

# Rajalakshmi Engineering College

Name: Mithran VT  
Email: 240701312@rajalakshmi.edu.in  
Roll no:  
Phone: 9952919350  
Branch: REC  
Department: CSE - Section 8  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### REC\_2028\_OOPS using Java\_Week 5\_CY

Attempt : 1  
Total Mark : 40  
Marks Obtained : 40

#### **Section 1 : Coding**

##### **1. Problem Statement**

Arjun is working as a developer for CityWater Supply Board, which wants to build a household water billing system.

Each household's water account has:

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

The water bill is calculated based on these rules:

For the first 500 liters    2 per liter  
For the next 500 liters (501–1000)    3 per liter  
For liters above 1000    5 per liter  
If the total bill exceeds 3000, a 10% discount is applied on the final bill.

Arjun 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 Liters 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

300

Output: Customer ID: 1001

Customer Name: Ravi Kumar

Final Bill: 600.0

#### ***Answer***

```
// You are using Java
```

```
import java.util.*;

class Customer {
    private int customerId;
    private String customerName;
    private double litersConsumed;

    public Customer(int customerId, String customerName, double
litersConsumed) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.litersConsumed = litersConsumed;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getLitersConsumed() {
        return litersConsumed;
    }

    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }

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

    public void setLitersConsumed(double litersConsumed) {
        this.litersConsumed = litersConsumed;
    }

    public double calculateBill() {
        double bill = 0.0;
```

```

        double liters = litersConsumed;

        if (liters <= 500) {
            bill = liters * 2;
        } else if (liters <= 1000) {
            bill = (500 * 2) + ((liters - 500) * 3);
        } else {
            bill = (500 * 2) + (500 * 3) + ((liters - 1000) * 5);
        }

        // Apply discount if bill > 3000
        if (bill > 3000) {
            bill = bill - (bill * 0.10);
        }

        return bill;
    }

    public void displayDetails() {
        System.out.println("Customer ID: " + customerId);
        System.out.println("Customer Name: " + customerName);
        System.out.printf("Final Bill: %.1f%n", calculateBill());
    }
}

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

        int N = Integer.parseInt(sc.nextLine());
        List<Customer> customers = new ArrayList<>();

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double liters = Double.parseDouble(sc.nextLine());

            Customer customer = new Customer(id, name, liters);
            customers.add(customer);
        }

        for (Customer c : customers) {

```

```
        c.displayDetails();
    }
}
}
```

**Status : Correct**

**Marks : 10/10**

## 2. Problem Statement

You are working as a developer for CityMobile, which wants to build a basic mobile data usage management system.

Each customer has:

A Customer ID (integer)A Customer Name (string)An Initial Data Balance (in GB, double)

The company allows two types of operations:

Recharge – increases the data balance. Usage – decreases the data balance only if enough data is available.

If the usage amount is greater than the available data balance, the usage should not happen, and the balance should remain the same.

You are required 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 after all operations.

### ***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 Initial Data Balance (double).
- The next line contains the Recharge Amount in GB (double).
- The next line contains the Usage Amount in GB (double).

#### ***Output Format***

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

Customer ID: <customer\_id>

Customer Name: <customer\_name>

Final Data Balance: <final\_data\_balance> GB (The final balance must be rounded to one decimal place.)

Refer to the sample output for formatting specifications.

#### ***Sample Test Case***

Input: 1

1234

Ravi Kumar

5.0

2.0

3.0

Output: Customer ID: 1234

Customer Name: Ravi Kumar

Final Data Balance: 4.0 GB

#### ***Answer***

```
import java.util.*;  
  
class Customer {  
    private int customerId;  
    private String customerName;  
    private double dataBalance;  
  
    // Constructor  
    public Customer(int customerId, String customerName, double dataBalance) {  
        this.customerId = customerId;
```

```
    this.customerName = customerName;
    this.dataBalance = dataBalance;
}

// Setters
public void setCustomerId(int customerId) {
    this.customerId = customerId;
}

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

public void setDataBalance(double dataBalance) {
    this.dataBalance = dataBalance;
}

// Getters
public int getCustomerId() {
    return customerId;
}

public String getCustomerName() {
    return customerName;
}

public double getDataBalance() {
    return dataBalance;
}

// Method to recharge
public void recharge(double amount) {
    if (amount >= 0)
        this.dataBalance += amount;
}

// Method to use data
public void useData(double amount) {
    if (amount <= this.dataBalance)
        this.dataBalance -= amount;
    // else usage ignored
}
```

```

}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());

        Customer[] customers = new Customer[n];

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double balance = Double.parseDouble(sc.nextLine());
            double recharge = Double.parseDouble(sc.nextLine());
            double usage = Double.parseDouble(sc.nextLine());

            // Create customer object
            Customer c = new Customer(id, name, balance);

            // Perform operations
            c.recharge(recharge);
            c.useData(usage);

            customers[i] = c;
        }

        sc.close();

        // Display output
        for (Customer c : customers) {
            System.out.println("Customer ID: " + c.getCustomerId());
            System.out.println("Customer Name: " + c.getCustomerName());
            System.out.printf("Final Data Balance: %.1f GB\n", c.getDataBalance());
        }
    }
}

```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Anjali is working as a developer for the City Basketball Association, which wants to build a system to track and find the top scorer among basketball players.

Each player's record has:

Player ID (integer) Player Name (string) An array of points scored in 5 matches (integers)

The system must calculate:

The total score of each player (sum of all match points). Identify the highest scorer among all players. If two or more players have the same total score, the one with the lower Player ID is considered the top scorer.

Anjali has been asked to implement this system using:

A class with attributes for player details. A constructor to initialize player details. Getter and Setter methods to retrieve and update player details if required. A method to calculate the total score. Objects of the class to represent players.

Finally, display each player's details and announce the Top Scorer.

#### ***Input Format***

The first line of input contains an integer N (number of players).

For each player:

- The next line contains the Player ID (integer).
- The following line contains the Player Name (string).
- The next line contains 5 integers separated by spaces (points scored in 5 matches).

#### ***Output Format***

For each player the output prints the following details:

- Player ID: <player\_id>
- Player Name: <player\_name>
- Total Score: <total\_score>

Finally, print "Top Scorer: <player\_name> with <total\_score> points"

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 1  
1001  
Ravi Kumar  
10 20 30 40 50

Output: Player ID: 1001  
Player Name: Ravi Kumar  
Total Score: 150  
Top Scorer: Ravi Kumar with 150 points

### ***Answer***

```
// You are using Java
import java.util.*;

class Player {
    private int playerId;
    private String playerName;
    private int[] points = new int[5];

    // Constructor
    public Player(int playerId, String playerName, int[] points) {
        this.playerId = playerId;
        this.playerName = playerName;
        this.points = points;
    }

    // Getters
    public int getPlayerId() {
        return playerId;
    }

    public String getPlayerName() {
```

```
        return playerName;
    }

    public int[] getPoints() {
        return points;
    }

    // Method to calculate total score
    public int getTotalScore() {
        int total = 0;
        for (int p : points) {
            total += p;
        }
        return total;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        Player[] players = new Player[n];

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String[] pointStr = sc.nextLine().split(" ");
            int[] points = new int[5];
            for (int j = 0; j < 5; j++) {
                points[j] = Integer.parseInt(pointStr[j]);
            }

            players[i] = new Player(id, name, points);
        }

        sc.close();

        // Find top scorer
        Player topScorer = players[0];
        for (Player p : players) {
            if (p.getTotalScore() > topScorer.getTotalScore()) {
                topScorer = p;
            }
        }
    }
}
```

```

        } else if (p.getTotalScore() == topScorer.getTotalScore() &&
p.getPlayerId() < topScorer.getPlayerId()) {
            topScorer = p;
        }
    }

    // Display all players
    for (Player p : players) {
        System.out.println("Player ID: " + p.getPlayerId());
        System.out.println("Player Name: " + p.getPlayerName());
        System.out.println("Total Score: " + p.getTotalScore());
    }

    // Display top scorer
    System.out.println("Top Scorer: " + topScorer.getPlayerName() + " with " +
topScorer.getTotalScore() + " points");
}
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Anjali is now working as a developer for the City Marathon Association, which wants to build a system to track and find the fastest runner among marathon participants.

Each runner's record has:

Runner ID (integer)  
 Runner Name (string)  
 An array of times (in minutes)  
 taken in 5 marathon events (integers)

The system must calculate:

The average time of each runner (sum of all times / 5). Identify the fastest runner (the one with the lowest average time). If two or more runners have the same average time, the one with the lower Runner ID is considered the fastest runner.

Anjali has been asked to implement this system using:

A class with attributes for runner details. A constructor to initialize runner details. Getter and Setter methods to retrieve and update runner details if required. A method to calculate the average time. Objects of the class to represent runners.

Finally, display each runner's details and announce the Fastest Runner.

### ***Input Format***

The first line of input contains an integer N (number of runners).

For each runner:

- The next line contains the Runner ID (integer).
- The following line contains the Runner Name (string).
- The next line contains 5 integers separated by spaces (times in minutes for 5 marathon events).

### ***Output Format***

For each runner the output prints the following details:

- Runner ID: <runner\_id>
- Runner Name: <runner\_name>
- Average Time: <average\_time>

Finally, print "Fastest Runner: <runner\_name> with <average\_time> minutes"

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 1

1001

Ravi Kumar

240 250 245 255 260

Output: Runner ID: 1001

Runner Name: Ravi Kumar

Average Time: 250  
Fastest Runner: Ravi Kumar with 250 minutes

### **Answer**

```
// You are using Java
import java.util.*;

class Runner {
    private int runnerId;
    private String runnerName;
    private int[] times = new int[5];

    // Constructor
    public Runner(int runnerId, String runnerName, int[] times) {
        this.runnerId = runnerId;
        this.runnerName = runnerName;
        this.times = times;
    }

    // Getters
    public int getRunnerId() {
        return runnerId;
    }

    public String getRunnerName() {
        return runnerName;
    }

    public int[] getTimes() {
        return times;
    }

    // Method to calculate average time
    public int getAverageTime() {
        int total = 0;
        for (int t : times) {
            total += t;
        }
        return total / 5;
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        Runner[] runners = new Runner[n];

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String[] timeStr = sc.nextLine().split(" ");
            int[] times = new int[5];
            for (int j = 0; j < 5; j++) {
                times[j] = Integer.parseInt(timeStr[j]);
            }

            runners[i] = new Runner(id, name, times);
        }

        sc.close();

        // Find fastest runner (lowest average time, tie  lower ID)
        Runner fastest = runners[0];
        for (Runner r : runners) {
            if (r.getAverageTime() < fastest.getAverageTime()) {
                fastest = r;
            } else if (r.getAverageTime() == fastest.getAverageTime() &&
r.getRunnerId() < fastest.getRunnerId()) {
                fastest = r;
            }
        }

        // Display all runners' details
        for (Runner r : runners) {
            System.out.println("Runner ID: " + r.getRunnerId());
            System.out.println("Runner Name: " + r.getRunnerName());
            System.out.println("Average Time: " + r.getAverageTime());
        }

        // Display fastest runner
        System.out.println("Fastest Runner: " + fastest.getRunnerName() + " with " +
fastest.getAverageTime() + " minutes");
    }
}
```

}

**Status :** Correct

**Marks :** 10/10