**Server Code**

package mnserver;

import java.io.\*;

import java.net.\*;

import java.sql.\*;

public class server implements Runnable {

private final Socket clientSocket;

public server(Socket clientSocket) {

this.clientSocket = clientSocket;

}

@Override

public void run() {

DataInputStream inputStream = null;

DataOutputStream outputStream = null;

Connection connection = null;

try {

inputStream = new DataInputStream(clientSocket.getInputStream());

int repID = inputStream.readInt();

int laptopsSold = inputStream.readInt();

double unitPrice = 90.0;

double salesProfit = laptopsSold \* unitPrice;

String chargeCode;

if (salesProfit > 20000) {

chargeCode = "CR1";

} else if (salesProfit > 10000) {

chargeCode = "CR2";

} else {

chargeCode = "CR3";

}

String dbURL = "jdbc:derby://localhost:1527/SalesCommissionDB";

String username = "Ad";

String password = "Ad"; // Replace with your database password

connection = DriverManager.getConnection(dbURL, username, password);

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

statement.setString(1, chargeCode);

ResultSet resultSet = statement.executeQuery();

double commissionRate = 0.0;

if (resultSet.next()) {

commissionRate = resultSet.getDouble("ChargeRate");

}

double commissionValue = salesProfit \* (commissionRate / 100);

outputStream = new DataOutputStream(clientSocket.getOutputStream());

outputStream.writeDouble(salesProfit);

outputStream.writeDouble(commissionRate);

outputStream.writeDouble(commissionValue);

} catch (IOException | SQLException e) {

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

} finally {

try {

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

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

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

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

} catch (IOException | SQLException e) {

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

}

}

}

public static void main(String[] args) {

try (ServerSocket serverSocket = new ServerSocket(5555)) {

System.out.println("Server is running on port 5555...");

while (true) {

Socket clientSocket = serverSocket.accept();

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

Thread serverThread = new Thread(new server(clientSocket));

serverThread.start();

}

} catch (IOException e) {

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

}

}

}

**Client Code**

package mnclient;

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class Client implements Runnable {

@Override

public void run() {

Socket sN = null;

DataInputStream DIS = null;

DataOutputStream DOS = null;

Scanner scanner = new Scanner(System.in);

try {

sN = new Socket("localhost", 5555);

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;

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

}

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

DOS.writeInt(repID);

DOS.writeInt(sSold);

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

DIS = new DataInputStream(sN.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 (sN != null) sN.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();

}

}

**Task 4:**

import java.util.\*;

class SalesRecord {

private final String OrderNumber;

private final String AccountName;

public SalesRecord(String OrderNumber, String customerName) {

this.OrderNumber = OrderNumber;

this.AccountName = customerName;

}

public String getOrderNumber() {

return OrderNumber;

}

public String getLaptopsSold() {

return AccountName;

}

@Override

public String toString() {

return "ID: " + OrderNumber + ", number of Laptops sold: " + AccountName;

}

}

class SalesRecordManager {

private final LinkedList<SalesRecord> salesRecords = new LinkedList<>();

public void addRecord(SalesRecord record) {

salesRecords.add(record); // Append to the end of the list

System.out.println("Record added successfully.");

}

public void deleteRecord(String billNumber) {

boolean found = false;

Iterator<SalesRecord> iterator = salesRecords.iterator();

while (iterator.hasNext()) {

SalesRecord record = iterator.next();

if (record.getOrderNumber().equalsIgnoreCase("OrderNumber")); {

iterator.remove();

System.out.println("Record deleted successfully.");

found = true;

break;

}

}

if (!found) {

System.out.println("Record not found.");

}

}

public void searchRecord(String OrderNumber) {

for (SalesRecord record : salesRecords) {

if (record.getOrderNumber().equalsIgnoreCase(OrderNumber)) {

System.out.println("Record found: " + record);

return;

}

}

System.out.println("Record not found.");

}

public void displayRecords() {

if (salesRecords.isEmpty()) {

System.out.println("No records to display.");

} else {

System.out.println("Sales Records:");

salesRecords.forEach((record) -> {

System.out.println(record);

});

}

}

}

class MenuDrivenProgram {

public static void main(String[] args) {

SalesRecordManager manager = new SalesRecordManager();

try (Scanner scanner = new Scanner(System.in)) {

int choice;

do {

System.out.println("\nMenu:");

System.out.println("1. Add a Laptops Solds");

System.out.println("2. Display All Laptops Solds");

System.out.println("3. Delete a Laptops Solds");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

scanner.nextLine();

System.out.print("Put the ID number: ");

String IDNumber = scanner.nextLine();

System.out.print("Put number of Laptops sold: ");

String Laptopssold = scanner.nextLine();

SalesRecord record = new SalesRecord(IDNumber, Laptopssold);

manager.addRecord(record);

break;

case 2:

manager.displayRecords();

break;

case 3:

scanner.nextLine();

System.out.print("Enter The ID to delete: ");

String deleteLaptopssold = scanner.nextLine();

manager.deleteRecord(deleteLaptopssold);

break;

case 4:

System.out.println("Goodbye!");

break;

default:

System.out.println("Invalid choice. Please try again.");

}

} while (choice != 4);

}

}

}