Rajalakshmi Engineering College

Name: Prajeet V
Email: 240701389@rajalakshmi.edu.in
Roll no: 240701389
Phone: 9363389322
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

Meera is working as a developer for CityGas Supply Board, which wants to build a household gas billing system.

Each household's gas account has:

A Customer ID (integer)A Customer Name (string)Units Consumed in cubic meters (double)

The gas bill is calculated based on these rules:

For the first 50 units 4 per unitFor the next 100 units (51–150) 6 per unitFor units above 150 8 per unitIf the total bill exceeds 2000, a 15% discount is applied on the final bill.

Meera 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> (The final bill must be rounded to one decimal place.)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
30
Output: Customer ID: 1001
Customer Name: Ravi Kumar
Final Bill: 120.0

Answer

// You are using Java

```
import java.util.*;

class Customer {
    private int customerId;
    private String customerName;
    private double units;
    private double finalBill;

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

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getFinalBill() {
        return finalBill;
    }

    private void calculateFinalBill() {
        double bill = 0;
        if (units <= 50) {
            bill = units * 4;
        } else if (units <= 150) {
            bill = (50 * 4) + ((units - 50) * 6);
        } else {
            bill = (50 * 4) + (100 * 6) + ((units - 150) * 8);
        }
        if (bill > 2000) {
            bill -= bill * 0.15;
        }
        finalBill = bill;
    }
}
```

```
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 units = Double.parseDouble(sc.nextLine());
            customers.add(new Customer(id, name, units));
        }
        for (Customer c : customers) {
            System.out.println("Customer ID: " + c.getCustomerId());
            System.out.println("Customer Name: " + c.getCustomerName());
            System.out.printf("Final Bill: %.1f\n", c.getFinalBill());
        }
    }
}
```

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

2. 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;
    private int averageTime;

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

    public int getRunnerId() {
        return runnerId;
    }

    public String getRunnerName() {
        return runnerName;
    }

    public int getAverageTime() {
        return averageTime;
    }

    private void calculateAverageTime() {
        int sum = 0;
        for (int t : times) {
            sum += t;
        }
        averageTime = sum / times.length;
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Runner> runners = new ArrayList<>();
        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.add(new Runner(id, name, times));
        }
        Runner fastest = runners.get(0);
        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());
            if (r.getAverageTime() < fastest.getAverageTime() ||
                (r.getAverageTime() == fastest.getAverageTime() && r.getRunnerId() <
fastest.getRunnerId())) {
                fastest = r;
            }
        }
        System.out.println("Fastest Runner: " + fastest.getRunnerName() + " with " +
fastest.getAverageTime() + " minutes");
    }
}
```

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

3. 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

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

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

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

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getDataBalance() {
        return dataBalance;
    }

    public void recharge(double amount) {
```

```
        if (amount > 0) {
            dataBalance += amount;
        }
    }

    public void useData(double amount) {
        if (amount <= dataBalance) {
            dataBalance -= amount;
        }
    }
}

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 initialBalance = Double.parseDouble(sc.nextLine());
            double rechargeAmount = Double.parseDouble(sc.nextLine());
            double usageAmount = Double.parseDouble(sc.nextLine());

            Customer c = new Customer(id, name, initialBalance);
            c.recharge(rechargeAmount);
            c.useData(usageAmount);

            customers.add(c);
        }

        for (Customer c : customers) {
            System.out.printf("Customer ID: %d\nCustomer Name: %s\nFinal Data
Balance: %.1f GB\n",
                    c.getCustomerId(), c.getCustomerName(), c.getDataBalance());
        }
    }
}
```

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

4. 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;

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

    public int getPlayerId() {
        return playerId;
    }

    public String getPlayerName() {
```

```
        return playerName;
    }

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

    public int getTotalScore() {
        int sum = 0;
        for (int p : points) {
            sum += p;
        }
        return sum;
    }
}

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

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

            players.add(new Player(id, name, points));
        }

        Player topScorer = players.get(0);

        for (Player p : players) {
            int total = p.getTotalScore();
            System.out.println("Player ID: " + p.getPlayerId());
            System.out.println("Player Name: " + p.getPlayerName());
            System.out.println("Total Score: " + total);
```

```
            if (total > topScorer.getTotalScore() ||
                (total == topScorer.getTotalScore() && p.getPlayerId() <
topScorer.getPlayerId())) {
                topScorer = p;
            }
        }

        System.out.println("Top Scorer: " + topScorer.getPlayerName() +
                           " with " + topScorer.getTotalScore() + " points");
    }
}
```

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