1. **Create a java program for dream job roll no, Name of Company,Country.**

public class DreamJob {

private int rollNo;

private String companyName;

private String country;

// Constructor to initialize the DreamJob object

public DreamJob(int rollNo, String companyName, String country) {

this.rollNo = rollNo;

this.companyName = companyName;

this.country = country;

}

// Method to display the details of the DreamJob

public void displayDetails()

{

System.out.println("Roll No: " + rollNo);

System.out.println("Company Name: " + companyName);

System.out.println("Country: " + country);

}

public static void main(String[] args) {

// Creating an instance of DreamJob

DreamJob dreamJob = new DreamJob(83, "Infosys", "India");

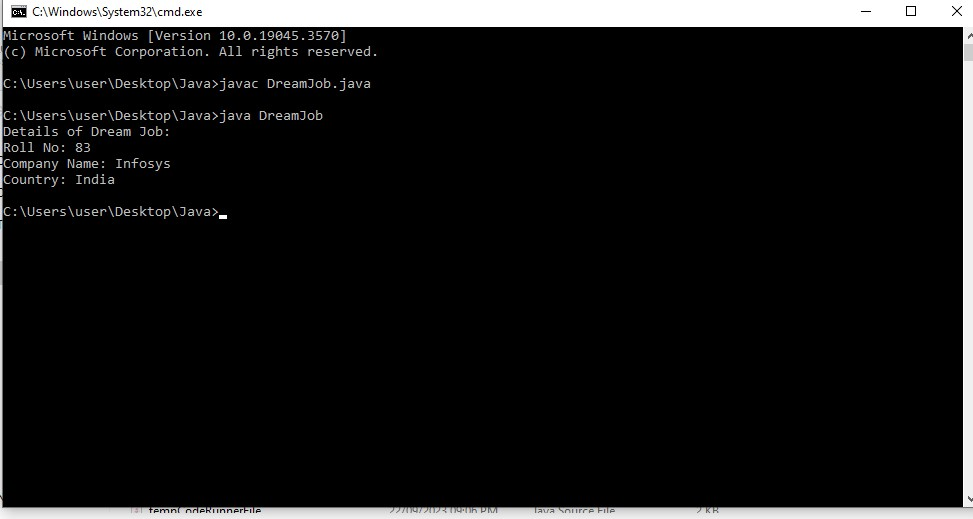
System.out.println("Details of Dream Job:");

dreamJob.displayDetails();

}

}

**Output:**



1. **Create a Java Superclass & subclass hierarchy for stowing information of IMS students for placement System IMS has following Courses: 1) MBA 2) MCA 3) B.Voc (Bachelor of vocation) B.Voc has 2 branches (finance) 2 (tours & travels). as well as it has short Courses like Event Management, Certification in Bharat Natyam , Certification in Kathak.**

**// Superclass: Student**

**class Student {**

**private int rollNo;**

**private String name;**

**private String course;**

**public Student(int rollNo, String name, String course) {**

**this.rollNo = rollNo;**

**this.name = name;**

**this.course = course;**

**}**

**// Getters and Setters for student information (roll number, name, course)**

**public void displayDetails() {**

**System.out.println("Roll No: " + rollNo);**

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

**System.out.println("Course: " + course);**

**}**

**}**

**// Subclass: MBAStudent**

**class MBAStudent extends Student {**

**public MBAStudent(int rollNo, String name) {**

**super(rollNo, name, "MBA");**

**}**

**}**

**// Subclass: MCAStudent**

**class MCAStudent extends Student {**

**public MCAStudent(int rollNo, String name) {**

**super(rollNo, name, "MCA");**

**}**

**}**

**// Subclass: BVocStudent**

**class BVocStudent extends Student {**

**private String branch;**

**public BVocStudent(int rollNo, String name, String branch) {**

**super(rollNo, name, "B.Voc");**

**this.branch = branch;**

**}**

**// Getters and Setters for branch information**

**@Override**

**public void displayDetails() {**

**super.displayDetails();**

**System.out.println("Branch: " + branch);**

**}**

**}**

**// Subclass: ShortCourseStudent**

**class ShortCourseStudent extends Student {**

**private String courseName;**

**public ShortCourseStudent(int rollNo, String name, String courseName) {**

**super(rollNo, name, "Short Course");**

**this.courseName = courseName;**

**}**

**// Getters and Setters for course name information**

**// ...**

**@Override**

**public void displayDetails() {**

**super.displayDetails();**

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

**}**

**}**

**// Main class to demonstrate the usage of the Student hierarchy**

**public class PlacementSystem {**

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

**MBAStudent mbaStudent = new MBAStudent(1, "Sara");**

**mbaStudent.displayDetails();**

**MCAStudent mcaStudent = new MCAStudent(2, "Faizan");**

**mcaStudent.displayDetails();**

**BVocStudent bvocStudent = new BVocStudent(3, "Arfat", "Finance");**

**bvocStudent.displayDetails();**

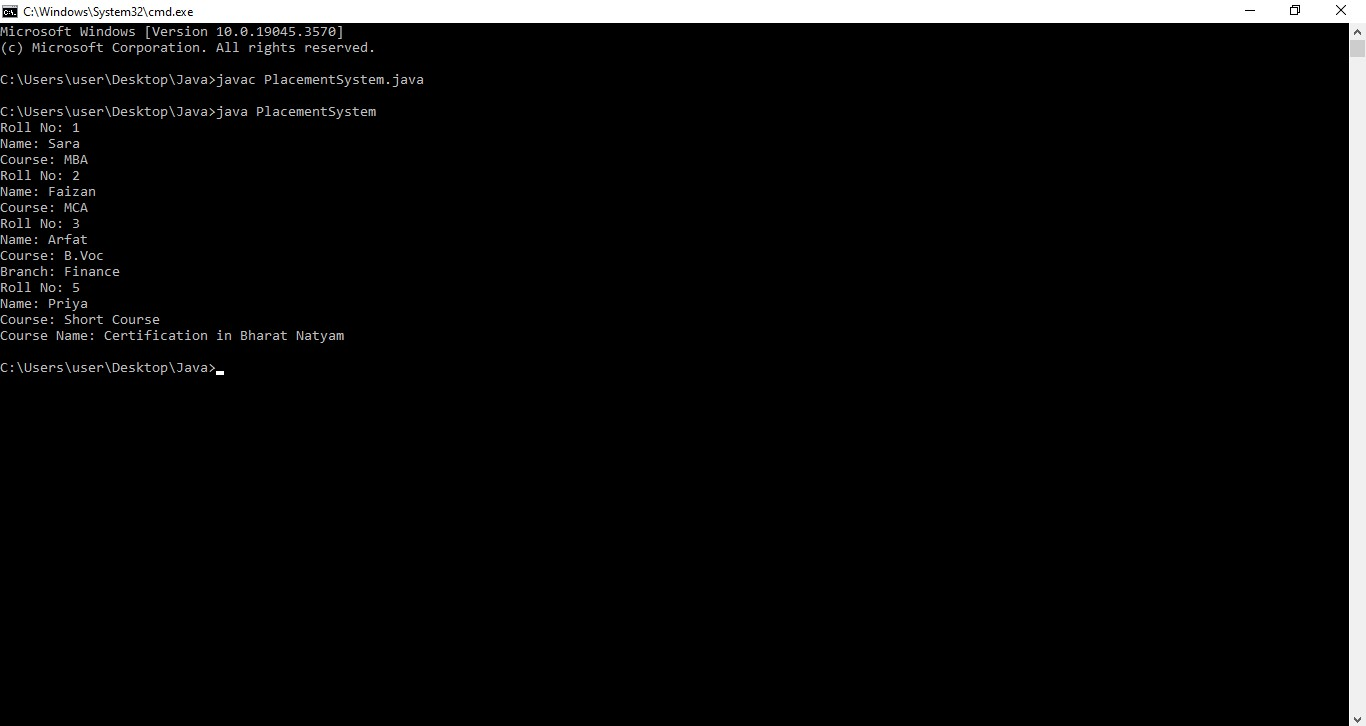
**ShortCourseStudent shortCourseStudent = new ShortCourseStudent(5, "Priya", "Certification in Bharat Natyam");**

**shortCourseStudent.displayDetails();**

**}**

**}**

**Output:**



1. **Design a java class to store information of employees of an Organization. There are 3 types. of employees: 1) Managers ii) Engineers iii) Daily Workers There Salaries are calculated as follows: Managers are getting Meeting Allowance dong with other Salary components. Engineers are getting Overtime & skill upgraded allowances workers are getting daily wages.**

**formula = Salary = basic (bay) + Dearness Allowance 2 House Rent Allowance.**

**// Superclass: Employee**

**class Employee {**

**private int employeeId;**

**private String name;**

**protected double basicSalary;**

**public Employee(int employeeId, String name, double basicSalary) {**

**this.employeeId = employeeId;**

**this.name = name;**

**this.basicSalary = basicSalary;**

**}**

**public void displayDetails() {**

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

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

**System.out.println("Basic Salary: " + basicSalary);**

**}**

**public double calculateSalary() {**

**return basicSalary;**

**}**

**}**

**// Subclass: Manager**

**class Manager extends Employee {**

**private double meetingAllowance;**

**public Manager(int employeeId, String name, double basicSalary, double meetingAllowance) {**

**super(employeeId, name, basicSalary);**

**this.meetingAllowance = meetingAllowance;**

**}**

**@Override**

**public void displayDetails() {**

**super.displayDetails();**

**System.out.println("Meeting Allowance: " + meetingAllowance);**

**}**

**@Override**

**public double calculateSalary() {**

**return basicSalary + meetingAllowance + calculateOtherAllowances();**

**}**

**private double calculateOtherAllowances() {**

**return 0;**

**}**

**}**

**// Subclass: Engineer**

**class Engineer extends Employee {**

**private double overtimeAllowance;**

**private double skillUpgradedAllowance;**

**public Engineer(int employeeId, String name, double basicSalary, double overtimeAllowance, double skillUpgradedAllowance) {**

**super(employeeId, name, basicSalary);**

**this.overtimeAllowance = overtimeAllowance;**

**this.skillUpgradedAllowance = skillUpgradedAllowance;**

**}**

**@Override**

**public void displayDetails() {**

**super.displayDetails();**

**System.out.println("Overtime Allowance: " + overtimeAllowance);**

**System.out.println("Skill Upgraded Allowance: " + skillUpgradedAllowance);**

**}**

**@Override**

**public double calculateSalary() {**

**return basicSalary + overtimeAllowance + skillUpgradedAllowance;**

**}**

**}**

**// Subclass: DailyWorker**

**class DailyWorker extends Employee {**

**private double dailyWage;**

**public DailyWorker(int employeeId, String name, double dailyWage) {**

**super(employeeId, name, 0); // Daily workers don't have a fixed basic salary**

**this.dailyWage = dailyWage;**

**}**

**@Override**

**public void displayDetails() {**

**super.displayDetails();**

**System.out.println("Daily Wage: " + dailyWage);**

**}**

**@Override**

**public double calculateSalary() {**

**return dailyWage;**

**}**

**}**

**public class EmployeeInfo {**

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

**Manager manager = new Manager(1, "John Doe", 50000, 5000);**

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

**manager.displayDetails();**

**System.out.println("Calculated Salary: " + manager.calculateSalary());**

**Engineer engineer = new Engineer(2, "Bob", 40000, 2000, 3000);**

**System.out.println("\nEngineer Details:");**

**engineer.displayDetails();**

**System.out.println("Calculated Salary: " + engineer.calculateSalary());**

**DailyWorker worker = new DailyWorker(3, "Yash", 1000);**

**System.out.println("\nDaily Worker Details:");**

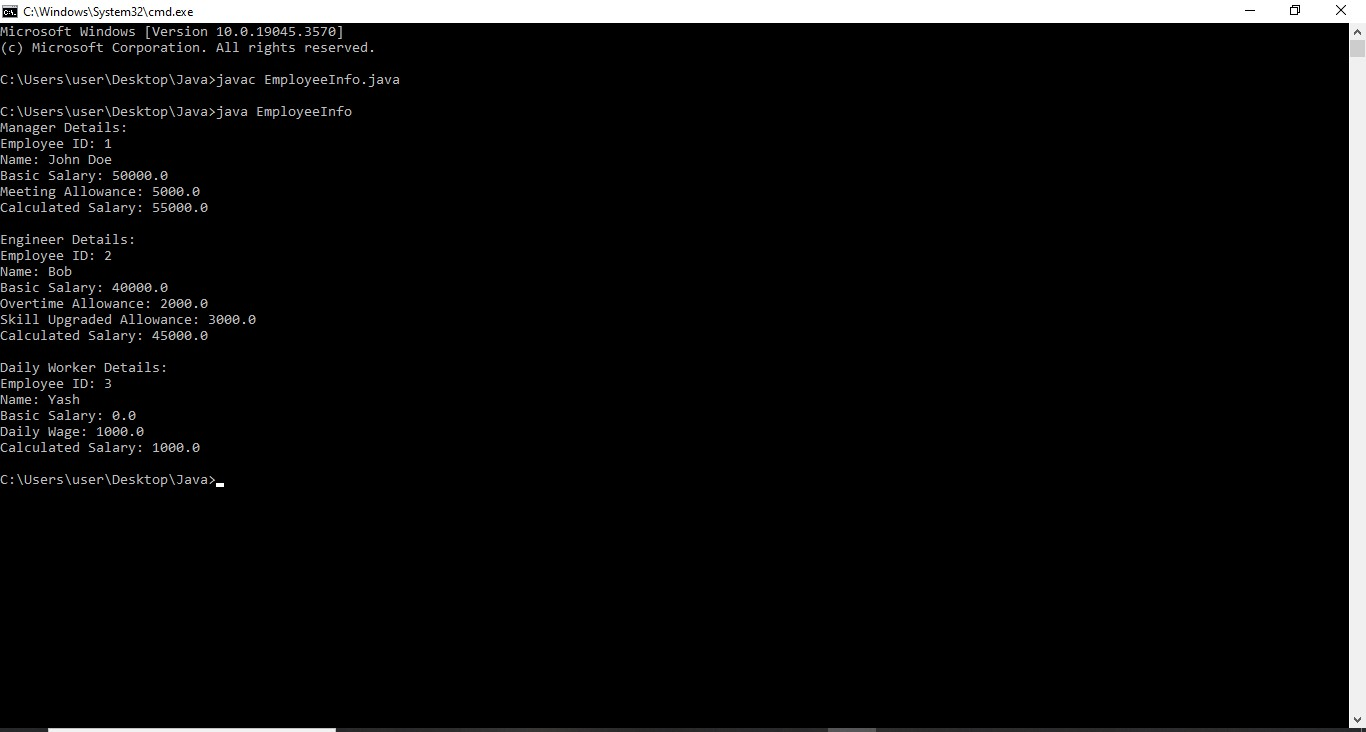
**worker.displayDetails();**

**System.out.println("Calculated Salary: " + worker.calculateSalary());**

**}**

**}**

**Output:**



**4.Create instance of car and call start, stop and move method(using interface)**

**interface Car {**

**void start();**

**void stop();**

**void move();**

**}**

**// Implement the interface in a Car class**

**class MyCar implements Car {**

**@Override**

**public void start() {**

**System.out.println("Car started");**

**}**

**@Override**

**public void stop() {**

**System.out.println("Car stopped");**

**}**

**@Override**

**public void move() {**

**System.out.println("Car is moving");**

**}**

**}**

**public class Main {**

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

**// Create an instance of the car**

**Car myCar = new MyCar();**

**// Call the methods**

**myCar.start();**

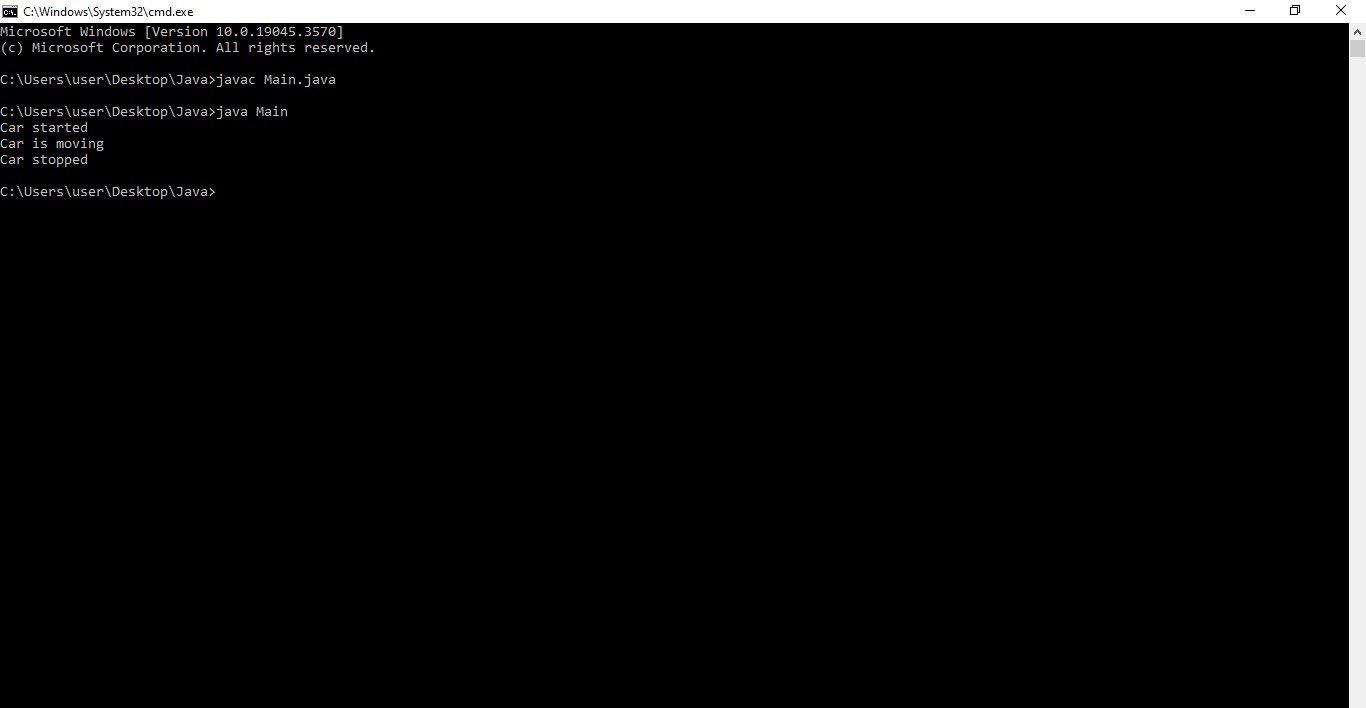
**myCar.move();**

**myCar.stop();**

**}**

**}**

**Output:**



**5.Create a Login GUI.**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.ActionEvent;**

**public class LoginForm extends JFrame {**

**JLabel username;**

**JLabel password;**

**JTextField usernameField;**

**JTextField passwordField;**

**JButton okbutton, canclebutton;**

**public LoginForm() {**

**setTitle("Shaikh Arfat");**

**setSize(300, 150);**

**setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**setLayout(null);**

**username = new JLabel("username");**

**username.setBounds(100, 80, 150, 30);**

**usernameField = new JTextField("Enter your Name");**

**usernameField.setBounds(200, 80, 200, 30);**

**password = new JLabel("password");**

**password.setBounds(100, 135, 150, 30);**

**passwordField = new JTextField("Enter your Password");**

**passwordField.setBounds(200, 135, 200, 30);**

**okbutton = new JButton("OK");**

**okbutton.setBounds(200, 200, 80, 30);**

**canclebutton = new JButton("CANCLE");**

**canclebutton.setBounds(300, 200, 100, 30);**

**add(username);**

**add(usernameField);**

**add(password);**

**add(passwordField);**

**add(okbutton);**

**add(canclebutton);**

**setVisible(true);**

**setBounds(100, 100, 500, 400);**

**}**

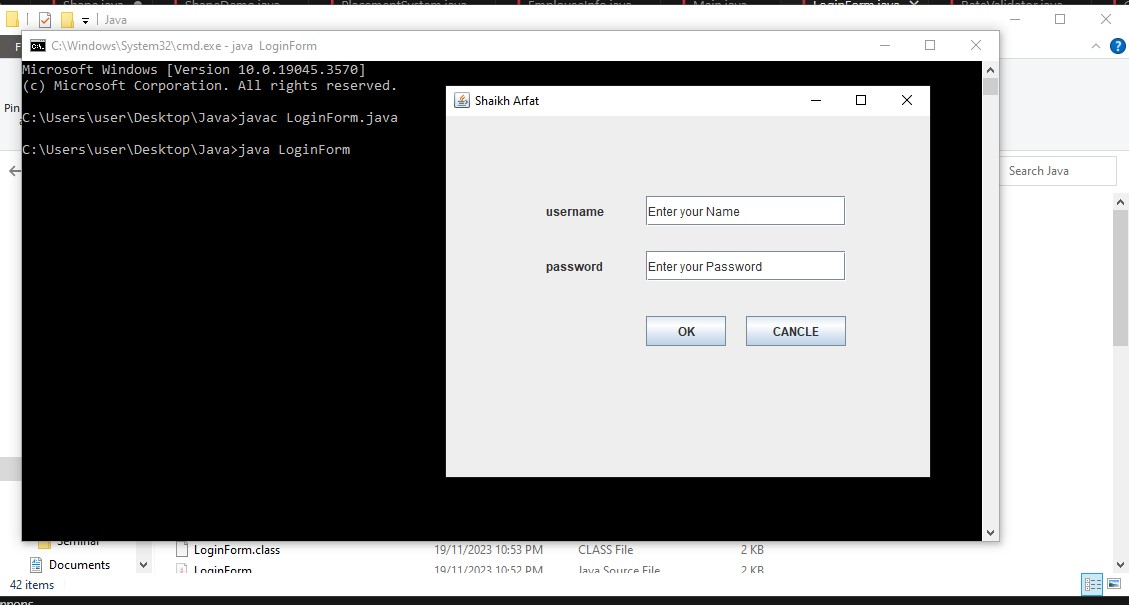
**public static void main(String[] args) {**

**new LoginForm();**

**}**

**}**

**Output:**



**6.** **Create a java jdbc program to insert record into StudentInfo database.**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**public class InsertStudent {**

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

**// JDBC URL, username, and password of MySQL server**

**String jdbcUrl = "jdbc:mysql://your\_database\_url/StudentInfo";**

**String username = "your\_username";**

**String password = "your\_password";**

**// JDBC variables for opening, closing, and managing connection**

**try (Connection connection = DriverManager.getConnection(jdbcUrl, username, password)) {**

**// The SQL query for inserting a new student record**

**String insertQuery = "INSERT INTO students (student\_id, student\_name, student\_age) VALUES (?, ?, ?)";**

**// Creating a PreparedStatement to execute the SQL query**

**try (PreparedStatement preparedStatement = connection.prepareStatement(insertQuery)) {**

**// Set values for parameters in the SQL query**

**preparedStatement.setInt(1, 1); // Replace with the actual student ID**

**preparedStatement.setString(2, "John Doe"); // Replace with the actual student name**

**preparedStatement.setInt(3, 20); // Replace with the actual student age**

**// Execute the SQL query to insert the record**

**int rowsAffected = preparedStatement.executeUpdate();**

**// Check if the record was successfully inserted**

**if (rowsAffected > 0) {**

**System.out.println("Record inserted successfully!");**

**} else {**

**System.out.println("Failed to insert the record.");**

**}**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**7. Create custom exception for invalid rate exception rate should be between 50-250Rs.**

**public class InvalidRateException extends Exception {**

**public InvalidRateException(String message) {**

**super(message);**

**}**

**}**

**public class RateValidator {**

**private static final int MIN\_RATE = 50;**

**private static final int MAX\_RATE = 250;**

**public static void validateRate(int rate) throws InvalidRateException {**

**if (rate < MIN\_RATE || rate > MAX\_RATE) {**

**throw new InvalidRateException("Invalid rate. Rate should be between " + MIN\_RATE + " and " + MAX\_RATE + " Rs.");**

**}**

**}**

**}**

**public class Example {**

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

**int customerRate = 300; // Replace with the actual rate**

**try {**

**RateValidator.validateRate(customerRate);**

**System.out.println("Rate is valid.");**

**// Perform other actions with the valid rate...**

**} catch (InvalidRateException e) {**

**System.out.println("Error: " + e.getMessage());**

**// Handle the exception or provide feedback to the user...**

**}**

**}**

**}**