# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

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 void setAccountNumber(int accountNumber) {
      this.accountNumber = accountNumber;
   }

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

   public void setBalance(double balance) {
      this.balance = balance;
   }

   public int getAccountNumber() {
      return accountNumber;
   }

   public String getCustomerName() {
      return customerName;
   }

   public double getBalance() {
      return balance;
   }

   public void deposit(double amount) {
      if (amount > 0) {
            balance += amount;
        }
    }

    public void withdraw(double amount) {
        if (amount <= balance) {
            balance -= amount;
        }
    }
}

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 accNo = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            double initBalance = Double.parseDouble(sc.nextLine());
            double depositAmt = Double.parseDouble(sc.nextLine());
            double withdrawAmt = Double.parseDouble(sc.nextLine());

            Account customer = new Account(accNo, name, initBalance);
            customer.deposit(depositAmt);
            customer.withdraw(withdrawAmt);

            System.out.printf("Account Number: %d%n",
customer.getAccountNumber());
            System.out.printf("Customer Name: %s%n",
customer.getCustomerName());
            System.out.printf("Final Balance: %.1f%n", customer.getBalance());
        }

        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

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

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

    public void setUnitsConsumed(double unitsConsumed) {
        this.unitsConsumed = unitsConsumed;
    }

    public int getCustomerId() {
        return customerId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getUnitsConsumed() {
        return unitsConsumed;
    }

    public double calculateBill() {
        double units = unitsConsumed;
        double bill = 0;
        if (units <= 100) {
            bill = units * 5;
        } else if (units <= 200) {
            bill = (100 * 5) + ((units - 100) * 7);
        } else {
            bill = (100 * 5) + (100 * 7) + ((units - 200) * 10);
        }
        if (bill > 2000) {
            bill = bill - (bill * 0.05);
        }
        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 units = Double.parseDouble(sc.nextLine());
            Customer c = new Customer(id, name, units);
            System.out.printf("Customer ID: %d%n", c.getCustomerId());
            System.out.printf("Customer Name: %s%n", c.getCustomerName());
            System.out.printf("Final Bill: %.1f%n", c.calculateBill());
        }
        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

## 2028_REC_OOPS using Java_Week 5_Q4

Attempt : 2
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 distance;

    public Booking(int bookingId, String customerName, double distance) {
        this.bookingId = bookingId;
        this.customerName = customerName;
        this.distance = distance;
    }

    public void setBookingId(int bookingId) {
        this.bookingId = bookingId;
    }

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

    public void setDistance(double distance) {
        this.distance = distance;
    }

    public int getBookingId() {
        return bookingId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getDistance() {
        return distance;
    }

    public double calculateFare() {
        double fare = 50 + (distance * 10);
        if (distance > 20) {
            fare = fare - (fare * 0.10);
        }
        return fare;
    }
}
```

```
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 dist = Double.parseDouble(sc.nextLine());
            Booking b = new Booking(id, name, dist);
            System.out.printf("Booking ID: %d%n", b.getBookingId());
            System.out.printf("Customer Name: %s%n", b.getCustomerName());
            System.out.printf("Final Fare: %.1f%n", b.calculateFare());
        }
        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

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***

```
import java.util.Scanner;

class Student {
   private int enrollmentId;
```

```
    private String studentName;
    private int numSubjects;

    public Student(int enrollmentId, String studentName, int numSubjects) {
        this.enrollmentId = enrollmentId;
        this.studentName = studentName;
        this.numSubjects = numSubjects;
    }

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

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

    public void setNumSubjects(int numSubjects) {
        this.numSubjects = numSubjects;
    }

    public int getEnrollmentId() {
        return enrollmentId;
    }

    public String getStudentName() {
        return studentName;
    }

    public int getNumSubjects() {
        return numSubjects;
    }

    public double calculateFee() {
        double fee = 1000 + (numSubjects * 800);
        if (numSubjects > 5) {
            fee = fee - (fee * 0.20);
        }
        return fee;
    }
}
```

```
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();
            int subjects = Integer.parseInt(sc.nextLine());
            Student s = new Student(id, name, subjects);
            System.out.printf("Enrollment ID: %d%n", s.getEnrollmentId());
            System.out.printf("Student Name: %s%n", s.getStudentName());
            System.out.printf("Final Fee: %.1f%n", s.calculateFee());
        }
        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

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 : 50

## Section 1 : Coding

1. 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*

```
import java.util.Scanner;

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

   public Booking(int bookingId, String customerName, int numberOfTickets,
String seatType) {
      this.bookingId = bookingId;
      this.customerName = customerName;
      this.numberOfTickets = numberOfTickets;
      this.seatType = seatType;
   }

   public int getBookingId() {
      return bookingId;
   }

   public String getCustomerName() {
      return customerName;
   }

   public int getNumberOfTickets() {
      return numberOfTickets;
   }

   public String getSeatType() {
      return seatType;
   }

   public void setNumberOfTickets(int numberOfTickets) {
        this.numberOfTickets = numberOfTickets;
    }

    public void setSeatType(String seatType) {
        this.seatType = seatType;
    }

    public double calculateFinalAmount() {
        double seatPrice = 0;

        switch (seatType) {
            case "Standard":
                seatPrice = 250;
                break;
            case "Premium":
                seatPrice = 400;
                break;
            case "VIP":
                seatPrice = 600;
                break;
        }

        double totalAmount = numberOfTickets * seatPrice;

        if (numberOfTickets > 4) {
            totalAmount *= 0.9;
        }

        if (seatType.equals("VIP") && totalAmount > 3000) {
            totalAmount *= 1.05;
        }

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

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

        int n = Integer.parseInt(sc.nextLine());
        Booking[] bookings = new Booking[n];
        for (int i = 0; i < n; i++) {
            int bookingId = Integer.parseInt(sc.nextLine());
            String customerName = sc.nextLine().trim();
            int numberOfTickets = Integer.parseInt(sc.nextLine());
            String seatType = sc.nextLine().trim();

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

        for (Booking booking : bookings) {
            System.out.println("Booking ID: " + booking.getBookingId());
            System.out.println("Customer Name: " + booking.getCustomerName());
            System.out.println("Final Ticket Amount: " +
booking.calculateFinalAmount());
        }

        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 !@#$%^&amp;*).

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*

```
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;
   }

   public int getUserId() {
      return userId;
   }

   public String getUserName() {
      return userName;
   }

   public String getPassword() {
      return password;
   }

   public void setUserName(String userName) {
      this.userName = userName;
   }

   public void setPassword(String password) {
      this.password = password;
   }

   public String checkPasswordStrength() {
      if (password.length() < 8)
```

```
            return "Weak";

        boolean hasUpper = false;
        boolean hasLower = false;
        boolean hasDigit = false;
        boolean hasSpecial = false;

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

        if (hasUpper && hasLower && hasDigit && hasSpecial)
            return "Strong";
        else
            return "Weak";
    }
}

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

        for (int i = 0; i < n; i++) {
            int userId = Integer.parseInt(sc.nextLine());
            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

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***

```
import java.util.*;

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

```
    private int totalScore;
    private int averageScore;

    public Participant(int participantId, String participantName, int[] scores) {
        this.participantId = participantId;
        this.participantName = participantName;
        this.scores = scores;
        this.totalScore = 0;
        this.averageScore = 0;
    }

    public int getParticipantId() {
        return participantId;
    }

    public String getParticipantName() {
        return participantName;
    }

    public int getTotalScore() {
        return totalScore;
    }

    public int getAverageScore() {
        return averageScore;
    }

    public void calculateScores() {
        boolean allAbove80 = true;
        for (int score : scores) {
            totalScore += score;
            if (score <= 80) {
                allAbove80 = false;
            }
        }
        if (allAbove80) {
            totalScore += 10;
        }
        averageScore = totalScore / 5;
    }
}
```

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

    Participant[] participants = new Participant[n];

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

    Participant topScorer = participants[0];

    for (Participant p : participants) {
      System.out.println("Participant ID: " + p.getParticipantId());
      System.out.println("Participant Name: " + p.getParticipantName());
      System.out.println("Total Score: " + p.getTotalScore());
      System.out.println("Average Score: " + p.getAverageScore());
      if (
        p.getTotalScore() > topScorer.getTotalScore() ||
        (p.getTotalScore() == topScorer.getTotalScore() && p.getParticipantId()
< topScorer.getParticipantId())
      ) {
        topScorer = p;
      }
    }

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

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

4. 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***

```
import java.util.*;

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

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

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            this.balance += amount;
        }
    }

    public void withdraw(double amount) {
        if (amount <= this.balance) {
            this.balance -= amount;
        }
    }
}

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

        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 withdrawalAmount = Double.parseDouble(sc.nextLine());

            BankAccount account = new BankAccount(accountNumber,
customerName, initialBalance);
            account.deposit(depositAmount);
            account.withdraw(withdrawalAmount);
            accounts.add(account);
        }

        for (BankAccount acc : accounts) {
            System.out.println("Account Number: " + acc.getAccountNumber());
            System.out.println("Customer Name: " + acc.getCustomerName());
            System.out.println("Final Balance: " + String.format("%.1f",
acc.getBalance()));
        }

        sc.close();
    }
}
```

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

5. 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*

```
import java.util.*;

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

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

    public int getMemberId() {
        return memberId;
    }

    public String getMemberName() {
        return memberName;
    }

    public String getMembershipType() {
        return membershipType;
    }

    public int getTrainingSessions() {
        return trainingSessions;
    }

    public void setMemberName(String name) {
        this.memberName = name;
    }

    public void setMembershipType(String type) {
        this.membershipType = type;
    }

    public void setTrainingSessions(int sessions) {
        this.trainingSessions = sessions;
    }

    public double calculateFinalFee() {
        int membershipFee = 0;
        if (membershipType.equals("Basic")) {
            membershipFee = 1000;
        } else if (membershipType.equals("Premium")) {
            membershipFee = 1500;
        } else if (membershipType.equals("Elite")) {
            membershipFee = 2000;
        }

        int trainingFee = trainingSessions * 500;
        double total = membershipFee + trainingFee;

        if (trainingSessions > 5) {
            total = total * 0.9;
        }

        if (membershipType.equals("Elite") && total > 4000) {
            total = total * 1.05;
        }

        return Math.round(total * 10.0) / 10.0;
    }
}
class GymMembership {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Member> members = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int memberId = Integer.parseInt(sc.nextLine());
            String memberName = sc.nextLine();
            String membershipType = sc.nextLine();
            int trainingSessions = Integer.parseInt(sc.nextLine());
            Member member = new Member(memberId, memberName,
membershipType, trainingSessions);
            members.add(member);
        }

        for (Member m : members) {
            System.out.println("Member ID: " + m.getMemberId());
            System.out.println("Member Name: " + m.getMemberName());
            System.out.println("Final Monthly Fee: " + m.calculateFinalFee());
        }

        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

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 : 50

## Section 1 : Coding

1. 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*

```
import java.util.Scanner;

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

   public Booking(int bookingId, String customerName, int numberOfTickets,
String seatType) {
      this.bookingId = bookingId;
      this.customerName = customerName;
      this.numberOfTickets = numberOfTickets;
      this.seatType = seatType;
   }

   public int getBookingId() {
      return bookingId;
   }

   public String getCustomerName() {
      return customerName;
   }

   public int getNumberOfTickets() {
      return numberOfTickets;
   }

   public String getSeatType() {
      return seatType;
   }

   public void setNumberOfTickets(int numberOfTickets) {
        this.numberOfTickets = numberOfTickets;
    }

    public void setSeatType(String seatType) {
        this.seatType = seatType;
    }

    public double calculateFinalAmount() {
        double seatPrice = 0;

        switch (seatType) {
            case "Standard":
                seatPrice = 250;
                break;
            case "Premium":
                seatPrice = 400;
                break;
            case "VIP":
                seatPrice = 600;
                break;
        }

        double totalAmount = numberOfTickets * seatPrice;

        if (numberOfTickets > 4) {
            totalAmount *= 0.9;
        }

        if (seatType.equals("VIP") && totalAmount > 3000) {
            totalAmount *= 1.05;
        }

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

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

        int n = Integer.parseInt(sc.nextLine());
        Booking[] bookings = new Booking[n];
        for (int i = 0; i < n; i++) {
            int bookingId = Integer.parseInt(sc.nextLine());
            String customerName = sc.nextLine().trim();
            int numberOfTickets = Integer.parseInt(sc.nextLine());
            String seatType = sc.nextLine().trim();

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

        for (Booking booking : bookings) {
            System.out.println("Booking ID: " + booking.getBookingId());
            System.out.println("Customer Name: " + booking.getCustomerName());
            System.out.println("Final Ticket Amount: " +
booking.calculateFinalAmount());
        }

        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 !@#$%^&amp;*).

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*

```
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;
    }

    public int getUserId() {
        return userId;
    }

    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String checkPasswordStrength() {
        if (password.length() < 8)
```

```
            return "Weak";

        boolean hasUpper = false;
        boolean hasLower = false;
        boolean hasDigit = false;
        boolean hasSpecial = false;

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

        if (hasUpper && hasLower && hasDigit && hasSpecial)
            return "Strong";
        else
            return "Weak";
    }
}

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

        for (int i = 0; i < n; i++) {
            int userId = Integer.parseInt(sc.nextLine());
            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

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***

```
import java.util.*;

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

    public Participant(int participantId, String participantName, int[] scores) {
        this.participantId = participantId;
        this.participantName = participantName;
        this.scores = scores;
        this.totalScore = 0;
        this.averageScore = 0;
    }

    public int getParticipantId() {
        return participantId;
    }

    public String getParticipantName() {
        return participantName;
    }

    public int getTotalScore() {
        return totalScore;
    }

    public int getAverageScore() {
        return averageScore;
    }

    public void calculateScores() {
        boolean allAbove80 = true;
        for (int score : scores) {
            totalScore += score;
            if (score <= 80) {
                allAbove80 = false;
            }
        }
        if (allAbove80) {
            totalScore += 10;
        }
        averageScore = totalScore / 5;
    }
}
```

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

    Participant[] participants = new Participant[n];

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

    Participant topScorer = participants[0];

    for (Participant p : participants) {
      System.out.println("Participant ID: " + p.getParticipantId());
      System.out.println("Participant Name: " + p.getParticipantName());
      System.out.println("Total Score: " + p.getTotalScore());
      System.out.println("Average Score: " + p.getAverageScore());
      if (
        p.getTotalScore() > topScorer.getTotalScore() ||
        (p.getTotalScore() == topScorer.getTotalScore() && p.getParticipantId()
< topScorer.getParticipantId())
      ) {
        topScorer = p;
      }
    }

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

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

4. 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***

```
import java.util.*;

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

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

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public void deposit(double amount) {
        if (amount > 0) {
            this.balance += amount;
        }
    }

    public void withdraw(double amount) {
        if (amount <= this.balance) {
            this.balance -= amount;
        }
    }
}

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

        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 withdrawalAmount = Double.parseDouble(sc.nextLine());

            BankAccount account = new BankAccount(accountNumber,
customerName, initialBalance);
            account.deposit(depositAmount);
            account.withdraw(withdrawalAmount);
            accounts.add(account);
        }

        for (BankAccount acc : accounts) {
            System.out.println("Account Number: " + acc.getAccountNumber());
            System.out.println("Customer Name: " + acc.getCustomerName());
            System.out.println("Final Balance: " + String.format("%.1f",
acc.getBalance()));
        }

        sc.close();
    }
}
```

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

5. 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***

```
import java.util.*;

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

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

   public int getMemberId() {
      return memberId;
   }

   public String getMemberName() {
      return memberName;
   }

   public String getMembershipType() {
      return membershipType;
   }

   public int getTrainingSessions() {
      return trainingSessions;
   }

   public void setMemberName(String name) {
      this.memberName = name;
   }

   public void setMembershipType(String type) {
      this.membershipType = type;
```

```
    }

    public void setTrainingSessions(int sessions) {
        this.trainingSessions = sessions;
    }

    public double calculateFinalFee() {
        int membershipFee = 0;
        if (membershipType.equals("Basic")) {
            membershipFee = 1000;
        } else if (membershipType.equals("Premium")) {
            membershipFee = 1500;
        } else if (membershipType.equals("Elite")) {
            membershipFee = 2000;
        }

        int trainingFee = trainingSessions * 500;
        double total = membershipFee + trainingFee;

        if (trainingSessions > 5) {
            total = total * 0.9;
        }

        if (membershipType.equals("Elite") && total > 4000) {
            total = total * 1.05;
        }

        return Math.round(total * 10.0) / 10.0;
    }
}
class GymMembership {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine());
        List<Member> members = new ArrayList<>();

        for (int i = 0; i < n; i++) {
            int memberId = Integer.parseInt(sc.nextLine());
            String memberName = sc.nextLine();
            String membershipType = sc.nextLine();
            int trainingSessions = Integer.parseInt(sc.nextLine());
            Member member = new Member(memberId, memberName,
membershipType, trainingSessions);
            members.add(member);
        }

        for (Member m : members) {
            System.out.println("Member ID: " + m.getMemberId());
            System.out.println("Member Name: " + m.getMemberName());
            System.out.println("Final Monthly Fee: " + m.calculateFinalFee());
        }

        sc.close();
    }
}
```

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

# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

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 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*

```
import java.util.*;

class Player {
    private int playerId;
    private String playerName;
    private int[] matchPoints;

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

    public int getPlayerId() {
        return playerId;
    }

    public String getPlayerName() {
        return playerName;
    }

    public int[] getMatchPoints() {
        return matchPoints;
    }

    public void setPlayerName(String name) {
        this.playerName = name;
    }

    public void setMatchPoints(int[] points) {
        this.matchPoints = points;
    }

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

class BasketballScoring {
    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 playerId = Integer.parseInt(sc.nextLine());
            String playerName = sc.nextLine();
            String[] pointsStr = sc.nextLine().split(" ");
            int[] matchPoints = new int[5];
            for (int j = 0; j < 5; j++) {
                matchPoints[j] = Integer.parseInt(pointsStr[j]);
            }
            players.add(new Player(playerId, playerName, matchPoints));
        }

        Player topScorer = null;
        int highestScore = -1;

        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 > highestScore) {
                highestScore = total;
                topScorer = p;
            } else if (total == highestScore && p.getPlayerId() <
topScorer.getPlayerId()) {
                topScorer = p;
            }
        }

        System.out.println("Top Scorer: " + topScorer.getPlayerName() + " with " +
highestScore + " points");
    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*

```
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 setCustomerId(int customerId) {
        this.customerId = customerId;
    }

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

    public void setDataBalance(double dataBalance) {
        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 >= 0 && amount <= dataBalance) {
            dataBalance -= amount;
        }
    }
}

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();
            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(id, name, initialBalance);
            customer.recharge(rechargeAmount);
            customer.useData(usageAmount);

            customers[i] = customer;
        }

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

        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 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.*;

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 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 liters = litersConsumed;
      double bill = 0;

      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 * 0.9;
      }

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

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

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

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

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

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

4. Problem Statement

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

Each runner's record has:

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

The system must calculate:

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

Anjali has been asked to implement this system using:

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

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

***Input Format***

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

For each runner:

- The next line contains the Runner ID (integer).
- The following line contains the Runner Name (string).

- The next line contains 5 integers separated by spaces (times in minutes for 5 marathon events).

***Output Format***

For each runner the output prints the following details:

- Runner ID: <runner_id>
- Runner Name: <runner_name>
- Average Time: <average_time>

Finally, print "Fastest Runner: <runner_name> with <average_time> minutes"

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
1001
Ravi Kumar
240 250 245 255 260

Output: Runner ID: 1001
Runner Name: Ravi Kumar
Average Time: 250
Fastest Runner: Ravi Kumar with 250 minutes

***Answer***

```
import java.util.*;

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

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

```
    }

    public int getRunnerId() {
        return runnerId;
    }

    public String getName() {
        return name;
    }

    public int calculateAverage() {
        int sum = 0;
        for (int time : times) {
            sum += time;
        }
        return sum / times.length;
    }
}

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

        Runner fastestRunner = runners.get(0);
        int fastestAvg = fastestRunner.calculateAverage();

        for (Runner runner : runners) {
            int avg = runner.calculateAverage();
            System.out.println("Runner ID: " + runner.getRunnerId());
            System.out.println("Runner Name: " + runner.getName());
            System.out.println("Average Time: " + avg);
            if (avg < fastestAvg || (avg == fastestAvg && runner.getRunnerId() <
fastestRunner.getRunnerId())) {
                fastestRunner = runner;
                fastestAvg = avg;
            }
        }

        System.out.println("Fastest Runner: " + fastestRunner.getName() + " with " +
fastestAvg + " minutes");
    }
}
```

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

# Rajalakshmi Engineering College

Name: Navin Muthia
Email: 241001151@rajalakshmi.edu.in
Roll no: 2116241001151
Phone: 9943885959
Branch: REC
Department: IT - Section 5
Batch: 2028
Degree: B.E - IT

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 : 14

## 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 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***

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***

Compilation error

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

4. 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***

Area = 20

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

5. 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***

6. 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***

7. 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***

8. 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***

9. 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***

10. 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*

11

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

11. 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***

12. 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***

25

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

13. 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***

14. 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***

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***