " is now ₹" + price);

}

}

//WebApp.java

package com.observer;

public class WebApp implements Observer {

private String name;

public WebApp(String name) {

this.name = name;

}

public void update(String stockName, double price) {

System.out.println("[" + name + " - Web] Stock Update: " + stockName + " is now ₹" + price);

}

}

//Main.java

package com.observer;

public class Main {

public static void main(String[] args) {

StockMarket stockMarket = new StockMarket();

Observer mobileUser = new MobileApp("Ram");

Observer webUser = new WebApp("Sri");

// Register observers

stockMarket.registerObserver(mobileUser);

stockMarket.registerObserver(webUser);

// Simulate price changes

stockMarket.setStockPrice("TCS", 3550.50);

stockMarket.setStockPrice("INFY", 1510.75);

// Remove one observer

stockMarket.removeObserver(mobileUser);

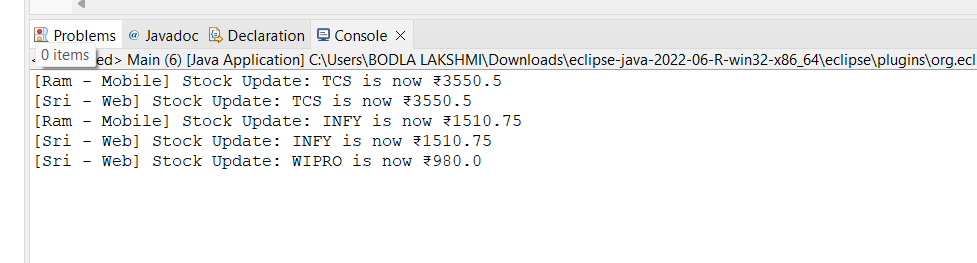
// Simulate another change

stockMarket.setStockPrice("WIPRO", 980.00);

}

}

OUTPUT:



EXERCISE 8:

//PaymentStrategy.java

package com.strategy;

public interface PaymentStrategy {

void pay(double amount);

}

//CreditCardPayment.java

package com.strategy;

public class CreditCardPayment implements PaymentStrategy {

private String cardNumber;

public CreditCardPayment(String cardNumber) {

this.cardNumber = cardNumber;

}

public void pay(double amount) {

System.out.println("Paid ₹" + amount + " using Credit Card ending with " + cardNumber.substring(cardNumber.length() - 4));

}

}

//PayPalPayment.java

package com.strategy;

public class PayPalPayment implements PaymentStrategy {

private String email;

public PayPalPayment(String email) {

this.email = email;

}

public void pay(double amount) {

System.out.println("Paid ₹" + amount + " using PayPal account: " + email);

}

}

//PaymentContext.java

package com.strategy;

public class PaymentContext {

private PaymentStrategy strategy;

public void setPaymentStrategy(PaymentStrategy strategy) {

this.strategy = strategy;

}

public void pay(double amount) {

if (strategy == null) {

System.out.println("No payment strategy selected.");

} else {

strategy.pay(amount);

}

}

}

//Main.java

package com.strategy;

public class Main {

public static void main(String[] args) {

PaymentContext context = new PaymentContext();

// Pay using Credit Card

context.setPaymentStrategy(new CreditCardPayment("1234567890123456"));

context.pay(1500.00);

// Pay using PayPal

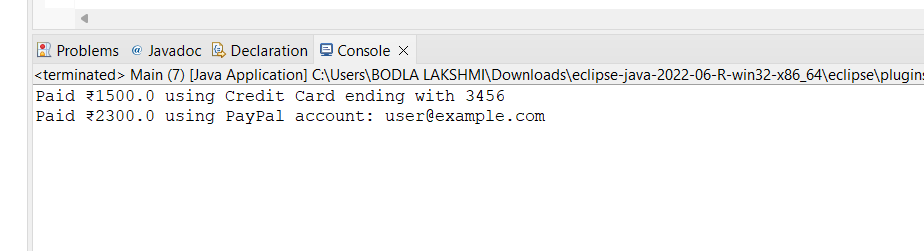
context.setPaymentStrategy(new PayPalPayment("user@example.com"));

context.pay(2300.00);

}

}

OUTPUT:



EXERCISE:9

//Command.java

package com.command;

public interface Command {

void execute();

}

//Light.java

package com.command;

public class Light {

public void turnOn() {

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

}

public void turnOff() {

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

}

}

//LightOnCommand.java

package com.command;

public class LightOnCommand implements Command {

private Light light;

public LightOnCommand(Light light) {

this.light = light;

}

public void execute() {

light.turnOn();

}

}

//LightOffCommand.java

package com.command;

public class LightOffCommand implements Command {

private Light light;

public LightOffCommand(Light light) {

this.light = light;

}

public void execute() {

light.turnOff();

}

}

//RemoteControl.java

package com.command;

public class RemoteControl {

private Command command;

public void setCommand(Command command) {

this.command = command;

}

public void pressButton() {

command.execute();

}

}

//Main.java

package com.command;

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 the light

remote.setCommand(lightOn);

remote.pressButton();

// Turn off the light

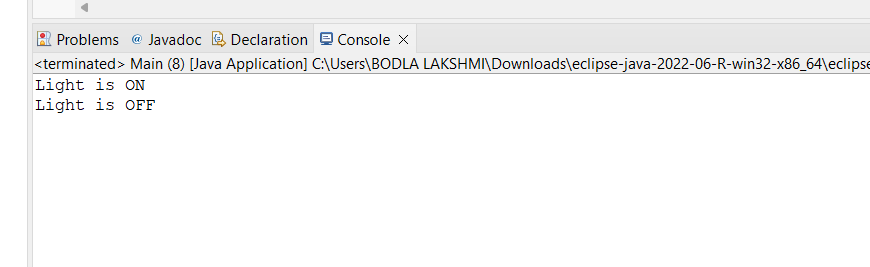
remote.setCommand(lightOff);

remote.pressButton();

}

}

OUTPUT:



EXERCISE:10