# Rajalakshmi Engineering College

Name: mohamed  hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

## 2028_REC_OOPS using Java_Week 5_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

### Section 1 : Coding

1. Problem Statement

You are working as a developer for CityBank, which wants to build a basic account management system.

Each customer at the bank has:

An Account Number (integer)A Customer Name (string)An Initial Balance (double)

The bank allows two types of transactions:

Deposit – increases the balance.Withdrawal – decreases the balance only if enough funds are available.

If the withdrawal amount is greater than the balance, the withdrawal should not happen, and the balance should remain the same.

You are required to implement this system using:

A class with attributes for account details.A constructor to initialize account details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer’s account details after all transactions.

***Input Format***

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the account number (integer).
- The following line contains the customer name (string).
- The next line contains the initial balance (double).
- The next line contains the deposit amount (double).
- The next line contains the withdrawal amount (double).

***Output Format***

For each customer, print the details in the following format:

1. Account Number: <account_number>
2. Customer Name: <customer_name>
3. Final Balance: <final_balance> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
1234
Rahul Sharma
5000
2000
3000

Output: Account Number: 1234
Customer Name: Rahul Sharma

Final Balance: 4000.0

***Answer***

```
import java.util.Scanner;

class BankAccount {
   private int accountNumber;
   private String customerName;
   private double balance;

   // Constructor to initialize account details
   public BankAccount(int accountNumber, String customerName, double
initialBalance) {
      this.accountNumber = accountNumber;
      this.customerName = customerName;
      this.balance = initialBalance;
   }

   // Getter methods
   public int getAccountNumber() {
      return accountNumber;
   }

   public String getCustomerName() {
      return customerName;
   }

   public double getBalance() {
      return balance;
   }

   // Setter methods
   public void setAccountNumber(int accountNumber) {
      this.accountNumber = accountNumber;
   }

   public void setCustomerName(String customerName) {
      this.customerName = customerName;
   }

   public void setBalance(double balance) {
      this.balance = balance;
```

```
    }

    // Method to deposit money
    public void deposit(double amount) {
        if (amount > 0) {
            balance += amount;
        }
    }

    // Method to withdraw money
    public void withdraw(double amount) {
        if (amount > 0 && amount <= balance) {
            balance -= amount;
        }
        // If amount > balance, withdrawal is ignored (balance remains same)
    }

    // Method to display account details
    public void displayAccountDetails() {
        System.out.println("Account Number: " + accountNumber);
        System.out.println("Customer Name: " + customerName);
        System.out.printf("Final Balance: %.1f%n", balance);
        System.out.println(); // Empty line after each customer's details
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read number of customers
        int n = scanner.nextInt();

        // Create array to store bank accounts
        BankAccount[] accounts = new BankAccount[n];

        // Process each customer
        for (int i = 0; i < n; i++) {
            // Read account details
            int accountNumber = scanner.nextInt();
            scanner.nextLine(); // Consume newline after integer input
            String customerName = scanner.nextLine();
```

```
            double initialBalance = scanner.nextDouble();
            double depositAmount = scanner.nextDouble();
            double withdrawalAmount = scanner.nextDouble();

            // Create bank account object
            accounts[i] = new BankAccount(accountNumber, customerName,
initialBalance);

            // Perform deposit
            accounts[i].deposit(depositAmount);

            // Perform withdrawal
            accounts[i].withdraw(withdrawalAmount);
        }

        // Display all customer account details
        for (int i = 0; i < n; i++) {
            accounts[i].displayAccountDetails();
        }

        scanner.close();
    }
}
```

***Status :*** Correct ***Marks : 10/10***

# Rajalakshmi Engineering College

Name: mohamed  hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

## 2028_REC_OOPS using Java_Week 5_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

### Section 1 : Coding

1. Problem Statement

Neha is working as a developer for CityElectricity Board, which wants to build a household electricity billing system.

Each customer’s electricity account has:

A Customer ID (integer)A Customer Name (string)Units Consumed (double)

The electricity bill is calculated based on these rules:

For the first 100 units  5 units charge per unitFor the next 100 units (101–200)  7 units charge per unitFor units above 200  10 units charge per unitIf the total bill exceeds 2000 units, a 5% discount is applied on the final bill.

Neha has been asked to implement this system using:

A class with attributes for customer details.A constructor to initialize customer details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer’s details and final bill amount.

***Input Format***

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Units Consumed (double).

***Output Format***

For each customer, print the details in the following format:

Customer ID: <customer_id>

Customer Name: <customer_name>

Final Bill: <final_bill> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
1001
Ravi Kumar
80
Output: Customer ID: 1001
Customer Name: Ravi Kumar
Final Bill: 400.0

***Answer***

// You are using Java

```
import java.util.Scanner;

class ElectricityAccount {
    private int customerId;
    private String customerName;
    private double unitsConsumed;

    // Constants for bill calculation
    private static final double FIRST_SLAB_RATE = 5.0;    // First 100 units
    private static final double SECOND_SLAB_RATE = 7.0;   // Next 100 units
(101-200)
    private static final double THIRD_SLAB_RATE = 10.0;   // Units above 200
    private static final double FIRST_SLAB_LIMIT = 100.0;
    private static final double SECOND_SLAB_LIMIT = 200.0;
    private static final double DISCOUNT_THRESHOLD = 2000.0; // Bills above
2000
    private static final double DISCOUNT_RATE = 0.05;      // 5% discount

    // Constructor to initialize customer details
    public ElectricityAccount(int customerId, String customerName, double
unitsConsumed) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.unitsConsumed = unitsConsumed;
    }

    // Getter methods
    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getUnitsConsumed() {
        return unitsConsumed;
    }

    // Setter methods
    public void setCustomerId(int customerId) {
        this.customerId = customerId;
```

```
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void setUnitsConsumed(double unitsConsumed) {
        this.unitsConsumed = unitsConsumed;
    }

    // Method to calculate final bill
    public double calculateFinalBill() {
        double totalBill = 0.0;
        double remainingUnits = unitsConsumed;

        // Calculate bill for first 100 units (0-100) at 5 units per unit
        if (remainingUnits > 0) {
            double unitsInFirstSlab = Math.min(remainingUnits, FIRST_SLAB_LIMIT);
            totalBill += unitsInFirstSlab * FIRST_SLAB_RATE;
            remainingUnits -= unitsInFirstSlab;
        }

        // Calculate bill for next 100 units (101-200) at 7 units per unit
        if (remainingUnits > 0) {
            double unitsInSecondSlab = Math.min(remainingUnits,
FIRST_SLAB_LIMIT);
            totalBill += unitsInSecondSlab * SECOND_SLAB_RATE;
            remainingUnits -= unitsInSecondSlab;
        }

        // Calculate bill for remaining units (above 200) at 10 units per unit
        if (remainingUnits > 0) {
            totalBill += remainingUnits * THIRD_SLAB_RATE;
        }

        // Apply 5% discount if total bill exceeds 2000
        if (totalBill > DISCOUNT_THRESHOLD) {
            totalBill = totalBill * (1 - DISCOUNT_RATE);
        }

        return totalBill;
    }
```

```
    // Method to display customer details and final bill
    public void displayCustomerDetails() {
        System.out.println("Customer ID: " + customerId);
        System.out.println("Customer Name: " + customerName);
        System.out.printf("Final Bill: %.1f%n", calculateFinalBill());
        System.out.println(); // Empty line after each customer's details
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read number of customers
        int n = scanner.nextInt();

        // Create array to store electricity accounts
        ElectricityAccount[] accounts = new ElectricityAccount[n];

        // Process each customer
        for (int i = 0; i < n; i++) {
            // Read customer details
            int customerId = scanner.nextInt();
            scanner.nextLine(); // Consume newline after integer input
            String customerName = scanner.nextLine();
            double unitsConsumed = scanner.nextDouble();

            // Create electricity account object
            accounts[i] = new ElectricityAccount(customerId, customerName,
unitsConsumed);
        }

        // Display all customer details and final bills
        for (int i = 0; i < n; i++) {
            accounts[i].displayCustomerDetails();
        }

        scanner.close();
    }
}
```

***Status :*** Correct ***Marks : 10/10***

# Rajalakshmi Engineering College

Name: mohamed  hafiz
Email: 241501115@rajalakshmi.edu.in
Roll no:
Phone: 9342701083
Branch: REC
Department: AI & ML - Section 3
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

### 2028_REC_OOPS using Java_Week 5_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

### Section 1 : Coding

1. Problem Statement

You are working as a developer for CityCab, a taxi service company that wants to build a ride fare management system.

Each customer booking has:

A Booking ID (integer)A Customer Name (string)A Distance Travelled in km (double)

The fare calculation rules are:

Base Fare = 50 units (flat charge for every ride).Per km charge = 10 units/km.If the distance is greater than 20 km, a 10% discount is applied on the total fare.

You are required to implement this system using:

A class with attributes for booking details.A constructor to initialize booking details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customer rides.

Finally, display each booking’s details and final fare.

***Input Format***

The first line of input contains an integer N, representing the number of bookings.

For each booking:

- The next line contains the booking ID (integer).
- The following line contains the customer's name (string).
- The next line contains the distance travelled (double).

***Output Format***

For each booking, print the details in the following format:

1. Booking ID: <booking_id>
2. Customer Name: <customer_name>
3. Final Fare: <final_fare> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
1234
Rahul Sharma
15

Output: Booking ID: 1234
Customer Name: Rahul Sharma
Final Fare: 200.0

***Answer***

```
import java.util.Scanner;

class RideBooking {
```

```
    private int bookingId;
    private String customerName;
    private double distanceTravelled;

    // Constants for fare calculation
    private static final int BASE_FARE = 50;
    private static final int PER_KM_CHARGE = 10;
    private static final double DISCOUNT_RATE = 0.10; // 10% discount
    private static final double DISCOUNT_THRESHOLD = 20.0; // More than 20 km

    // Constructor to initialize booking details
    public RideBooking(int bookingId, String customerName, double
distanceTravelled) {
        this.bookingId = bookingId;
        this.customerName = customerName;
        this.distanceTravelled = distanceTravelled;
    }

    // Getter methods
    public int getBookingId() {
        return bookingId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getDistanceTravelled() {
        return distanceTravelled;
    }

    // Setter methods
    public void setBookingId(int bookingId) {
        this.bookingId = bookingId;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public void setDistanceTravelled(double distanceTravelled) {
        this.distanceTravelled = distanceTravelled;
```

```
    }

    // Method to calculate final fare
    public double calculateFinalFare() {
        // Calculate base fare = Base Fare + (Distance * Per km charge)
        double totalFare = BASE_FARE + (distanceTravelled * PER_KM_CHARGE);

        // Apply 10% discount if distance is greater than 20 km
        if (distanceTravelled > DISCOUNT_THRESHOLD) {
            totalFare = totalFare * (1 - DISCOUNT_RATE);
        }

        return totalFare;
    }

    // Method to display booking details and final fare
    public void displayBookingDetails() {
        System.out.println("Booking ID: " + bookingId);
        System.out.println("Customer Name: " + customerName);
        System.out.printf("Final Fare: %.1f%n", calculateFinalFare());
        System.out.println(); // Empty line after each booking's details
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Read number of bookings
        int n = scanner.nextInt();

        // Create array to store ride bookings
        RideBooking[] bookings = new RideBooking[n];

        // Process each booking
        for (int i = 0; i < n; i++) {
            // Read booking details
            int bookingId = scanner.nextInt();
            scanner.nextLine(); // Consume newline after integer input
            String customerName = scanner.nextLine();
            double distanceTravelled = scanner.nextDouble();
```

```
            // Create ride booking object
            bookings[i] = new RideBooking(bookingId, customerName,
distanceTravelled);
        }

        // Display all booking details and final fares
        for (int i = 0; i < n; i++) {
            bookings[i].displayBookingDetails();
        }

        scanner.close();
    }
}
```

***Status :*** Correct ***Marks : 10/10***