Rajalakshmi Engineering College

Name: Mohit Jha
Email: 241901056@rajalakshmi.edu.in
Roll no:
Phone: 9445934493
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

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

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

```
import java.util.*;

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

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

    public int getRunnerId() {
        return runnerId;
    }

    public String getRunnerName() {
        return runnerName;
    }

    public double getAverageTime() {
        int sum = 0;
        for (int t : times) {
            sum += t;
        }
        return sum / 5.0;
    }
}

class MarathonTracker {
    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();
            int[] times = new int[5];
            String[] parts = sc.nextLine().split(" ");
            for (int j = 0; j < 5; j++) {
                times[j] = Integer.parseInt(parts[j]);
            }
            runners[i] = new Runner(id, name, times);
        }

        Runner fastest = runners[0];
        for (Runner r : runners) {
            System.out.println("Runner ID: " + r.getRunnerId());
            System.out.println("Runner Name: " + r.getRunnerName());
            System.out.println("Average Time: " + (int) 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 " +
(int) fastest.getAverageTime() + " minutes");
    }
}
```

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.Scanner;

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 void recharge(double amount) {
        dataBalance += amount;
    }

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

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
```

```
    }

    public double getDataBalance() {
        return dataBalance;
    }
}

class CityMobile {
    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 initialBalance = Double.parseDouble(sc.nextLine());
            double recharge = Double.parseDouble(sc.nextLine());
            double usage = Double.parseDouble(sc.nextLine());
            customers[i] = new Customer(id, name, initialBalance);
            customers[i].recharge(recharge);
            customers[i].useData(usage);
        }
        for (Customer c : customers) {
            System.out.printf("Customer ID: %d%n", c.getCustomerId());
            System.out.printf("Customer Name: %s%n", c.getCustomerName());
            System.out.printf("Final Data Balance: %.1f GB%n", c.getDataBalance());
        }
    }
}
```

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

3. 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 literFor the next 500 liters (501–1000) 3 per literFor liters above 1000 5 per literIf 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

```
import java.util.Scanner;

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

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

   public String getCustomerName() {
      return customerName;
   }
```

```
    public double getLitersConsumed() {
        return litersConsumed;
    }

    public double calculateBill() {
        double bill = 0;
        double units = litersConsumed;
        if (units <= 500) {
            bill = units * 2;
        } else if (units <= 1000) {
            bill = 500 * 2 + (units - 500) * 3;
        } else {
            bill = 500 * 2 + 500 * 3 + (units - 1000) * 5;
        }
        if (bill > 3000) {
            bill = bill - (bill * 0.1);
        }
        return bill;
    }
}
class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double liters = Double.parseDouble(sc.nextLine());
            WaterCustomer customer = new WaterCustomer(id, name, liters);
            System.out.println("Customer ID: " + customer.getCustomerId());
            System.out.println("Customer Name: " + customer.getCustomerName());
            System.out.printf("Final Bill: %.1f\n", customer.calculateBill());
        }
        sc.close();
    }
}
```

Status : Correct **_Marks : 10/10_**

4. 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

```
import java.util.Scanner;

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

   public GasCustomer(int customerId, String customerName, double
unitsConsumed) {
      this.customerId = customerId;
      this.customerName = customerName;
      this.unitsConsumed = unitsConsumed;
   }

   public int getCustomerId() {
      return customerId;
   }

   public String getCustomerName() {
      return customerName;
   }

   public double getUnitsConsumed() {
      return unitsConsumed;
   }

   public double calculateBill() {
```

```
        double bill = 0;
        if (unitsConsumed <= 50) {
            bill = unitsConsumed * 4;
        } else if (unitsConsumed <= 150) {
            bill = 50 * 4 + (unitsConsumed - 50) * 6;
        } else {
            bill = 50 * 4 + 100 * 6 + (unitsConsumed - 150) * 8;
        }
        if (bill > 2000) {
            bill = bill - (bill * 0.15);
        }
        return bill;
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        GasCustomer[] customers = new GasCustomer[n];
        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double units = Double.parseDouble(sc.nextLine());
            customers[i] = new GasCustomer(id, name, units);
        }
        for (int i = 0; i < n; i++) {
            System.out.println("Customer ID: " + customers[i].getCustomerId());
            System.out.println("Customer Name: " +
customers[i].getCustomerName());
            System.out.println("Final Bill: " + String.format("%.1f",
customers[i].calculateBill()));
        }
    }
}
```

Status : Correct *Marks : 10/10*