Task server:

package Mainserver;

import java.io.\*;

import java.net.\*;

import java.sql.\*;

import java.util.concurrent.\*;

public class Server implements Runnable {

private final Socket COSS;

public Server(Socket clientSocket) {

this.COSS = clientSocket;

}

@Override

public void run() {

DataInputStream DIS = null;

DataOutputStream DOS = null;

Connection connection = null;

try {

DIS = new DataInputStream(COSS.getInputStream());

int repID = DIS.readInt();

int Sold = DIS.readInt();

if (Sold < 0) {

throw new IllegalArgumentException("Number of laptops sold cannot be negative.");

}

double Price = 90.0;

double sales = Sold \* Price;

String chargeCode = (sales > 20000) ? "CR1" :

(sales > 10000) ? "CR2" : "CR3";

String DN = "jdbc:derby://localhost:1527/Sharaf DG";

String Username = "Sharaf";

String Password = "12345";

connection = DriverManager.getConnection(DN, Username, Password);

PreparedStatement statement = connection.prepareStatement("SELECT ChargeRate FROM ChargeRates WHERE ChargeCode = ?");

statement.setString(1,chargeCode);

ResultSet resultSet = statement.executeQuery();

double Rate = 0.0;

if (resultSet.next()) {

Rate = resultSet.getDouble("ChargeRate");

}

double Value = sales \* (Rate / 100);

DOS = new DataOutputStream(COSS.getOutputStream());

DOS.writeDouble(sales);

DOS.writeDouble(Rate);

DOS.writeDouble(Value);

System.out.printf("Processed SalesRep ID: %d | Profit: %.2f | Rate: %.2f%% | Commission: %.2f%n",

repID, sales, Rate, Value);

} catch (IOException | SQLException | IllegalArgumentException e) {

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

} finally {

try {

if (DIS != null) DIS.close();

if (DOS != null) DOS.close();

if (COSS != null) COSS.close();

if (connection != null) connection.close();

} catch (IOException | SQLException e) {

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

}

}

}

public static void main(String[] args) {

ExecutorService executor = Executors.newFixedThreadPool(10);

try (ServerSocket serverSocket = new ServerSocket(7946)) {

System.out.println("Server is running on port 7946...");

while (true) {

Socket clientSocket = serverSocket.accept();

System.out.println("Client connected.");

executor.execute(new Server(clientSocket));

}

} catch (IOException e) {

System.out.println("Server is runn");

} finally {

executor.shutdown();

}

}

}

Task client:

package mnclient;

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class Client implements Runnable {

@Override

public void run() {

Socket SocketS = null;

DataInputStream DIS = null;

DataOutputStream DOS = null;

Scanner scanner = new Scanner(System.in);

try {

SocketS = new Socket("localhost", 7946);

System.out.println("Connected to the server.");

System.out.print("Enter Sales Representative ID: ");

int repID = scanner.nextInt();

int sSold;

while (true) {

System.out.print("Enter the number of laptops sold: ");

sSold = scanner.nextInt();

if (sSold >= 0) break; // Ensure non-negative input

System.out.println("Invalid input. Please enter a non-negative number.");

}

DOS = new DataOutputStream(SocketS.getOutputStream());

DOS.writeInt(repID);

DOS.writeInt(sSold);

System.out.println("Data sent to the server.");

DIS = new DataInputStream(SocketS.getInputStream());

double salesProfit = DIS.readDouble();

double commissionRate = DIS.readDouble();

double commissionValue = DIS.readDouble();

System.out.println("\n=== Commission Details ===");

System.out.println("Sales Representative ID: " + repID);

System.out.printf("Sales Profit: %.2f OMR%n", salesProfit);

System.out.printf("Commission Rate: %.2f%%%n", commissionRate);

System.out.printf("Commission Value: %.2f OMR%n", commissionValue);

} catch (IOException e) {

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

} finally {

try {

if (DIS != null) DIS.close();

if (DOS != null) DOS.close();

if (SocketS != null) SocketS.close();

System.out.println("Connection closed.");

} catch (IOException e) {

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

}

}

}

public static void main(String[] args) {

Thread clientThread = new Thread(new Client());

clientThread.start();

}

}