Rajalakshmi Engineering College

Name: Nivedithaa S
Email: 241901076@rajalakshmi.edu.in
Roll no: 241901076
Phone: 6374185608
Branch: REC
Department: CSE (CS) - Section 2
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 11

Section 1 : MCQ

1. What will be the output of the following code?

```
class Alpha {
    void greet(String name) {
        System.out.println("Hello " + name);
    }
}

public class Main {
    public static void main(String[] args) {
        Alpha obj = new Alpha();
        obj.greet("Anu");
    }
}
```

Answer

Hello Anu

Status : Correct ***Marks : 1/1***

2. What will be the output of the following code?

```
class Person {
   int age = 18;
}

public class Main {
   public static void main(String[] args) {
      Person p = new Person();
      p.age += 2;
      System.out.println("Age: " + p.age);
   }
}
```

Answer

Age: 20

Status : Correct ***Marks : 1/1***

3. What will be the output of the following code?

```
class A {
   int y = 30;
}

public class Main {
   public static void main(String[] args) {
      A a1 = new A();
      A a2 = new A();
      a1.y = 50;
      System.out.println(a2.y);
   }
}
```

Answer

50

Status : Wrong **Marks : 0/1**

4. What will be the output of the following code?

```
class Ball {
   int size = 11;
}

class Game {
   public static void main(String[] args) {
      Ball b1 = new Ball();
      Ball b2 = new Ball();
      b2.size = 10;
      System.out.println(b1.size);
   }
}
```

Answer

10

Status : Wrong **Marks : 0/1**

5. What will be the output of the following code?

```
class Test {
   private int value;
   Test(int value) {
      this.value = value;
   }
   public int getValue() {
      return value;
   }
}
public class Main {
   public static void main(String[] args) {
      Test obj = new Test(10);
      System.out.println(obj.value);
```

```
    }
}
```

Answer

Compile-time error

Status : Correct ***Marks : 1/1***

6. What is the output of the following code?

```
class Box {
    int height;
    Box(int height) {
        this.height = height;
    }
    void modifyHeight(Box b) {
        b.height += 10;
    }
}
public class Main {
    public static void main(String[] args) {
        Box b1 = new Box(20);
        b1.modifyHeight(b1);
        System.out.println(b1.height);
    }
}
```

Answer

30

Status : Correct ***Marks : 1/1***

7. What will be the output of the following code?

```
class Sample {
    int x = 10;

    void display() {
        System.out.println("x = " + x);
```

```
    }

    public static void main(String[] args) {
        Sample s = new Sample();
        s.display();
    }
}
```

Answer

x = 10

Status : Correct ***Marks : 1/1***

8. What will be the output of the following code?

```
class Demo {
    void printMessage() {
        System.out.println("Hello from Demo");
    }
}

public class Main {
    public static void main(String[] args) {
        Demo d = new Demo();
        d.printMessage();
    }
}
```

Answer

Hello from Demo

Status : Correct ***Marks : 1/1***

9. What will be the output of the following code?

```
class A {
    int x = 50;
}
```

```
public class Main {
    public static void main(String[] args) {
        A obj1 = new A();
        A obj2 = obj1;
        obj2.x = 100;
        System.out.println(obj1.x);
    }
}
```

Answer

100

Status : Correct ***Marks : 1/1***

10. What will be the output of the following code?

```
class A {
    int val = 20;
}

public class Main {
    public static void main(String[] args) {
        A obj1 = new A();
        A obj2 = obj1;
        obj2.val += 5;
        System.out.println(obj1.val);
    }
}
```

Answer

5

Status : Wrong ***Marks : 0/1***

11. What will be the output of the following code?

```
class Box {
    int volume(int l, int b, int h) {
        return l * b * h;
```

```
    }
}

public class Main {
    public static void main(String[] args) {
        Box b = new Box();
        System.out.println(b.volume(2, 3, 4));
    }
}
```

Answer

24

Status : Correct ***Marks : 1/1***

12. What will be the output of the following code?

```
class Box {
    int length = 5;
    int width = 4;

    int area() {
        return length * width;
    }

    public static void main(String[] args) {
        Box b = new Box();
        System.out.println("Area = " + b.area());
    }
}
```

Answer

Compilation error

Status : Wrong ***Marks : 0/1***

13. What will be the output of the following code?

```
class MathUtils {
```

```
    int add(int x) {
        return x + x;
    }
}

public class Main {
    public static void main(String[] args) {
        MathUtils m = new MathUtils();
        System.out.println(m.add(5));
    }
}
```

Answer

10

Status : Correct ***Marks : 1/1***

14. What will be the output of the following code?

```
class A {
    int p = 5;
    int q = 2;
}

class Main {
    public static void main(String[] args) {
        A obj = new A();
        System.out.println(obj.p + obj.q);
    }
}
```

Answer

7

Status : Correct ***Marks : 1/1***

15. What will be the output of the following code?

```
class Person {
```

```
    String name;
    void setName(String n) {
        name = n;
    }
    void printName() {
        System.out.println(name);
    }
}

class Test {
    public static void main(String[] args) {
        Person p = new Person();
        p.printName();
    }
}
```

Answer

null

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Nivedithaa S
Email: 241901076@rajalakshmi.edu.in
Roll no: 241901076
Phone: 6374185608
Branch: REC
Department: CSE (CS) - Section 2
Batch: 2028
Degree: B.E - CSE (CS)

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 Account{
   private int accountNumber;
   private String customerName;
   private double balance;
   public Account(int accountNumber, String customerName, double balance){
      this.accountNumber = accountNumber;
      this.customerName = customerName;
      this.balance = balance;
   }
   public int getAccountNumber(){
      return accountNumber;
   }
   public String getCustomerName(){
      return customerName;
   }
   public double getBalance(){
      return balance;
   }
   public void setAccountNumber(int accountNumber){
      this.accountNumber = accountNumber;
   }
   public void setCustomerName(String customerName){
      this.customerName = customerName;
   }
   public void setBalance(double balance){
      this.balance = balance;
   }
   public void deposit(double amount){
      if (amount >= 0){
         balance += amount;
      }
   }
   public void withdraw(double amount){
      if (amount <= balance){
         balance -= amount;
      }
   }
}
```

```
public 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 accountNumber = Integer.parseInt(sc.nextLine());
            String customerName = sc.nextLine();
            double initialBalance = Double.parseDouble(sc.nextLine());
            double depositAmount = Double.parseDouble(sc.nextLine());
            double withdrawAmount = Double.parseDouble(sc.nextLine());
            Account customer = new Account(accountNumber, customerName,
initialBalance);
            customer.deposit(depositAmount);
            customer.withdraw(withdrawAmount);
            System.out.println("Account Number: " + customer.getAccountNumber());
            System.out.println("Customer Name: " + customer.getCustomerName());
            System.out.println("Final Balance: " + String.format("%.1f",
customer.getBalance()));
        }
        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Nivedithaa S
Email: 241901076@rajalakshmi.edu.in
Roll no: 241901076
Phone: 6374185608
Branch: REC
Department: CSE (CS) - Section 2
Batch: 2028
Degree: B.E - CSE (CS)

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

```
import java.util.Scanner;
```

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

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

    public double calculateBill() {
        double bill = 0.0;
        double remaining = unitsConsumed;

        if (remaining > 200) {
            bill += (remaining - 200) * 10;
            remaining = 200;
        }
        if (remaining > 100) {
            bill += (remaining - 100) * 7;
            remaining = 100;
        }
        bill += remaining * 5;

        if (bill > 2000) {
            bill *= 0.95;
        }
        return Math.round(bill * 10) / 10.0;
    }

    public int getCustomerId() { return customerId; }
    public String getCustomerName() { return customerName; }
}

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().trim();
            double units = Double.parseDouble(sc.nextLine());
            customers[i] = 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.calculateBill());
        }
    }
}
```

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

Rajalakshmi Engineering College

Name: Nivedithaa S
Email: 241901076@rajalakshmi.edu.in
Roll no: 241901076
Phone: 6374185608
Branch: REC
Department: CSE (CS) - Section 2
Batch: 2028
Degree: B.E - CSE (CS)

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 Booking {
```

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

    // Constructor to initialize booking details
    public Booking(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;
    }

    // Method to calculate final fare
    public double calculateFare() {
        double baseFare = 50.0;
        double perKmCharge = 10.0;
        double totalFare = baseFare + (perKmCharge * distanceTravelled);

        // Apply discount if distance is greater than 20 km
        if (distanceTravelled > 20) {
            totalFare *= 0.9; // Applying 10% discount
        }

        return Math.round(totalFare * 10.0) / 10.0; // Round to one decimal place
    }
}

class CityCabFareManagement {
    public static void main(String[] args) {
```

```
        Scanner scanner = new Scanner(System.in);

        // Read number of bookings
        int N = Integer.parseInt(scanner.nextLine());
        Booking[] bookings = new Booking[N];

        // Read booking details
        for (int i = 0; i < N; i++) {
            int bookingId = Integer.parseInt(scanner.nextLine());
            String customerName = scanner.nextLine().trim();
            double distanceTravelled = Double.parseDouble(scanner.nextLine());
            bookings[i] = new Booking(bookingId, customerName, distanceTravelled);
        }

        // Display booking details and final fare
        for (Booking booking : bookings) {
            System.out.println("Booking ID: " + booking.getBookingId());
            System.out.println("Customer Name: " + booking.getCustomerName());
            System.out.printf("Final Fare: %.1f%n", booking.calculateFare());
        }

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Nivedithaa S
Email: 241901076@rajalakshmi.edu.in
Roll no: 241901076
Phone: 6374185608
Branch: REC
Department: CSE (CS) - Section 2
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ram is working as a developer for BrightEdu Coaching Center, which wants to build a student fee management system.

Each student's enrollment has:

An Enrollment ID (integer)A Student Name (string)The Number of Subjects (integer)

The fee calculation rules are:

Registration Fee = 1000 units (flat for every student).Per Subject Fee = 800 units.If the student enrolls in more than 5 subjects, a 20% scholarship (discount) is applied on the total fee.

Ram has been asked to implement this system using:

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

Finally, display each student's details and final fee.

Input Format

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

For each student:

- The next line contains the Enrollment ID (integer).
- The following line contains the student's name (string).
- The next line contains the Number of subjects (integer).

Output Format

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

- Enrollment ID: <enrollment_id>
- Student Name: <student_name>
- Final Fee: <final_fee> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Ravi Kumar
3
Output: Enrollment ID: 1234
Student Name: Ravi Kumar
Final Fee: 3400.0

Answer

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

class Student {
```

```
private int enrollmentId;
private String studentName;
private int numberOfSubjects;

// Constructor
public Student(int enrollmentId, String studentName, int numberOfSubjects) {
    this.enrollmentId = enrollmentId;
    this.studentName = studentName;
    this.numberOfSubjects = numberOfSubjects;
}

// Setters
public void setEnrollmentId(int enrollmentId) {
    this.enrollmentId = enrollmentId;
}

public void setStudentName(String studentName) {
    this.studentName = studentName;
}

public void setNumberOfSubjects(int numberOfSubjects) {
    this.numberOfSubjects = numberOfSubjects;
}

// Getters
public int getEnrollmentId() {
    return enrollmentId;
}

public String getStudentName() {
    return studentName;
}

public int getNumberOfSubjects() {
    return numberOfSubjects;
}

// Fee calculation
public double calculateFinalFee() {
    double baseFee = 1000.0;
    double subjectFee = 800.0 * numberOfSubjects;
    double total = baseFee + subjectFee;
```

```
        if (numberOfSubjects > 5) {
            total *= 0.8; // 20% discount
        }

        return Math.round(total * 10) / 10.0;
    }
}

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

        Student[] students = new Student[n];

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(scanner.nextLine());
            String name = scanner.nextLine().trim();
            int subjects = Integer.parseInt(scanner.nextLine());
            students[i] = new Student(id, name, subjects);
        }

        for (Student s : students) {
            System.out.println("Enrollment ID: " + s.getEnrollmentId());
            System.out.println("Student Name: " + s.getStudentName());
            System.out.printf("Final Fee: %.1f%n", s.calculateFinalFee());
        }

        scanner.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Nivedithaa S
Email: 241901076@rajalakshmi.edu.in
Roll no: 241901076
Phone: 6374185608
Branch: REC
Department: CSE (CS) - Section 2
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Anjali is working as a developer for CityFitness Gym, which wants to build a system to calculate monthly membership fees for gym members based on the type of membership and the number of personal training sessions booked.

Each member's record has:

Member ID (integer)Member Name (string)Membership Type (string: "Basic", "Premium", "Elite")Number of Personal Training Sessions (integer)

The monthly fees are:

Basic – 1000 unitsPremium – 1500 unitsElite – 2000 units

The cost of personal training sessions is 500 units per session.

The calculation rules:

Total Amount = Membership Fee + (Number of Personal Training Sessions × 500)If the number of sessions is more than 5, a 10% discount is applied on the total amount.If the member has Elite membership and the total amount exceeds 4000, an additional 5% service tax is added after discount.

Anjali has been asked to implement this system using:

A class with attributes for member details.A constructor to initialize member details.Getter and Setter methods to retrieve and update member details if required.A method to calculate the final monthly fee.Objects of the class to represent members.

Finally, display each member's details and the final monthly fee.

Input Format

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

For each member:

- Next line contains Member ID (integer)
- Next line contains Member Name (string)
- Next line contains Membership Type ("Basic", "Premium", "Elite")
- Next line contains Number of Personal Training Sessions (integer)

Output Format

For each member, print:

- Member ID: <member_id>
- Member Name: <member_name>
- Final Monthly Fee: <final_fee> (The final fee must be rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1

1001
Ravi Kumar
Basic
3

Output: Member ID: 1001
Member Name: Ravi Kumar
Final Monthly Fee: 2500.0

Answer

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

class Member {
    private int memberId;
    private String memberName;
    private String membershipType;
    private int trainingSessions;

    public Member(int id, String name, String type, int sessions) {
        this.memberId = id;
        this.memberName = name;
        this.membershipType = type;
        this.trainingSessions = sessions;
    }

    public double calculateFinalFee() {
        double baseFee;
        switch (membershipType) {
            case "Basic":
                baseFee = 1000;
                break;
            case "Premium":
                baseFee = 1500;
                break;
            case "Elite":
                baseFee = 2000;
                break;
            default:
                throw new IllegalArgumentException("Invalid membership type");
        }

        double total = baseFee + (trainingSessions * 500);
```

```
        if (trainingSessions > 5) {
            total *= 0.9;  // 10% discount
        }

        if (membershipType.equals("Elite") && total > 4000) {
            total *= 1.05;  // 5% service tax
        }

        return Math.round(total * 10) / 10.0;  // Round to 1 decimal
    }

    // Getters
    public int getMemberId() { return memberId; }
    public String getMemberName() { return memberName; }
}

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

        for (int i = 0; i < n; i++) {
            int id = sc.nextInt();
            sc.nextLine();  // Consume newline
            String name = sc.nextLine().trim();
            String type = sc.nextLine().trim();
            int sessions = sc.nextInt();
            if (i < n-1) sc.nextLine();  // Consume newline for next member

            members[i] = new Member(id, name, type, sessions);
        }

        for (Member m : members) {
            System.out.println("Member ID: " + m.getMemberId());
            System.out.println("Member Name: " + m.getMemberName());
            System.out.printf("Final Monthly Fee: %.1f\n", m.calculateFinalFee());
        }

        sc.close();
    }
```

}

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

2. Problem Statement

Ravi is working as a developer for SecureLogin Systems, which wants to build a system to evaluate the strength of user passwords.

Each user record has:

User ID (integer)User Name (string)Password (string)

The system must calculate whether a password is strong or weak.

A password is considered strong if it meets all of the following conditions:

At least 8 characters long.Contains at least one uppercase letter.Contains at least one lowercase letter.Contains at least one digit.Contains at least one special character (from !@#$%^&*).

Ravi has been asked to implement this system using:

A class with attributes for user details.A constructor to initialize user details.Getter and setter methods to retrieve or update user details.A method to check whether the password is strong.Objects of the class to represent users.

Finally, display each user's details and indicate whether their password is Strong or Weak.

Input Format

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

For each user:

The next line contains the User ID (integer).

The next line contains the User Name (string).

The next line contains the Password (string).

Output Format

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

User ID: <user_id>

User Name: <user_name>

Password: <password>

Password Strength: <Strong/Weak>

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
Abc@1234

Output: User ID: 1001
User Name: Ravi Kumar
Password: Abc@1234
Password Strength: Strong

Answer

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

class User {
    private int userId;
    private String userName;
    private String password;

    public User(int userId, String userName, String password) {
        this.userId = userId;
        this.userName = userName;
        this.password = password;
    }
```

```
    // Getters
    public int getUserId() {
        return userId;
    }

    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }

    // Method to check password strength
    public String checkPasswordStrength() {
        if (password.length() < 8) {
            return "Weak";
        }

        boolean hasUpperCase = false;
        boolean hasLowerCase = false;
        boolean hasDigit = false;
        boolean hasSpecialChar = false;

        for (char c : password.toCharArray()) {
            if (Character.isUpperCase(c)) {
                hasUpperCase = true;
            } else if (Character.isLowerCase(c)) {
                hasLowerCase = true;
            } else if (Character.isDigit(c)) {
                hasDigit = true;
            } else if ("!@#$%^&*".indexOf(c) != -1) {
                hasSpecialChar = true;
            }
        }

        if (hasUpperCase && hasLowerCase && hasDigit && hasSpecialChar) {
            return "Strong";
        } else {
            return "Weak";
        }
```

```
    }
}

class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine(); // Consume the newline

        User[] users = new User[n];

        for (int i = 0; i < n; i++) {
            int userId = sc.nextInt();
            sc.nextLine(); // Consume the newline
            String userName = sc.nextLine().trim();
            String password = sc.nextLine().trim();

            users[i] = new User(userId, userName, password);
        }

        for (User user : users) {
            System.out.println("User ID: " + user.getUserId());
            System.out.println("User Name: " + user.getUserName());
            System.out.println("Password: " + user.getPassword());
            System.out.println("Password Strength: " +
user.checkPasswordStrength());
        }

        sc.close();
    }
}
```

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

3. Problem Statement

Each customer at the bank has an Account Number, Customer Name, and an Initial Balance. The bank allows two types of transactions:

Deposit – Increases the balance.Withdrawal – Decreases the balance, but

only if enough funds are available. If the withdrawal amount exceeds the available balance, the transaction should be skipped, and the balance should remain unchanged.

You are required to implement this banking system by:

Creating a class with the necessary attributes to store account details.

Using a constructor to initialize the account details when a new account is created.Providing setter methods to update the details if required.Providing getter methods to retrieve account details.Creating objects of this class to represent different customers, where each customer can perform deposits and withdrawals.

Instructions:

Implement the class to store account details.Implement the logic for performing deposit and withdrawal transactions.Ensure that withdrawals don't exceed the available balance.After performing the transactions, print the account number, customer name, and final balance.

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

```
// You are using Java

import java.util.Scanner;

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

    public BankAccount(int accountNumber, String customerName, double
initialBalance) {
        this.accountNumber = accountNumber;
        this.customerName = customerName;
        this.balance = initialBalance;
    }

    // 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 <= balance) {
            balance -= amount;
        }
```

```
    }

    // Getters for account details
    public int getAccountNumber() {
        return accountNumber;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getFinalBalance() {
        return balance;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine(); // Consume the newline

        BankAccount[] accounts = new BankAccount[n];

        for (int i = 0; i < n; i++) {
            int accountNumber = sc.nextInt();
            sc.nextLine(); // Consume the newline
            String customerName = sc.nextLine().trim();
            double initialBalance = sc.nextDouble();
            double depositAmount = sc.nextDouble();
            double withdrawalAmount = sc.nextDouble();
            sc.nextLine(); // Consume the newline after the last double input

            // Create a new bank account
            BankAccount account = new BankAccount(accountNumber,
customerName, initialBalance);
            // Perform transactions
            account.deposit(depositAmount);
            account.withdraw(withdrawalAmount);
            accounts[i] = account;
        }
```

```
        // Print account details
        for (BankAccount account : accounts) {
            System.out.println("Account Number: " + account.getAccountNumber());
            System.out.println("Customer Name: " + account.getCustomerName());
            System.out.printf("Final Balance: %.1f\n", account.getFinalBalance());
        }

        sc.close();
    }
}
```

Status : Wrong ***Marks : 0/10***

4. Problem Statement

Neha is working as a developer for CityMovie Theatre, which wants to build a system to calculate total ticket cost for movie-goers based on the number of tickets and type of seats booked.

Each customer’s booking has:

Booking ID (integer)Customer Name (string)Number of Tickets (integer)Seat Type (string: "Standard", "Premium", "VIP")

The ticket prices are:

Standard – 250 units per ticketPremium – 400 units per ticketVIP – 600 units per ticket

The calculation rules:

Total Amount = Number of Tickets × Seat Price

If a customer books more than 4 tickets, they get a 10% discount on the total amount.

If the booking is for VIP seats and the total amount exceeds 3000 units, a 5% luxury tax is added after any discount.

Neha has been asked to implement this system using:

A class with attributes for booking details.A constructor to initialize booking details.Getter and Setter methods to retrieve and update booking details if required.A method to calculate the final ticket cost.Objects of the class to represent bookings.

Finally, display each customer's details and final ticket amount.

Input Format

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

For each booking:

- The next line contains the Booking ID (integer).
- The next line contains the Customer Name (string).
- The next line contains Number of Tickets (integer).
- The next line contains Seat Type ("Standard", "Premium", or "VIP").

Output Format

For each booking, print:

- Booking ID: <booking_id>
- Customer Name: <customer_name>
- Final Ticket Amount: <final_amount> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
3
Standard
Output: Booking ID: 1001
Customer Name: Ravi Kumar
Final Ticket Amount: 750.0

Answer

// You are using Java

```
import java.util.Scanner;

class Booking {
    private int bookingId;
    private String customerName;
    private int numberOfTickets;
    private String seatType;

    // Ticket prices
    private static final double STANDARD_PRICE = 250.0;
    private static final double PREMIUM_PRICE = 400.0;
    private static final double VIP_PRICE = 600.0;

    // Constructor to initialize booking details
    public Booking(int bookingId, String customerName, int numberOfTickets,
String seatType) {
        this.bookingId = bookingId;
        this.customerName = customerName;
        this.numberOfTickets = numberOfTickets;
        this.seatType = seatType;
    }

    // Method to calculate the final ticket cost
    public double calculateFinalAmount() {
        double seatPrice = 0.0;

        // Determine seat price based on seat type
        switch (seatType) {
            case "Standard":
                seatPrice = STANDARD_PRICE;
                break;
            case "Premium":
                seatPrice = PREMIUM_PRICE;
                break;
            case "VIP":
                seatPrice = VIP_PRICE;
                break;
            default:
                throw new IllegalArgumentException("Invalid seat type");
        }

        // Calculate total amount
```

```
        double totalAmount = numberOfTickets * seatPrice;

        // Apply discount if more than 4 tickets are booked
        if (numberOfTickets > 4) {
            totalAmount *= 0.9; // 10% discount
        }

        // Apply luxury tax if the booking is for VIP seats and exceeds 3000 units
        if ("VIP".equals(seatType) && totalAmount > 3000) {
            totalAmount *= 1.05; // 5% luxury tax
        }

        return totalAmount;
    }

    // Getters for booking details
    public int getBookingId() {
        return bookingId;
    }

    public String getCustomerName() {
        return customerName;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt(); // Read number of bookings
        sc.nextLine(); // Consume the newline character

        Booking[] bookings = new Booking[n];

        for (int i = 0; i < n; i++) {
            int bookingId = sc.nextInt();
            sc.nextLine(); // Consume the newline character
            String customerName = sc.nextLine().trim();
            int numberOfTickets = sc.nextInt();
            sc.nextLine(); // Consume the newline character
            String seatType = sc.nextLine().trim();

            // Create a new booking
```

```
            Booking booking = new Booking(bookingId, customerName,
numberOfTickets, seatType);
            bookings[i] = booking;
        }

        // Print booking details and final ticket amount
        for (Booking booking : bookings) {
            System.out.println("Booking ID: " + booking.getBookingId());
            System.out.println("Customer Name: " + booking.getCustomerName());
            System.out.printf("Final Ticket Amount: %.1f\n",
booking.calculateFinalAmount());
        }

        sc.close(); // Close the scanner
    }
}
```

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

5. Problem Statement

Neha is working as a developer for CityQuiz Platform, which wants to build a system to calculate quiz scores and identify top scorers among participants.

Each participant’s record has:

Participant ID (integer)Participant Name (string)An array of scores in 5 quiz rounds (integers, each between 0 and 100)

The system must calculate:

Total Score = sum of scores in all 5 rounds.Average Score = Total Score ÷ 5.If a participant scores above 80 in all rounds, a bonus of 10 points is added to the total score.Identify the Top Scorer among all participants. If two participants have the same total score, the one with the lower Participant ID is considered the top scorer.

Neha has been asked to implement this system using:

A class with attributes for participant details.A constructor to initialize participant details.Getter and setter methods to retrieve or update participant details.A method to calculate total score and average score (including bonus if applicable).Objects of the class to represent participants.

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

Input Format

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

For each participant:

- Next line: Participant ID (integer)
- Next line: Participant Name (string)
- Next line: 5 integers separated by spaces (scores for 5 quiz rounds)

Output Format

For each participant:

- Participant ID: <participant_id>
- Participant Name: <participant_name>
- Total Score: <total_score>
- Average Score: <average_score>

Finally, print "Top Scorer: <participant_name> with <total_score> points"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
85 90 88 92 87

Output: Participant ID: 1001
Participant Name: Ravi Kumar
Total Score: 452
Average Score: 90
Top Scorer: Ravi Kumar with 452 points

Answer

```
// You are using Java

import java.util.Scanner;

class Participant {
    private int participantId;
    private String participantName;
    private int[] scores;

    public Participant(int participantId, String participantName, int[] scores) {
        this.participantId = participantId;
        this.participantName = participantName;
        this.scores = scores;
    }

    public int calculateTotalScore() {
        int total = 0;
        for (int score : scores) total += score;
        boolean allAbove80 = true;
        for (int score : scores) {
            if (score <= 80) {
                allAbove80 = false;
                break;
            }
        }
        return total + (allAbove80 ? 10 : 0);
    }

    public int calculateAverageScore() {
        return calculateTotalScore() / 5;
    }

    public int getParticipantId() { return participantId; }
    public String getParticipantName() { return participantName; }
}
```

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

        Participant topScorer = null;

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

            Participant p = new Participant(id, name, scores);

            if (topScorer == null ||
                p.calculateTotalScore() > topScorer.calculateTotalScore() ||
                (p.calculateTotalScore() == topScorer.calculateTotalScore() &&
                 p.getParticipantId() < topScorer.getParticipantId())) {
                topScorer = p;
            }

            System.out.println("Participant ID: " + p.getParticipantId());
            System.out.println("Participant Name: " + p.getParticipantName());
            System.out.println("Total Score: " + p.calculateTotalScore());
            System.out.println("Average Score: " + p.calculateAverageScore());
        }

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

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

Rajalakshmi Engineering College

Name: Nivedithaa S
Email: 241901076@rajalakshmi.edu.in
Roll no: 241901076
Phone: 6374185608
Branch: REC
Department: CSE (CS) - Section 2
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 : 0

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.Scanner;

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[] getTimes() {
        return times;
    }

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

public class Marathon {
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int N = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character after the integer input
        Runner[] runners = new Runner[N];

        for (int i = 0; i < N; i++) {
            int runnerId = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character after the integer
input
            String runnerName = scanner.nextLine();
            int[] times = new int[5];
            for (int j = 0; j < 5; j++) {
                times[j] = scanner.nextInt();
            }
            scanner.nextLine(); // Consume the newline character after the last
integer input
            runners[i] = new Runner(runnerId, runnerName, times);
        }

        Runner fastestRunner = runners[0];
        for (Runner runner : runners) {
            double averageTime = runner.calculateAverageTime();
            System.out.printf("Runner ID: %d\nRunner Name: %s\nAverage Time: %.0f
\n",
                    runner.getRunnerId(), runner.getRunnerName(), averageTime);
            if (runner.calculateAverageTime() <
fastestRunner.calculateAverageTime() ||
                (runner.calculateAverageTime() ==
fastestRunner.calculateAverageTime() && runner.getRunnerId() <
fastestRunner.getRunnerId())) {
                fastestRunner = runner;
            }
        }
        System.out.printf("Fastest Runner: %s with %.0f minutes\n",
                fastestRunner.getRunnerName(),
fastestRunner.calculateAverageTime());
        scanner.close(); // Close the scanner to avoid resource leaks
    }
}
```

Status : Wrong **Marks : 0/10**

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) {
        dataBalance += amount;
    }
```

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

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getFinalDataBalance() {
        return dataBalance;
    }
}

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

        // Read number of customers
        int N = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        Customer[] customers = new Customer[N];

        for (int i = 0; i < N; i++) {
            // Read customer details
            int customerId = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character

            String customerName = scanner.nextLine();
            double initialBalance = scanner.nextDouble();
            double rechargeAmount = scanner.nextDouble();
            double usageAmount = scanner.nextDouble();
            scanner.nextLine(); // Consume the newline character

            // Create customer object and perform operations
            Customer customer = new Customer(customerId, customerName,
initialBalance);
```

```
            customer.recharge(rechargeAmount);
            customer.useData(usageAmount);
            customers[i] = customer;
        }

        // Display customer details
        for (Customer customer : customers) {
            System.out.printf("Customer ID: %d\nCustomer Name: %s\nFinal Data
Balance: %.1f GB\n",
                    customer.getCustomerId(), customer.getCustomerName(),
                    customer.getFinalDataBalance());
        }

        scanner.close(); // Close the scanner to avoid resource leaks
    }
}
```

Status : Wrong **Marks : 0/10**

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.Scanner;

class Player {
    private int playerId;
    private String playerName;
    private int[] points; // Array to store points scored in 5 matches

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

    // Method to calculate the total score
    public int calculateTotalScore() {
        int totalScore = 0;
        for (int point : points) {
            totalScore += point;
        }
        return totalScore;
    }

    // Getter methods
    public int getPlayerId() {
        return playerId;
    }

    public String getPlayerName() {
        return playerName;
    }
}

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

        // Read number of players
        int N = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character
```

```
        Player[] players = new Player[N];

        // Read player details
        for (int i = 0; i < N; i++) {
            int playerId = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character
            String playerName = scanner.nextLine();
            int[] points = new int[5];

            // Read points scored in 5 matches
            for (int j = 0; j < 5; j++) {
                points[j] = scanner.nextInt();
            }
            scanner.nextLine(); // Consume the newline character

            players[i] = new Player(playerId, playerName, points);
        }

        // Variables to find the top scorer
        Player topScorer = players[0];
        int maxScore = topScorer.calculateTotalScore();

        // Display player details and find the top scorer
        for (Player player : players) {
            int totalScore = player.calculateTotalScore();
            System.out.printf("Player ID: %d\nPlayer Name: %s\nTotal Score: %d\n",
                    player.getPlayerId(), player.getPlayerName(), totalScore);

            // Determine the top scorer
            if (totalScore > maxScore || (totalScore == maxScore &&
player.getPlayerId() < topScorer.getPlayerId())) {
                topScorer = player;
                maxScore = totalScore;
            }
        }

        // Print the top scorer details
        System.out.printf("Top Scorer: %s with %d points\n",
topScorer.getPlayerName(), maxScore);

        scanner.close(); // Close the scanner to avoid resource leaks
```

```
    }
}
```

Status : Wrong ***Marks : 0/10***

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

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

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

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

    // Method to calculate the final bill
    public double calculateFinalBill() {
        double bill = 0.0;
```

```
        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);
        }

        // Apply discount if applicable
        if (bill > 2000) {
            bill *= 0.85; // 15% discount
        }

        return bill;
    }

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

    public String getCustomerName() {
        return customerName;
    }
}

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

        // Read number of customers
        int N = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        Customer[] customers = new Customer[N];

        // Read customer details
        for (int i = 0; i < N; i++) {
            int customerId = scanner.nextInt();
            scanner.nextLine(); // Consume the newline character
            String customerName = scanner.nextLine();
```

```
            double unitsConsumed = scanner.nextDouble();
            scanner.nextLine(); // Consume the newline character

            customers[i] = new Customer(customerId, customerName,
unitsConsumed);
        }

        // Display customer details and final bill
        for (Customer customer : customers) {
            double finalBill = customer.calculateFinalBill();
            System.out.printf("Customer ID: %d\nCustomer Name: %s\nFinal Bill: %.1f
\n",
                              customer.getCustomerId(), customer.getCustomerName(),
finalBill);
        }

        scanner.close(); // Close the scanner to avoid resource leaks
    }
}
```

Status : Wrong **Marks : 0/10**