



 slington college
(इसलिंग्टन कलेज)

CS4001NI Programming

60% Individual Coursework

2025 Spring

Student Name: Alisha Maharjan

London Met ID: 24046561

College ID: NP01AI4A240078

Group: L1 AI 3

Assignment Due Date: Friday, May 16, 2025

Assignment Submission Date: Friday, May 16, 2025

I confirm that I understand my coursework needs to be submitted online via MySecondTeacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

24046561AlishaMaharjan (2).docx

 Islington College, Nepal

Document Details

Submission ID

trn:oid::3618:96183826

Submission Date

May 16, 2025, 11:20 AM GMT+5:45

Download Date

May 16, 2025, 11:23 AM GMT+5:45

File Name

24046561AlishaMaharjan (2).docx

File Size

78.1 KB

90 Pages





7,255 Words

48,245 Characters




11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

-  **63 Not Cited or Quoted 11%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **2 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 0%  Internet sources
- 0%  Publications
- 11%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

- **63 Not Cited or Quoted 11%**
Matches with neither in-text citation nor quotation marks
- **0 Missing Quotations 0%**
Matches that are still very similar to source material
- **2 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
- **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 0% ■ Internet sources
- 0% ■ Publications
- 11% ■ Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Submitted works	islingtoncollege on 2025-05-16	5%
2	Submitted works	iacademy on 2025-05-14	4%
3	Submitted works	Chester College of Higher Education on 2016-05-10	<1%

Table of Contents

Table of Tables	8
1.Introduction	9
1.1 Java GUI.....	9
1.2 Aims.....	9
1.3 Objectives	9
1.4 Tools and Technologies used.....	9
2. GUI Wireframes.....	12
3. Class Diagram.....	13
3.1 Gym Member.....	13
3.2 Regular Member	13
3.3 Premium Member	14
3.4 Gym GUI.....	15
3.5 Complete Class diagram	16
4. Pseudocode	17
4.1 Gym Member.....	17
4.2 Regular Member	20
4.3 Premium Member.....	23
4.4 Gym GUI.....	27
5. Method Description	78
5.1 Gym Member.....	78
5.2 Regular Member	79
5.3 Premium Member	80
5.4 Gym GUI.....	81
6. Testing	83
6.1 Test 1: Compile and run using terminal	83
6.2 Test 2: Add Regular Member and Premium Member	84
6.2.1 Test 2.1: Adding Regular Member	84
6.2.2 Test 2.2: Adding Premium Member.....	86
6.3 Test 3: Mark Attendance and Upgrade Plan.....	87
6.3.1 Test 3.1: Mark Attendance	87
6.3.2 Test 3.2: Upgrade Plan	89

6.4 Test 4: Calculate Discount, pay due amount, revert members	92
6.4.1 Test 4.1: Calculate Discount	92
6.4.2 Test 4.2: Pay Due Amount	94
6.4.3 Test 4.3: Revert Members.....	96
6.5 Test 5: Save to and read from file.....	100
6.5.1 Test 5.1: Saving to file.....	100
6.5.2 Reading from file	103
7. Error Detection and Correction.....	105
7.1 Syntax Error.....	105
7.2 Runtime Error	106
7.3 Logical Error	108
8. Conclusion	110
9. Appendix	110
9.1 Gym Member.....	110
9.2 Regular Member	115
9.3 Premium Member	122
9.4 GymGUI.....	127

Table of Figures

Figure 1: BlueJ	10
Figure 2: draw.io	11
Figure 3: Notepad	11
Figure 4: GUI Wireframes	12
Figure 5: Class diagram for Gym Member	13
Figure 6: Class diagram for Regular Member	14
Figure 7: Class diagram for Premium Member.....	15
Figure 8: Class diagram for Gym GUI	15
Figure 9: Complete class diagram.....	16
Figure 10: Command to open the Gym GUI.....	83
Figure 11: Opening from terminal.....	84
Figure 12: Adding Regular Member	85
Figure 13: Adding Premium Member.....	87
Figure 14: Marking attendance for regular member	88
Figure 15: Marking attendance for premium member	89
Figure 16: Upgrading member in progress.....	90
Figure 17: Successfully upgraded member from basic to standard.....	91
Figure 18: Successfully upgraded member from basic to deluxe	91
Figure 19: Calculating discount	93
Figure 20: Full amount was not paid by member	93
Figure 21: Discount applied after full payment	94
Figure 22: Payment not fully paid.....	95
Figure 23: Full payment done.....	96
Figure 24: Before reverting regular member	97
Figure 25: Reverting regular member.....	97
Figure 26: Reverting regular member.....	98
Figure 27: After reverting regular member	98
Figure 28: Before reverting premium member.....	99
Figure 29: Reverting premium member.....	100
Figure 30: After reverting premium member.....	100
Figure 31: Adding all necessary information	102
Figure 32: Member details saved	102
Figure 33: Save to text file.....	103
Figure 34: Read from file	104
Figure 35: Syntax error.....	105
Figure 36: Solving Syntax error.....	106
Figure 37: Runtime error	107
Figure 38: Solving Runtime error	108
Figure 39: Creating logical error	108
Figure 40: Logical error	109

Table of Tables

Table 1: Test 1: Compiling and running using terminal	83
Table 2: Test 2.1: Adding Regular Member.....	84
Table 3: Test 2.2: Adding Premium Member	86
Table 4: Test 3.1: Mark Attendance	87
Table 5: Test 3.2 Upgrade Plan	89
Table 6: Test 4.1: Calculate Discount.....	92
Table 7: Test 4.2: Pay Due Amount	94
Table 8: Test 4.3.1: Revert Members	96
Table 9: Test 4.3.2: Reverting Premium Member.....	98
Table 10: Test 5.1: Saving to file	100
Table 11: Test 5.2: Reading from file	103

1.Introduction

1.1 Java GUI

This project flashes on designing and implementing a graphical user interface (GUI) which manages a gym membership. This system helps a gym staff to keep the track and record about the members by handling tasks like adding new members, activating or deactivating memberships, tracking attendance and calculating discounts. Gym membership class includes two sub classes regular members and premium members. Both these classes have its own unique features and functions. This system also ensures about the duplicate membership IDs.

1.2 Aims

The aim of this project is to use object-oriented concept of java by implementing in a real-world scenario which includes a class to represent a gym member with its two subclasses.

1.3 Objectives

The objectives of this project are as follows:

- Simple Interface: It builds user friendly interface using Java Swing which makes it easy to use.
- Member Management: This allows staff to add, update, manage data of both regular and premium members ensuring about duplicate IDs.
- Track Attendance: This project marks the attendance of the members and provides the loyalty points to the members.
- Error Message: It shows error messages for invalid inputs.

1.4 Tools and Technologies used

The tools and technologies used for developing this project are as follows:

- BlueJ

BlueJ is simple and beginner friendly graphical user interface. BlueJ is used for developing this entire project about Management of Gym Membership for coding purpose.



Figure 1: BlueJ

- Draw.io

Draw.io is used for developing class diagram for this project about Management of Gym Membership.



draw.io

Figure 2: draw.io

- Notepad

Notepad is used in this project for generating text file about member details.



Figure 3: Notepad

2. GUI Wireframes

The wireframe for developing GUI is as given below:

Gym Membership

Add a Regular Member

Name:

ID:

Location:

Mobile:

Email:

Gender: ☐ Male ☐ Female ☐ Other

DOB:

Day

Month

Year

Membership Start Date:

Day

Month

Year

Referral Source:

Add a Regular Member

Add a Premium Member

Name:

ID:

Location:

Mobile:

Email:

Gender: ☐ Male ☐ Female ☐ Other

DOB:

Day

Month

Year

Membership Start Date:

Day

Month

Year

Personal Trainer:

Add a Regular Member

Activate Membership

Membership ID:

Activate

Deactivate Membership

Membership ID:

Deactivate

Mark Attendance

Membership ID:

Is Member Active:

Mark Attendance

Revert Member

Membership ID:

Revert Regular Member

Revert Premium Member

Upgrade Plan

Membership ID:

Is Member Valid:

Upgrade Plan

Calculate Discount

Membership ID:

Is Full Payment Done?:

Calculate Discount

Pay Due Amount

Membership ID:

Remaining Amount:

Pay Due Amount

Display Members

Clear Fields

Save to File

Read from File

Figure 4: GUI Wireframes

3. Class Diagram

The class diagram for all the members classes are as follows:

3.1 Gym Member

The class Diagram for Gym Member is as follows:

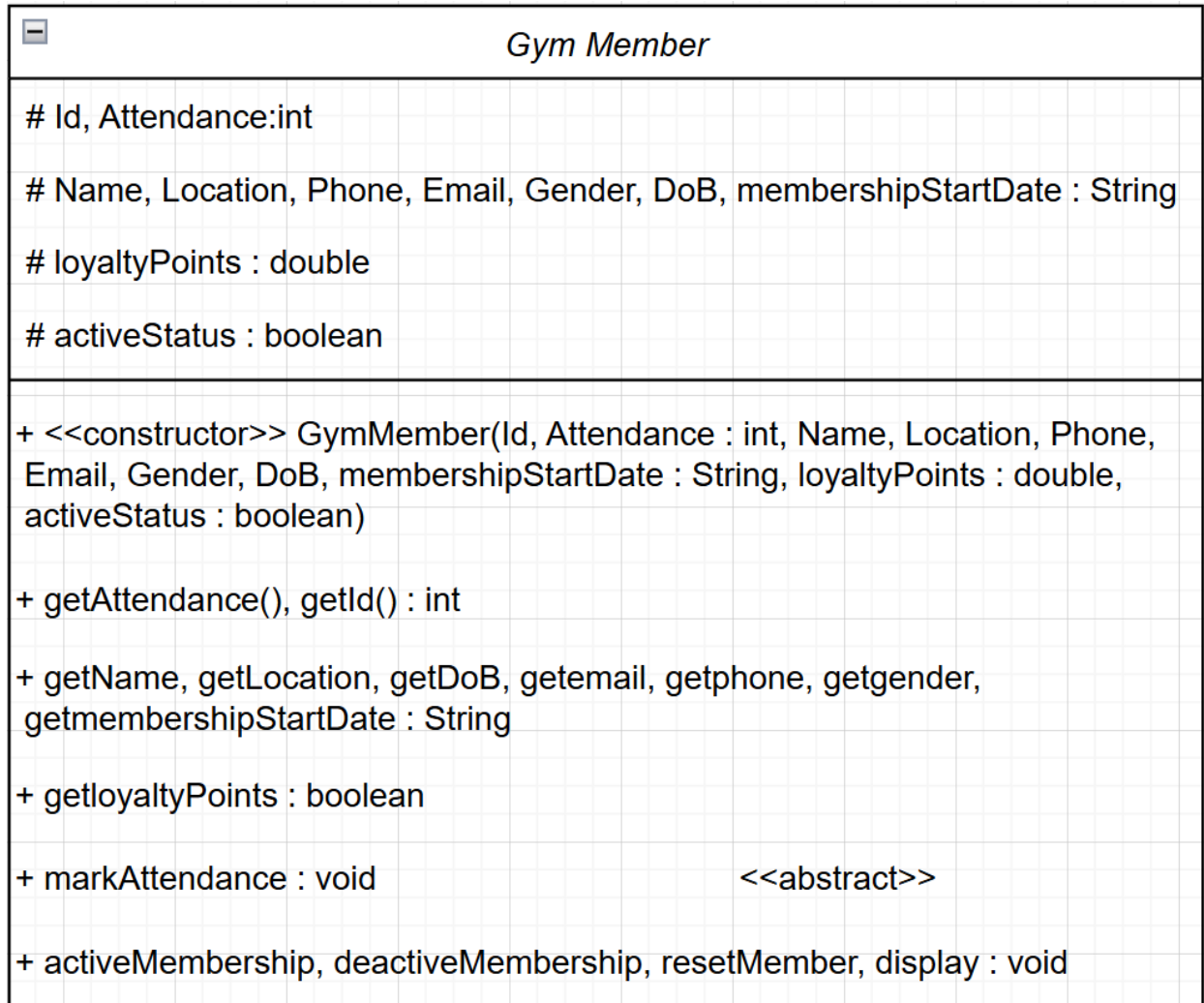


Figure 5: Class diagram for Gym Member

3.2 Regular Member

The class diagram for Regular Member is as follows:

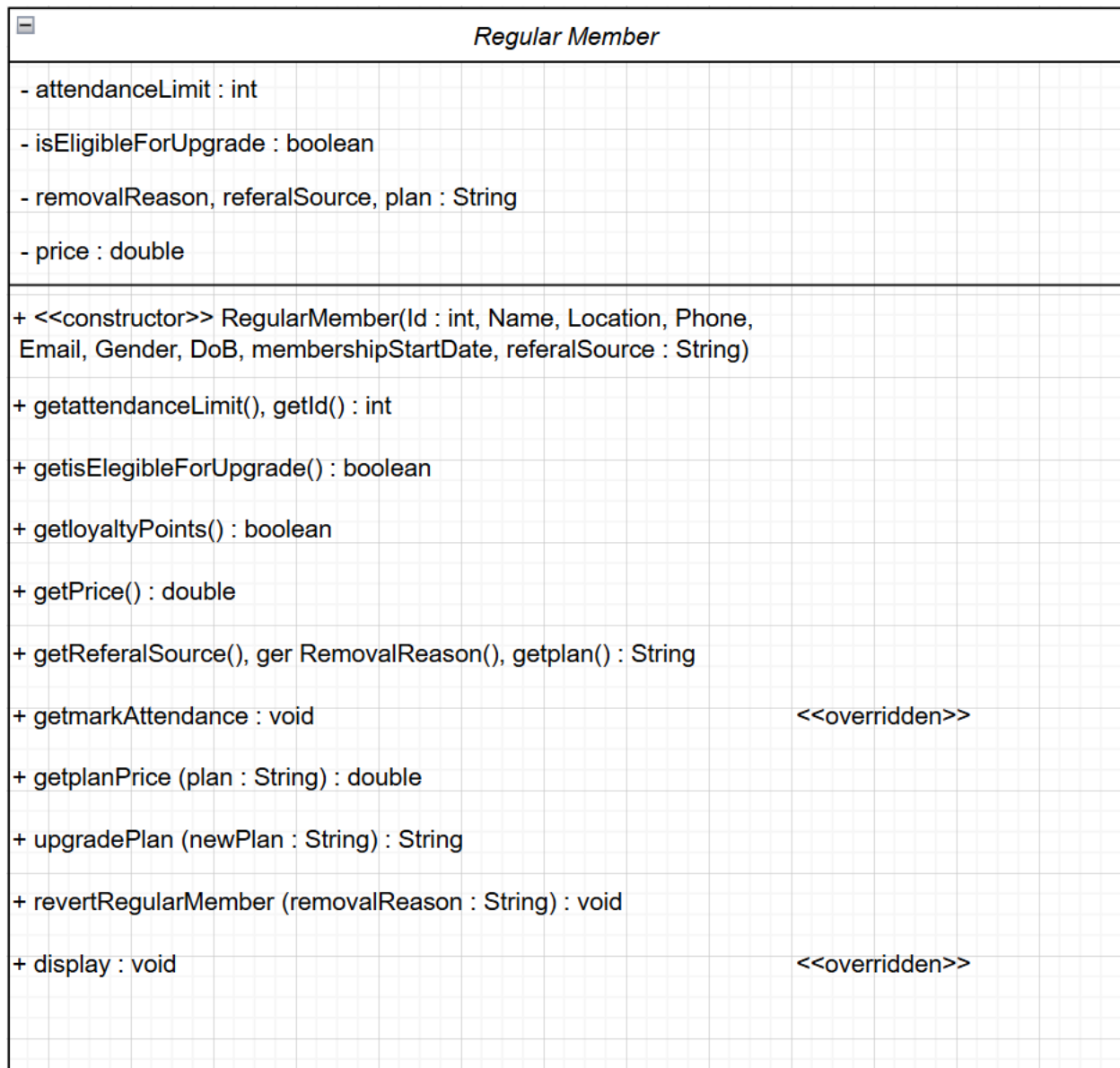


Figure 6: Class diagram for Regular Member

3.3 Premium Member

The class diagram for Premium Member is as follows:

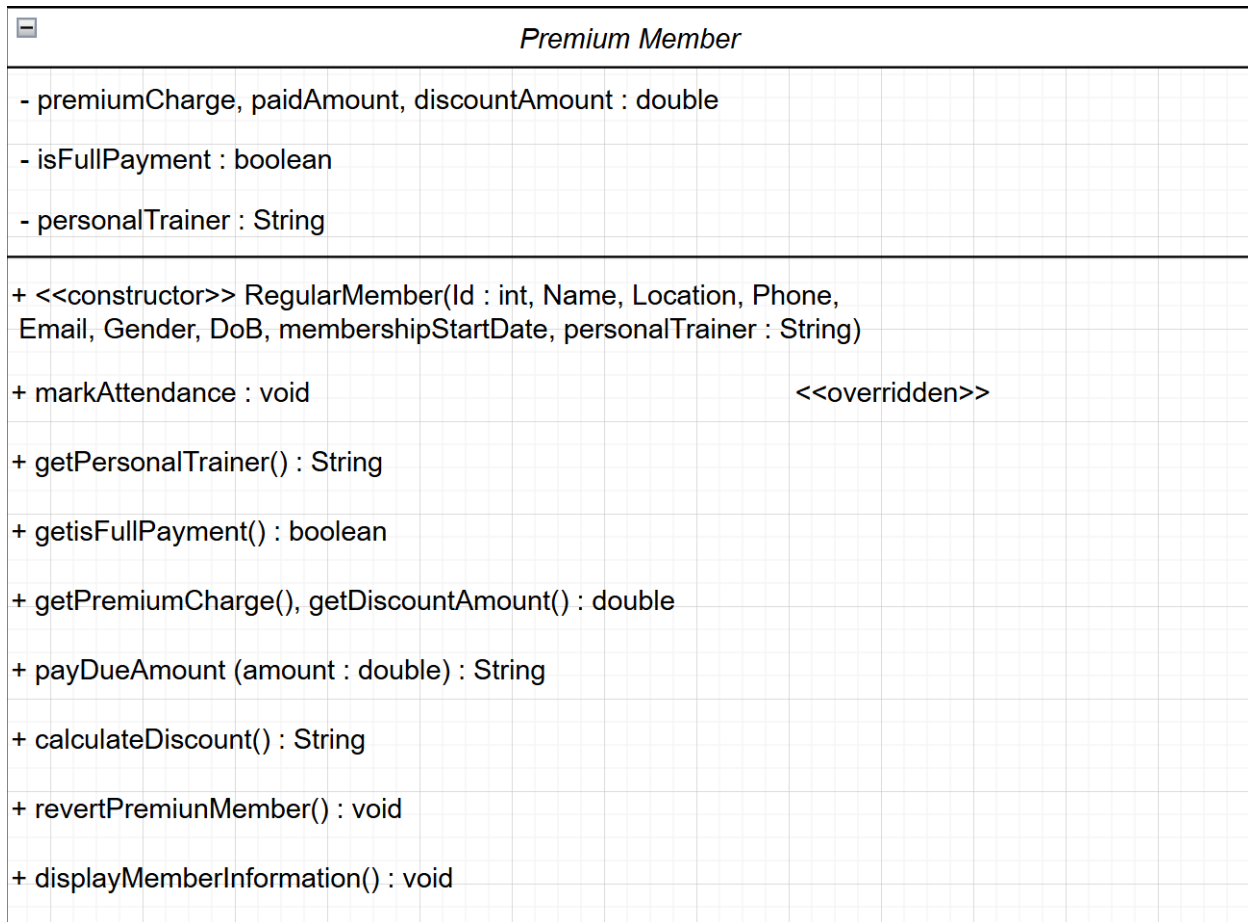


Figure 7: Class diagram for Premium Member

3.4 Gym GUI

The class diagram for Gym GUI is as follows:

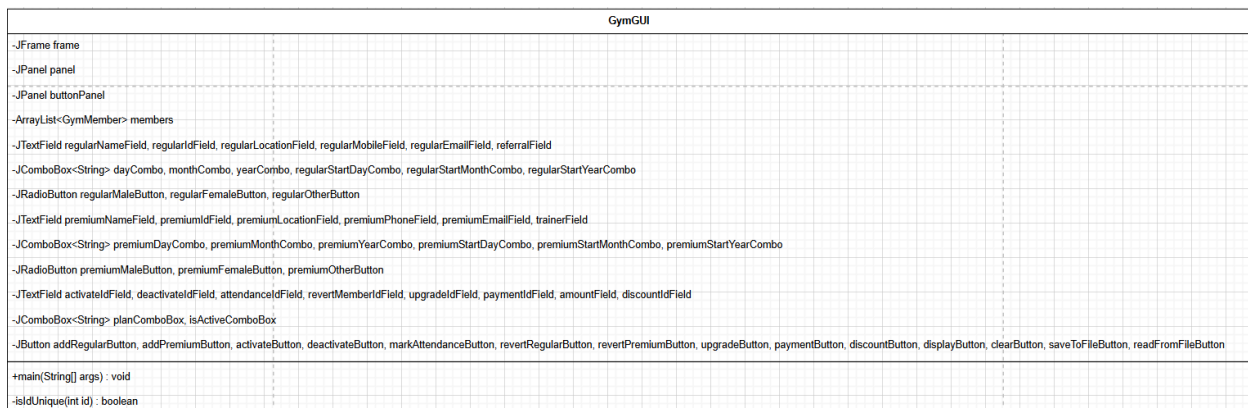


Figure 8: Class diagram for Gym GUI

3.5 Complete Class diagram

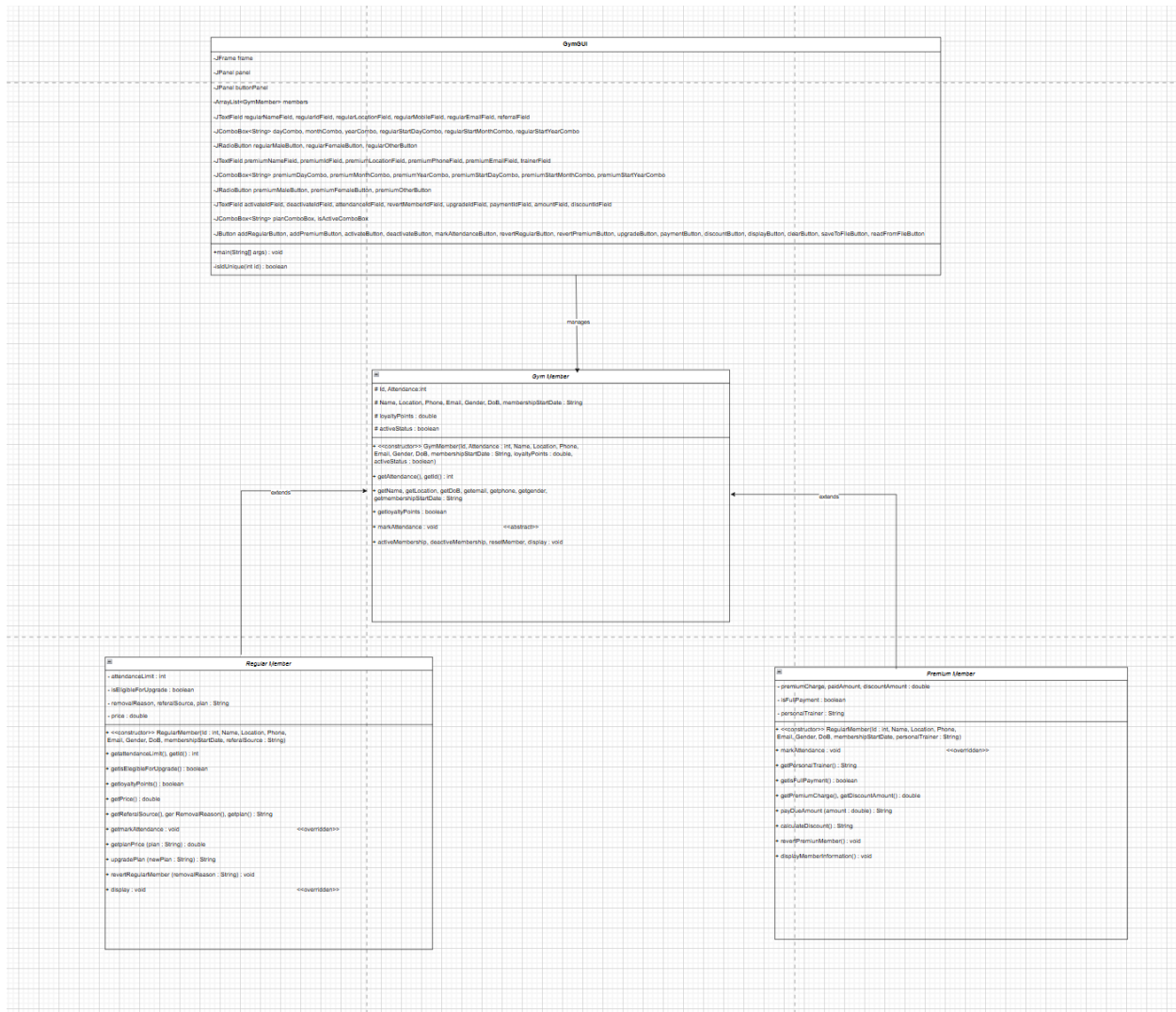


Figure 9: Complete class diagram

4. Pseudocode

The pseudocode for all the members classes are as follows:

4.1 Gym Member

The pseudocode of Gym Member is as follows:

```
CREATE CLASS GymMember AS ABSTRACT
```

```
DO
```

```
    // Declare attributes with protected access
```

```
    DECLARE id, attendance AS INTEGER
```

```
    DECLARE name, location, phone, email, gender, DOB, membershipStartDate AS  
    STRING
```

```
    DECLARE loyaltyPoints AS DOUBLE
```

```
    DECLARE activeStatus AS BOOLEAN
```

```
    <<CONSTRUCTOR>> GymMember(id, name, location, phone, email, gender, DOB,  
    membershipStartDate)
```

```
    DO
```

```
        SET this.id TO id
```

```
        SET this.name TO name
```

```
        SET this.location TO location
```

```
        SET this.phone TO phone
```

```
        SET this.email TO email
```

```
        SET this.gender TO gender
```

```
        SET this.DOB TO DOB
```

```
        SET this.membershipStartDate TO membershipStartDate
```

```
    SET this.attendance TO 0

    SET this.loyaltyPoints TO 0.0

    SET this.activeStatus TO FALSE

END


// Abstract method for marking attendance
ABSTRACT METHOD markAttendance()


// Membership activation/deactivation methods
METHOD activateMembership()

DO

    SET activeStatus TO TRUE

END


METHOD deactivateMembership()

DO

    IF activeStatus IS TRUE THEN

        SET activeStatus TO FALSE

    ELSE

        PRINT "The membership is already inactive"

    END IF

END
```

```
// Method to reset member data
```

```
METHOD resetMember()
```

```
DO
```

```
    SET activeStatus TO FALSE
```

```
    SET attendance TO 0
```

```
    SET loyaltyPoints TO 0.0
```

```
END
```

```
// Method to display member details
```

```
METHOD display()
```

```
DO
```

```
    PRINT "Id: " + id
```

```
    PRINT "Name: " + name
```

```
    PRINT "Location: " + location
```

```
    PRINT "Phone: " + phone
```

```
    PRINT "Email: " + email
```

```
    PRINT "Gender: " + gender
```

```
    PRINT "DOB: " + DOB
```

```
    PRINT "Membership Start Date: " + membershipStartDate
```

```
    PRINT "Attendance: " + attendance
```

```
    PRINT "Loyalty Points: " + loyaltyPoints
```

```
    PRINT "Active Status: " + activeStatus
```

```
END
```

END CLASS

4.2 Regular Member

The pseudocode of Regular Member is as follows:

```
CREATE CLASS RegularMember EXTENDS GymMember
```

```
DO
```

```
    // Attributes
```

```
    DECLARE attendanceLimit AS INTEGER CONSTANT = 30
```

```
    DECLARE isEligibleForUpgrade AS BOOLEAN
```

```
    DECLARE removalReason, referralSource, plan AS STRING
```

```
    DECLARE price AS DOUBLE
```

```
    <<CONSTRUCTOR>> RegularMember(id, name, location, phone, email, gender,  
    DOB, membershipStartDate, referralSource)
```

```
    DO
```

```
        CALL SUPER(id, name, location, phone, email, gender, DOB,  
membershipStartDate)
```

```
        SET isEligibleForUpgrade TO FALSE
```

```
        SET plan TO "basic"
```

```
        SET price TO 6500
```

```
        SET removalReason TO ""
```

```
        SET this.referralSource TO referralSource
```

```
    END
```

```

// Core Methods

METHOD markAttendance() OVERRIDE

DO

    INCREMENT attendance BY 1

    INCREMENT loyaltyPoints BY 5

END

METHOD upgradePlan(newPlan) RETURNING STRING

DO

    IF attendance >= attendanceLimit THEN

        SET isEligibleForUpgrade TO TRUE

    END IF

CASE newPlan OF

    "basic":

        IF plan EQUALS "basic" THEN

            RETURN "Already on basic plan"

        END IF

        SET plan TO "basic"

        SET price TO 6500

        RETURN "Downgraded to basic plan"

    "standard":

```

```

    IF plan EQUALS "standard" THEN
        RETURN "Already on standard plan"
    END IF

    SET plan TO "standard"

    SET price TO 12500

    RETURN "Upgraded to standard plan"

"deluxe":

    IF plan EQUALS "deluxe" THEN
        RETURN "Already on deluxe plan"
    END IF

    SET plan TO "deluxe"

    SET price TO 18500

    RETURN "Upgraded to deluxe plan"

DEFAULT:

    RETURN "Invalid plan selection"

END CASE

END

METHOD revertRegularMember(reason)

DO

    CALL resetMember()

```

```

    SET isEligibleForUpgrade TO FALSE

    SET plan TO "basic"

    SET price TO 6500

    SET removalReason TO reason

END

METHOD display() OVERRIDE
DO
    CALL SUPER.display()

    PRINT "Plan Type: " + plan

    PRINT "Monthly Fee: " + price

    IF removalReason IS NOT EMPTY THEN
        PRINT "Removal Reason: " + removalReason
    END IF

END

END CLASS

```

4.3 Premium Member

The pseudocode for Premium Member is as follows:

```

CREATE CLASS PremiumMember EXTENDS GymMember

DO
    // Attributes

    DECLARE premiumCharge AS DOUBLE CONSTANT = 50000

```

DECLARE personalTrainer AS STRING

DECLARE isFullPayment AS BOOLEAN

DECLARE paidAmount, discountAmount AS DOUBLE

<<CONSTRUCTOR>> PremiumMember(id, name, location, phone, email, gender,
DOB, membershipStartDate, trainer)

DO

CALL SUPER(id, name, location, phone, email, gender, DOB,
membershipStartDate)

SET personalTrainer TO trainer

SET isFullPayment TO FALSE

SET paidAmount TO 0

SET discountAmount TO 0

END

// Core Methods

METHOD markAttendance() OVERRIDE

DO

INCREMENT attendance BY 1

INCREMENT loyaltyPoints BY 10

END

METHOD payDueAmount(amount) RETURNING STRING

DO

IF isFullPayment THEN

RETURN "Payment already completed"

END IF

SET newTotal TO paidAmount + amount

IF newTotal > premiumCharge THEN

RETURN "Payment exceeds total due"

END IF

SET paidAmount TO newTotal

IF paidAmount EQUALS premiumCharge THEN

SET isFullPayment TO TRUE

RETURN "Payment completed. Membership fully paid"

ELSE

SET remaining TO premiumCharge - paidAmount

RETURN "Payment received. Remaining: " + remaining

END IF

END

METHOD calculateDiscount()

DO

IF isFullPayment THEN

SET discountAmount TO premiumCharge * 0.10

PRINT "10% discount applied: " + discountAmount

ELSE

PRINT "Complete payment to qualify for discount"

END IF

END

METHOD revertPremiumMember()

DO

CALL resetMember()

SET personalTrainer TO ""

SET isFullPayment TO FALSE

SET paidAmount TO 0

SET discountAmount TO 0

END

METHOD display() OVERRIDE

DO

CALL SUPER.display()

PRINT "Trainer: " + personalTrainer

PRINT "Paid: " + paidAmount + "/" + premiumCharge

```

    PRINT "Payment Status: " + (isFullPayment ? "FULL" : "PARTIAL")

    IF isFullPayment THEN

        PRINT "Discount Earned: " + discountAmount

    END IF

END

END CLASS

```

4.4 Gym GUI

The pseudocode of Gym GUI is as follows:

```

CREATE CLASS GymGUI

DO

    // 1. DECLARE ALL COMPONENTS

    // Main Frame

    DECLARE frame AS JFrame

    // Panels

    DECLARE regularPanel, premiumPanel, activatePanel, deactivatePanel,
        attendancePanel, revertMemberPanel, upgradePanel, paymentPanel,
        discountPanel AS JPanel

    // Regular Member Components

    DECLARE regularNameLabel, regularIdLabel, regularLocationLabel,

```

```

        regularMobileLabel, regularEmailLabel, regularDobLabel,
        regularGenderLabel, regularStartDateLabel, referralLabel AS JLabel
DECLARE regularNameField, regularIdField, regularLocationField,
        regularMobileField, regularEmailField, referralField AS JTextField
DECLARE regularGenderGroup AS ButtonGroup
DECLARE regularMaleButton, regularFemaleButton, regularOtherButton AS
JRadioButton
DECLARE dayCombo, monthCombo, yearCombo, regularStartDayCombo,
        regularStartMonthCombo, regularStartYearCombo AS JComboBox
DECLARE addRegularButton AS JButton

// Premium Member Components
DECLARE premiumNameLabel, premiumIdLabel, premiumLocationLabel,
        premiumPhoneLabel, premiumEmailLabel, premiumDobLabel,
        premiumGenderLabel, premiumStartDateLabel, trainerLabel AS JLabel
DECLARE premiumNameField, premiumIdField, premiumLocationField,
        premiumPhoneField, premiumEmailField, trainerField AS JTextField
DECLARE premiumGenderGroup AS ButtonGroup
DECLARE premiumMaleButton, premiumFemaleButton, premiumOtherButton AS
JRadioButton
DECLARE premiumDayCombo, premiumMonthCombo, premiumYearCombo,
        premiumStartDayCombo, premiumStartMonthCombo, premiumStartYearCombo
AS JComboBox

```

```
DECLARE addPremiumButton AS JButton
```

```
// Action Panels Components
```

```
DECLARE activateIdLabel, deactivateIdLabel, attendanceIdLabel,  
    isActiveLabel, revertMemberIdLabel, upgradeIdLabel,  
    planLabel, paymentIdLabel, amountLabel, discountIdLabel AS JLabel
```

```
DECLARE activateIdField, deactivateIdField, attendanceIdField,  
    revertMemberIdField, upgradeIdField, paymentIdField,  
    amountField, discountIdField AS JTextField
```

```
DECLARE activateButton, deactivateButton, markAttendanceButton,  
    revertRegularMemberButton, revertPremiumMemberButton,  
    upgradeButton, paymentButton, discountButton AS JButton
```

```
DECLARE isActiveComboBox, planComboBox AS JComboBox
```

```
// Utility Buttons
```

```
DECLARE displayButton, clearButton, saveToFileButton, readFromFileButton AS  
JButton
```

```
// Data Storage
```

```
DECLARE members AS ArrayList<GymMember>
```

```
DECLARE regularMembers AS ArrayList<RegularMember>
```

```
DECLARE premiumMembers AS ArrayList<PremiumMember>
```

```

// 2. INITIALIZE ALL COMPONENTS IN CONSTRUCTOR

<<CONSTRUCTOR>> GymGUI()

DO

    // Initialize main frame

    SET frame TO NEW JFrame("Gym Membership")

    SET frame.size TO (1500, 1500)

    SET frame.defaultCloseOperation TO JFrame.EXIT_ON_CLOSE

    SET frame.layout TO null


    // REGULAR MEMBER PANEL

    SET regularPanel TO NEW JPanel()

    SET regularPanel.bounds TO (20, 20, 650, 370)

    SET regularPanel.layout TO null

    SET regularPanel.border TO "Add a Regular Member"

    frame.add(regularPanel)


    // Name Components

    SET regularNameLabel TO NEW JLabel("Name:")

    SET regularNameLabel.bounds TO (20, 30, 100, 25)

    regularPanel.add(regularNameLabel)


    SET regularNameField TO NEW JTextField()

    SET regularNameField.bounds TO (120, 30, 200, 25)

```

```
regularPanel.add(regularNameField)
```

```
// ID Components
```

```
SET regularIdLabel TO NEW JLabel("ID:")
```

```
SET regularIdLabel.bounds TO (340, 30, 100, 25)
```

```
regularPanel.add(regularIdLabel)
```

```
SET regularIdField TO NEW JTextField()
```

```
SET regularIdField.bounds TO (440, 30, 200, 25)
```

```
regularPanel.add(regularIdField)
```

```
// Location Components
```

```
SET regularLocationLabel TO NEW JLabel("Location:")
```

```
SET regularLocationLabel.bounds TO (20, 70, 100, 25)
```

```
regularPanel.add(regularLocationLabel)
```

```
SET regularLocationField TO NEW JTextField()
```

```
SET regularLocationField.bounds TO (120, 70, 200, 25)
```

```
regularPanel.add(regularLocationField)
```

```
// Mobile Components
```

```
SET regularMobileLabel TO NEW JLabel("Mobile:")
```

```
SET regularMobileLabel.bounds TO (340, 70, 100, 25)
```

```
regularPanel.add(regularMobileLabel)
```

```
SET regularMobileField TO NEW JTextField()
```

```
SET regularMobileField.bounds TO (440, 70, 200, 25)
```

```
regularPanel.add(regularMobileField)
```

```
// Email Components
```

```
SET regularEmailLabel TO NEW JLabel("Email:")
```

```
SET regularEmailLabel.bounds TO (20, 110, 100, 25)
```

```
regularPanel.add(regularEmailLabel)
```

```
SET regularEmailField TO NEW JTextField()
```

```
SET regularEmailField.bounds TO (120, 110, 200, 25)
```

```
regularPanel.add(regularEmailField)
```

```
// DOB Components
```

```
SET regularDobLabel TO NEW JLabel("DOB:")
```

```
SET regularDobLabel.bounds TO (20, 150, 100, 25)
```

```
regularPanel.add(regularDobLabel)
```

```
// Day Combo
```

```
SET days TO ["1", "2", "3", ..., "31"]
```

```
SET dayCombo TO NEW JComboBox(days)
```



```
SET dayCombo.bounds TO (120, 150, 60, 25)
```

```
regularPanel.add(dayCombo)
```

```
// Month Combo
```

```
SET months TO ["Jan","Feb","Mar",,...,"Dec"]
```

```
SET monthCombo TO NEW JComboBox(months)
```

```
SET monthCombo.bounds TO (190, 150, 60, 25)
```

```
regularPanel.add(monthCombo)
```

```
// Year Combo
```

```
SET years TO ["2000","2001",,...,"2025"]
```

```
SET yearCombo TO NEW JComboBox(years)
```

```
SET yearCombo.bounds TO (260, 150, 80, 25)
```

```
regularPanel.add(yearCombo)
```

```
// Gender Components
```

```
SET regularGenderLabel TO NEW JLabel("Gender:")
```

```
SET regularGenderLabel.bounds TO (340, 110, 100, 25)
```

```
regularPanel.add(regularGenderLabel)
```

```
SET regularFemaleButton TO NEW JRadioButton("Female")
```

```
SET regularFemaleButton.bounds TO (440, 110, 70, 25)
```

```
regularPanel.add(regularFemaleButton)
```

```
SET regularMaleButton TO NEW JRadioButton("Male")
```

```
SET regularMaleButton.bounds TO (520, 110, 60, 25)
```

```
regularPanel.add(regularMaleButton)
```

```
SET regularOtherButton TO NEW JRadioButton("Other")
```

```
SET regularOtherButton.bounds TO (590, 110, 140, 25)
```

```
regularPanel.add(regularOtherButton)
```

```
SET regularGenderGroup TO NEW ButtonGroup()
```

```
regularGenderGroup.add(regularFemaleButton)
```

```
regularGenderGroup.add(regularMaleButton)
```

```
regularGenderGroup.add(regularOtherButton)
```

```
// Membership Start Date
```

```
SET regularStartDateLabel TO NEW JLabel("Membership Start Date:")
```

```
SET regularStartDateLabel.bounds TO (20, 190, 200, 25)
```

```
regularPanel.add(regularStartDateLabel)
```

```
SET regularStartDayCombo TO NEW JComboBox(days)
```

```
SET regularStartDayCombo.bounds TO (190, 200, 60, 25)
```

```
regularPanel.add(regularStartDayCombo)
```

```
SET regularStartMonthCombo TO NEW JComboBox(months)
SET regularStartMonthCombo.bounds TO (260, 200, 60, 25)
regularPanel.add(regularStartMonthCombo)
```

```
SET regularStartYearCombo TO NEW JComboBox(years)
SET regularStartYearCombo.bounds TO (330, 200, 80, 25)
regularPanel.add(regularStartYearCombo)
```

```
// Referral Source
```

```
SET referralLabel TO NEW JLabel("Referral Source:")
SET referralLabel.bounds TO (20, 230, 200, 25)
regularPanel.add(referralLabel)
```

```
SET referralField TO NEW JTextField()
SET referralField.bounds TO (220, 230, 200, 25)
regularPanel.add(referralField)
```

```
// Add Regular Member Button
```

```
SET addRegularButton TO NEW JButton("Add a Regular Member")
SET addRegularButton.bounds TO (220, 280, 200, 30)
regularPanel.add(addRegularButton)
```

```
// PREMIUM MEMBER PANEL
```

```
SET premiumPanel TO NEW JPanel()  
SET premiumPanel.bounds TO (20, 400, 650, 420)  
SET premiumPanel.layout TO null  
SET premiumPanel.border TO "Add a Premium Member"  
frame.add(premiumPanel)
```

```
// ID Components
```

```
SET premiumIdLabel TO NEW JLabel("ID:")  
SET premiumIdLabel.bounds TO (20, 30, 100, 25)  
premiumPanel.add(premiumIdLabel)
```

```
SET premiumIdField TO NEW JTextField()  
SET premiumIdField.bounds TO (120, 30, 200, 25)  
premiumPanel.add(premiumIdField)
```

```
// Name Components
```

```
SET premiumNameLabel TO NEW JLabel("Name:")  
SET premiumNameLabel.bounds TO (340, 30, 100, 25)  
premiumPanel.add(premiumNameLabel)
```

```
SET premiumNameField TO NEW JTextField()  
SET premiumNameField.bounds TO (440, 30, 200, 25)  
premiumPanel.add(premiumNameField)
```

```
// Location Components
```

```
SET premiumLocationLabel TO NEW JLabel("Location:")
```

```
SET premiumLocationLabel.bounds TO (20, 70, 100, 25)
```

```
premiumPanel.add(premiumLocationLabel)
```

```
SET premiumLocationField TO NEW JTextField()
```

```
SET premiumLocationField.bounds TO (120, 70, 200, 25)
```

```
premiumPanel.add(premiumLocationField)
```

```
// Phone Components
```

```
SET premiumPhoneLabel TO NEW JLabel("Phone:")
```

```
SET premiumPhoneLabel.bounds TO (340, 70, 100, 25)
```

```
premiumPanel.add(premiumPhoneLabel)
```

```
SET premiumPhoneField TO NEW JTextField()
```

```
SET premiumPhoneField.bounds TO (440, 70, 200, 25)
```

```
premiumPanel.add(premiumPhoneField)
```

```
// Email Components
```

```
SET premiumEmailLabel TO NEW JLabel("Email:")
```

```
SET premiumEmailLabel.bounds TO (20, 110, 100, 25)
```

```
premiumPanel.add(premiumEmailLabel)
```

```
SET premiumEmailField TO NEW JTextField()  
SET premiumEmailField.bounds TO (120, 110, 200, 25)  
premiumPanel.add(premiumEmailField)
```

```
// DOB Components
```

```
SET premiumDobLabel TO NEW JLabel("DOB:")  
SET premiumDobLabel.bounds TO (20, 150, 100, 25)  
premiumPanel.add(premiumDobLabel)
```

```
SET premiumDayCombo TO NEW JComboBox(days)  
SET premiumDayCombo.bounds TO (120, 150, 60, 25)  
premiumPanel.add(premiumDayCombo)
```

```
SET premiumMonthCombo TO NEW JComboBox(months)  
SET premiumMonthCombo.bounds TO (190, 150, 60, 25)  
premiumPanel.add(premiumMonthCombo)
```

```
SET premiumYearCombo TO NEW JComboBox(years)  
SET premiumYearCombo.bounds TO (260, 150, 80, 25)  
premiumPanel.add(premiumYearCombo)
```

```
// Gender Components
```

```
SET premiumGenderLabel TO NEW JLabel("Gender:")
```

```
SET premiumGenderLabel.bounds TO (340, 110, 100, 25)
```

```
premiumPanel.add(premiumGenderLabel)
```

```
SET premiumFemaleButton TO NEW JRadioButton("Female")
```

```
SET premiumFemaleButton.bounds TO (440, 110, 70, 25)
```

```
premiumPanel.add(premiumFemaleButton)
```

```
SET premiumMaleButton TO NEW JRadioButton("Male")
```

```
SET premiumMaleButton.bounds TO (520, 110, 60, 25)
```

```
premiumPanel.add(premiumMaleButton)
```

```
SET premiumOtherButton TO NEW JRadioButton("Other")
```

```
SET premiumOtherButton.bounds TO (590, 110, 140, 25)
```

```
premiumPanel.add(premiumOtherButton)
```

```
SET premiumGenderGroup TO NEW ButtonGroup()
```

```
premiumGenderGroup.add(premiumFemaleButton)
```

```
premiumGenderGroup.add(premiumMaleButton)
```

```
premiumGenderGroup.add(premiumOtherButton)
```

```
// Membership Start Date
```

```
SET premiumStartDateLabel TO NEW JLabel("Membership Start Date:")
```

```
SET premiumStartDateLabel.bounds TO (20, 190, 200, 25)
premiumPanel.add(premiumStartDateLabel)
```

```
SET premiumStartDayCombo TO NEW JComboBox(days)
SET premiumStartDayCombo.bounds TO (190, 200, 60, 25)
premiumPanel.add(premiumStartDayCombo)
```

```
SET premiumStartMonthCombo TO NEW JComboBox(months)
SET premiumStartMonthCombo.bounds TO (260, 200, 60, 25)
premiumPanel.add(premiumStartMonthCombo)
```

```
SET premiumStartYearCombo TO NEW JComboBox(years)
SET premiumStartYearCombo.bounds TO (330, 200, 80, 25)
premiumPanel.add(premiumStartYearCombo)
```

```
// Personal Trainer
```

```
SET trainerLabel TO NEW JLabel("Personal Trainer:")
SET trainerLabel.bounds TO (20, 230, 200, 25)
premiumPanel.add(trainerLabel)
```

```
SET trainerField TO NEW JTextField()
SET trainerField.bounds TO (220, 230, 200, 25)
premiumPanel.add(trainerField)
```



```
// Add Premium Member Button

SET addPremiumButton TO NEW JButton("Add a Premium Member")

SET addPremiumButton.bounds TO (220, 280, 200, 30)

premiumPanel.add(addPremiumButton)
```

```
// ACTION PANELS
```

```
// 1. ACTIVATE MEMBERSHIP PANEL
```

```
SET activatePanel TO NEW JPanel()

SET activatePanel.bounds TO (700, 20, 380, 120)

SET activatePanel.layout TO null

SET activatePanel.border TO "Activate Membership"

frame.add(activatePanel)
```

```
SET activateIdLabel TO NEW JLabel("Membership ID:")

SET activateIdLabel.bounds TO (20, 30, 120, 25)

activatePanel.add(activateIdLabel)
```

```
SET activateIdField TO NEW JTextField()

SET activateIdField.bounds TO (140, 30, 200, 25)

activatePanel.add(activateIdField)
```

```
SET activateButton TO NEW JButton("Activate")
```

```
SET activateButton.bounds TO (140, 70, 100, 25)
```

```
activatePanel.add(activateButton)
```

```
// 2. DEACTIVATE MEMBERSHIP PANEL
```

```
SET deactivatePanel TO NEW JPanel()
```

```
SET deactivatePanel.bounds TO (700, 150, 380, 120)
```

```
SET deactivatePanel.layout TO null
```

```
SET deactivatePanel.border TO "Deactivate Membership"
```

```
frame.add(deactivatePanel)
```

```
SET deactivateIdLabel TO NEW JLabel("Membership ID:")
```

```
SET deactivateIdLabel.bounds TO (20, 30, 120, 25)
```

```
deactivatePanel.add(deactivateIdLabel)
```

```
SET deactivateIdField TO NEW JTextField()
```

```
SET deactivateIdField.bounds TO (140, 30, 200, 25)
```

```
deactivatePanel.add(deactivateIdField)
```

```
SET deactivateButton TO NEW JButton("Deactivate")
```

```
SET deactivateButton.bounds TO (140, 70, 100, 25)
```

```
deactivatePanel.add(deactivateButton)
```

```
// 3. ATTENDANCE PANEL
```

```
SET attendancePanel TO NEW JPanel()
```

```
SET attendancePanel.bounds TO (700, 290, 380, 150)
```

```
SET attendancePanel.layout TO null
```

```
SET attendancePanel.border TO "Mark Attendance"
```

```
frame.add(attendancePanel)
```

```
SET attendanceIdLabel TO NEW JLabel("Membership ID:")
```

```
SET attendanceIdLabel.bounds TO (20, 30, 120, 25)
```

```
attendancePanel.add(attendanceIdLabel)
```

```
SET attendanceIdField TO NEW JTextField()
```

```
SET attendanceIdField.bounds TO (140, 30, 200, 25)
```

```
attendancePanel.add(attendanceIdField)
```

```
SET isActiveLabel TO NEW JLabel("Is Member Active:")
```

```
SET isActiveLabel.bounds TO (20, 70, 120, 25)
```

```
attendancePanel.add(isActiveLabel)
```

```
SET isActiveComboBox TO NEW JComboBox(["Yes", "No"])
```

```
SET isActiveComboBox.bounds TO (140, 70, 200, 25)
```

```
attendancePanel.add(isActiveComboBox)
```

```
SET markAttendanceButton TO NEW JButton("Mark Attendance")
```

```
SET markAttendanceButton.bounds TO (140, 110, 150, 25)
```

```
attendancePanel.add(markAttendanceButton)
```

```
// 4. REVERT MEMBER PANEL
```

```
SET revertMemberPanel TO NEW JPanel()
```

```
SET revertMemberPanel.bounds TO (700, 450, 400, 150)
```

```
SET revertMemberPanel.layout TO null
```

```
SET revertMemberPanel.border TO "Revert Member"
```

```
frame.add(revertMemberPanel)
```

```
SET revertMemberIdLabel TO NEW JLabel("Member ID:")
```

```
SET revertMemberIdLabel.bounds TO (20, 30, 100, 25)
```

```
revertMemberPanel.add(revertMemberIdLabel)
```

```
SET revertMemberIdField TO NEW JTextField()
```

```
SET revertMemberIdField.bounds TO (120, 30, 200, 25)
```

```
revertMemberPanel.add(revertMemberIdField)
```

```
SET revertRegularMemberButton TO NEW JButton("Revert RegularMember")
```

```
SET revertRegularMemberButton.bounds TO (20, 70, 180, 30)
```

```
revertMemberPanel.add(revertRegularMemberButton)
```

```
SET revertPremiumMemberButton TO NEW JButton("Revert PremiumMember")  
SET revertPremiumMemberButton.bounds TO (210, 70, 180, 30)  
revertMemberPanel.add(revertPremiumMemberButton)
```

```
// 5. UPGRADE PLAN PANEL
```

```
SET upgradePanel TO NEW JPanel()  
SET upgradePanel.bounds TO (1100, 20, 380, 150)  
SET upgradePanel.layout TO null  
SET upgradePanel.border TO "Upgrade Plan"  
frame.add(upgradePanel)
```

```
SET upgradeldLabel TO NEW JLabel("Membership ID:")  
SET upgradeldLabel.bounds TO (20, 30, 120, 25)  
upgradePanel.add(upgradeldLabel)
```

```
SET upgradeldField TO NEW JTextField()  
SET upgradeldField.bounds TO (140, 30, 200, 25)  
upgradePanel.add(upgradeldField)
```

```
SET planLabel TO NEW JLabel("Select Plan:")  
SET planLabel.bounds TO (20, 70, 120, 25)  
upgradePanel.add(planLabel)
```

```
SET planComboBox TO NEW JComboBox(["basic", "standard", "deluxe"])
```

```
SET planComboBox.bounds TO (140, 70, 200, 25)
```

```
upgradePanel.add(planComboBox)
```

```
SET upgradeButton TO NEW JButton("Upgrade Plan")
```

```
SET upgradeButton.bounds TO (140, 110, 150, 25)
```

```
upgradePanel.add(upgradeButton)
```

```
// 6. PAYMENT PANEL
```

```
SET paymentPanel TO NEW JPanel()
```

```
SET paymentPanel.bounds TO (1100, 310, 380, 150)
```

```
SET paymentPanel.layout TO null
```

```
SET paymentPanel.border TO "Pay Due Amount"
```

```
frame.add(paymentPanel)
```

```
SET paymentIdLabel TO NEW JLabel("Membership ID:")
```

```
SET paymentIdLabel.bounds TO (20, 30, 120, 25)
```

```
paymentPanel.add(paymentIdLabel)
```

```
SET paymentIdField TO NEW JTextField()
```

```
SET paymentIdField.bounds TO (140, 30, 200, 25)
```

```
paymentPanel.add(paymentIdField)
```

```
SET amountLabel TO NEW JLabel("Amount to Pay:")
```

```
SET amountLabel.bounds TO (20, 70, 120, 25)
```

```
paymentPanel.add(amountLabel)
```

```
SET amountField TO NEW JTextField()
```

```
SET amountField.bounds TO (140, 70, 200, 25)
```

```
paymentPanel.add(amountField)
```

```
SET paymentButton TO NEW JButton("Pay Due Amount")
```

```
SET paymentButton.bounds TO (140, 110, 150, 25)
```

```
paymentPanel.add(paymentButton)
```

```
// 7. DISCOUNT PANEL
```

```
SET discountPanel TO NEW JPanel()
```

```
SET discountPanel.bounds TO (1100, 180, 380, 120)
```

```
SET discountPanel.layout TO null
```

```
SET discountPanel.border TO "Calculate Discount"
```

```
frame.add(discountPanel)
```

```
SET discountIdLabel TO NEW JLabel("Membership ID:")
```

```
SET discountIdLabel.bounds TO (20, 30, 120, 25)
```

```
discountPanel.add(discountIdLabel)
```

```
SET discountIdField TO NEW JTextField()  
SET discountIdField.bounds TO (140, 30, 200, 25)  
discountPanel.add(discountIdField)
```

```
SET discountButton TO NEW JButton("Calculate Discount")  
SET discountButton.bounds TO (140, 70, 150, 25)  
discountPanel.add(discountButton)
```

```
// UTILITY BUTTONS
```

```
SET displayButton TO NEW JButton("Display Members")  
SET displayButton.bounds TO (500, 900, 140, 30)  
frame.add(displayButton)
```

```
SET clearButton TO NEW JButton("Clear Fields")  
SET clearButton.bounds TO (700, 900, 140, 30)  
frame.add(clearButton)
```

```
SET saveToFileButton TO NEW JButton("Save to File")  
SET saveToFileButton.bounds TO (500, 950, 140, 30)  
frame.add(saveToFileButton)
```

```
SET readFromFileButton TO NEW JButton("Read from File")  
SET readFromFileButton.bounds TO (700, 950, 140, 30)
```



```
frame.add(readFromFileButton)
```

```
// DATA STORAGE INIT
```

```
SET members TO NEW ArrayList<GymMember>()
```

```
SET regularMembers TO NEW ArrayList<RegularMember>()
```

```
SET premiumMembers TO NEW ArrayList<PremiumMember>()
```

```
// SET ACTION LISTENERS
```

```
SET addRegularButton.actionListener TO handleAddRegularMember()
```

```
SET addPremiumButton.actionListener TO handleAddPremiumMember()
```

```
SET activateButton.actionListener TO handleActivateMembership()
```

```
SET deactivateButton.actionListener TO handleDeactivateMembership()
```

```
SET markAttendanceButton.actionListener TO handleMarkAttendance()
```

```
SET revertRegularMemberButton.actionListener TO  
handleRevertRegularMember()
```

```
SET revertPremiumMemberButton.actionListener TO  
handleRevertPremiumMember()
```

```
SET upgradeButton.actionListener TO handleUpgradePlan()
```

```
SET paymentButton.actionListener TO handlePayment()
```

```
SET discountButton.actionListener TO handleCalculateDiscount()
```

```
SET displayButton.actionListener TO handleDisplayMembers()
```

```
SET clearButton.actionListener TO handleClearFields()
```

```

SET saveToFileButton.actionListener TO handleSaveToFile()

SET readFromFileButton.actionListener TO handleReadFromFile()


// SHOW FRAME

SET frame.visible TO true

END

METHOD handleAddRegularMember()

DO

    // Validate all required fields

    IF regularIdField.text IS empty OR
        regularNameField.text IS empty OR
        regularLocationField.text IS empty OR
        regularMobileField.text IS empty OR
        regularEmailField.text IS empty OR
        referralField.text IS empty THEN

        SHOW "All fields are required" WITH ERROR_ICON

        RETURN

    END IF


    // Validate ID is numeric

    TRY

        SET id TO INTEGER(regularIdField.text)

    CATCH

```

```
    SHOW "ID must be a number" WITH ERROR_ICON

    RETURN

END TRY

// Check ID uniqueness

IF NOT isIdUnique(id) THEN

    SHOW "Member ID already exists" WITH WARNING_ICON

    RETURN

END IF

// Get gender selection

SET gender TO ""

IF regularMaleButton.selected THEN

    SET gender TO "Male"

ELSE IF regularFemaleButton.selected THEN

    SET gender TO "Female"

ELSE IF regularOtherButton.selected THEN

    SET gender TO "Other"

ELSE

    SHOW "Please select gender" WITH ERROR_ICON

    RETURN

END IF
```

```

// Create date strings

SET dob TO dayCombo.selectedItem + "-" + monthCombo.selectedItem + "-" +
yearCombo.selectedItem

SET startDate TO regularStartDayCombo.selectedItem + "-" +
        regularStartMonthCombo.selectedItem + "-" +
        regularStartYearCombo.selectedItem


// Create new RegularMember

SET newMember TO NEW RegularMember(
    id,
    regularNameField.text,
    regularLocationField.text,
    regularMobileField.text,
    regularEmailField.text,
    gender,
    dob,
    startDate,
    referralField.text
)


// Add to collections

members.add(newMember)

regularMembers.add(newMember)

```

```
// Show success and clear fields  
  
SHOW "Regular member added successfully!" WITH INFO_ICON  
  
clearRegularMemberFields()  
  
END
```

```
METHOD handleAddPremiumMember()  
  
DO  
  
    // Validate all required fields  
  
    IF premiumIdField.text IS empty OR  
       premiumNameField.text IS empty OR  
       premiumLocationField.text IS empty OR  
       premiumPhoneField.text IS empty OR  
       premiumEmailField.text IS empty OR  
       trainerField.text IS empty THEN  
        SHOW "All fields are required" WITH ERROR_ICON  
        RETURN  
    END IF
```

```
    // Validate ID is numeric  
  
    TRY  
        SET id TO INTEGER(premiumIdField.text)  
    CATCH
```

```
    SHOW "ID must be a number" WITH ERROR_ICON

    RETURN

END TRY

// Check ID uniqueness

IF NOT isIdUnique(id) THEN

    SHOW "Member ID already exists" WITH WARNING_ICON

    RETURN

END IF

// Get gender selection

SET gender TO ""

IF premiumMaleButton.selected THEN

    SET gender TO "Male"

ELSE IF premiumFemaleButton.selected THEN

    SET gender TO "Female"

ELSE IF premiumOtherButton.selected THEN

    SET gender TO "Other"

ELSE

    SHOW "Please select gender" WITH ERROR_ICON

    RETURN

END IF
```

```

// Create date strings

SET dob TO premiumDayCombo.selectedItem + "-" +
premiumMonthCombo.selectedItem + "-" + premiumYearCombo.selectedItem

SET startDate TO premiumStartDayCombo.selectedItem + "-" +
    premiumStartMonthCombo.selectedItem + "-" +
    premiumStartYearCombo.selectedItem


// Create new PremiumMember

SET newMember TO NEW PremiumMember(
    id,
    premiumNameField.text,
    premiumLocationField.text,
    premiumPhoneField.text,
    premiumEmailField.text,
    gender,
    dob,
    startDate,
    trainerField.text
)


// Add to collections

members.add(newMember)

premiumMembers.add(newMember)

```

```

// Show success and clear fields

SHOW "Premium member added successfully!" WITH INFO_ICON

clearPremiumMemberFields()

END

METHOD handleActivateMembership()

DO

    // Get and validate input

    SET idInput TO activateIdField.text.trim()

    IF idInput IS empty THEN

        SHOW "Please enter Member ID" WITH ERROR_ICON

        RETURN

    END IF

TRY

    SET id TO INTEGER(idInput)

    // Search for member

    FOR EACH member IN members

        IF member.id == id THEN

            IF member.activeStatus THEN

                SHOW "Membership is already active" WITH INFO_ICON

```



```

        ELSE

            CALL member.activeMembership()

            SHOW "Membership activated successfully!" WITH INFO_ICON

        END IF

        RETURN

    END IF

END FOR

SHOW "Member ID not found" WITH WARNING_ICON

CATCH

    SHOW "Invalid Member ID format" WITH ERROR_ICON

END TRY

END

METHOD handleDeactivateMembership()

DO

    // Get and validate input

    SET idInput TO deactivateIdField.text.trim()

    IF idInput IS empty THEN

        SHOW "Please enter Member ID" WITH ERROR_ICON

        RETURN

    END IF

```

```

TRY

    SET id TO INTEGER(idInput)

    // Search for member

    FOR EACH member IN members

        IF member.id == id THEN

            IF NOT member.activeStatus THEN

                SHOW "Membership is already inactive" WITH INFO_ICON

            ELSE

                CALL member.deactivateMembership()

                SHOW "Membership deactivated successfully!" WITH INFO_ICON

            END IF

            RETURN

        END IF

    END FOR

    SHOW "Member ID not found" WITH WARNING_ICON

CATCH

    SHOW "Invalid Member ID format" WITH ERROR_ICON

END TRY

END

METHOD handleMarkAttendance()

```

DO

// Get and validate input

SET idInput TO attendanceldField.text.trim()

IF idInput IS empty THEN

 SHOW "Please enter Member ID" WITH ERROR_ICON

 RETURN

END IF

TRY

 SET id TO INTEGER(idInput)

 // Search for member

 FOR EACH member IN members

 IF member.id == id THEN

 IF isActiveComboBox.selectedItem == "No" AND member.activeStatus

THEN

 SHOW "Cannot mark attendance for inactive member" WITH
ERROR_ICON

 RETURN

 END IF

 CALL member.markAttendance()

 SHOW "Attendance marked successfully!" WITH INFO_ICON

```

        RETURN

    END IF

END FOR

    SHOW "Member ID not found" WITH WARNING_ICON

CATCH

    SHOW "Invalid Member ID format" WITH ERROR_ICON

END TRY

END

METHOD handleRevertRegularMember()

DO

    // Get and validate input

    SET idInput TO revertMemberIdField.text.trim()

    IF idInput IS empty THEN

        SHOW "Please enter Member ID" WITH ERROR_ICON

        RETURN

    END IF

    // Search for member

    FOR EACH member IN members

        IF STRING(member.id) == idInput THEN

            IF member IS INSTANCE OF RegularMember THEN

```

```

    SET removalReason TO SHOW_INPUT_DIALOG("Enter removal reason:")

    IF removalReason IS empty THEN

        SHOW "Removal reason is required" WITH ERROR_ICON

        RETURN

    END IF

    CALL member.revertRegularMember(removalReason)

    SHOW "Regular member reverted successfully" WITH INFO_ICON

ELSE

    SHOW "This member is not a Regular member" WITH ERROR_ICON

END IF

RETURN

END IF

END FOR

SHOW "Member ID not found" WITH WARNING_ICON

END

METHOD handleRevertPremiumMember()

DO

    // Get and validate input

    SET idInput TO revertMemberIdField.text.trim()

    IF idInput IS empty THEN

```

```

        SHOW "Please enter Member ID" WITH ERROR_ICON

        RETURN
    END IF

    // Search for member

    FOR EACH member IN members

        IF STRING(member.id) == idInput THEN

            IF member IS INSTANCE OF PremiumMember THEN

                CALL member.revertPremiumMember()

                SHOW "Premium member reverted successfully" WITH INFO_ICON

            ELSE

                SHOW "This member is not a Premium member" WITH ERROR_ICON

            END IF

            RETURN

        END IF

    END FOR

    SHOW "Member ID not found" WITH WARNING_ICON

END

METHOD handleUpgradePlan()

DO

    // Get and validate input

```

```

SET idInput TO upgradeldField.text.trim()

IF idInput IS empty THEN

    SHOW "Please enter Member ID" WITH ERROR_ICON

    RETURN

END IF


TRY

    SET id TO INTEGER(idInput)

    SET selectedPlan TO planComboBox.selectedItem


    // Search for member

    FOR EACH member IN members

        IF member.id == id THEN

            IF member IS INSTANCE OF RegularMember THEN

                SET result TO member.upgradePlan(selectedPlan)


                // Prepare detailed status message

                SET statusMsg TO "Current Plan: " + member.plan + NEWLINE +
                    "Attendance: " + member.attendance + "/" +
member.attendanceLimit + NEWLINE +
                    "Status: " + (member.activeStatus ? "Active" : "Inactive") +
NEWLINE + NEWLINE +
                    "Upgrade Result: " + result

```

```

        SHOW statusMsg WITH INFO_ICON
    ELSE
        SHOW "Only Regular Members can upgrade plans" WITH
ERROR_ICON
    END IF
    RETURN
END IF
END FOR

    SHOW "Member ID not found" WITH WARNING_ICON
CATCH
    SHOW "Invalid Member ID format" WITH ERROR_ICON
END TRY
END

METHOD handlePayment()
DO
    // Get and validate inputs
    SET idInput TO paymentIdField.text.trim()
    SET amountInput TO amountField.text.trim()

    IF idInput IS empty OR amountInput IS empty THEN

```



```

    SHOW "Please enter both Member ID and Amount" WITH ERROR_ICON

    RETURN

END IF

TRY

    SET id TO INTEGER(idInput)

    SET amount TO DOUBLE(amountInput)

    IF amount <= 0 THEN

        SHOW "Amount must be positive" WITH ERROR_ICON

        RETURN

    END IF

    // Search for member

    FOR EACH member IN members

        IF member.id == id THEN

            IF member IS INSTANCE OF PremiumMember THEN

                SET result TO member.payDueAmount(amount)

                // Show detailed payment information

                SET paymentMsg TO "Payment Amount: " + amount + NEWLINE +

                    "Previous Due: " + (member.premiumCharge -
member.amountPaid) + NEWLINE +

```

```
        "New Due: " + (member.premiumCharge -  
member.amountPaid) + NEWLINE +
```

```
        "Payment Status: " + result
```

```
    SHOW paymentMsg WITH INFO_ICON
```

```
    amountField.text = "" // Clear amount field after payment
```

```
ELSE
```

```
    SHOW "Only Premium Members can make payments" WITH  
ERROR_ICON
```

```
END IF
```

```
RETURN
```

```
END IF
```

```
END FOR
```

```
    SHOW "Member ID not found" WITH WARNING_ICON
```

```
CATCH
```

```
    SHOW "Invalid Member ID or Amount format" WITH ERROR_ICON
```

```
END TRY
```

```
END
```

```
METHOD handleCalculateDiscount()
```

```
DO
```

```
    // Get and validate input
```

```

SET idInput TO discountIdField.text.trim()

IF idInput IS empty THEN

    SHOW "Please enter Member ID" WITH ERROR_ICON

    RETURN

END IF


TRY

    SET id TO INTEGER(idInput)


    // Search for member

    FOR EACH member IN members

        IF member.id == id THEN

            IF member IS INSTANCE OF PremiumMember THEN

                IF NOT member.isFullPayment() THEN

                    SHOW "Full payment not completed yet" WITH WARNING_ICON

                    RETURN

                END IF

            CALL member.calculateDiscount()


            // Show discount details

            SET discountMsg TO "Premium Charge: " + member.premiumCharge +
NEWLINE +

```

```

        "Amount Paid: " + member.amountPaid + NEWLINE +
        "Discount Applied: " + member.discountAmount + NEWLINE
+
        "Final Amount: " + (member.premiumCharge -
member.discountAmount)

```

```

        SHOW discountMsg WITH INFO_ICON
    ELSE
        SHOW "Only Premium Members can calculate discounts" WITH
ERROR_ICON
    END IF
    RETURN
END IF
END FOR

```

```

        SHOW "Member ID not found" WITH WARNING_ICON
    CATCH
        SHOW "Invalid Member ID format" WITH ERROR_ICON
    END TRY
END

```

```

METHOD handleDisplayMembers()
DO

```

```

IF members IS empty THEN

    SHOW "No members to display" WITH INFO_ICON

    RETURN

END IF


// Create display text with header

SET displayText TO ""

SET header TO FORMAT("%-5s %-15s %-10s %-15s %-10s %-15s %-15s %-10s",
                    "ID", "Name", "Type", "Plan", "Status", "Attendance", "Loyalty",
                    "Balance")

displayText = displayText + header + NEWLINE + NEWLINE


// Add each member's information

FOR EACH member IN members

    SET type TO (member IS INSTANCE OF RegularMember) ? "Regular" :
    "Premium"

    SET plan TO (member IS INSTANCE OF RegularMember) ? member.plan :
    "Premium"

    SET status TO member.activeStatus ? "Active" : "Inactive"

    SET balance TO (member IS INSTANCE OF PremiumMember) ?
    (member.premiumCharge - member.amountPaid) : 0.0

```

```
        SET row TO FORMAT("%-5d %-15s %-10s %-15s %-10s %-15d %-15.2f %-10.2f",
```

```
            member.id, member.name, type, plan, status,
```

```
            member.attendance, member.loyaltyPoints, balance)
```

```
        displayText = displayText + row + NEWLINE
```

```
    END FOR
```

```
    // Show in scrollable dialog
```

```
    SHOW_SCROLLABLE_DIALOG("All Members", displayText)
```

```
END
```

```
METHOD handleClearFields()
```

```
DO
```

```
    // Clear all input fields
```

```
    clearRegularMemberFields()
```

```
    clearPremiumMemberFields()
```

```
    // Clear action panels
```

```
    activateIdField.text = ""
```

```
    deactivateIdField.text = ""
```

```
    attendanceIdField.text = ""
```

```
    revertMemberIdField.text = ""
```

```

upgradeldField.text = ""

paymentldField.text = ""

amountField.text = ""

discountldField.text = ""


// Reset combo boxes

isActiveComboBox.selectedIndex = 0

planComboBox.selectedIndex = 0


SHOW "All fields cleared successfully" WITH INFO_ICON

END


METHOD handleSaveToFile()

DO

    IF members IS empty THEN

        SHOW "No members to save" WITH WARNING_ICON

        RETURN

    END IF


TRY

    // Open file for writing

    SET writer TO NEW FileWriter("MemberDetails.txt")

```

```

// Write header

writer.write(FORMAT("%-5s %-15s %-15s %-15s %-25s %-20s %-10s %-10s %-
10s %-15s %-10s %-15s %-15s",

                "ID", "Name", "Location", "Phone", "Email", "Membership Start",

                "Plan", "Price", "Attendance", "Loyalty", "Active", "Full Pay",

                "Discount"))

writer.write(NEWLINE)


// Write each member's details

FOR EACH member IN members

    IF member IS INSTANCE OF RegularMember THEN

        writer.write(FORMAT("%-5d %-15s %-15s %-15s %-25s %-20s %-10s %-
10.2f %-10d %-15.2f %-10s %-15s %-15.2f",

                            member.id, member.name, member.location, member.phone,
member.email,

                            member.membershipStartDate, member.plan, member.price,
member.attendance,

                            member.loyaltyPoints, (member.activeStatus ? "Yes" : "No"),
"N/A", 0.0))

    ELSE IF member IS INSTANCE OF PremiumMember THEN

        // Ensure discount is calculated if payment is complete

        IF member.isFullPayment() AND member.discountAmount == 0 THEN

            CALL member.calculateDiscount()

        END IF

```



```

        writer.write(FORMAT("%-5d %-15s %-15s %-15s %-25s %-20s %-10s %-
10.2f %-10d %-15.2f %-10s %-15s %-15.2f",

            member.id, member.name, member.location, member.phone,
member.email,

            member.membershipStartDate, "Premium",
member.premiumCharge, member.attendance,

            member.loyaltyPoints, (member.activeStatus ? "Yes" : "No"),
            (member.isFullPayment() ? "Yes" : "No"),
member.discountAmount))

        END IF

        writer.write(NEWLINE)

    END FOR

    writer.close()

    SHOW "Member details saved to MemberDetails.txt" WITH INFO_ICON

    CATCH IOException AS e

        SHOW "Error saving file: " + e.message WITH ERROR_ICON

    END TRY

END

METHOD handleReadFromFile()

DO

    TRY

```

```

// Check if file exists

IF NOT FILE_EXISTS("MemberDetails.txt") THEN

    SHOW "File not found: MemberDetails.txt" WITH ERROR_ICON

    RETURN

END IF


// Open file for reading

SET reader TO NEW BufferedReader(NEW FileReader("MemberDetails.txt"))

SET content TO ""

SET line TO reader.readLine()


// Read file line by line

WHILE line IS NOT null

    content = content + line + NEWLINE

    line = reader.readLine()

END WHILE


reader.close()


// Show content in scrollable dialog

SHOW_SCROLLABLE_DIALOG("Member Details from File", content)

CATCH FileNotFoundException

    SHOW "File not found: MemberDetails.txt" WITH ERROR_ICON

```

```

    CATCH IOException AS e

        SHOW "Error reading file: " + e.message WITH ERROR_ICON

    END TRY

END

// HELPER METHOD

METHOD isIdUnique(id)

DO

    FOR EACH member IN members

        IF member.id == id THEN

            RETURN false

        END IF

    END FOR

    RETURN true

END

METHOD clearRegularMemberFields()

DO

    regularIdField.text = ""

    regularNameField.text = ""

    regularLocationField.text = ""

    regularMobileField.text = ""

    regularEmailField.text = ""

    referralField.text = ""

```

```
regularGenderGroup.clearSelection()

dayCombo.selectedIndex = 0

monthCombo.selectedIndex = 0

yearCombo.selectedIndex = 0

regularStartDayCombo.selectedIndex = 0

regularStartMonthCombo.selectedIndex = 0

regularStartYearCombo.selectedIndex = 0

END
```

```
METHOD clearPremiumMemberFields()
```

```
DO
```

```
premiumIdField.text = ""

premiumNameField.text = ""

premiumLocationField.text = ""

premiumPhoneField.text = ""

premiumEmailField.text = ""

trainerField.text = ""

premiumGenderGroup.clearSelection()

premiumDayCombo.selectedIndex = 0

premiumMonthCombo.selectedIndex = 0

premiumYearCombo.selectedIndex = 0

premiumStartDayCombo.selectedIndex = 0

premiumStartMonthCombo.selectedIndex = 0
```

```
    premiumStartYearCombo.selectedIndex = 0  
  
END  
  
METHOD SHOW_SCROLLABLE_DIALOG(title, content)  
  
DO  
  
    // Implementation to show content in scrollable dialog  
  
    // This would create a JTextArea inside JScrollPane inside JOptionPane  
  
END  
  
END CLASS
```

5. Method Description

The method description for all the classes is given below:

5.1 Gym Member

The method description for gym member is as follows:

- The constructor, `public GymMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate)`, initializes a new gym member with default values. Attendance is set to 0, loyalty points to 0, and active status to false. The parameters include the member's ID, name, location, phone number, email, gender, date of birth, and membership start date.
- The `getId()` method retrieves the member's unique ID.
- The `getName()` method provides the member's name.
- The `getLocation()` method retrieves the location of the member.
- The `getPhone()` method returns the member's phone number.
- The `getEmail()` method gives the email address of the member.
- The `getGender()` method retrieves the member's gender.
- The `getDOB()` method provides the member's date of birth.
- The `getMembershipStartDate()` method retrieves the date the membership started.
- The `getAttendance()` method gives the total attendance count.
- The `getLoyaltyPoints()` method retrieves the loyalty points earned by the member.
- The `getActiveStatus()` method indicates whether the membership is currently active.
- The `markAttendance()` method is an abstract method that subclasses must implement to increment attendance and loyalty points.
- The `activateMembership()` method activates the membership by setting the active status to true.
- The `deactivateMembership()` method deactivates the membership by setting the active status to false.

- The `resetMembership()` method resets the member's data by deactivating the membership and setting loyalty points to 0.
- The `display()` method prints all the member's details, including ID, name, location, phone, email, gender, date of birth, membership start date, attendance, loyalty points, and active status.

5.2 Regular Member

The method description for regular member is as follows:

- The constructor, `public RegularMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate, String referralSource)`, initializes a new `RegularMember` object. It takes parameters including the member's ID, name, location, phone number, email, gender, date of birth, membership start date, and referral source.
- The `getIsEligibleForUpgrade()` method returns a boolean value, indicating whether the member is eligible for a plan upgrade.
- The `getRemovalReason()` method returns a string providing the reason for the member's removal.
- The `getReferralSource()` method retrieves the referral source of the member.
- The `getPlan()` method returns the member's current plan as a string.
- The `getPrice()` method retrieves the price of the member's current plan as a double value.
- The `markAttendance()` method increments the member's attendance by 1 and adds 5 loyalty points. This method overrides the parent class's `markAttendance()` method.
- The `getPlanPrice(String plan)` method takes the plan type as a parameter and returns the corresponding price of the plan.
- The `upgradePlan(String plan)` method allows upgrading the member's plan if the attendance is greater than or equal to the attendance limit. It takes the plan name as a parameter and returns a string indicating the upgrade status.

- The `revertRegularMember(String removalReason)` method resets the member's data by calling the `resetMember()` method from the superclass and sets the removal reason.
- The `display()` method prints the details of the `RegularMember`, including the current plan, price, and any additional information specific to regular members.

5.3 Premium Member

- The constructor `PremiumMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate, String personalTrainer)` initializes a new `PremiumMember` object with details such as ID, name, location, phone, email, gender, date of birth, membership start date, and the assigned personal trainer.
- The `getPersonalTrainer()` method retrieves the name of the personal trainer assigned to the member.
- The `getIsFullyPayment()` method returns a boolean value indicating whether the member has completed full payment.
- The `getPaidAmount()` method retrieves the total amount the member has paid so far.
- The `getDiscountAmount()` method calculates and returns the discount amount (10% of the premium charge).
- The `markAttendance()` method, which overrides the parent class method, increments the member's attendance and loyalty points if the member's status is active.
- The `payDueAmount()` method processes the member's payment, updates the payment status, and returns a message indicating the payment result.
- The `revertPremiumMember()` method resets premium-specific fields, such as the assigned trainer and discount details, by reverting the member's state.
- The `display()` method, which overrides the parent class method, displays all premium member details, including the trainer's name, the total paid amount, and the discount received.

5.4 Gym GUI

The method description for Gym GUI is as follows:

- `main(String[] args)` - The entry point that initializes the GUI window and sets up all components
- regular member panel setup: configures input fields, dropdowns, and the “Add Regular Member” button.
- Premium Member Panel Setup: Sets up fields for premium members, including personal trainer selection.
- Action Panel Setup: Creates panels for membership management functions like activation and payments.
- `addRegularButton ActionListener`: Validates input, creates a regular member, and adds it to the system.
- `addPremiumButton ActionListener`: Validates input, creates a premium member, and adds it to the system.
- `activateButton ActionListener`: Enables a membership by ID if found.
- `DeactiveButton ActionListener`: Disables a membership by ID if active.
- `markAttendanceButton ActionListener`: Increases attendance count for a member by ID.
- `revertRegularButton ActionListener`: Removes a regular member after recording a reason.
- `revertPremiumButton ActionListener`: Removes a premium member and clears their data.
- `upgradeButton ActionListener`: Upgrades a regular member's plan (basic → standard → deluxe).
- `paymentButton ActionListener`: Processes a premium member's payment and updates dues.
- `discountButton ActionListener`: Calculates a discount for a premium member.
- `displayButton ActionListener`: Shows all members in a scrollable table format.
- `clearButton ActionListener`: Resets all form fields to default values.

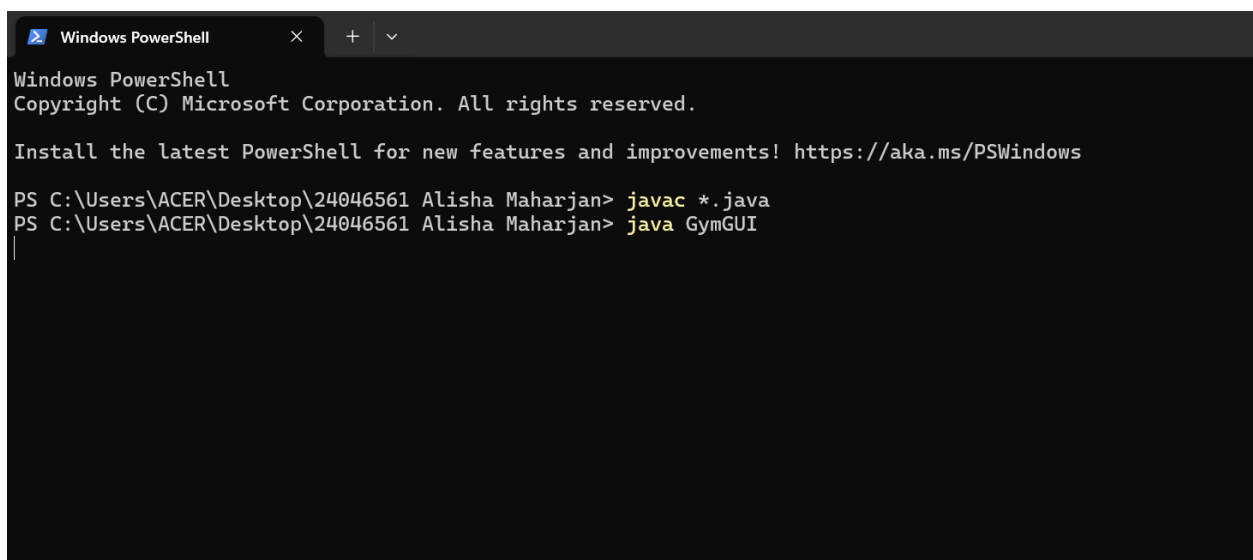
- `saveToFileButton` ActionListener: Exports member data to `MemberDetails.txt`.
- `readFromFileButton` ActionListener: Imports and displays member data from `MemberDetails.txt`.
- `isUnique(int id)`: Checks if a member ID is already in use.

6. Testing

6.1 Test 1: Compile and run using terminal

Table 1: Test 1: Compiling and running using terminal

Objective	To compile and run the program in the terminal.
Action	The program was opened in the terminal and necessary command was provided.
Expected Output	The GymGUI frame should be opened.
Actual Output	The GymGUI frame was opened.
Result	The test was successful.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ACER\Desktop\24046561 Alisha Maharjan> javac *.java
PS C:\Users\ACER\Desktop\24046561 Alisha Maharjan> java GymGUI
```

Figure 10: Command to open the Gym GUI

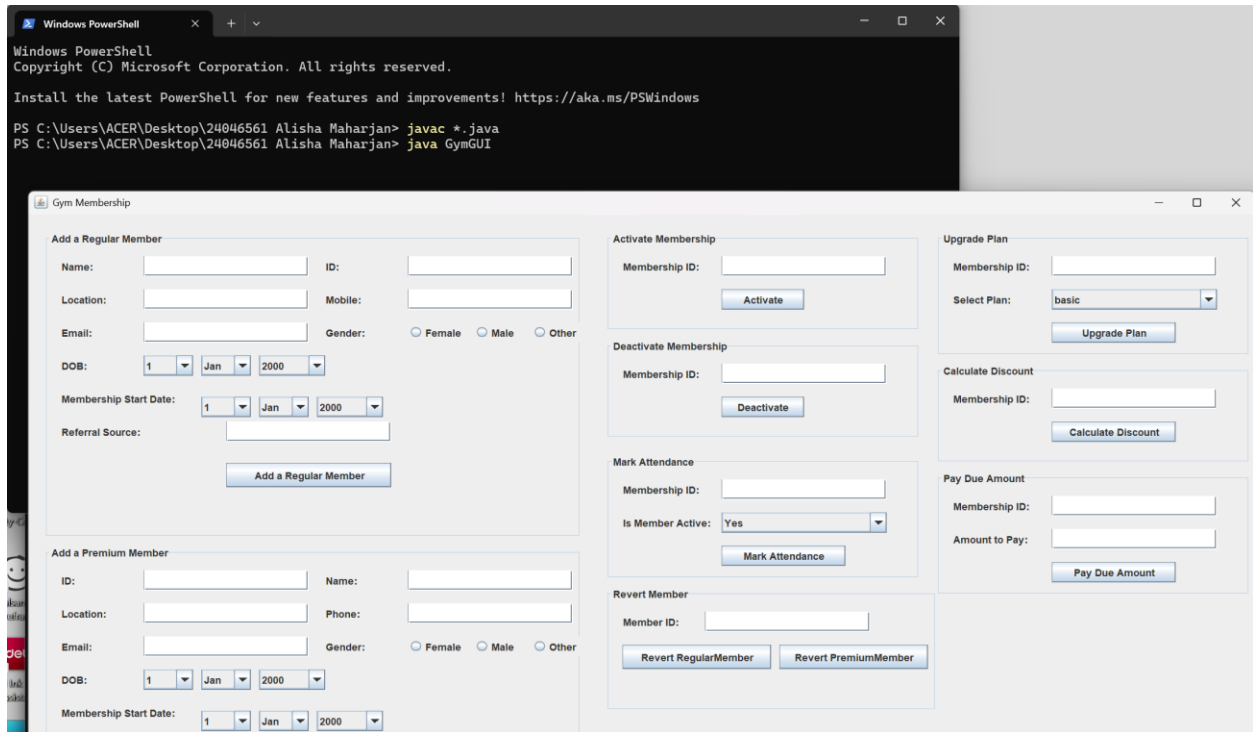


Figure 11: Opening from terminal

6.2 Test 2: Add Regular Member and Premium Member

6.2.1 Test 2.1: Adding Regular Member

Table 2: Test 2.1: Adding Regular Member

Objective	To add a regular member.
Action	All the required fields were filled and "Add Regular Member" Button was clicked.

Expected Output	The regular members details should be added successfully with an appropriate message dialog box.
Actual Output	The regular member details were added successfully with an appropriate message dialog box.
Result	The test was successful.

The screenshot displays the 'Gym Membership' application interface. The main form is titled 'Add a Regular Member' and contains the following fields:

- Name:** Ayira
- Location:** Patan
- Email:** ayira@gmail.com
- ID:** 1
- Mobile:** 9841225532
- Gender:** ☒ Female ☐ Male ☐ Other
- DOB:** 1 Jan 2004
- Membership Start Date:** 1 Jan 2025
- Referral Source:** Arman

Below the form is a button labeled 'Add a Regular Member'. To the right of the main form are three sections:

- Activate Membership:** Includes a 'Membership ID' field and an 'Activ' button.
- Deactivate Membership:** Includes a 'Membership ID' field and a 'Deacti' button.
- Mark Attendance:** Includes a 'Membership ID' field, an 'Is Member Active' dropdown set to 'Yes', and a 'Mark' button.

At the bottom of the screen, there is a 'Message' dialog box with the text: 'Regular member is added succesfully.' and an 'OK' button.

Figure 12: Adding Regular Member

6.2.2 Test 2.2: Adding Premium Member

Table 3: Test 2.2: Adding Premium Member

Objective	To add premium member.
Action	All the required fields were filled and “Add Premium Member” Button was clicked.
Expected Output	The premium member details should be added successfully with and appropriate message dialog box.
Actual Output	The premium member details were added successfully with an appropriate message dialog box.
Result	The test was successful.

The screenshot shows a web application interface for adding a premium member. The form is titled "Add a Premium Member" and contains the following fields:

- ID:** 2
- Name:** Ayira
- Location:** Patan
- Phone:** 9841225532
- Email:** ayira@gmail.com
- Gender:** ☒ Female ☐ Male
- DOB:** 1 Jan 2000
- Membership Start Date:** 1 Jan 2000
- Personal Trainer:** Arman

At the bottom of the form is a button labeled "Add a Premium Member". Overlaid on the right side of the form is a "Message" dialog box with the text "Premium member is added succesfully." and an "OK" button.

Figure 13: Adding Premium Member

6.3 Test 3: Mark Attendance and Upgrade Plan

6.3.1 Test 3.1: Mark Attendance

Table 4: Test 3.1: Mark Attendance

Objective	To mark the attendance.
Action	The required fields were filled, and mark attendance button was clicked.
Expected Output	The attendance should be marked successfully with an appropriate message dialog box.

Actual Output	The attendance was marked successfully with an appropriate dialog box.
Result	The test was successful.

The screenshot displays the 'Gym Membership' application interface. It features several sections for managing members and their attendance:

- Add a Regular Member:** Includes fields for Name (Aylira), ID (1), Location (patan), Mobile (9851223346), Email (ayira@gmail.com), Gender (Female selected), DOB (1 Jan 2000), Membership Start Date (1 Jan 2007), and Referral Source (Arman). A button 'Add a Regular Member' is present.
- Add a Premium Member:** Includes fields for ID (2), Name (Aylira), Location (lalitpur), Phone (9841223356), Email (ayira@gmail.com), Gender (Female selected), and DOB (1 Jan 2000).
- Activate Membership:** Includes a field for Membership ID and an 'Activate' button.
- Deactivate Membership:** Includes a field for Membership ID and a 'Deactivate' button.
- Mark Attendance:** Includes a field for Membership ID (1), a dropdown for 'Is Member Active' (Yes), and a 'Mark Attendance' button.

A success dialog box is overlaid on the 'Mark Attendance' section, displaying the message: 'Success Attendance marked successfully!' with an 'OK' button.

Figure 14: Marking attendance for regular member

The screenshot displays a web application interface. On the left, there is a form titled 'Add a Premium Member'. It contains the following fields: ID (2), Name (Ayira), Location (lalitpur), Phone (9841223356), Email (ayira@gmail.com), Gender (Female selected), DOB (1 Jan 2000), Membership Start Date (1 Jan 2000), and Personal Trainer (Arman). Below the form is a button labeled 'Add a Premium Member'. On the right, there is a 'Mark Attendance' dialog box. It contains 'Membership ID: 2' and 'Is Member Active: Yes'. Below these fields is a button labeled 'Mark Attendance'. A 'Success' message box is overlaid on the dialog, stating 'Attendance marked successfully!' with an 'OK' button. In the background, there are buttons for 'Revert Regular Member' and 'Revert Premium Member'.

Figure 15: Marking attendance for premium member

6.3.2 Test 3.2: Upgrade Plan

Table 5: Test 3.2 Upgrade Plan

Objective	To upgrade the plan from basic to standard or deluxe of a regular member.
Action	When the attendance reached to 30 days and the required field is filled then regular member was upgraded to standard from basic.
Expected Output	The regular member should be upgraded from basic to standard.

Actual Output	The regular member was upgraded from basic to standard.
Result	The test was successful.

The screenshot displays the 'Gym Membership' application window. On the left, there are two forms: 'Add a Regular Member' and 'Add a Premium Member'. The 'Add a Regular Member' form is filled with data for member 'Ayira' (ID: 1, Location: patan, Mobile: 9851223346, Email: ayira@gmail.com, Gender: Female, DOB: 1 Jan 2000, Membership Start Date: 1 Jan 2007, Referral Source: Arman). The 'Add a Premium Member' form is also filled with data for member 'Ayira' (ID: 2, Location: lallipur, Phone: 9841223356, Email: ayira@gmail.com, Gender: Female, DOB: 1 Jan 2000, Membership Start Date: 1 Jan 2000, Personal Trainer: Arman). On the right, there are three panels: 'Activate Membership' (Membership ID: 1, Activate button), 'Deactivate Membership' (Membership ID: , Deactivate button), and 'Upgrade Plan' (Membership ID: 1, Select Plan: standard, Upgrade Plan button). Below these panels are 'Calculate Discount' and 'Pay Due Amount' sections. A modal dialog titled 'Upgrade Status' is open in the center, displaying the following information: 'Current Plan: basic', 'Attendance: 1/30', 'Status: Active', and 'Need 29 more attendances to upgrade'. The dialog has an 'OK' button.

Figure 16: Upgrading member in progress

The screenshot displays the 'Gym Membership' application interface. On the left, there are two forms: 'Add a Regular Member' and 'Add a Premium Member'. The 'Add a Regular Member' form is filled with details for a member named Ayira, ID 1, located at patan, with a mobile number 9851223346, email ayira@gmail.com, gender Female, DOB 1 Jan 2000, and referral source Arman. The 'Add a Premium Member' form is also filled with details for a member named Ayira, ID 2, located at lallipur, with a phone number 9841223356, email ayira@gmail.com, gender Female, DOB 1 Jan 2000, and personal trainer Arman. On the right, there are several functional panels: 'Activate Membership' (Membership ID: 1, Activate button), 'Deactivate Membership' (Membership ID: , Deactivate button), 'Mark Attendance' (Membership ID: 1, Is Member Active: Yes), 'Upgrade Plan' (Membership ID: 1, Select Plan: standard, Upgrade Plan button), 'Calculate Discount' (Membership ID: , Calculate Discount button), and 'Pay Due Amount' (Membership ID: , Amount to Pay: , Pay Due Amount button). A central 'Upgrade Status' dialog box is open, displaying the following information: Current Plan: basic, Attendance: 30/30, Status: Active, and Upgraded to Standard plan (Rs. 12,500/month). The dialog box has an 'OK' button.

Figure 17: Successfully upgraded member from basic to standard

The screenshot displays the 'Gym Membership' application interface, similar to Figure 17. The 'Add a Regular Member' and 'Add a Premium Member' forms are the same. The 'Upgrade Plan' panel now shows 'Select Plan: deluxe' instead of 'standard'. The 'Upgrade Status' dialog box is open, displaying the following information: Current Plan: Standard, Attendance: 30/30, Status: Active, and Upgraded to Deluxe plan (Rs. 18,500/month). The dialog box has an 'OK' button.

Figure 18: Successfully upgraded member from basic to deluxe

6.4 Test 4: Calculate Discount, pay due amount, revert members

6.4.1 Test 4.1: Calculate Discount

Table 6: Test 4.1: Calculate Discount

Objective	To calculate the discount for premium members if full payment is done.
Action	Firstly, it was checked if the member has paid full amount or not and then the required fields were filled.
Expected Output	If the amount is fully paid, then discount of 10% is applied and if the amount is not fully paid then no discount is provided.
Actual Output	When the member had paid full amount then discount of 10% was given and when the amount was not paid then no discount was provided.
Result	The test was successful.

The screenshot displays the 'Gym Membership' application interface. On the left, there are two forms: 'Add a Regular Member' and 'Add a Premium Member'. The 'Add a Regular Member' form is filled with the following data: Name: Ayira, ID: 1, Location: patan, Mobile: 9851223346, Email: ayira@gmail.com, Gender: Female, DOB: 1 Jan 2000, Membership Start Date: 1 Jan 2007, and Referral Source: Aman. The 'Add a Premium Member' form is also filled: ID: 2, Name: Ayira, Location: lalitpur, Phone: 9841223356, Email: ayira@gmail.com, Gender: Female, DOB: 1 Jan 2000, Membership Start Date: 1 Jan 2000, and Personal Trainer: Aman. On the right, there are three panels: 'Activate Membership' (Membership ID: 2), 'Deactivate Membership' (Membership ID:), and 'Mark Attendance' (Membership ID: 2, Is Member Active: Yes). Below these is the 'Upgrade Plan' panel (Membership ID: , Select Plan: deluxe) and the 'Calculate Discount' panel (Membership ID: 2). A 'Pay Due Amount' panel shows Membership ID: 2 and Amount to Pay: . A 'Success' dialog box is open in the center, displaying the message 'Discount calculated successfully!' with an 'OK' button.

Figure 19: Calculating discount

The screenshot displays the 'Gym Membership' application interface, similar to Figure 19. The forms and panels are the same, but the 'Calculate Discount' panel now shows 'Amount to Pay: 45000'. The 'Pay Due Amount' panel also shows 'Amount to Pay: 45000'. A 'Payment Result' dialog box is open in the center, displaying the message 'Payment successful. Remaining Amount: 5000.0' with an 'OK' button.

Figure 20: Full amount was not paid by member

The screenshot displays the 'Gym Membership' application window. It contains several functional panels:

- Add a Regular Member:** Fields for Name (Ayira), ID (1), Location (patan), Mobile (9851223346), Email (ayira@gmail.com), Gender (Female selected), DOB (1 Jan 2000), Membership Start Date (1 Jan 2007), and Referral Source (Aman). A button 'Add a Regular Member' is at the bottom.
- Add a Premium Member:** Fields for ID (2), Name (Ayira), Location (lailpur), Phone (9841223356), Email (ayira@gmail.com), Gender (Female selected), DOB (1 Jan 2000), Membership Start Date (1 Jan 2000), and Personal Trainer (Aman). A button 'Add a Premium Member' is at the bottom.
- Activate Membership:** Field for Membership ID (2) and an 'Activate' button.
- Deactivate Membership:** Field for Membership ID and a 'Deactivate' button.
- Mark Attendance:** Fields for Membership ID (2) and 'Is Member Active' (Yes selected), with a 'Mark Attendance' button.
- Upgrade Plan:** Fields for Membership ID, a 'Select Plan' dropdown (deluxe), and an 'Upgrade Plan' button.
- Calculate Discount:** Field for Membership ID (2) and a 'Calculate Discount' button.
- Pay Due Amount:** Fields for Membership ID (2) and 'Amount to Pay' (5000), with a 'Pay Due Amount' button.

 A 'Payment Result' dialog box is overlaid in the center, displaying an information icon, the text 'Full payment completed. Payment successful. Discount applied: 5000.0', and an 'OK' button.

Figure 21: Discount applied after full payment

6.4.2 Test 4.2: Pay Due Amount

Table 7: Test 4.2: Pay Due Amount

Objective	To pay the remaining amount of premium member.
Action	All the necessary fields were filled and then amount to pay was written.
Expected Output	If the payment is paid fully then payment successful and discount amount message should be generated in dialog box

	otherwise remaining amount should be generated.
Actual Output	When the full payment was done then “Payment Successful. Discount applied” message was generated and when payment was left “Payment Successful. Remaining amount” was generated.
Result	The test was successful.

The screenshot displays the 'Gym Membership' application window. It contains several functional areas:

- Add a Regular Member:** Fields for Name (Ayira), ID (1), Location (patan), Mobile (9851223346), Email (ayira@gmail.com), Gender (Female selected), DOB (1 Jan 2000), Membership Start Date (1 Jan 2007), and Referral Source (Arman). A button 'Add a Regular Member' is present.
- Add a Premium Member:** Fields for ID (2), Name (Ayira), Location (lailpur), Phone (9841223356), Email (ayira@gmail.com), Gender (Female selected), DOB (1 Jan 2000), Membership Start Date (1 Jan 2000), and Personal Trainer (Arman). A button 'Add a Premium Member' is present.
- Activate Membership:** Field for Membership ID (2) and an 'Activate' button.
- Deactivate Membership:** Field for Membership ID and a 'Deactivate' button.
- Mark Attendance:** Fields for Membership ID (2) and Is Member Active (Yes selected), with a 'Mark Attendance' button.
- Upgrade Plan:** Fields for Membership ID and a dropdown for Select Plan (deluxe selected), with an 'Upgrade Plan' button.
- Calculate Discount:** Field for Membership ID and a 'Calculate Discount' button.
- Pay Due Amount:** Fields for Membership ID (2) and Amount to Pay (5000), with a 'Pay Due Amount' button.

A 'Payment Result' dialog box is overlaid in the center, displaying an information icon, the text 'Payment successfull. Remaining Amount: 45000.0', and an 'OK' button.

Figure 22: Payment not fully paid

The screenshot displays the 'Gym Membership' application window. It contains several forms for managing members: 'Add a Regular Member', 'Add a Premium Member', 'Activate Membership', 'Deactivate Membership', 'Mark Attendance', 'Upgrade Plan', 'Calculate Discount', and 'Pay Due Amount'. A 'Payment Result' dialog box is overlaid in the center, displaying the message: 'Full payment completed. Payment successful. Discount applied: 5000.0'. The dialog has an 'OK' button. The background forms show data for a member named 'Ayira' with ID '1' or '2', location 'patan' or 'lalitpur', and a membership start date of '1 Jan 2007'.

Figure 23: Full payment done

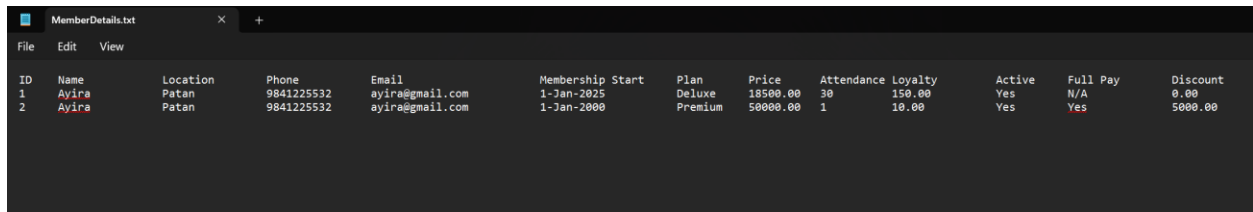
6.4.3 Test 4.3: Revert Members

6.4.3 Test 4.3.1: Reverting Regular Member

Table 8: Test 4.3.1: Revert Members

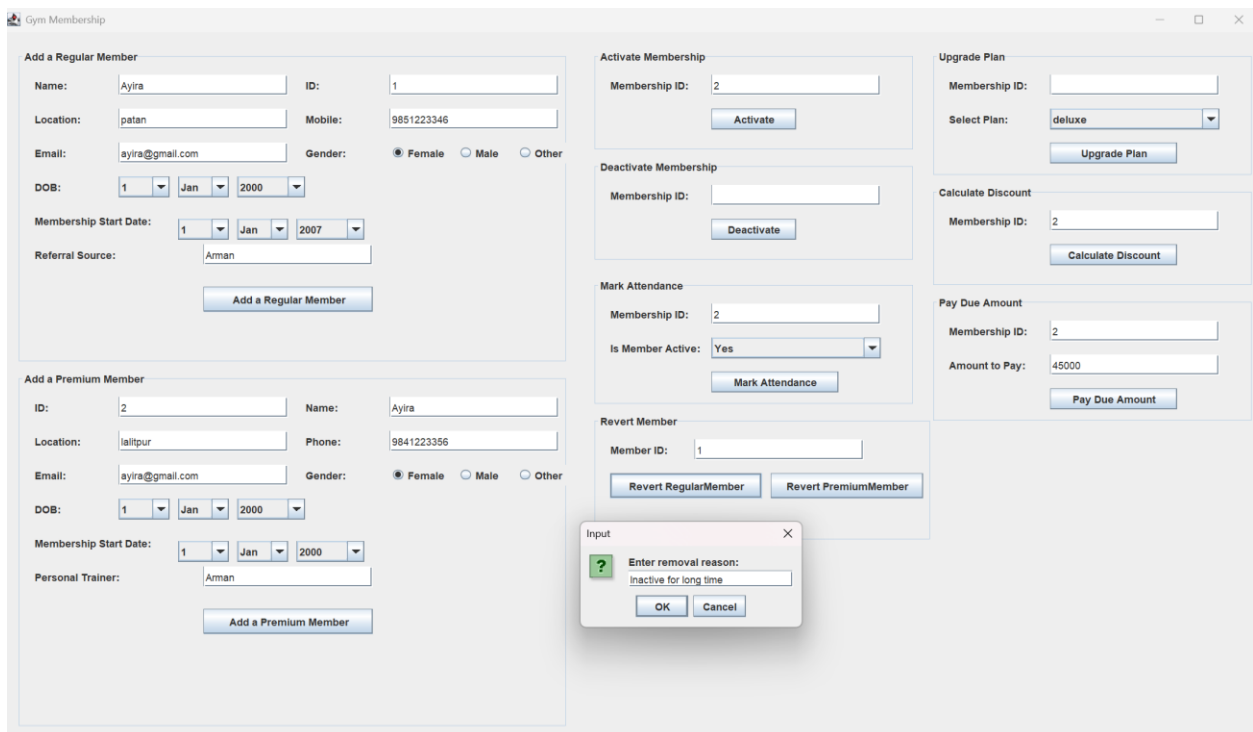
Objective	To revert the regular member.
Action	All the necessary fields were filled and “Revert Regular Member” button was clicked.
Expected Output	All the saved data should be deleted/reverted with removal reason dialog box.

Actual Output	All the saved data was deleted/reverted with the question of removal reason.
Result	The test was successful.



ID	Name	Location	Phone	Email	Membership Start	Plan	Price	Attendance	Loyalty	Active	Full Pay	Discount
1	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2025	Deluxe	18500.00	30	150.00	Yes	N/A	0.00
2	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2000	Premium	50000.00	1	10.00	Yes	Yes	5000.00

Figure 24: Before reverting regular member



The screenshot shows the 'Gym Membership' application interface. On the left, there are two forms: 'Add a Regular Member' and 'Add a Premium Member'. The 'Add a Regular Member' form has fields for Name, ID, Location, Mobile, Email, Gender, DOB, Membership Start Date, and Referral Source. The 'Add a Premium Member' form has fields for ID, Name, Location, Phone, Email, Gender, DOB, Membership Start Date, and Personal Trainer. On the right, there are several buttons and sections: 'Activate Membership', 'Deactivate Membership', 'Mark Attendance', 'Revert Member', 'Upgrade Plan', 'Calculate Discount', and 'Pay Due Amount'. The 'Revert Member' section has a 'Member ID' field and two buttons: 'Revert RegularMember' and 'Revert PremiumMember'. A small 'Input' dialog box is open in the foreground, asking for a removal reason. The dialog box has a green question mark icon and the text 'Enter removal reason:'. Below the text is a text input field containing 'Inactive for long time'. There are 'OK' and 'Cancel' buttons at the bottom of the dialog box.

Figure 25: Reverting regular member

The screenshot shows a web application for gym membership management. A modal message box is centered on the screen, displaying the text "Regular membership reverted successfully!" with an "OK" button. In the background, there are several forms: "Add a Regular Member" with fields for Name, ID, Location, Mobile, Email, Gender, DOB, Membership Start Date, and Referral Source; "Add a Premium Member" with fields for ID, Name, Location, Phone, Email, Gender, DOB, Membership Start Date, and Personal Trainer; "Activate Membership" with a Membership ID field and an "Activate" button; "Deactivate Membership" with a Membership ID field and a "Deactivate" button; "Mark Attendance" with a Membership ID field, an "Is Member Active" dropdown, and a "Mark Attendance" button; "Upgrade Plan" with a Membership ID field, a "Select Plan" dropdown, and an "Upgrade Plan" button; "Calculate Discount" with a Membership ID field and a "Calculate Discount" button; and "Pay Due Amount" with a Membership ID field, an "Amount to Pay" field, and a "Pay Due Amount" button.

Figure 26: Reverting regular member

ID	Name	Location	Phone	Email	Membership Start	Plan	Price	Attendance	Loyalty	Active	Full Pay	Discount
1	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2025	basic	6500.00	0	0.00	No	N/A	0.00
2	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2000	Premium	50000.00	1	10.00	Yes	Yes	5000.00

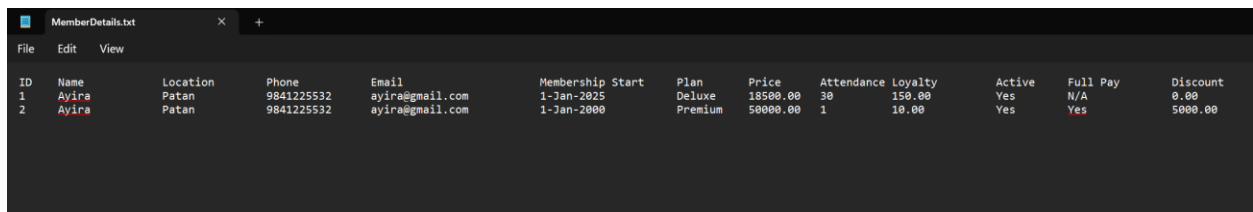
Figure 27: After reverting regular member

6.4.4 Test 4.3.2: Reverting Premium Member

Table 9: Test 4.3.2: Reverting Premium Member

Objective	To revert the premium member.
Action	All the necessary fields were filled and "Revert Premium Member" button was clicked.

Expected Output	All the saved data should be deleted/reverted with removal reason dialog box.
Actual Output	All the saved data was deleted/reverted with the question of removal reason.
Result	The test was successful.



ID	Name	Location	Phone	Email	Membership Start	Plan	Price	Attendance	Loyalty	Active	Full Pay	Discount
1	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2025	Deluxe	18500.00	30	150.00	Yes	N/A	0.00
2	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2000	Premium	50000.00	1	10.00	Yes	Yes	5000.00

Figure 28: Before reverting premium member

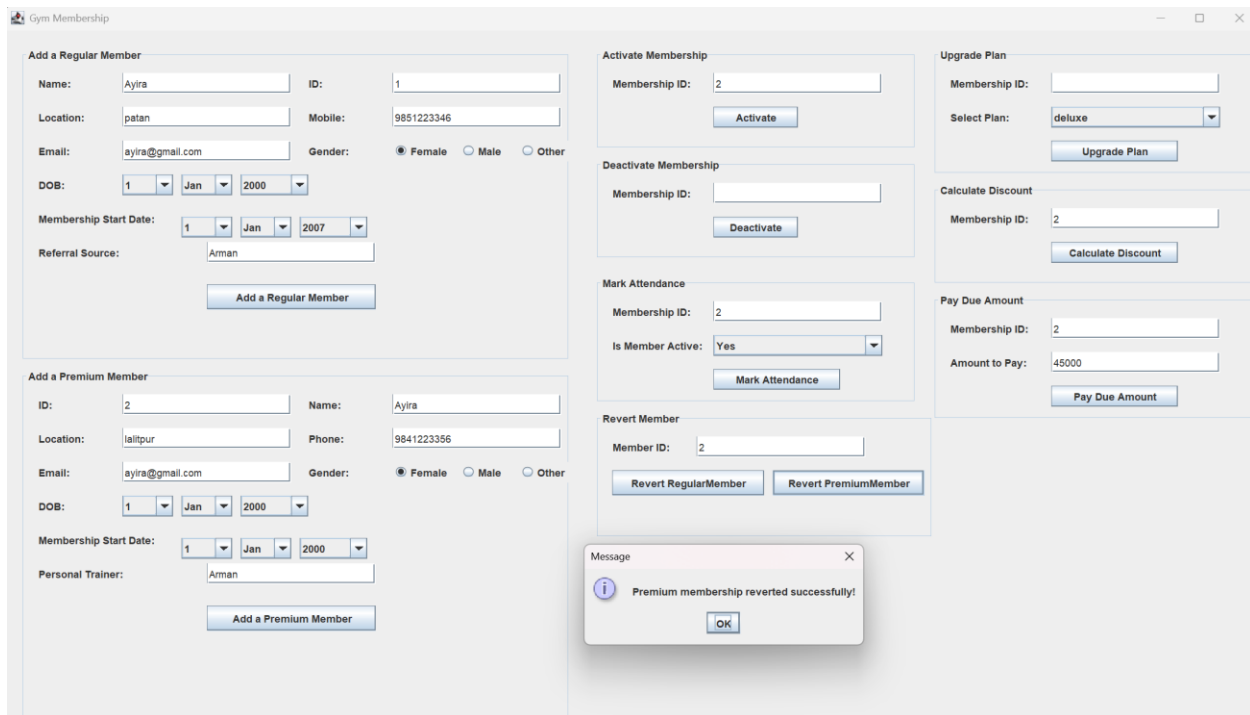


Figure 29: Reverting premium member

MemberDetails.txt												
ID	Name	Location	Phone	Email	Membership Start	Plan	Price	Attendance	Loyalty	Active	Full Pay	Discount
2	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2000	Premium	50000.00	0	0.00	No	No	0.00

Figure 30: After reverting premium member

6.5 Test 5: Save to and read from file

6.5.1 Test 5.1: Saving to file

Table 10: Test 5.1: Saving to file

Objective	To save the entered data of regular and premium member.
-----------	---

Action	All the required fields were filled of both regular and premium members and save to file button was clicked.
Expected Output	All the information of regular and premium members should be saved in text file named "MemberDetails.txt".
Actual Output	All the information of regular and premium members was saved in text file named "MemberDetails.txt".
Result	The test was successful.

Gym Membership

Add a Regular Member

Name: ID:

Location: Mobile:

Email: Gender: ☒ Female ☐ Male ☐ Other

DOB:

Membership Start Date:

Referral Source:

Add a Premium Member

ID: Name:

Location: Phone:

Email: Gender: ☐ Female ☒ Male ☐ Other

DOB:

Membership Start Date:

Personal Trainer:

Activate Membership

Membership ID:

Deactivate Membership

Membership ID:

Mark Attendance

Membership ID:

Is Member Active:

Revert Member

Member ID:

Upgrade Plan

Membership ID:

Select Plan:

Calculate Discount

Membership ID:

Pay Due Amount

Membership ID:

Amount to Pay:

Figure 31: Adding all necessary information

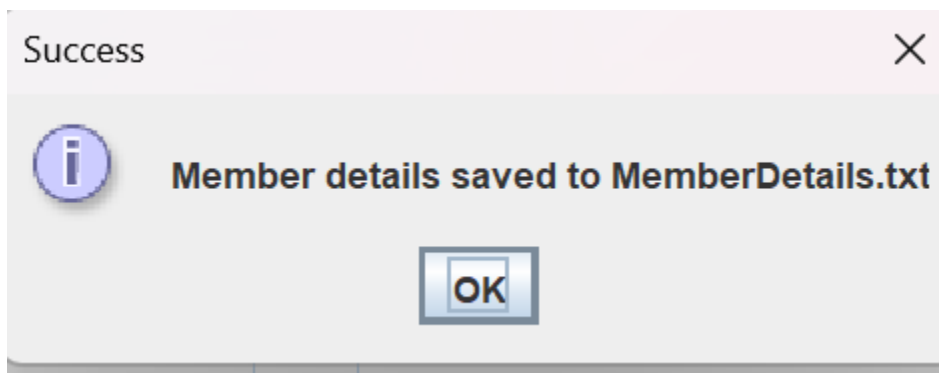
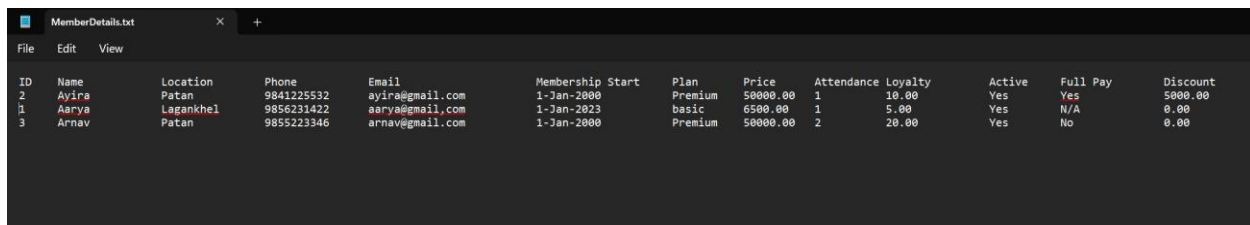


Figure 32: Member details saved



ID	Name	Location	Phone	Email	Membership Start	Plan	Price	Attendance	Loyalty	Active	Full Pay	Discount
2	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2000	Premium	50000.00	1	10.00	Yes	Yes	5000.00
1	Aarya	Lagankhel	9856231422	aarya@gmail.com	1-Jan-2023	basic	6500.00	1	5.00	Yes	N/A	0.00
3	Arnav	Patan	9855223346	arnav@gmail.com	1-Jan-2000	Premium	50000.00	2	20.00	Yes	No	0.00

Figure 33: Save to text file

6.5.2 Reading from file

Table 11: Test 5.2: Reading from file

Objective	To read the data from the file.
Action	After saving all the data in a text file then read from file button was clicked.
Expected Output	The system should display all member information from "MemberDetails.txt" in a readable format within a scrollable dialog box.
Actual Output	The system displayed all member information from "MemberDetails.txt" in a readable format within a scrollable dialog box.
Result	The test was successful.

Gym Membership

Add a Regular Member

Name: ID:

Location: Mobile:

Email: Gender: ☐ Female ☐ Male ☐ Other

DOB: 1 Jan 2000

Membership Start Date: 1 Jan 2000

Referral Source:

Add a Regular Member

Activate Membership

Membership ID:

Activate

Deactivate Membership

Membership ID:

Deactivate

Upgrade Plan

Membership ID:

Select Plan: basic

Upgrade Plan

Calculate Discount

Membership ID:

Calculate Discount

Add a Premium Member

ID:

Location:

Email:

DOB: 1 Jan 2000

Membership Start Date: 1 Jan

Personal Trainer:

Add a Premium Member

Member Details from File

ID	Name	Location	Phone	Email	Membership Start	Plan	Price	Attendance	Loyalty	Active
2	Ayira	Patan	9841225532	ayira@gmail.com	1-Jan-2000	Premium	50000.00	1	10.00	Yes
1	Aarya	Lagankhel	9856231422	aarya@gmail.com	1-Jan-2023	basic	6500.00	1	5.00	Yes
3	Amav	Patan	9855223346	amav@gmail.com	1-Jan-2000	Premium	50000.00	2	20.00	Yes

OK

Pay Due Amount

Membership ID: 3

Amount to Pay: 45000

Pay Due Amount

Display Members **Clear Fields**

Save to File **Read from File**

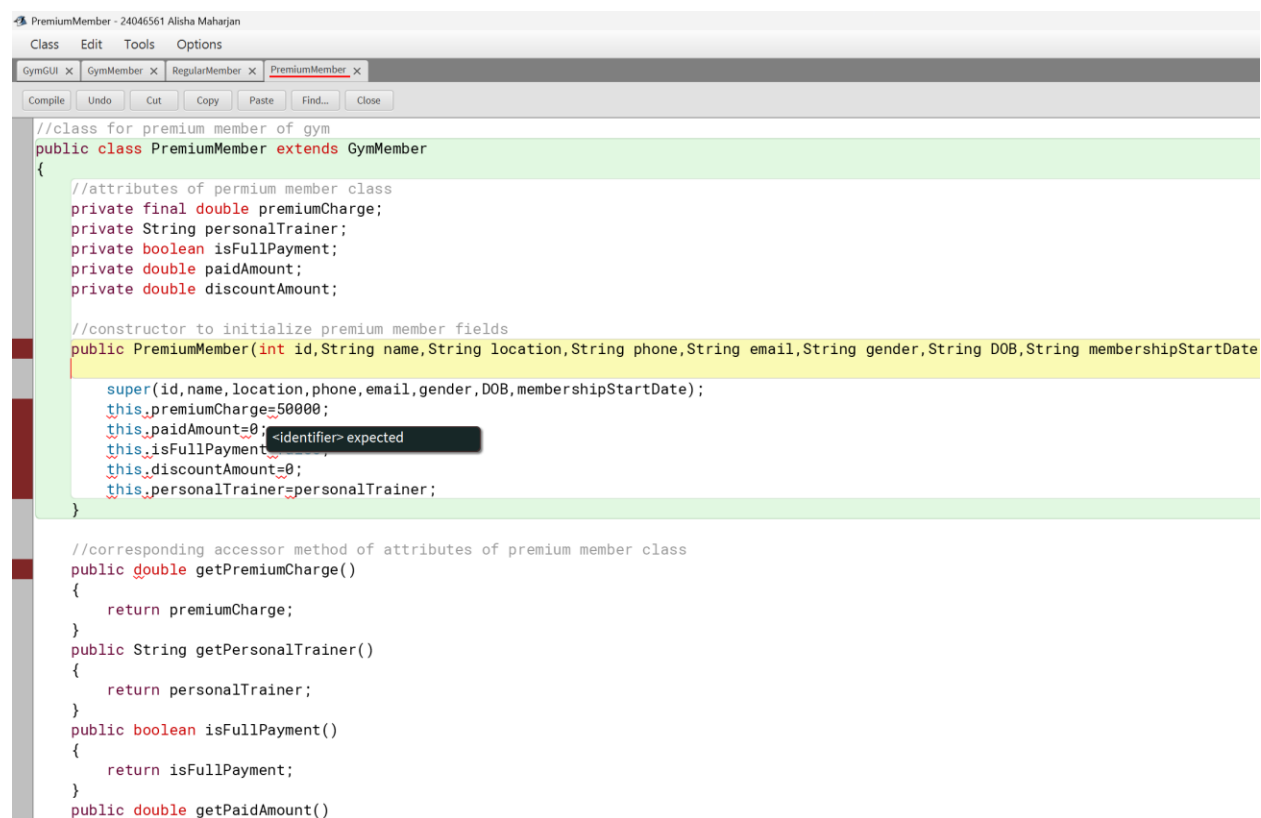
Figure 34: Read from file

7. Error Detection and Correction

During developing this program many errors occurred during the execution. Few errors that occurred during this development process are listed below:

7.1 Syntax Error

The very common error that can occur during developing application through Java. Here, in this project of mine I introduced a syntax error by removing a curly bracket in my code which is fundamental part of syntax in Java.



```
PremiumMember - 24046561 Alisha Maharjan
Class Edit Tools Options
GymGUI x GymMember x RegularMember x PremiumMember x
Compile Undo Cut Copy Paste Find... Close

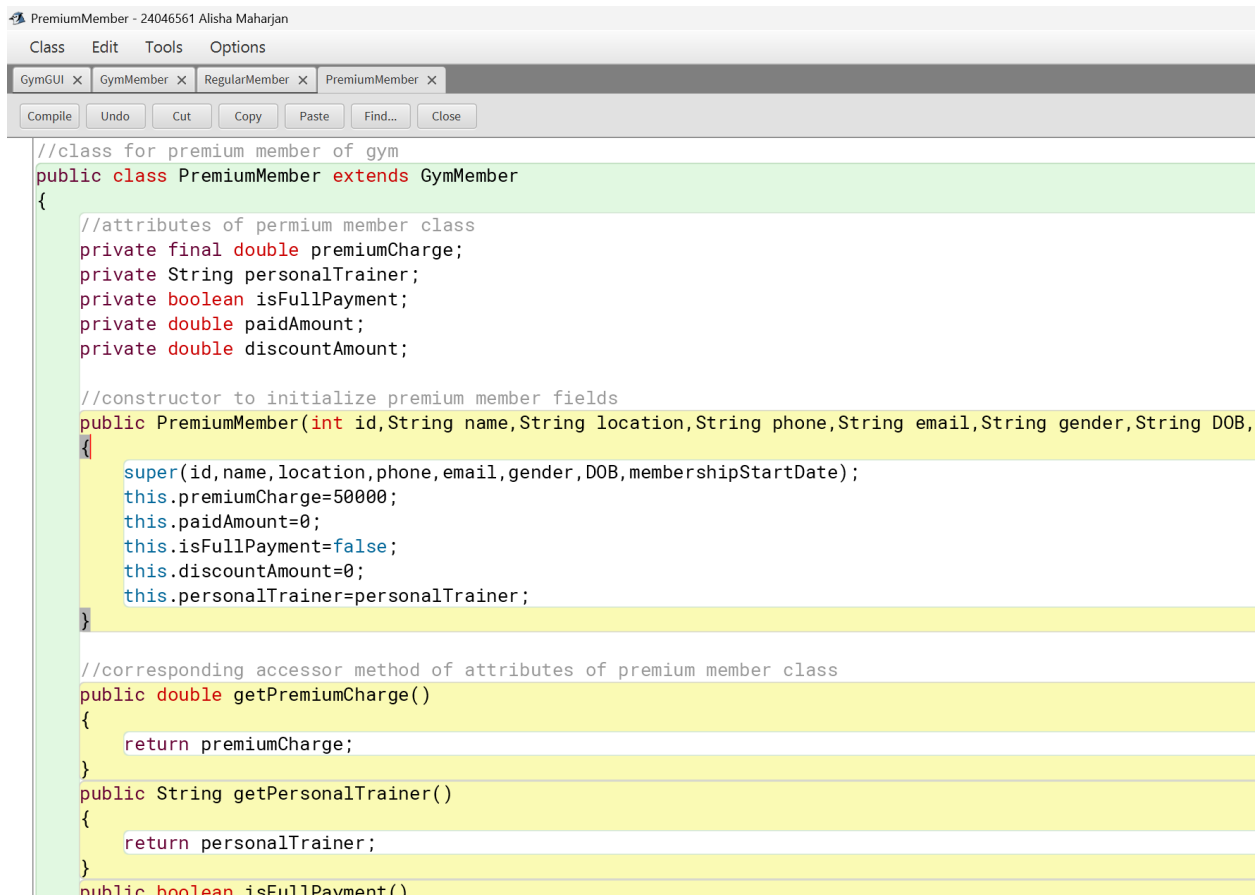
//class for premium member of gym
public class PremiumMember extends GymMember
{
    //attributes of permium member class
    private final double premiumCharge;
    private String personalTrainer;
    private boolean isFullPayment;
    private double paidAmount;
    private double discountAmount;

    //constructor to initialize premium member fields
    public PremiumMember(int id,String name,String location,String phone,String email,String gender,String DOB,String membershipStartDate)
    {
        super(id,name,location,phone,email,gender,DOB,membershipStartDate);
        this.premiumCharge=50000;
        this.paidAmount=0;
        this.isFullPayment;
        this.discountAmount=0;
        this.personalTrainer=personalTrainer;
    }

    //corresponding accessor method of attributes of premium member class
    public double getPremiumCharge()
    {
        return premiumCharge;
    }
    public String getPersonalTrainer()
    {
        return personalTrainer;
    }
    public boolean isFullPayment()
    {
        return isFullPayment;
    }
    public double getPaidAmount()
}
```

Figure 35: Syntax error

For solving this error, I wrote the syntax properly that is by adding the curly brackets. And as soon as I introduced the curly brackets again in this code the error was solved and the program got compiled.

The image shows a screenshot of an IDE window titled "PremiumMember - 24046561 Alisha Maharjan". The window has a menu bar with "Class", "Edit", "Tools", and "Options". Below the menu bar is a tab bar with four tabs: "GymGUI x", "GymMember x", "RegularMember x", and "PremiumMember x". The "PremiumMember x" tab is selected. Below the tab bar is a toolbar with buttons for "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The main editor area displays the following Java code:

```
//class for premium member of gym
public class PremiumMember extends GymMember
{
    //attributes of permium member class
    private final double premiumCharge;
    private String personalTrainer;
    private boolean isFullPayment;
    private double paidAmount;
    private double discountAmount;

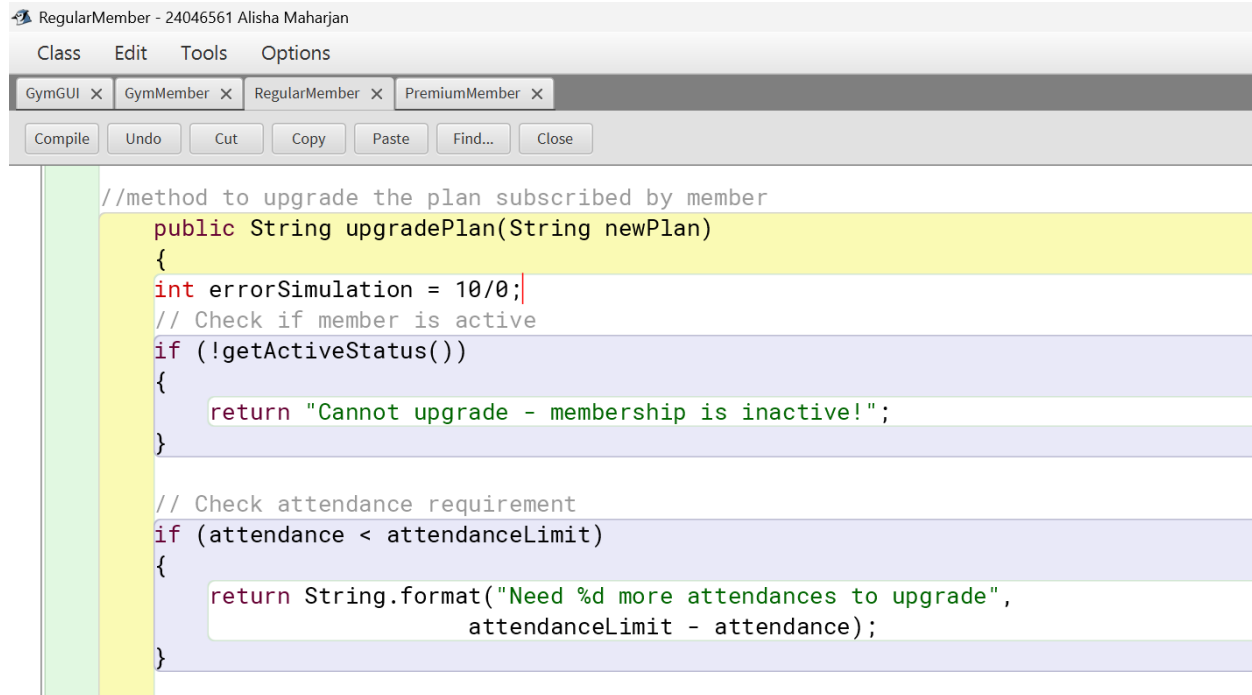
    //constructor to initialize premium member fields
    public PremiumMember(int id,String name,String location,String phone,String email,String gender,String DOB,
    {
        super(id, name, location, phone, email, gender, DOB, membershipStartDate);
        this.premiumCharge=50000;
        this.paidAmount=0;
        this.isFullPayment=false;
        this.discountAmount=0;
        this.personalTrainer=personalTrainer;
    }

    //corresponding accessor method of attributes of premium member class
    public double getPremiumCharge()
    {
        return premiumCharge;
    }
    public String getPersonalTrainer()
    {
        return personalTrainer;
    }
    public boolean isFullPayment()
```

Figure 36: Solving Syntax error

7.2 Runtime Error

Some of the error does not occur during compiling and it occurs during execution, so such type of error is known as runtime error. So, here I introduced this error by adding a extra line of code which creates arithmetic exception.



The screenshot shows a Java IDE window titled "RegularMember - 24046561 Alisha Maharjan". The menu bar includes "Class", "Edit", "Tools", and "Options". The tab bar shows "GymGUI", "GymMember", "RegularMember", and "PremiumMember". The toolbar contains "Compile", "Undo", "Cut", "Copy", "Paste", "Find...", and "Close". The code editor displays the following Java code for the `RegularMember` class:

```
//method to upgrade the plan subscribed by member
public String upgradePlan(String newPlan)
{
    int errorSimulation = 10/0;
    // Check if member is active
    if (!getActiveStatus())
    {
        return "Cannot upgrade - membership is inactive!";
    }

    // Check attendance requirement
    if (attendance < attendanceLimit)
    {
        return String.format("Need %d more attendances to upgrade",
                               attendanceLimit - attendance);
    }
}
```

The line `int errorSimulation = 10/0;` is highlighted in yellow, indicating a runtime error (ArithmeticException: / by zero).

Figure 37: Runtime error

Here, when this line is added in the code the code compiles, but it will show arithmetic error since 10 cannot be divided by zero. For solving this problem, I commented that line so that it won't affect the code.

```
//method to upgrade the plan subscribed by member
public String upgradePlan(String newPlan)
{
    //int errorSimulation = 10/0;
    // Check if member is active
    if (!getActiveStatus())
    {
        return "Cannot upgrade - membership is inactive!";
    }

    // Check attendance requirement
    if (attendance < attendanceLimit)
    {
        return String.format("Need %d more attendances to upgrade",
            attendanceLimit - attendance);
    }

    // Check if already on this plan
    if (newPlan.equalsIgnoreCase(plan))
    {
        return "Already subscribed to " + plan + " plan";
    }
}
```

Figure 38: Solving Runtime error

7.3 Logical Error

Logical error are the errors which compiles and runs but gives unexpected results. So, in my project I created a logical error by changing the sign in this attendance < attendanceLimit to attendance > attendanceLimit.

```
// Check attendance requirement
if (attendance < attendanceLimit)
{
    return String.format("Need %d more attendances to upgrade",
        attendanceLimit - attendance);
}
```

Figure 39: Creating logical error

```
// Check attendance requirement
if (attendance > attendanceLimit)
{
    return String.format("Need %d more attendances to upgrade",
        attendanceLimit - attendance);
}
```

Figure 40: Logical error

This error just created the error while marking attendance since it is not logically valid so for solving this error, I again used the proper logic for this code.

8. Conclusion

This whole project was based on developing a program about Gym Management System which helps the gym staffs to keep the record both regular and premium members. This project handled the gym memberships, tracked attendance and processed the payments with features like upgrading plans and discounts.

While doing this project I got to learn about object-oriented programming, designing user interfaces and managing data in Java. The difficulties and challenges I faced during this project was while developing user interface and making sure that the UI was easy to handle and is smooth enough to use. By testing, debugging and doing various research on development process I was able to overcome the difficulties. So, in conclusion this project helped me to develop technical skills.

9. Appendix

9.1 Gym Member

```
//abstract class for gym member

public abstract class GymMember
{

    //protected access modifier for the attributes

    protected int id;

    protected String name;

    protected String location;

    protected String phone;

    protected String email;
```

```
protected String gender;  
  
protected String DOB;  
  
protected String membershipStartDate;  
  
protected int attendance;  
  
protected double loyaltyPoints;  
  
protected boolean activeStatus;
```

```
//constructor to initialize Gym member fields
```

```
public GymMember(int id, String name, String location, String phone, String email,  
String gender, String DOB, String membershipStartDate)
```

```
{  
  
    this.id=id;  
  
    this.name=name;  
  
    this.location=location;  
  
    this.phone=phone;  
  
    this.email=email;  
  
    this.gender=gender;  
  
    this.DOB=DOB;  
  
    this.membershipStartDate=membershipStartDate;  
  
    this.attendance=0;  
  
    this.loyaltyPoints=0;  
  
    this.activeStatus=false;  
  
}
```

//corresponding accessor methods

public int getId()

{

 return id;

}

public String getName()

{

 return name;

}

public String getLocation()

{

 return location;

}

public String getPhone()

{

 return phone;

}

public String getEmail()

{

 return email;

}

public String getDOB()

{


```

        return DOB;
    }

    public String MembershipStartDate()
    {
        return membershipStartDate;
    }

    public int getAttendance()
    {
        return attendance;
    }

    public double getLoyaltyPoints()
    {
        return loyaltyPoints;
    }

    public boolean getActiveStatus()
    {
        return activeStatus;
    }

```

//an abstract method markAttendance to track the attendance of gym member

```
public abstract void markAttendance();
```

//method to track active status of the member

```
public void activeMembership()
```

```

{
    this.activeStatus=true;
}

public void deactivateMembership()
{
    if(activeStatus)
    {
        this.activeStatus = false;
    }
    else
    {
        System.out.println("The membership is already inactive");
    }
}

//method to set
public void resetMember()
{
    this.activeStatus=false;
    this.attendance=0;
    this.loyaltyPoints=0;
}

//method to display the output
public void display()

```

```

{
    System.out.println("Id: " + id);
    System.out.println("Name: " + name);
    System.out.println("Location: " + location);
    System.out.println("Phone: " + phone);
    System.out.println("Email: " + email);
    System.out.println("Gender: " + gender);
    System.out.println("DOB: " + DOB);
    System.out.println("Memebership Start Date: " + membershipStartDate);
    System.out.println("Attendance: " + attendance);
    System.out.println("Loyalty Points: " + loyaltyPoints);
    System.out.println("Active Status: " + activeStatus);
}
}

```

9.2 Regular Member

```

//class to track the regular memeber of gym
public class RegularMember extends GymMember
{
    //private attributes of regular member class
    private final int attendanceLimit;
    private boolean isEligibleForUpgrade;
}

```

```
private String removalReason;
```

```
private String referralSource;
```

```
private String plan;
```

```
private double price;
```

```
//constructor to initialize the regular member fields
```

```
public RegularMember(int id,String name,String location,String phone,String  
email,String gender,String DOB,String membershipStartDate,String referralSource)
```

```
{
```

```
    super(id,name,location,phone,email,gender,DOB,membershipStartDate);
```

```
    this.isEligibleForUpgrade=false;
```

```
    this.attendanceLimit=30;
```

```
    this.plan="basic";
```

```
    this.price=6500;
```

```
    this.removalReason=" ";
```

```
    this.referralSource=referralSource;
```

```
}
```

```
//corresponding accessor method of attributes of regular member class
```

```
public int getAttendanceLimit()
```

```
{
```

```
    return attendanceLimit;
```

```
}
```

```
public boolean isEligibleForUpgrade()
```

```
{
```

```
    return isEligibleForUpgrade;
```

```
}
```

```
public String getPlan()
```

```
{
```

```
    return plan;
```

```
}
```

```
public double getPrice()
```

```
{
```

```
    return price;
```

```
}
```

```
public String getRemovalReason()
```

```
{
```

```
    return removalReason;
```

```
}
```

```
public String getReferralSource()
```

```
{
```

```
    return referralSource;
```

```
}
```

```
//method to mark the attendance of regular class
```

```
@Override
```

```
public void markAttendance()
{
    attendance++; //increament in attendance by 1
    loyaltyPoints += 5; //increament in loyalty point by 5 points
}
```

//method to get plan price based on plan type

```
public double getPlanPrice(String plan)
```

```
{
    switch(plan)
    {
        case "Basic":
            return 6500;
        case "Standard":
            return 12500;
        case "Deluxe":
            return 18500;
        default:
            return -1;
    }
}
```

//method to upgrade the plan subscribed by member

```

public String upgradePlan(String newPlan)

{
    // Check if member is active
    if (!getActiveStatus())
    {
        return "Cannot upgrade - membership is inactive!";
    }

    // Check attendance requirement
    if (attendance < attendanceLimit)
    {
        return String.format("Need %d more attendances to upgrade",
                               attendanceLimit - attendance);
    }

    // Check if already on this plan
    if (newPlan.equalsIgnoreCase(plan))
    {
        return "Already subscribed to " + plan + " plan";
    }

    // Validate and set new plan
    switch(newPlan.toLowerCase())

```

```

{
    case "standard":
        this.plan = "Standard";
        this.price = 12500;
        isEligibleForUpgrade = false; // Reset eligibility
        return "Upgraded to Standard plan (Rs. 12,500/month)";

    case "deluxe":
        this.plan = "Deluxe";
        this.price = 18500;
        isEligibleForUpgrade = false; // Reset eligibility
        return "Upgraded to Deluxe plan (Rs. 18,500/month)";

    case "basic":
        return "Cannot downgrade to Basic plan";

    default:
        return "Invalid plan selection!";
}
}

```

//method to revert regular member details

```
public void revertRegularMember(String removalReason)
```



```

{
    super.resetMember();

    this.isEligibleForUpgrade=false;

    this.plan="basic";

    this.price=6500;

    this.removalReason=removalReason;
}

//method to display regular memeber details

@Override

public void display()
{
    super.display(); //displays all the common member details

    System.out.println("Plan: " + plan);

    System.out.println("Price: " + price);

    if(!removalReason.isEmpty())
    {
        System.out.println("Removal Reason: " + removalReason);
    }
}
}

```

9.3 Premium Member

```
//class for premium member of gym

public class PremiumMember extends GymMember
{
    //attributes of permium member class

    private final double premiumCharge;

    private String personalTrainer;

    private boolean isFullPayment;

    private double paidAmount;

    private double discountAmount;


    //constructor to initialize premium member fields

    public PremiumMember(int id,String name,String location,String phone,String
email,String gender,String DOB,String membershipStartDate,String personalTrainer)
    {
        super(id,name,location,phone,email,gender,DOB,membershipStartDate);

        this.premiumCharge=50000;

        this.paidAmount=0;

        this.isFullPayment=false;

        this.discountAmount=0;

        this.personalTrainer=personalTrainer;
    }
}
```

//corresponding accessor method of attributes of premium member class

```
public double getPremiumCharge()
```

```
{
```

```
    return premiumCharge;
```

```
}
```

```
public String getPersonalTrainer()
```

```
{
```

```
    return personalTrainer;
```

```
}
```

```
public boolean isFullPayment()
```

```
{
```

```
    return isFullPayment;
```

```
}
```

```
public double getPaidAmount()
```

```
{
```

```
    return paidAmount;
```

```
}
```

```
public double getDiscountAmount()
```

```
{
```

```
    return discountAmount;
```

```
}
```

```
//method to mark the attendance of premium class
```

```
@Override
```

```
public void markAttendance()
```

```
{
```

```
    attendance++; //increament by 1
```

```
    loyaltyPoints += 10; //increament by 10 points
```

```
}
```

```
//method to pay the due amount
```

```
public String payDueAmount(double paidAmount)
```

```
{
```

```
    if(this.isFullPayment==true)
```

```
    {
```

```
        return "Payment is already completed.";
```

```
    }
```

```
    this.paidAmount += paidAmount;
```

```
    if(this.paidAmount > premiumCharge)
```

```
    {
```

```
        //this.paidAmount -= paidAmount;
```

```
        return "Paid Amount exceeds premium charge. Payment unsuccessful.";
```

```
    }
```

```
    if(this.paidAmount == premiumCharge)
```

```
    {
```

```

        isFullPayment = true;

        this.calculateDiscount(); // Automatically calculate discount on full payment

        return "Full payment completed. Payment successfull.Discount applied: " +
this.discountAmount;

    }

    else

    {

        double remainingAmount = premiumCharge - this.paidAmount;

        return "Payment successfull. Remaining Amount: " + remainingAmount;

    }

}

```

//method to calculate discount amount

```

public void calculateDiscount()

{

    if(isFullPayment==true)

    {

        discountAmount = 0.1 * premiumCharge;

        System.out.println("Discount : " + discountAmount);

    }

    else

    {

        discountAmount = 0;

    }

}

```

```
        System.out.println("No discount is applied since you have not paid full  
amount.");
```

```
    }
```

```
}
```

```
//method to revert premium member details
```

```
public void revertPremiumMember()
```

```
{
```

```
    super.resetMember();
```

```
    personalTrainer=" ";
```

```
    isFullPayment=false;
```

```
    paidAmount=0;
```

```
    discountAmount=0;
```

```
}
```

```
//method to display premium member details
```

```
@Override
```

```
public void display()
```

```
{
```

```
    super.display(); //displays the common member details
```

```
    System.out.println("Premium charge : " + premiumCharge);
```

```
    System.out.println("Personal Trainer : " + personalTrainer);
```

```
    System.out.println("Is Full Payment : " + isFullPayment);
```

```
        System.out.println("Paid Amount : " + paidAmount);

        if(isFullPayment==true)

        {

            System.out.println("Discount Amount : " + discountAmount);

        }

    }

}
```

9.4 GymGUI

```
import java.util.ArrayList;

import javax.swing.*;

import java.awt.*;

import java.awt.Insets;

import javax.swing.JFrame;

import javax.swing.JLabel;

import java.awt.event.*;

import javax.swing.border.Border;

import java.io.FileNotFoundException;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.FileWriter;

import java.io.FileReader;
```

```

public class GymGUI
{
    private JFrame frame;

    private JPanel panel,buttonPanel;

    private JLabel
titleLabel,nameLabel,phoneLabel,genderLabel,dobLabel,addressLabel;

    private JTextField nameField,phoneField;

    private JRadioButton maleButton,femaleButton;

    private JComboBox<String> daysCombo,monthsCombo,yearCombo;

    private JTextArea addressField;

    private JCheckBox termCheckBox;

    private JButton submitButton,resetButton;


    public static void main(String[] args)
    {
        //creating an array list to store object pf type regular member and premium
member
        ArrayList<GymMember>members=new ArrayList<GymMember>();

        ArrayList<RegularMember>regularMembers=new ArrayList<RegularMember>();

        ArrayList<PremiumMember>premiumMembers=new
ArrayList<PremiumMember>();

```



```

//creating JFrame

JFrame frame=new JFrame("Gym Membership");

frame.setSize(1500,1500);

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

frame.setLayout(null);


// Regular Member Panel

JPanel regularPanel = new JPanel();

regularPanel.setBounds(20, 20, 650, 370);

regularPanel.setLayout(null);

regularPanel.setBorder(BorderFactory.createTitledBorder("Add a Regular
Member"));

frame.add(regularPanel);


// Regular Member Fields

// Name

JLabel regularNameLabel = new JLabel("Name:");

regularNameLabel.setBounds(20, 30, 100, 25);

regularPanel.add(regularNameLabel);


JTextField regularNameField = new JTextField();

regularNameField.setBounds(120, 30, 200, 25);

regularPanel.add(regularNameField);

```

```
// ID

JLabel regularIdLabel = new JLabel("ID:");
regularIdLabel.setBounds(340, 30, 100, 25);
regularPanel.add(regularIdLabel);


JTextField regularIdField = new JTextField();
regularIdField.setBounds(440, 30, 200, 25);
regularPanel.add(regularIdField);


// Location

JLabel regularLocationLabel = new JLabel("Location:");
regularLocationLabel.setBounds(20, 70, 100, 25);
regularPanel.add(regularLocationLabel);


JTextField regularLocationField = new JTextField();
regularLocationField.setBounds(120, 70, 200, 25);
regularPanel.add(regularLocationField);


// Mobile

JLabel regularMobileLabel = new JLabel("Mobile:");
regularMobileLabel.setBounds(340, 70, 100, 25);
regularPanel.add(regularMobileLabel);
```

```

    JTextField regularMobileField = new JTextField();

    regularMobileField.setBounds(440, 70, 200, 25);

    regularPanel.add(regularMobileField);


    // Email

    JLabel regularEmailLabel = new JLabel("Email:");

    regularEmailLabel.setBounds(20, 110, 100, 25);

    regularPanel.add(regularEmailLabel);


    JTextField regularEmailField = new JTextField();

    regularEmailField.setBounds(120, 110, 200, 25);

    regularPanel.add(regularEmailField);


    // DOB

    JLabel regularDobLabel = new JLabel("DOB:");

    regularDobLabel.setBounds(20, 150, 100, 25);

    String[]
days={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","31"};

    String[] months={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};

    String[]
year={"2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010","

```

```
2011","2012","2013","2014","2015","2016","2017","2018","2019","2020","2021","2022","  
2023","2024","2025"};
```

```
JComboBox<String> dayCombo = new JComboBox<>(days);  
dayCombo.setBounds(120, 150, 60, 25);  
regularPanel.add(dayCombo);
```

```
JComboBox<String> monthCombo = new JComboBox<>(months);  
monthCombo.setBounds(190, 150, 60, 25);  
regularPanel.add(monthCombo);
```

```
JComboBox<String> yearCombo = new JComboBox<>(year);  
yearCombo.setBounds(260, 150, 80, 25);  
regularPanel.add(yearCombo);  
regularPanel.add(regularDobLabel);
```

```
// Gender
```

```
JLabel regularGenderLabel = new JLabel("Gender:");  
regularGenderLabel.setBounds(340, 110, 100, 25);  
regularPanel.add(regularGenderLabel);
```

```
JRadioButton regularFemaleButton = new JRadioButton("Female");  
regularFemaleButton.setBounds(440, 110, 70, 25);
```

```
regularPanel.add(regularFemaleButton);
```

```
JRadioButton regularMaleButton = new JRadioButton("Male");
```

```
regularMaleButton.setBounds(520, 110, 60, 25);
```

```
regularPanel.add(regularMaleButton);
```

```
JRadioButton regularOtherButton = new JRadioButton("Other");
```

```
regularOtherButton.setBounds(590, 110, 140, 25);
```

```
regularPanel.add(regularOtherButton);
```

```
ButtonGroup regularGenderGroup = new ButtonGroup();
```

```
regularGenderGroup.add(regularFemaleButton);
```

```
regularGenderGroup.add(regularMaleButton);
```

```
regularGenderGroup.add(regularOtherButton);
```

```
// Membership Start Date
```

```
JLabel regularStartDateLabel = new JLabel("Membership Start Date:");
```

```
regularStartDateLabel.setBounds(20, 190, 200, 25);
```

```
regularPanel.add(regularStartDateLabel);
```

```
String[]
```

```
startDays={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","31"};
```

```
String[] startMonths={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug",  
"Sep","Oct","Nov","Dec"};
```

```
String[]  
startYear={"2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","20  
10","2011","2012","2013","2014","2015","2016","2017","2018","2019","2020","2021","20  
22","2023","2024","2025"};
```

```
JComboBox<String> regularStartDayCombo = new JComboBox<>(days);  
regularStartDayCombo.setBounds(190, 200, 60, 25);  
regularPanel.add(regularStartDayCombo);
```

```
JComboBox<String> regularStartMonthCombo = new JComboBox<>(months);  
regularStartMonthCombo.setBounds(260, 200, 60, 25);  
regularPanel.add(regularStartMonthCombo);
```

```
JComboBox<String> regularStartYearCombo = new JComboBox<>(year);  
regularStartYearCombo.setBounds(330, 200, 80, 25);  
regularPanel.add(regularStartYearCombo);
```

```
// Referral Source
```

```
JLabel referralLabel = new JLabel("Referral Source:");  
referralLabel.setBounds(20, 230, 200, 25);  
regularPanel.add(referralLabel);
```

```

JTextField referralField = new JTextField();

referralField.setBounds(220, 230, 200, 25);

regularPanel.add(referralField);


// Add Regular Member Button

JButton addRegularButton = new JButton("Add a Regular Member");

addRegularButton.setBounds(220, 280, 200, 30);

regularPanel.add(addRegularButton);


// Add Regular Member ActionListener

addRegularButton.addActionListener(new ActionListener()

{

@Override

public void actionPerformed(ActionEvent e)

{

String id=regularIdField.getText();

String name=regularNameField.getText();

String location=regularLocationField.getText();

String mobile=regularMobileField.getText();

String email=regularEmailField.getText();

String referralSource=referralField.getText();


//Get selected gender

```

```

String gender = "";

if (regularMaleButton.isSelected()) {

    gender = "Male";

} else if (regularFemaleButton.isSelected()) {

    gender = "Female";

} else if (regularOtherButton.isSelected()) {

    gender = "Other";

}

```

```

String dob=dayCombo.getSelectedItemId() + "-" + monthCombo.getSelectedItemId()
+ "-" + yearCombo.getSelectedItemId();

```

```

String startDate = regularStartDayCombo.getSelectedItemId() + "-" +
regularStartMonthCombo.getSelectedItemId() + "-" +
regularStartYearCombo.getSelectedItemId();

```

```

//Validation

if (regularIdField.getText().isEmpty())

{

    JOptionPane.showMessageDialog(null, "The ID field cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);

    return;

}

```

```

if (regularNameField.getText().isEmpty())

```



```
{  
    JOptionPane.showMessageDialog(null, "The Name field cannot be empty!",  
"Error", JOptionPane.ERROR_MESSAGE);  
    return;  
}
```

```
if (regularLocationField.getText().isEmpty())  
{  
    JOptionPane.showMessageDialog(null, "The Location field cannot be  
empty!", "Error", JOptionPane.ERROR_MESSAGE);  
    return;  
}
```

```
if (regularMobileField.getText().isEmpty())  
{  
    JOptionPane.showMessageDialog(null, "The Mobile field cannot be empty!",  
"Error", JOptionPane.ERROR_MESSAGE);  
    return;  
}
```

```
if (regularEmailField.getText().isEmpty())  
{  
    JOptionPane.showMessageDialog(null, "The Email field cannot be empty!",  
"Error", JOptionPane.ERROR_MESSAGE);
```

```

        return;
    }

    if (referralField.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(null, "The Referral field cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);

        return;
    }

    int Id;

    try {
        Id = Integer.parseInt(regularIdField.getText());
    } catch (NumberFormatException ex) {
        JOptionPane.showMessageDialog(null, "Please enter a valid numeric ID.");

        return;
    }

    if (!isIdUnique(Id))
    {
        JOptionPane.showMessageDialog(frame, "Member ID already exists.");

        return;
    }

```

```

        RegularMember member = new
RegularMember(Id,name,location,mobile,email,gender,dob,startDate,referralSource);

        members.add(member);

        JOptionPane.showMessageDialog(frame,"Regular member is added
successfully.");

//Clear Inputs

regularIdField.setText("");
regularNameField.setText("");
regularLocationField.setText("");
regularMobileField.setText("");
regularEmailField.setText("");
referralField.setText("");
regularGenderGroup.clearSelection();
yearCombo.setSelectedIndex(0);
monthCombo.setSelectedIndex(0);
dayCombo.setSelectedIndex(0);
regularStartDayCombo.setSelectedIndex(0);
regularStartMonthCombo.setSelectedIndex(0);
regularStartYearCombo.setSelectedIndex(0);
}

```

```

public boolean isIdUnique(int id)
{
    for (GymMember member : members)
    {
        if(member.getId() == id)
        {
            return false;
        }
    }
    return true;
}

});

// Premium Member Panel

JPanel premiumPanel = new JPanel();

premiumPanel.setBounds(20, 400, 650, 420);

premiumPanel.setLayout(null);

premiumPanel.setBorder(BorderFactory.createTitledBorder("Add a Premium
Member"));

frame.add(premiumPanel);

```

```
// Premium Member Fields

// ID

JLabel premiumIdLabel = new JLabel("ID:");
premiumIdLabel.setBounds(20, 30, 100, 25);
premiumPanel.add(premiumIdLabel);


JTextField premiumIdField = new JTextField();
premiumIdField.setBounds(120, 30, 200, 25);
premiumPanel.add(premiumIdField);


// Name

JLabel premiumNameLabel = new JLabel("Name:");
premiumNameLabel.setBounds(340, 30, 100, 25);
premiumPanel.add(premiumNameLabel);


JTextField premiumNameField = new JTextField();
premiumNameField.setBounds(440, 30, 200, 25);
premiumPanel.add(premiumNameField);


// Location

JLabel premiumLocationLabel = new JLabel("Location:");
premiumLocationLabel.setBounds(20, 70, 100, 25);
premiumPanel.add(premiumLocationLabel);
```

```
JTextField premiumLocationField = new JTextField();  
premiumLocationField.setBounds(120, 70, 200, 25);  
premiumPanel.add(premiumLocationField);
```

```
// Phone
```

```
JLabel premiumPhoneLabel = new JLabel("Phone:");  
premiumPhoneLabel.setBounds(340, 70, 100, 25);  
premiumPanel.add(premiumPhoneLabel);
```

```
JTextField premiumPhoneField = new JTextField();  
premiumPhoneField.setBounds(440, 70, 200, 25);  
premiumPanel.add(premiumPhoneField);
```

```
// Email
```

```
JLabel premiumEmailLabel = new JLabel("Email:");  
premiumEmailLabel.setBounds(20, 110, 100, 25);  
premiumPanel.add(premiumEmailLabel);
```

```
JTextField premiumEmailField = new JTextField();  
premiumEmailField.setBounds(120, 110, 200, 25);  
premiumPanel.add(premiumEmailField);
```

```
// DOB
```

```
JLabel premiumDobLabel = new JLabel("DOB:");
```

```
premiumDobLabel.setBounds(20, 150, 100, 25);
```

```
premiumPanel.add(premiumDobLabel);
```

```
String[]
```

```
day={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19","20",  
"21","22","23","24","25","26","27","28","29","30","31"};
```

```
String[] month={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug",  
"Sep","Oct","Nov","Dec"};
```

```
String[]
```

```
years={"2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010",  
"2011","2012","2013","2014","2015","2016","2017","2018","2019","2020","2021","2022",  
"2023","2024","2025"};
```

```
JComboBox<String> premiumDayCombo = new JComboBox<>(day);
```

```
premiumDayCombo.setBounds(120, 150, 60, 25);
```

```
premiumPanel.add(premiumDayCombo);
```

```
JComboBox<String> premiumMonthCombo = new JComboBox<>(month);
```

```
premiumMonthCombo.setBounds(190, 150, 60, 25);
```

```
premiumPanel.add(premiumMonthCombo);
```

```
JComboBox<String> premiumYearCombo = new JComboBox<>(year);
```

```
premiumYearCombo.setBounds(260, 150, 80, 25);

premiumPanel.add(premiumYearCombo);

premiumPanel.add(premiumDobLabel);


// Gender

JLabel premiumGenderLabel = new JLabel("Gender:");

premiumGenderLabel.setBounds(340, 110, 100, 25);

premiumPanel.add(premiumGenderLabel);


JRadioButton premiumFemaleButton = new JRadioButton("Female");

premiumFemaleButton.setBounds(440, 110, 70, 25);

premiumPanel.add(premiumFemaleButton);


JRadioButton premiumMaleButton = new JRadioButton("Male");

premiumMaleButton.setBounds(520, 110, 60, 25);

premiumPanel.add(premiumMaleButton);


JRadioButton premiumOtherButton = new JRadioButton("Other");

premiumOtherButton.setBounds(590, 110, 140, 25);

premiumPanel.add(premiumOtherButton);


ButtonGroup premiumGenderGroup = new ButtonGroup();

premiumGenderGroup.add(premiumFemaleButton);
```



```

premiumGenderGroup.add(premiumMaleButton);

premiumGenderGroup.add(premiumOtherButton);


// Membership Start Date

JLabel premiumStartDateLabel = new JLabel("Membership Start Date:");

premiumStartDateLabel.setBounds(20, 190, 200, 25);

premiumPanel.add(premiumStartDateLabel);


String[]
startDay={"1","2","3","4","5","6","7","8","9","10","11","12","13","14","15","16","17","18","19",
"20","21","22","23","24","25","26","27","28","29","30","31"};

String[] startMonth={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug",
"Sep","Oct","Nov","Dec"};

String[]
startYears={"2000","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010",
"2011","2012","2013","2014","2015","2016","2017","2018","2019","2020","2021","2022",
"2023","2024","2025"};


JComboBox<String> premiumStartDayCombo = new JComboBox<>(days);

premiumStartDayCombo.setBounds(190, 200, 60, 25);

premiumPanel.add(premiumStartDayCombo);


JComboBox<String> premiumStartMonthCombo = new JComboBox<>(months);

premiumStartMonthCombo.setBounds(260, 200, 60, 25);

```

```

premiumPanel.add(premiumStartMonthCombo);

JComboBox<String> premiumStartYearCombo = new JComboBox<>(year);
premiumStartYearCombo.setBounds(330, 200, 80, 25);
premiumPanel.add(premiumStartYearCombo);

// Personal Trainer

JLabel trainerLabel = new JLabel("Personal Trainer:");
trainerLabel.setBounds(20, 230, 200, 25);
premiumPanel.add(trainerLabel);

JTextField trainerField = new JTextField();
trainerField.setBounds(220, 230, 200, 25);
premiumPanel.add(trainerField);

// Add Premium Member Button

JButton addPremiumButton = new JButton("Add a Premium Member");
addPremiumButton.setBounds(220, 280, 200, 30);
premiumPanel.add(addPremiumButton);

//Action listeners to premium member class
addPremiumButton.addActionListener(new ActionListener()
{

```

```

@Override

public void actionPerformed(ActionEvent e)
{
    String name = premiumNameField.getText();
    String id = premiumIdField.getText();
    String location = premiumLocationField.getText();
    String phone = premiumPhoneField.getText();
    String email = premiumEmailField.getText();
    String trainer = trainerField.getText();


    //Get selected gender
    String gender = "";
    if (regularMaleButton.isSelected())
    {
        gender = "Male";
    }
    else if (regularFemaleButton.isSelected())
    {
        gender = "Female";
    }
    else if (regularOtherButton.isSelected())
    {
        gender = "Other";
    }
}

```

```

    }

    String dob = premiumDayCombo.getSelectedItemId() + "-" +
premiumMonthCombo.getSelectedItemId() + "-" +
premiumYearCombo.getSelectedItemId();

    String startDate = regularStartDayCombo.getSelectedItemId() + "-" +
regularStartMonthCombo.getSelectedItemId() + "-" +
regularStartYearCombo.getSelectedItemId();

    //Validation

    if (premiumIdField.getText().isEmpty())

    {

        JOptionPane.showMessageDialog(null, "The ID field cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);

        return;

    }

    if (premiumNameField.getText().isEmpty())

    {

        JOptionPane.showMessageDialog(null, "The Name field cannot be
empty!", "Error", JOptionPane.ERROR_MESSAGE);

        return;

    }

```

```

        if (premiumLocationField.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(null, "The Location field cannot be
empty!", "Error", JOptionPane.ERROR_MESSAGE);

            return;
        }

        if (premiumPhoneField.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(null, "The Mobile field cannot be
empty!", "Error", JOptionPane.ERROR_MESSAGE);

            return;
        }

        if (premiumEmailField.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(null, "The Email field cannot be empty!",
"Error", JOptionPane.ERROR_MESSAGE);

            return;
        }

        if (trainerField.getText().isEmpty())
        {

```

```

        JOptionPane.showMessageDialog(null, "The Trainer field cannot be
empty!", "Error", JOptionPane.ERROR_MESSAGE);

        return;
    }

    int Id;

    try
    {
        Id = Integer.parseInt(premiumIdField.getText());
    }

    catch (NumberFormatException ex)
    {
        JOptionPane.showMessageDialog(null, "Please enter a valid numeric ID.");

        return;
    }

    if (!isIdUnique(Id))
    {
        JOptionPane.showMessageDialog(frame, "Member ID already exists.");

        return;
    }

```

```
PremiumMember member = new
PremiumMember(Id,name,location,phone,email,gender,dob,startDate,trainer);

members.add(member);

JOptionPane.showMessageDialog(frame,"Premium member is added
successfully.");
```

```
//Clear Inputs

premiumIdField.setText("");
premiumNameField.setText("");
premiumLocationField.setText("");
premiumPhoneField.setText("");
premiumEmailField.setText("");
trainerField.setText("");

premiumGenderGroup.clearSelection();
premiumYearCombo.setSelectedIndex(0);
premiumMonthCombo.setSelectedIndex(0);
premiumDayCombo.setSelectedIndex(0);
premiumStartDayCombo.setSelectedIndex(0);
premiumStartMonthCombo.setSelectedIndex(0);
premiumStartYearCombo.setSelectedIndex(0);
}
```

```
public boolean isIdUnique(int id)
```

```

    {
        for (GymMember member : members)
        {
            if(member.getId() == id)
            {
                return false;
            }
        }
        return true;
    }

});

// Panel for Activate Membership

JPanel activatePanel = new JPanel();

activatePanel.setLayout(null);

activatePanel.setBounds(700, 20, 380, 120);

activatePanel.setBorder(BorderFactory.createTitledBorder("Activate
Membership"));

frame.add(activatePanel);

JLabel activateIdLabel = new JLabel("Membership ID:");

activateIdLabel.setBounds(20, 30, 120, 25);

```



```

activatePanel.add(activateIdLabel);

JTextField activateIdField = new JTextField();
activateIdField.setBounds(140, 30, 200, 25);
activatePanel.add(activateIdField);

//Activate Button

JButton activateButton = new JButton("Activate");
activateButton.setBounds(140, 70, 100, 25);
activatePanel.add(activateButton);

//adding action listener
activateButton.addActionListener(e ->
{
    // Retrieve the entered Member ID from the text field
    String idInput = activateIdField.getText();

    // Validate if the ID field is empty
    if (idInput.isEmpty())
    {
        JOptionPane.showMessageDialog(null, "Please enter a Member ID!", "Error",
JOptionPane.ERROR_MESSAGE);

        return;
    }
}

```

```

    }

    try {

        // Parse the input to an integer

        int id = Integer.parseInt(idInput);

        // Check if the ID exists in the members list

        boolean found = false;

        for (GymMember member : members)

        {

            if (member.getId() == id)

            {

                member.activeMembership();    // Call the method to activate
membership

                JOptionPane.showMessageDialog(null, "Membership activated
successfully!", "Success", JOptionPane.INFORMATION_MESSAGE);

                found = true;

                break;

            }

        }

        // If the ID is not found in the list

        if (!found) {

```

```

        JOptionPane.showMessageDialog(null, "Member ID not found!", "Error",
JOptionPane.ERROR_MESSAGE);

    }

}

catch (NumberFormatException ex)

{

    // Handle invalid ID format

    JOptionPane.showMessageDialog(null, "Please enter a valid numeric
Member ID!", "Error", JOptionPane.ERROR_MESSAGE);

}

});

```

```

// Panel for Deactivate Membership

JPanel deactivatePanel = new JPanel();

deactivatePanel.setLayout(null);

deactivatePanel.setBounds(700, 150, 380, 120);

deactivatePanel.setBorder(BorderFactory.createTitledBorder("Deactivate
Membership"));

frame.add(deactivatePanel);

JLabel deactivateIdLabel = new JLabel("Membership ID:");

deactivateIdLabel.setBounds(20, 30, 120, 25);

```

```
deactivatePanel.add(deactivateIdLabel);
```

```
JTextField deactivateIdField = new JTextField();
```

```
deactivateIdField.setBounds(140, 30, 200, 25);
```

```
deactivatePanel.add(deactivateIdField);
```

```
JButton deactivateButton = new JButton("Deactivate");
```

```
deactivateButton.setBounds(140, 70, 100, 25);
```

```
deactivatePanel.add(deactivateButton);
```

```
//adding action listener
```

```
deactivateButton.addActionListener(e ->
```

```
{
```

```
    String idInput = deactivateIdField.getText();
```

```
    if (idInput.isEmpty())
```

```
    {
```

```
        JOptionPane.showMessageDialog(null, "Please enter a Member ID!", "Error",  
JOptionPane.ERROR_MESSAGE);
```

```
        return;
```

```
    }
```

```
try {
```

```

int id = Integer.parseInt(idInput);

boolean found = false;

for (GymMember member : members)
{
    if (member.getId() == id)
    {
        if (!member.getActiveStatus())
        {
            JOptionPane.showMessageDialog(null, "Membership is already
inactive!", "Info", JOptionPane.INFORMATION_MESSAGE);
        }
        else
        {
            member.deactivateMembership(); // Use the provided method

            JOptionPane.showMessageDialog(null, "Membership deactivated
successfully!", "Success", JOptionPane.INFORMATION_MESSAGE);
        }

        found = true;

        break;
    }
}

```

```

        if (!found)
        {
            JOptionPane.showMessageDialog(null, "Member ID not found!", "Error",
JOptionPane.ERROR_MESSAGE);
        }
    }

    catch (NumberFormatException ex)
    {
        JOptionPane.showMessageDialog(null, "Please enter a valid numeric
Member ID!", "Error", JOptionPane.ERROR_MESSAGE);
    }
});

```

```

// Panel for Mark Attendance

JPanel attendancePanel = new JPanel();

attendancePanel.setLayout(null);

attendancePanel.setBounds(700, 290, 380, 150);

attendancePanel.setBorder(BorderFactory.createTitledBorder("Mark
Attendance"));

frame.add(attendancePanel);

```

```

JLabel attendanceIdLabel = new JLabel("Membership ID:");

attendanceIdLabel.setBounds(20, 30, 120, 25);

```

```
attendancePanel.add(attendanceIdLabel);
```

```
JTextField attendanceIdField = new JTextField();
```

```
attendanceIdField.setBounds(140, 30, 200, 25);
```

```
attendancePanel.add(attendanceIdField);
```

```
JLabel isActiveLabel = new JLabel("Is Member Active:");
```

```
isActiveLabel.setBounds(20, 70, 120, 25);
```

```
attendancePanel.add(isActiveLabel);
```

```
JComboBox<String> isActiveComboBox = new JComboBox<>(new String[]{"Yes",  
"No"});
```

```
isActiveComboBox.setBounds(140, 70, 200, 25);
```

```
attendancePanel.add(isActiveComboBox);
```

```
JButton markAttendanceButton = new JButton("Mark Attendance");
```

```
markAttendanceButton.setBounds(140, 110, 150, 25);
```

```
attendancePanel.add(markAttendanceButton);
```

```
//mark attendance action listener
```

```
markAttendanceButton.addActionListener(e ->
```

```
{
```

```
    String idInput = attendanceIdField.getText();
```

```

        if (idInput == null || idInput.isEmpty())
        {
            JOptionPane.showMessageDialog(null, "Please enter a Member ID!", "Error",
JOptionPane.ERROR_MESSAGE);

            return;
        }

        try
        {
            int id = Integer.parseInt(idInput); // Parse the ID directly

            boolean found = false;

            for (GymMember member : members)
            {
                if (member.getId() == id)
                {
                    member.markAttendance(); // Calls the overridden method

                    JOptionPane.showMessageDialog(null, "Attendance marked
successfully!", "Success", JOptionPane.INFORMATION_MESSAGE);

                    found = true;

                    break;
                }
            }

```



```

    }

    if (!found)
    {
        JOptionPane.showMessageDialog(null, "Member ID not found!", "Error",
JOptionPane.ERROR_MESSAGE);
    }
}

catch (NumberFormatException ex)
{
    JOptionPane.showMessageDialog(null, "Please enter a valid numeric
Member ID!", "Error", JOptionPane.ERROR_MESSAGE);
}

});

```

```

// Revert Member Panel

JPanel revertMemberPanel = new JPanel();

revertMemberPanel.setLayout(null);

revertMemberPanel.setBorder(BorderFactory.createTitledBorder("Revert
Member"));

revertMemberPanel.setBounds(700, 450, 400, 150);

frame.add(revertMemberPanel);

```

```

// Member ID Label and Text Field

JLabel revertMemberIdLabel = new JLabel("Member ID:");

revertMemberIdLabel.setBounds(20, 30, 100, 25);

revertMemberPanel.add(revertMemberIdLabel);


JTextField revertMemberIdField = new JTextField();

revertMemberIdField.setBounds(120, 30, 200, 25);

revertMemberPanel.add(revertMemberIdField);


// Button for Revert Regular Member

JButton revertRegularMemberButton = new JButton("Revert RegularMember");

revertRegularMemberButton.setBounds(20, 70, 180, 30);

revertMemberPanel.add(revertRegularMemberButton);


//adding action listener

revertRegularMemberButton.addActionListener(e ->

{

    String id = revertMemberIdField.getText(); // Get ID from text field

    boolean memberFound = false;


    for (GymMember member : members)

    {

```

```

        if (String.valueOf(member.getId()).equals(id))

        { // Match the member ID

            memberFound = true;


            if (member instanceof RegularMember)

            {

                String removalReason = JOptionPane.showInputDialog("Enter removal
reason:");

                ((RegularMember) member).revertRegularMember(removalReason); //
Cast and call method

                JOptionPane.showMessageDialog(frame, "Regular membership reverted
successfully!");

            }

            else

            {

                JOptionPane.showMessageDialog(frame, "Member is not a Regular
Member!", "Error", JOptionPane.ERROR_MESSAGE);

            }

            return; // Exit the loop once the member is processed

        }

    }

    if (!memberFound)

    {

```

```
        JOptionPane.showMessageDialog(frame, "Member ID not found!", "Error",  
JOptionPane.ERROR_MESSAGE);
```

```
    }  
});
```

```
// Button for Revert Premium Member
```

```
JButton revertPremiumMemberButton = new JButton("Revert PremiumMember");  
revertPremiumMemberButton.setBounds(210, 70, 180, 30);  
revertMemberPanel.add(revertPremiumMemberButton);
```

```
//adding action listener
```

```
revertPremiumMemberButton.addActionListener(e ->
```

```
{
```

```
    String id = revertMemberIdField.getText(); // Get ID from text field
```

```
    boolean memberFound = false;
```

```
    for (GymMember member : members)
```

```
    {
```

```
        if (String.valueOf(member.getId()).equals(id))
```

```
        { // Match the member ID
```

```
            memberFound = true;
```

```

        if (member instanceof PremiumMember)
        {
            ((PremiumMember) member).revertPremiumMember(); // Cast and call
method
            JOptionPane.showMessageDialog(frame, "Premium membership
reverted successfully!");
        }
        else
        {
            JOptionPane.showMessageDialog(frame, "Member is not a Premium
Member!", "Error", JOptionPane.ERROR_MESSAGE);
        }
        return; // Exit the loop once the member is processed
    }
}

if (!memberFound)
{
    JOptionPane.showMessageDialog(frame, "Member ID not found!", "Error",
JOptionPane.ERROR_MESSAGE);
}
});

// Panel for Upgrade Plan

```

```
JPanel upgradePanel = new JPanel();  
upgradePanel.setLayout(null);  
upgradePanel.setBounds(1100, 20, 380, 150);  
upgradePanel.setBorder(BorderFactory.createTitledBorder("Upgrade Plan"));  
frame.add(upgradePanel);
```

```
JLabel upgradeldLabel = new JLabel("Membership ID:");  
upgradeldLabel.setBounds(20, 30, 120, 25);  
upgradePanel.add(upgradeldLabel);
```

```
JTextField upgradeldField = new JTextField();  
upgradeldField.setBounds(140, 30, 200, 25);  
upgradePanel.add(upgradeldField);
```

```
JLabel planLabel = new JLabel("Select Plan:");  
planLabel.setBounds(20, 70, 120, 25);  
upgradePanel.add(planLabel);
```

```
JComboBox<String> planComboBox = new JComboBox<>(new String[]{"basic",  
"standard", "deluxe"});  
planComboBox.setBounds(140, 70, 200, 25);  
upgradePanel.add(planComboBox);
```

```

JButton upgradeButton = new JButton("Upgrade Plan");

upgradeButton.setBounds(140, 110, 150, 25);

upgradePanel.add(upgradeButton);


// Upgrade Plan ActionListener

upgradeButton.addActionListener(e -> {

    String idInput = upgradeldField.getText();


    if (idInput.isEmpty()) {

        JOptionPane.showMessageDialog(null, "Please enter Member ID", "Error",
JOptionPane.ERROR_MESSAGE);

        return;

    }


    try {

        int id = Integer.parseInt(idInput);

        boolean found = false;


        for (GymMember member : members) {

            if (member.getId() == id) {

                found = true;


                if (!(member instanceof RegularMember)) {

```

```

JOptionPane.showMessageDialog(null,

    "Only Regular Members can upgrade plans!",

    "Error", JOptionPane.ERROR_MESSAGE);

return;
}

```

```

RegularMember regularMember = (RegularMember) member;

String selectedPlan = (String) planComboBox.getSelectedItemId();

```

```

// Show current status before upgrading

```

```

String statusMessage = "Current Plan: " + regularMember.getPlan() +
"\n" +

    "Attendance: " + regularMember.getAttendance() + "/" +

    regularMember.getAttendanceLimit() + "\n" +

    "Status: " + (regularMember.getActiveStatus() ? "Active" :
"Inactive");

```

```

// Attempt upgrade

```

```

String upgradeResult = regularMember.upgradePlan(selectedPlan);

```

```

// Show complete result

```

```

JOptionPane.showMessageDialog(null,

    statusMessage + "\n\n" + upgradeResult,

```



```

        "Upgrade Status",
        JOptionPane.INFORMATION_MESSAGE);

        break;
    }
}

if (!found) {
    JOptionPane.showMessageDialog(null, "Member ID not found!", "Error",
JOptionPane.ERROR_MESSAGE);
}

} catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(null, "Invalid Member ID format", "Error",
JOptionPane.ERROR_MESSAGE);
}

});

// Panel for Pay Due Amount

JPanel paymentPanel = new JPanel();

paymentPanel.setLayout(null);

paymentPanel.setBounds(1100, 310, 380, 150);

paymentPanel.setBorder(BorderFactory.createTitledBorder("Pay Due Amount"));

frame.add(paymentPanel);

```

```
JLabel paymentIdLabel = new JLabel("Membership ID:");  
paymentIdLabel.setBounds(20, 30, 120, 25);  
paymentPanel.add(paymentIdLabel);
```

```
JTextField paymentIdField = new JTextField();  
paymentIdField.setBounds(140, 30, 200, 25);  
paymentPanel.add(paymentIdField);
```

```
JLabel amountLabel = new JLabel("Amount to Pay:");  
amountLabel.setBounds(20, 70, 120, 25);  
paymentPanel.add(amountLabel);
```

```
JTextField amountField = new JTextField();  
amountField.setBounds(140, 70, 200, 25);  
paymentPanel.add(amountField);
```

```
JButton paymentButton = new JButton("Pay Due Amount");  
paymentButton.setBounds(140, 110, 150, 25);  
paymentPanel.add(paymentButton);
```

```
// Pay Due Amount ActionListener  
paymentButton.addActionListener(e -> {
```

```

String idInput = paymentIdField.getText();

String amountInput = amountField.getText();


    if (idInput == null || idInput.isEmpty() || amountInput == null ||
amountInput.isEmpty()) {

        JOptionPane.showMessageDialog(null, "Please enter both Member ID and
Amount!", "Error", JOptionPane.ERROR_MESSAGE);

        return;

    }


    try {

        int id = Integer.parseInt(idInput);

        double amount = Double.parseDouble(amountInput);

        boolean found = false;


        for (GymMember member : members) {

            if (member.getId() == id) {

                found = true;

                if (member instanceof PremiumMember) {

                    String result = ((PremiumMember) member).payDueAmount(amount);

                    JOptionPane.showMessageDialog(null, result, "Payment Result",
JOptionPane.INFORMATION_MESSAGE);

                } else {

```

```

        JOptionPane.showMessageDialog(null, "Only Premium Members can
make payments!", "Error", JOptionPane.ERROR_MESSAGE);

    }

    break;

}

}

if (!found) {

    JOptionPane.showMessageDialog(null, "Member ID not found!", "Error",
JOptionPane.ERROR_MESSAGE);

}

} catch (NumberFormatException ex) {

    JOptionPane.showMessageDialog(null, "Please enter valid numeric values for
ID and Amount!", "Error", JOptionPane.ERROR_MESSAGE);

}

});

// Panel for Calculate Discount

JPanel discountPanel = new JPanel();

discountPanel.setLayout(null);

discountPanel.setBounds(1100, 180, 380, 120);

discountPanel.setBorder(BorderFactory.createTitledBorder("Calculate Discount"));

frame.add(discountPanel);

```

```

JLabel discountIdLabel = new JLabel("Membership ID:");
discountIdLabel.setBounds(20, 30, 120, 25);
discountPanel.add(discountIdLabel);

JTextField discountIdField = new JTextField();
discountIdField.setBounds(140, 30, 200, 25);
discountPanel.add(discountIdField);

JButton discountButton = new JButton("Calculate Discount");
discountButton.setBounds(140, 70, 150, 25);
discountPanel.add(discountButton);

// Calculate Discount ActionListener
discountButton.addActionListener(e -> {
    String idInput = discountIdField.getText();

    if (idInput == null || idInput.isEmpty()) {
        JOptionPane.showMessageDialog(null, "Please enter a Member ID!", "Error",
JOptionPane.ERROR_MESSAGE);

        return;
    }
}

```

```

try {

    int id = Integer.parseInt(idInput);

    boolean found = false;

    for (GymMember member : members) {

        if (member.getId() == id) {

            found = true;

            if (member instanceof PremiumMember) {

                ((PremiumMember) member).calculateDiscount();

                JOptionPane.showMessageDialog(null, "Discount calculated
successfully!", "Success", JOptionPane.INFORMATION_MESSAGE);

            } else {

                JOptionPane.showMessageDialog(null, "Only Premium Members can
calculate discounts!", "Error", JOptionPane.ERROR_MESSAGE);

            }

            break;

        }

    }

    if (!found) {

        JOptionPane.showMessageDialog(null, "Member ID not found!", "Error",
JOptionPane.ERROR_MESSAGE);

    }
}

```

```

    } catch (NumberFormatException ex) {

        JOptionPane.showMessageDialog(null, "Please enter a valid numeric
Member ID!", "Error", JOptionPane.ERROR_MESSAGE);

    }

});

```

```

// Display Button

```

```

JButton displayButton = new JButton("Display Members");

displayButton.setBounds(500, 900, 140, 30);

frame.add(displayButton);

```

```

// Display Button ActionListener

```

```

displayButton.addActionListener(e -> {

    if (members.isEmpty()) {

        JOptionPane.showMessageDialog(frame, "No members found!",
"Information", JOptionPane.INFORMATION_MESSAGE);

        return;

    }
}

```

```

// Create a text area with scroll pane

```

```

JTextArea textArea = new JTextArea(20, 80);

textArea.setEditable(false);

JScrollPane scrollPane = new JScrollPane(textArea);

```

```

// Build the display content

String header = String.format("%-5s %-15s %-10s %-15s %-10s %-15s %-15s\n",
    "ID", "Name", "Type", "Plan", "Status", "Attendance", "Loyalty");

textArea.append(header + "\n");

for (GymMember member : members) {

    String type = member instanceof RegularMember ? "Regular" : "Premium";

    String plan = member instanceof RegularMember ?
        ((RegularMember)member).getPlan() : "Premium";

    String status = member.getActiveStatus() ? "Active" : "Inactive";

    String row = String.format("%-5d %-15s %-10s %-15s %-10s %-15d %-15.2f\n",
        member.getId(),
        member.getName(),
        type,
        plan,
        status,
        member.getAttendance(),
        member.getLoyaltyPoints());

    textArea.append(row);
}

```



```

    }

    // Show in a dialog

    JOptionPane.showMessageDialog(frame, scrollPane, "All Members",
JOptionPane.INFORMATION_MESSAGE);

    });

// Clear Button

JButton clearButton = new JButton("Clear Fields");

clearButton.setBounds(700, 900, 140, 30);

frame.add(clearButton);

//Clear Button ActionListener

clearButton.addActionListener(e -> {

    // Clear all text fields

    regularIdField.setText("");

    regularNameField.setText("");

    regularLocationField.setText("");

    regularMobileField.setText("");

    regularEmailField.setText("");

    referralField.setText("");

    regularGenderGroup.clearSelection();

```

```
premiumIdField.setText("");  
premiumNameField.setText("");  
premiumLocationField.setText("");  
premiumPhoneField.setText("");  
premiumEmailField.setText("");  
trainerField.setText("");  
premiumGenderGroup.clearSelection();
```

```
// Clear action fields
```

```
activateIdField.setText("");  
deactivateIdField.setText("");  
attendancIdField.setText("");  
revertMemberIdField.setText("");  
upgradIdField.setText("");  
discountIdField.setText("");  
paymentIdField.setText("");  
amountField.setText("");
```

```
// Reset combo boxes
```

```
dayCombo.setSelectedIndex(0);  
monthCombo.setSelectedIndex(0);  
yearCombo.setSelectedIndex(0);
```

```
regularStartDayCombo.setSelectedIndex(0);
regularStartMonthCombo.setSelectedIndex(0);
regularStartYearCombo.setSelectedIndex(0);
premiumDayCombo.setSelectedIndex(0);
premiumMonthCombo.setSelectedIndex(0);
premiumYearCombo.setSelectedIndex(0);
premiumStartDayCombo.setSelectedIndex(0);
premiumStartMonthCombo.setSelectedIndex(0);
premiumStartYearCombo.setSelectedIndex(0);
planComboBox.setSelectedIndex(0);
isActiveComboBox.setSelectedIndex(0);
```

```
JOptionPane.showMessageDialog(frame, "All fields cleared successfully!",
"Clear", JOptionPane.INFORMATION_MESSAGE);
});
```

```
// Save to File Button
```

```
JButton saveToFileButton = new JButton("Save to File");
saveToFileButton.setBounds(500, 950, 140, 30);
frame.add(saveToFileButton);
```

```
// Save to File Button ActionListener
```

```
saveToFileButton.addActionListener(e -> {
```

```

        if (members.isEmpty()) {

            JOptionPane.showMessageDialog(frame, "No members to save!", "Error",
JOptionPane.ERROR_MESSAGE);

            return;

        }

        try (FileWriter writer = new FileWriter("MemberDetails.txt")) {

            // Write header

            writer.write(String.format("%-5s %-15s %-15s %-15s %-25s %-20s %-10s %-
10s %-10s %-15s %-10s %-15s %-15s\n",

                "ID", "Name", "Location", "Phone", "Email", "Membership Start", "Plan",
"Price",

                "Attendance", "Loyalty", "Active", "Full Pay", "Discount"));

            // Write member details

            for (GymMember member : members) {

                if (member instanceof RegularMember) {

                    RegularMember rm = (RegularMember) member;

                    writer.write(String.format("%-5d %-15s %-15s %-15s %-25s %-20s %-
10s %-10.2f %-10d %-15.2f %-10s %-15s %-15.2f\n",

                        rm.getId(), rm.getName(), rm.getLocation(), rm.getPhone(),
rm.getEmail(),

                        rm.MembershipStartDate(), rm.getPlan(), rm.getPrice(),
rm.getAttendance(),

```

```

        rm.getLoyaltyPoints(), rm.getActiveStatus() ? "Yes" : "No", "N/A",
0.0));

    } else if (member instanceof PremiumMember) {

        PremiumMember pm = (PremiumMember) member;

        // Ensure discount is calculated if payment is complete

        if(pm.isFullPayment() && pm.getDiscountAmount() == 0) {

            pm.calculateDiscount();

        }

        writer.write(String.format("%-5d %-15s %-15s %-15s %-25s %-20s %-
10s %-10.2f %-10d %-15.2f %-10s %-15s %-15.2f\n",

            pm.getId(),

            pm.getName(),

            pm.getLocation(),

            pm.getPhone(),

            pm.getEmail(),

            pm.MembershipStartDate(),

            "Premium",

            pm.getPremiumCharge(),

            pm.getAttendance(),

            pm.getLoyaltyPoints(),

            pm.getActiveStatus() ? "Yes" : "No",

            pm.isFullPayment() ? "Yes" : "No",

```

```

        pm.getDiscountAmount()); // This will now show the discount
    }

}

JOptionPane.showMessageDialog(frame, "Member details saved to
MemberDetails.txt", "Success", JOptionPane.INFORMATION_MESSAGE);

} catch (IOException ex) {

    JOptionPane.showMessageDialog(frame, "Error saving file: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);

}

});

// Read from File Button

JButton readFromFileButton = new JButton("Read from File");

readFromFileButton.setBounds(700, 950, 140, 30);

frame.add(readFromFileButton);

// Read from File Button ActionListener

readFromFileButton.addActionListener(e -> {

    try (BufferedReader reader = new BufferedReader(new
FileReader("MemberDetails.txt"))) {

        // Create text area with scroll pane

```

```

        JTextArea textArea = new JTextArea(20, 80);

        textArea.setEditable(false);

        JScrollPane scrollPane = new JScrollPane(textArea);

        // Read file line by line

        String line;

        while ((line = reader.readLine()) != null) {

            textArea.append(line + "\n");

        }

        // Show in dialog

        JOptionPane.showMessageDialog(frame, scrollPane, "Member Details from
File", JOptionPane.INFORMATION_MESSAGE);

    } catch (FileNotFoundException ex) {

        JOptionPane.showMessageDialog(frame, "File not found: MemberDetails.txt",
"Error", JOptionPane.ERROR_MESSAGE);

    } catch (IOException ex) {

        JOptionPane.showMessageDialog(frame, "Error reading file: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);

    }

});

```

```
    frame.setVisible(true);  
    }  
}
```