Rajalakshmi Engineering College

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

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. 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.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 CityBankApp {
    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 initBal = Double.parseDouble(sc.nextLine());
            double deposit = Double.parseDouble(sc.nextLine());
            double withdraw = Double.parseDouble(sc.nextLine());

            Account acc = new Account(accNo, name, initBal);
            acc.deposit(deposit);
            acc.withdraw(withdraw);

            System.out.println("Account Number: " + acc.getAccountNumber());
            System.out.println("Customer Name: " + acc.getCustomerName());
            System.out.println("Final Balance: " + acc.getBalance());
        }
        sc.close();
    }
}
```

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

2. 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.Scanner;

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;
    }

    public int getParticipantId() {
        return participantId;
    }

    public String getParticipantName() {
        return participantName;
    }
```

```
    public int getTotalScore() {
        return totalScore;
    }

    public int getAverageScore() {
        return averageScore;
    }

    public void setParticipantId(int participantId) {
        this.participantId = participantId;
    }

    public void setParticipantName(String participantName) {
        this.participantName = participantName;
    }

    public void setScores(int[] scores) {
        this.scores = scores;
    }

    public void calculateScores() {
        totalScore = 0;
        boolean allAbove80 = true;
        for (int score : scores) {
            totalScore += score;
            if (score <= 80) {
                allAbove80 = false;
            }
        }
        if (allAbove80) {
            totalScore += 10;
        }
        averageScore = totalScore / scores.length;
    }
}

class QuizScoreSystem {
    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();
            String[] scoreStr = sc.nextLine().split(" ");
            int[] scores = new int[5];
            for (int j = 0; j < 5; j++) {
                scores[j] = Integer.parseInt(scoreStr[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");
    }
}
```

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

3. 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 setBookingId(int bookingId) {
        this.bookingId = bookingId;
    }

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

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

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

    public double calculateFinalAmount() {
        double pricePerTicket = 0;
        if (seatType.equalsIgnoreCase("Standard")) pricePerTicket = 250;
        else if (seatType.equalsIgnoreCase("Premium")) pricePerTicket = 400;
        else if (seatType.equalsIgnoreCase("VIP")) pricePerTicket = 600;
        double total = pricePerTicket * numberOfTickets;
        if (numberOfTickets > 4) total *= 0.9;
        if (seatType.equalsIgnoreCase("VIP") && total > 3000) total *= 1.05;
        return total;
    }
}

class Main {
    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();
            int numberOfTickets = Integer.parseInt(sc.nextLine());
```

```
            String seatType = sc.nextLine();
            bookings[i] = new Booking(bookingId, customerName, numberOfTickets,
seatType);
        }
        for (int i = 0; i < n; i++) {
            System.out.println("Booking ID: " + bookings[i].getBookingId());
            System.out.println("Customer Name: " +
bookings[i].getCustomerName());
            System.out.println("Final Ticket Amount: " + String.format("%.1f",
bookings[i].calculateFinalAmount()));
        }
    }
}
```

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

4. 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.Scanner;

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

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

    public int getMemberId() {
        return memberId;
    }

    public String getMemberName() {
        return memberName;
    }

    public String getMembershipType() {
        return membershipType;
    }

    public int getPersonalTrainingSessions() {
        return personalTrainingSessions;
    }

    public void setMemberId(int memberId) {
        this.memberId = memberId;
    }

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

    public void setMembershipType(String membershipType) {
```

```
        this.membershipType = membershipType;
    }

    public void setPersonalTrainingSessions(int personalTrainingSessions) {
        this.personalTrainingSessions = personalTrainingSessions;
    }

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

        double total = membershipFee + personalTrainingSessions * 500;
        if (personalTrainingSessions > 5) total *= 0.9;
        if (membershipType.equalsIgnoreCase("Elite") && total > 4000) total *=
1.05;
        return total;
    }
}

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

        for (int i = 0; i < n; i++) {
            int id = Integer.parseInt(sc.nextLine());
            String name = sc.nextLine();
            String type = sc.nextLine();
            int sessions = Integer.parseInt(sc.nextLine());
            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.println("Final Monthly Fee: " + String.format("%.1f",
m.calculateFinalFee()));
```

```
    }
  }
}
```

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

5. 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

```
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 void setUserId(int userId) {
        this.userId = userId;
    }

    public String getUserName() {
        return userName;
    }

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

    public String getPassword() {
        return password;
    }

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

    public boolean isStrongPassword() {
        if (password.length() < 8) return false;
        boolean hasUpper = false, hasLower = false, hasDigit = false, 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) >= 0) hasSpecial = true;
        }
        return hasUpper && hasLower && hasDigit && hasSpecial;
    }
}

class Main {
    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();
            String password = sc.nextLine();
            users[i] = new User(userId, userName, password);
        }
        for (User u : users) {
            System.out.println("User ID: " + u.getUserId());
            System.out.println("User Name: " + u.getUserName());
            System.out.println("Password: " + u.getPassword());
            System.out.println("Password Strength: " + (u.isStrongPassword() ?
"Strong" : "Weak"));
        }
    }
}
```

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