**1)**

**// Logger Class**

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

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

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

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

{

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

}

**return** *instance*;

}

**public** **static** **void** log(String msg)

{

System.***out***.println("LOG : "+msg);

}

}

**//Test Class**

**Public** **class** test {

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

{

Logger l1=Logger.*getInstance*();

Logger l2=Logger.*getInstance*();

l1.log("This is the first object");

l2.log("This is the second object");

**if**(l1==l2)

{

System.***out***.println("Both the instance are the same.");

}

**else**

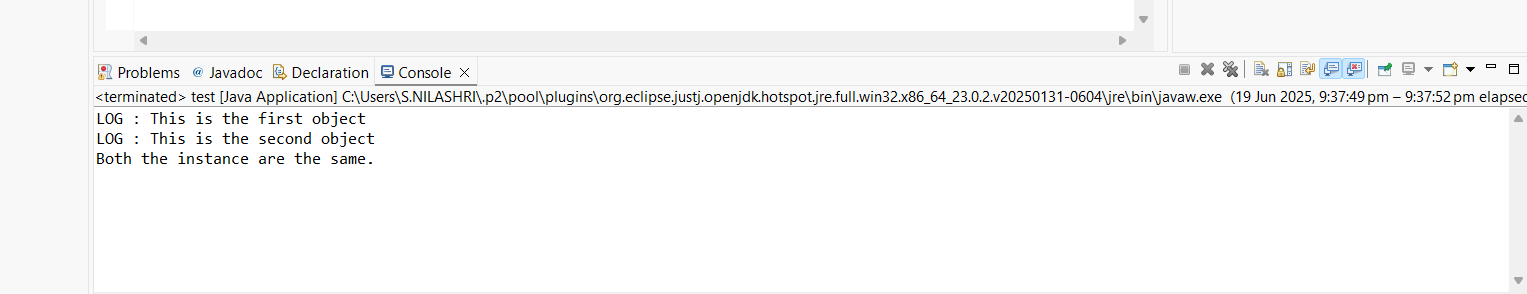
{

System.***out***.println("The instances are different.");

}

}

}



**2)**

**//DocumentFactory**

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

**public** **abstract** Documents createDocument();

}

**//Documents**

**public** **interface** Documents {

**public** **void** open();

}

**// ExcelDocument**

**public** **class** ExcelDocument **implements** Documents{

@Override

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

{

System.***out***.println("Open Excel Document");

}

}

**//** **ExcelDocumentFactory**

**public** **class** ExcelDocumentFactory **extends** DocumentFactory{

@Override

**public** Documents createDocument() {

**return** **new** ExcelDocument();

}

}

**//PdfDocument**

**public** **class** PdfDocument **implements** Documents{

@Override

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

{

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

}

}

**//PdfDocumentFactory**

**public** **class** PdfDocumentFactory **extends** DocumentFactory{

@Override

**public** Documents createDocument() {

**return** **new** PdfDocument();

}

}

**// WordDocument**

**public** **class** WordDocument **implements** Documents{

@Override

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

{

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

}

}

**//WordDocumentFactory**

**public** **class** WordDocumentFactory **extends** DocumentFactory{

@Override

**public** Documents createDocument() {

**return** **new** WordDocument();

}

}

**//Main**

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

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

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

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

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

Documents wordDoc = wordFactory.createDocument();

Documents pdfDoc = pdfFactory.createDocument();

Documents excelDoc = excelFactory.createDocument();

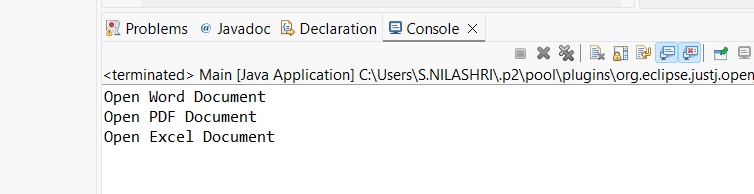
wordDoc.open();

pdfDoc.open();

excelDoc.open();

}

}



**3)**

**//Computer**

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

**private** String CPU;

**private** String RAM;

**private** String Storage;

**private** String Display;

**private** Computer(Builder builder)

{

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

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

**this**.Storage = builder.Storage;

**this**.Display= builder.Display;

}

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

{

**private** String CPU;

**private** String RAM;

**private** String Storage;

**private** String Display;

**public** Builder(String CPU, String RAM) {

**this**.CPU = CPU;

**this**.RAM = RAM;

}

**public** Builder setStorage(String Storage)

{

**this**.Storage=Storage;

**return** **this**;

}

**public** Builder setDisplay(String Display)

{

**this**.Display=Display;

**return** **this**;

}

**public** Computer build() {

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

}

}

@Override

**public** String toString() {

**return** "Computer [CPU=" + CPU + ", RAM=" + RAM + ", Storage=" + Storage

+ ", Display=" + Display + "]";

}

}

**//TestBuilder**

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

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

Computer basicComputer = **new** Computer.Builder("Intel i5", "8GB")

.build();

Computer gamingComputer = **new** Computer.Builder("Intel i9", "32GB")

.setStorage("1TB SSD")

.setDisplay("16 inch")

.build();

Computer workstation = **new** Computer.Builder("AMD Ryzen 9", "64GB")

.setStorage("2TB SSD")

.setDisplay("14 inch")

.build();

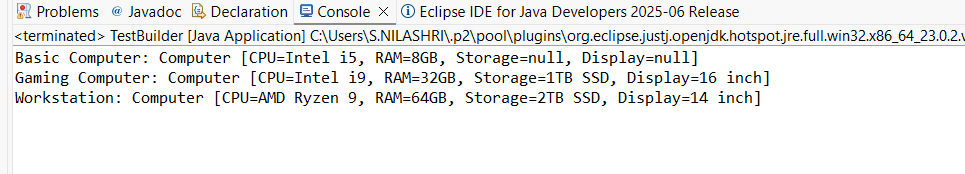
System.***out***.println("Basic Computer: " + basicComputer);

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

System.***out***.println("Workstation: " + workstation);

}

}



**4)**

**//PaymentProcessor**

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

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

}

**//PayPal**

**public** **class** PayPal {

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

System.***out***.println("Processing payment of $" + amountInUSD + " through PayPal.");

}

}

**//PayPalAdapter**

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

**private** PayPalGateway payPalGateway;

**public** PayPalAdapter(PayPalGateway payPalGateway) {

**this**.payPalGateway = payPalGateway;

}

@Override

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

payPalGateway.makePayment(amount);

}

}

**//PayPalGateway**

**public** **class** PayPalGateway **implements** PaymentProcessor{

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

System.***out***.println("Processing payment of $" + amountInUSD + " through PayPal.");

}

}

**//StripeAdapter**

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

**private** StripeGateway stripeGateway;

**public** StripeAdapter(StripeGateway stripeGateway) {

**this**.stripeGateway = stripeGateway;

}

@Override

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

stripeGateway.charge(amount);

}

}

**//StripeGateway**

**public** **class** StripeGateway **implements** PaymentProcessor{

**public** **void** charge(**double** amountInUSD) {

System.***out***.println("Charging $" + amountInUSD + " using Stripe.");

}

}

**//PaymentTest**

**public** **class** PaymentTest {

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

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

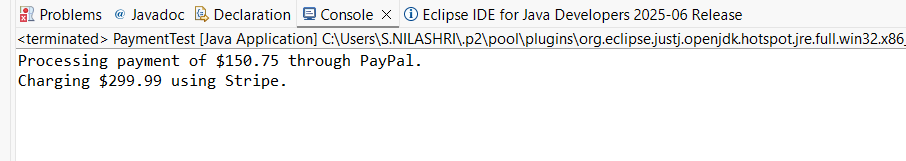
paypalProcessor.processPayment(150.75);

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

stripeProcessor.processPayment(299.99);

}

}



**5)**

**//Notifier**

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

**void** send(String message);

}

**//EmailNotifier**

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

@Override

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

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

}

}

**//NotifierDecorator**

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

**protected** Notifier wrappee;

**public** NotifierDecorator(Notifier notifier) {

**this**.wrappee = notifier;

}

@Override

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

wrappee.send(message);

}

}

**//SlackNotifierDecorator**

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

**public** SlackNotifierDecorator(Notifier notifier) {

**super**(notifier);

}

@Override

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

**super**.send(message);

sendSlack(message);

}

**private** **void** sendSlack(String message) {

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

}

}

**//SMSNotifierDecorator**

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

**public** SMSNotifierDecorator(Notifier notifier) {

**super**(notifier);

}

@Override

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

**super**.send(message);

sendSMS(message);

}

**private** **void** sendSMS(String message) {

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

}

}

**//Main**

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

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

{

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

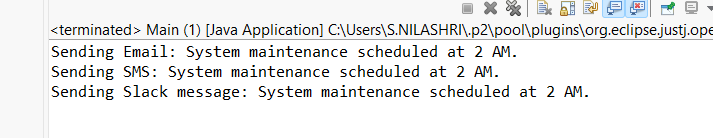
Notifier smsNotifier = **new** SMSNotifierDecorator(emailNotifier);

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

multiChannelNotifier.send("System maintenance scheduled at 2 AM.");

}

}



**6)**

**//Image**

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

**void** display();

}

**//ProxyImage**

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

**private** String filename;

**private** RealImage realImage;

**public** ProxyImage(String filename) {

**this**.filename = filename;

}

@Override

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

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

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

}

realImage.display();

}

}

**//RealImage**

**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);

// Simulated delay (optional)

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

Thread.*currentThread*().interrupt();

}

}

@Override

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

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

}

}

**//Main**

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

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

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

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

System.***out***.println("\nDisplaying image1 first time:");

image1.display();

System.***out***.println("\nDisplaying image1 second time:");

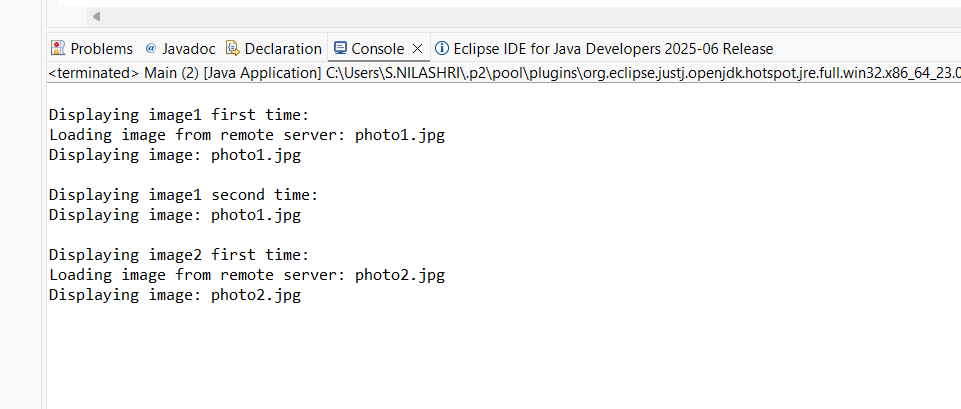
image1.display();

System.***out***.println("\nDisplaying image2 first time:");

image2.display();

}

}



**7)**

**//MobileApp**

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

**private** String appName;

**public** MobileApp(String appName) {

**this**.appName = appName;

}

@Override

**public** **void** update(**double** stockPrice) {

System.***out***.println(appName + " (Mobile): New stock price = $" + stockPrice);

}

}

**//Observer**

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

**void** update(**double** stockPrice);

}

**//Stock**

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

**void** registerObserver(Observer observer);

**void** removeObserver(Observer observer);

**void** notifyObservers();

}

**//StockMarket**

**import** java.util.ArrayList;

**import** java.util.List;

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

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

**private** **double** stockPrice;

@Override

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

observers.add(observer);

}

@Override

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

observers.remove(observer);

}

@Override

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

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

observer.update(stockPrice);

}

}

**public** **void** setStockPrice(**double** price) {

**this**.stockPrice = price;

System.***out***.println("StockMarket: Stock price updated to $" + price);

notifyObservers();

}

}

**//WebApp**

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

**private** String appName;

**public** WebApp(String appName) {

**this**.appName = appName;

}

@Override

**public** **void** update(**double** stockPrice) {

System.***out***.println(appName + " (Web): New stock price = $" + stockPrice);

}

}

**//Main**

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

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

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

Observer mobileApp = **new** MobileApp("StockTracker");

Observer webApp = **new** WebApp("StockViewer");

stockMarket.registerObserver(mobileApp);

stockMarket.registerObserver(webApp);

stockMarket.setStockPrice(100.50);

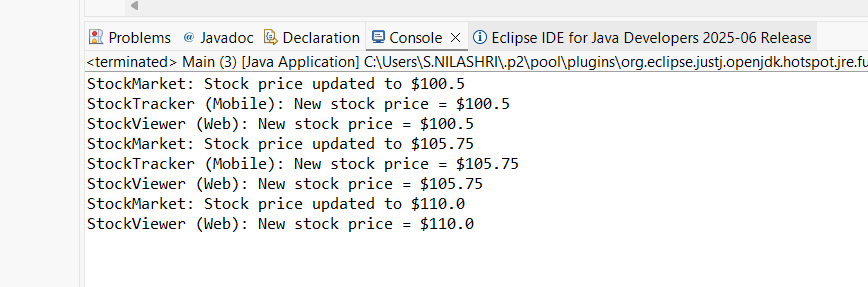
stockMarket.setStockPrice(105.75);

stockMarket.removeObserver(mobileApp);

stockMarket.setStockPrice(110.00);

}

}



**8)**

**//CreditCardPayment**

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

**private** String cardNumber;

**private** String cardHolder;

**public** CreditCardPayment(String cardNumber, String cardHolder) {

**this**.cardNumber = cardNumber;

**this**.cardHolder = cardHolder;

}

@Override

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

System.***out***.println("Paid Rs" + amount + " using Credit Card: " + cardNumber + " (Cardholder: " + cardHolder + ")");

}

}

**//PaymentContext**

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

**private** PaymentStrategy paymentStrategy;

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

**this**.paymentStrategy = paymentStrategy;

}

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

**if** (paymentStrategy == **null**) {

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

} **else** {

paymentStrategy.pay(amount);

}

}

}

**//PaymentStrategy**

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

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

}

**//PayPalPayment**

**public** **class** PayPalPayment **implements** PaymentStrategy{

**private** String email;

**public** PayPalPayment(String email) {

**this**.email = email;

}

@Override

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

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

}

}

**//Main**

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

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

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

context.setPaymentStrategy(**new** CreditCardPayment("1234-5678-9012-3456", "Arun"));

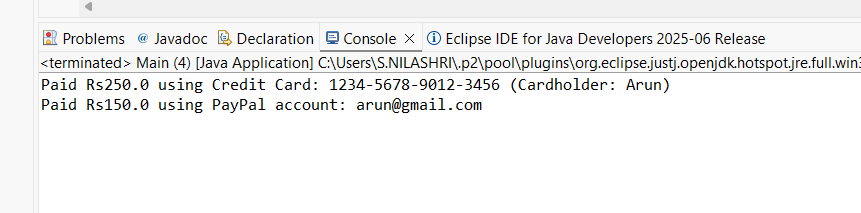
context.executePayment(250.00);

context.setPaymentStrategy(**new** PayPalPayment("arun@gmail.com"));

context.executePayment(150.00);

}

}



**9)**

**//Command**

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

**void** execute();

}

**//Light**

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

**private** String location;

**public** Light(String location) {

**this**.location = location;

}

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

System.***out***.println(location + " light is ON.");

}

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

System.***out***.println(location + " light is OFF.");

}

}

**//LightOffCommand**

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

**private** Light light;

**public** LightOffCommand(Light light) {

**this**.light = light;

}

@Override

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

light.turnOff();

}

}

**//LightOnCommand**

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

**private** Light light;

**public** LightOnCommand(Light light) {

**this**.light = light;

}

@Override

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

light.turnOn();

}

}

**//RemoteControl**

**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 assigned.");

}

}

}

**//Main**

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

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

Light livingRoomLight = **new** Light("Living Room");

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

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

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

remote.setCommand(lightOn);

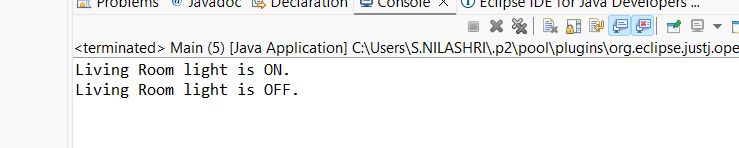
remote.pressButton();

remote.setCommand(lightOff);

remote.pressButton();

}

}



**10)**

**//Student**

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

**private** String id;

**private** String name;

**private** String grade;

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

**this**.id = id;

**this**.name = name;

**this**.grade = grade;

}

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

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

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

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

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

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

}

**//StudentController**

**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** setStudentGrade(String grade) {

model.setGrade(grade);

}

**public** String getStudentName() {

**return** model.getName();

}

**public** String getStudentGrade() {

**return** model.getGrade();

}

**public** String getStudentId() {

**return** model.getId();

}

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

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

}

}

**//StudentView**

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

**public** **void** displayStudentDetails(String studentId, String studentName, String studentGrade) {

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

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

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

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

}

}

**//Main**

**public** **class** Mian {

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

Student student = **new** Student("S001", "Alice", "A");

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

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

controller.updateView();

controller.setStudentName("Alice Johnson");

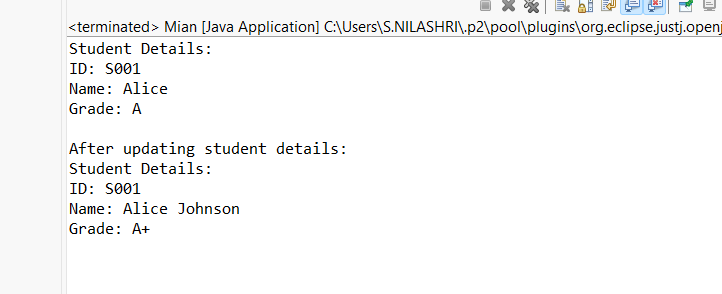
controller.setStudentGrade("A+");

System.***out***.println("\nAfter updating student details:");

controller.updateView();

}

}



**11)**

**//CustomerRepository**

**public** **interface** CustomerRepository {

Customer findCustomerById(String id);

}

**//Customer**

**public** **class** Customer {

**private** String id;

**private** String name;

**private** String email;

**public** Customer(String id, String name, String email) {

**this**.id = id;

**this**.name = name;

**this**.email = email;

}

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

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

**public** String getEmail() { **return** email; }

@Override

**public** String toString() {

**return** "Customer ID: " + id + ", Name: " + name + ", Email: " + email;

}

}

**//CustomerRepositoryImpl**

**public** **class** CustomerRepositoryImpl **implements** CustomerRepository {

@Override

**public** Customer findCustomerById(String id) {

**return** **new** Customer(id, "John", "john@gmail.com");

}

}

**//CustomerService**

**public** **class** CustomerService {

**private** CustomerRepository customerRepository;

**public** CustomerService(CustomerRepository customerRepository) {

**this**.customerRepository = customerRepository;

}

**public** **void** displayCustomerById(String id) {

Customer customer = customerRepository.findCustomerById(id);

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

}

}

**//Main**

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

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

CustomerRepository repository = **new** CustomerRepositoryImpl();

CustomerService service = **new** CustomerService(repository);

service.displayCustomerById("C001");

}

}

