

Rajalakshmi Engineering College

Name: MRIDHULA DEVI M
Email: 240701337@rajalakshmi.edu.in
Roll no:
Phone: 9840329629
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

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;

    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 int getAverageTime() {
        int sum = 0;
        for (int time : times) {
            sum += time;
        }
        return sum / 5;
    }
}

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

```

int n = Integer.parseInt(sc.nextLine().trim());
List<Runner> runners = new ArrayList<>();

for (int i = 0; i < n; i++) {
    int runnerId = Integer.parseInt(sc.nextLine().trim());
    String runnerName = sc.nextLine().trim();
    int[] times = new int[5];
    String[] timeInputs = sc.nextLine().trim().split(" ");
    for (int j = 0; j < 5; j++) {
        times[j] = Integer.parseInt(timeInputs[j]);
    }
    runners.add(new Runner(runnerId, runnerName, times));
}

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

Runner fastest = runners.get(0);
for (int i = 1; i < runners.size(); i++) {
    Runner current = runners.get(i);
    if (current.getAverageTime() < fastest.getAverageTime()) {
        fastest = current;
    } else if (current.getAverageTime() == fastest.getAverageTime()) {
        if (current.getRunnerId() < fastest.getRunnerId()) {
            fastest = current;
        }
    }
}

System.out.println("Fastest Runner: " + fastest.getRunnerName()
    + " with " + fastest.getAverageTime() + " minutes");

sc.close();
}
}

```

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

```
// You are using Java
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) {
        if (amount >= 0) {
            this.dataBalance += amount;
        }
    }
}
```

```
}

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

public int getCustomerId() {
    return customerId;
}

public String getCustomerName() {
    return customerName;
}

public double getFinalDataBalance() {
    return Math.round(dataBalance * 10.0) / 10.0;
}
}

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

        int n = Integer.parseInt(sc.nextLine().trim());
        Customer[] customers = new Customer[n];

        for (int i = 0; i < n; i++) {
            int customerId = Integer.parseInt(sc.nextLine().trim());
            String customerName = sc.nextLine().trim();
            double initialBalance = Double.parseDouble(sc.nextLine().trim());
            double rechargeAmount = Double.parseDouble(sc.nextLine().trim());
            double usageAmount = Double.parseDouble(sc.nextLine().trim());

            Customer customer = new Customer(customerId, customerName,
initialBalance);
            customer.recharge(rechargeAmount);
            customer.useData(usageAmount);

            customers[i] = customer;
        }
    }
}
```

```

        for (Customer c : customers) {
            System.out.println("Customer ID: " + c.getCustomerId()
                + " Customer Name: " + c.getCustomerName()
                + " Final Data Balance: " + c.getFinalDataBalance() + " GB");
        }

        sc.close();
    }
}

```

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 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.Scanner;
```

```
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 getFinalBill() {
        double bill = 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);
        }

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

        return Math.round(bill * 10.0) / 10.0;
    }
}

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

        int n = Integer.parseInt(sc.nextLine().trim());
        Customer[] customers = new Customer[n];

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine().trim());
            String name = sc.nextLine().trim();
```

```

        double liters = Double.parseDouble(sc.nextLine().trim());
        customers[i] = new Customer(id, name, liters);
    }

    for (Customer c : customers) {
        System.out.println("Customer ID: " + c.getCustomerId()
            + " Customer Name: " + c.getCustomerName()
            + " Final Bill: " + c.getFinalBill());
    }

    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 unit
For the next 100 units (51–150) 6 per unit
For units above 150 8 per unit
If 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.Scanner;
```

```
class Customer {  
    private int customerId;  
    private String customerName;  
    private double unitsConsumed;
```

```
public Customer(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 getFinalBill() {
    double bill = 0;
    double units = unitsConsumed;

    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 - (bill * 0.15);
    }

    return Math.round(bill * 10.0) / 10.0;
}

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

        int n = Integer.parseInt(sc.nextLine().trim());
        Customer[] customers = new Customer[n];
```

```
for (int i = 0; i < n; i++) {  
    int id = Integer.parseInt(sc.nextLine().trim());  
    String name = sc.nextLine().trim();  
    double units = Double.parseDouble(sc.nextLine().trim());  
  
    customers[i] = new Customer(id, name, units);  
}  
  
for (Customer c : customers) {  
    System.out.println("Customer ID: " + c.getCustomerId()  
        + " Customer Name: " + c.getCustomerName()  
        + " Final Bill: " + c.getFinalBill());  
}  
  
sc.close();  
}  
}
```

Status : Correct

Marks : 10/10