CS4001NI Programming

60% Individual Coursework

2025 Spring

Student Name: AnushaShrestha

London Met ID: 24046567

College ID: NP01AI4A240061

Group: L1AI3

Assignment Due Date: Friday, May 16, 2025

Assignment Submission Date: Friday, May 16, 2025

24046567 Anusha Shrestha (1).docx

Islington College,Nepal

Document Details

Submission ID

trn:oid:::3618:96195612

Submission Date

May 16, 2025, 12:27 PM GMT+5:45

Download Date

May 16, 2025, 12:28 PM GMT+5:45

File Name

24046567 Anusha Shrestha (1).docx

File Size

26.2 KB

41 Pages

3,557 Words

19,543 Characters

14% Overall Similarity

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

Match Groups

- **45** Not Cited or Quoted 13%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **3** Missing Citation 1%
  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
14%	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

45 Not Cited or Quoted 13%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

3 Missing Citation 1%
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
14%	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	8%
2	Submitted works	iacademy on 2025-05-14	3%
3	Submitted works	University of Technology, Sydney on 2006-09-21	<1%
4	Submitted works	islingtoncollege on 2024-12-31	<1%
5	Submitted works	MCAST on 2014-01-12	<1%
6	Submitted works	Greenwich School of Management on 2014-08-13	<1%
7	Submitted works	AlHussein Technical University on 2024-06-02	<1%
8	Submitted works	National University of Ireland, Galway on 2023-04-30	<1%
9	Submitted works	Swinburne University of Technology on 2025-04-29	<1%
10	Submitted works	University of Hertfordshire on 2025-04-03	<1%

Table of Contents

Table of Figures

Introduction

The aim of this project is to develop a gym membership management system using Java's Object-Oriented Programming (OOP). This project will help us to engage in the implementation of the superclass 'GymMember' and two subclasses 'RegularMember' and 'PremiumMember', which will be used to show the different membership plans. To improv user's involvement, Graphical User Interface (GUI) will also be used, using Java's AWT and Swing.

Details about gym members, such as personal information, membership type, attendance and loyalty points will be recorded by the system in Array List. Users will be able to alter members' information with ease due to GUI, which helps with effective member record management. The system will also allow users to add, update, activate, deactivate and track gym members. Member data including ID, Name, Location, Phone, Email, Gender, Date of Birth, Membership Start Date and additional data on membership type will be stored in an ArrayList. This project will be created in BlueJ while following the core concepts of OOP like Abstraction, Encapsulation, Inheritance and Polymorphism. Additionally, the project will help in developing the skills of Java with the help of our lecturers and tutors.

Aim and Objectives

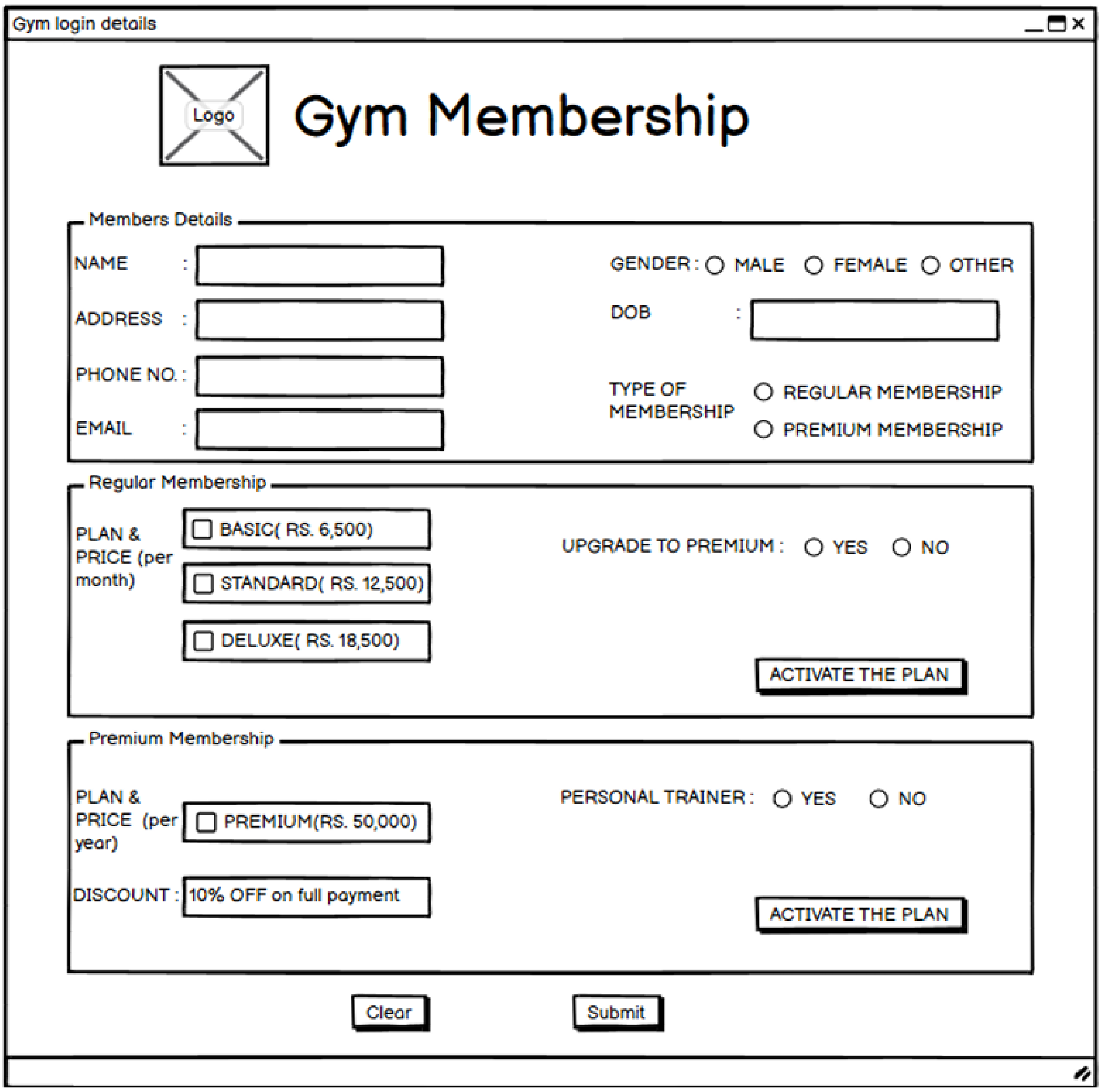
Aim: The aim of this project is to design a gym membership system using Java and Object-Oriented Principle (OOP) principles. The system will also maintain Unique Members IDs, activate attendance tracking, facilitate membership upgrades and calculating discounts for premium members. The system will help with users' engagement to the webpage and effective handling of members data, ensuring in organizing and handling the data effectively.

Objectives:

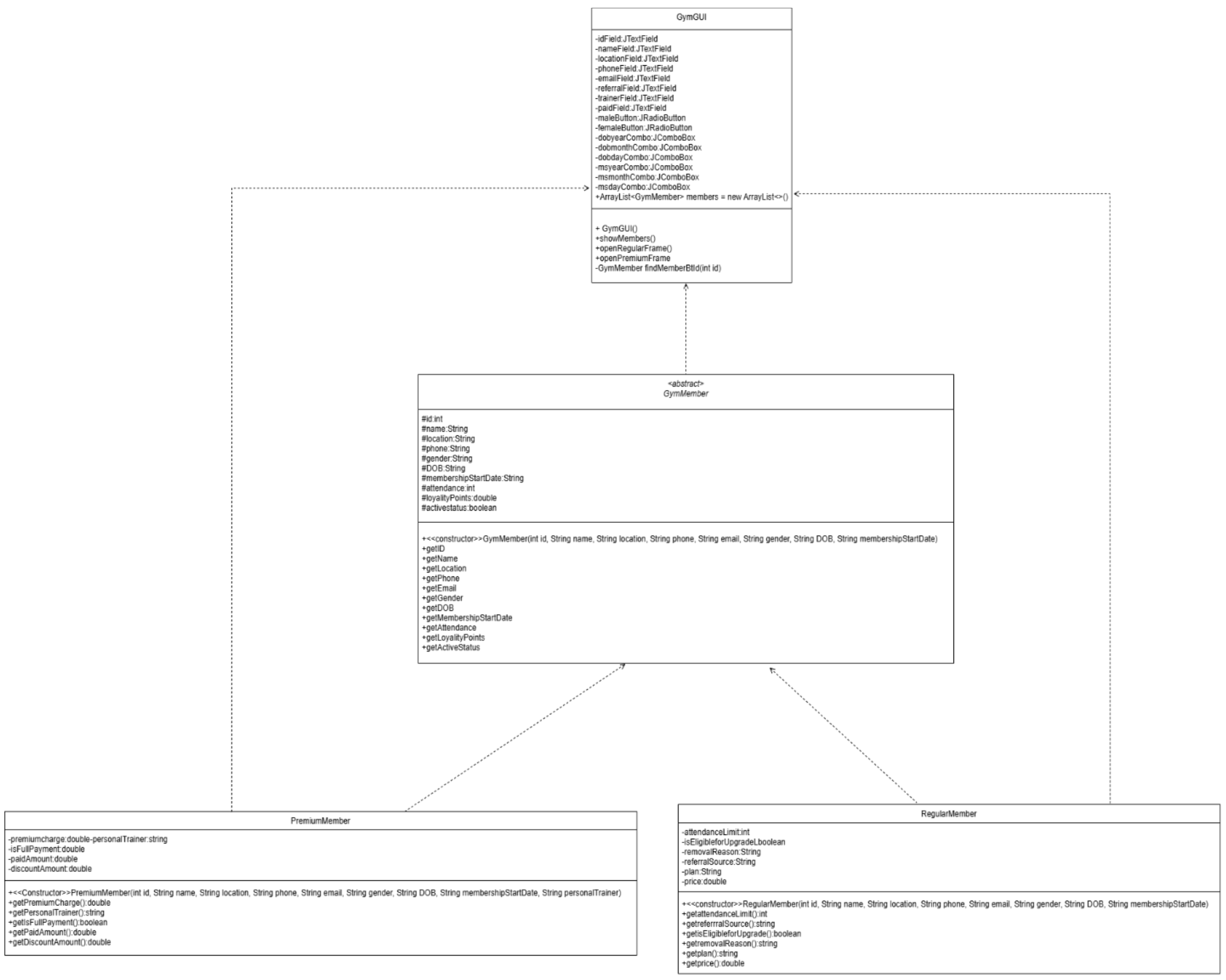
- Implementation of Object-oriented Concepts
- Development of Core Classes

- Tracking Member Information
- Implementation of GUI
- Ensuring Code Readability
- Documentation of System Functionality

GUI Wireframe



Class Diagram


GymGUI
-idField:JTextField
-nameField:JTextField
-locationField:JTextField
-phoneField:JTextField
-emailField:JTextField
-referralField:JTextField
-trainerField:JTextField
-paidField:JTextField
-maleButton:JRadioButton
-femaleButton:JRadioButton
-dobyearCombo:JComboBox
-dobmonthCombo:JComboBox
-dobdayCombo:JComboBox
-msyearCombo:JComboBox
-msmonthCombo:JComboBox
-msdayCombo:JComboBox
+ArrayList<GymMember> members = new ArrayList<>()
+ GymGUI()
+showMembers()
+openRegularFrame()
+openPremiumFrame
-GymMember findMemberBtId(int id)
<abstract>
GymMember
#id:int
#name:String
#location:String
#phone:String
#gender:String
#DOB:String
#membershipStartDate:String
#attendance:int
#loyalityPoints:double
#activestatus:boolean
+<<constructor>>GymMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate)
+getID
+getName
+getLocation
+getPhone
+getEmail
+getGender
+getDOB
+getMembershipStartDate
+getAttendance
+getLoyalityPoints
+getActiveStatus
PremiumMember
-premiumcharge:double-personalTrainer:string
-isFullPayment:double
-paidAmount:double
-discountAmount:double
+<<Constructor>>PremiumMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate, String personalTrainer)
+getPremiumCharge():double
+getPersonalTrainer():string
+getIsFullPayment():boolean
+getPaidAmount():double
+getDiscountAmount():double
RegularMember
-attendanceLimit:int
-isEligibleforUpgradeLboolean
-removalReason:String
-referralSource:String
-plan:String
-price:double
+<<constructor>>RegularMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate)
+getattendanceLimit():int
+getreferrralSource():string
+getisEligibleforUpgrade():boolean
+getremovalReason():string
+getplan():string
+getprice():double

Pseudocode

```
# Pseudocode for GymMember Abstract Class


CREATE an abstract parent class GymMember

DO


- DECLARE protected instance variable id as int

- DECLARE protected instance variable name as string

- DECLARE protected instance variable location as string

- DECLARE protected instance variable phone as string

- DECLARE protected instance variable email as string

- DECLARE protected instance variable gender as string

- DECLARE protected instance variable DOB as string

- DECLARE protected instance variable membershipStartDate as string

- DECLARE protected instance variable attendance as int

- DECLARE protected instance variable loyaltyPoints as double (initialize to 0)

- DECLARE protected instance variable activeStatus as boolean (initialize to false)
- CREATE constructor with parameters:
  id (int), name (string), location (string), phone (string),
  email (string), gender (string), DOB (string), membershipStartDate (string)
  DO
    - SET id = parameter id
    - SET name = parameter name
    - SET location = parameter location
    - SET phone = parameter phone
    - SET email = parameter email
    - SET gender = parameter gender
    - SET DOB = parameter DOB
    - SET membershipStartDate = parameter membershipStartDate
  END


- CREATE method markAttendance with no parameters
  DO
    - IF activeStatus is true THEN
        - INCREMENT attendance by 1
```

```
      - ADD 10 to loyaltyPoints

      - PRINT "Attendance marked."

     ELSE

      - PRINT "Cannot mark attendance. Membership not active."

     END IF

   END



- CREATE method activateMembership with no parameters

   DO

    - SET activeStatus = true

   END



- CREATE method deactivateMembership with no parameters

   DO

    - IF activeStatus is true THEN

      - SET activeStatus = false

     ELSE

      - PRINT "Membership is inactive"
    END IF
  END

- CREATE method resetMember with no parameters
  DO
    - SET activeStatus = false
    - SET attendance = 0
    - SET loyaltyPoints = 0
  END

- CREATE method display returning string
  DO
    - RETURN concatenated string of all member details
  END
END CLASS
```

# Pseudocode for PremiumMember Class

```
CREATE class PremiumMember that EXTENDS GymMember

DO


- DECLARE private constant premiumCharge as double (initialize to 50000)

- DECLARE private instance variable personalTrainer as string

- DECLARE private instance variable isFullPayment as boolean (initialize to false)

- DECLARE private instance variable paidAmount as double (initialize to 0)

- DECLARE private instance variable discountAmount as double (initialize to 0)


- CREATE constructor with parameters:

  id (int), name (string), location (string), phone (string),

  email (string), gender (string), DOB (string),

  membershipStartDate (string), personalTrainer (string)

  DO

    - CALL superclass constructor with first 8 parameters

    - SET personalTrainer = parameter personalTrainer

  END
```

```
- OVERRIDE method markAttendance with no parameters

  DO

    - INCREMENT attendance by 1

    - ADD 10 to loyaltyPoints

  END


- CREATE method payDueAmount with parameter paidAmount (double) returning
string

  DO

    - IF isFullPayment is true THEN

        - RETURN "Payment done."

      END IF


    - CALCULATE totalPaid = this.paidAmount + paidAmount

    - IF totalPaid > premiumCharge THEN

        - RETURN "Error: Total paid amount exceeds the required charge."

      END IF
  - SET this.paidAmount = totalPaid

  - IF this.paidAmount equals premiumCharge THEN

    - SET isFullPayment = true

   END IF


  - CALCULATE remainingAmount = premiumCharge - this.paidAmount

  - RETURN "Payment done. Remaining payment:" + remainingAmount

 END


- CREATE method calculateDiscount with no parameters

 DO

  - IF isFullPayment is true THEN

    - SET discountAmount = 0.1 * premiumCharge

    - PRINT "Discount:" + discountAmount

   ELSE

    - PRINT "No discount. Full Payment is to be done."

   END IF

 END
```

```
- CREATE method revertPremiumMember with no parameters
  DO
    - CALL super.resetMember()
    - SET personalTrainer = ""
    - SET isFullPayment = false
    - SET paidAmount = 0
    - SET discountAmount = 0
  END

- OVERRIDE method display returning string
  DO
    - RETURN concatenated string of all PremiumMember details including:
      ID, name, location, phone, email, gender, DOB, membership start date,
      personal trainer, plan type ("Premium"), active status,
      attendance status, paid amount, full payment status, discount amount
  END
END CLASS
```

```
# Pseudocode for RegularMember Class

CREATE class RegularMember that EXTENDS GymMember
DO

- DECLARE private constant attendanceLimit as int (initialize to 30)
- DECLARE private instance variable isEligibleforUpgrade as boolean (initialize to false)
- DECLARE private instance variable removalReason as string
- DECLARE private instance variable referralSource as string
- DECLARE private instance variable plan as string (initialize to "basic")
- DECLARE private instance variable price as double (initialize to 6500)
- DECLARE private instance variable attendanceMarked as boolean (initialize to false)

- CREATE constructor with parameters:
  id (int), name (string), location (string), phone (string),
  email (string), gender (string), DOB (string), membershipStartDate (string)

  DO

    - CALL superclass constructor with all parameters

    - SET attendanceLimit = 30

    - SET isEligibleforUpgrade = false

    - SET plan = "basic"

    - SET price = 6500

  END


- CREATE method markAttendance with no parameters

  DO

    - SET attendanceMarked = true

  END


- CREATE method getAttendanceStatus returning string

  DO

    - IF attendanceMarked is true THEN

        - RETURN "Present"
    ELSE

      - RETURN "Absent"

    END IF

  END



- CREATE method LoyalityPoints with no parameters

  DO

    - INCREMENT attendance by 1

    - ADD 5 to loyaltyPoints

    - IF attendance >= attendanceLimit THEN

      - SET isEligibleforUpgrade = true

    END IF

  END



- CREATE method getPlanPrice with parameter plan (string) returning double

  DO

    - SWITCH plan.toLowerCase()

    CASE "basic":
      - RETURN 6500

     CASE "standard":

      - RETURN 12500

     CASE "deluxe":

      - RETURN 18500

     DEFAULT:

      - RETURN -1

    END SWITCH

  END



- CREATE method upgradePlan with parameter newPlan (string) returning string

  DO

    - SET newPrice = CALL getPlanPrice(newPlan)

    - IF newPrice equals -1 THEN

      - RETURN "invalid plan"

     END IF



    - IF isEligibleforUpgrade is false THEN
```

```
      - RETURN "You cannot have an upgrade."
    END IF


  - IF this.plan equals newPlan (case insensitive) THEN
      - RETURN "You have been subscribed to the plan."
    END IF


  - SET this.plan = newPlan
  - SET this.price = newPrice
  - RETURN "Plan has been upgraded to " + newPlan
 END


- CREATE method revertRegularmember with parameter removalReason (string)
 DO
  - CALL super.resetMember()
  - SET isEligibleforUpgrade = false
  - SET plan = "Basic"
  - SET price = 6500
```

```
    - SET removalReason = parameter removalReason
  END


- OVERRIDE method display returning string
  DO
    - RETURN concatenated string of all RegularMember details including:
      ID, name, location, phone, email, gender, DOB,
      referral source, membership type ("Regular"), active status,
      and attendance status
  END
END CLASS
# Pseudocode for Gym Membership GUI System


CREATE CLASS GymGUI
DO
    // Declare all GUI components (as shown previously)
    - DECLARE PRIVATE idField AS JTextField
    - DECLARE PRIVATE nameField AS JTextField
```

```
[...] (all other component declarations from previous example)

// Main method
CREATE METHOD main(String[] args)
DO
    - CREATE NEW GymGUI()
END METHOD

// Constructor
CREATE METHOD GymGUI()
DO
    - CALL initializeMainFrame()
END METHOD

// Initialize main window
CREATE PRIVATE METHOD initializeMainFrame()
DO
    - CREATE frame AS NEW JFrame("Gym Membership System")
    - SET frame.size = (800, 600)

    - SET frame.defaultCloseOperation = JFrame.EXIT_ON_CLOSE

    - SET frame.layout = null



    - CALL addTitleLabel(frame)

    - CALL addNavigationButtons(frame)



    - SET frame.visible = true

  END METHOD



  // Add title label

  CREATE PRIVATE METHOD addTitleLabel(frame: JFrame)

  DO

    - CREATE titleLabel AS NEW JLabel("Welcome to the Gym Membership",
JLabel.CENTER)

    - SET titleLabel.bounds = (100, 10, 500, 100)

    - SET titleLabel.font = NEW Font("Arial", Font.BOLD, 16)

    - CALL frame.add(titleLabel)

  END METHOD
```

```
// Add navigation buttons

CREATE PRIVATE METHOD addNavigationButtons(frame: JFrame)

DO

    - CREATE regularButton AS NEW JButton("Regular Member")

    - SET regularButton.bounds = (125, 100, 500, 100)

    - SET regularButton.action = CALL openRegularFrame()

    - CALL frame.add(regularButton)



    - CREATE premiumButton AS NEW JButton("Premium Member")

    - SET premiumButton.bounds = (125, 250, 500, 100)

    - SET premiumButton.action = CALL openPremiumFrame()

    - CALL frame.add(premiumButton)



    - CREATE showButton AS NEW JButton("View Members")

    - SET showButton.bounds = (125, 400, 500, 100)

    - SET showButton.action = CALL showMembers()

    - CALL frame.add(showButton)
```

```
END METHOD



// Show all members

CREATE PRIVATE METHOD showMembers()

DO

   - CREATE frame AS NEW JFrame("All Members")

   - SET frame.size = (800, 600)

   - SET frame.layout = NEW GridLayout(1, 2)



   - CREATE regularTextArea AS NEW JTextArea()

   - CREATE premiumTextArea AS NEW JTextArea()

   - SET regularTextArea.editable = false

   - SET premiumTextArea.editable = false



   - FOR EACH member IN members

      DO

         - IF member INSTANCEOF RegularMember

            DO
```

```
                - CALL regularTextArea.append(member.display() + "\n\n")
            END
        - ELSE IF member INSTANCEOF PremiumMember
            DO
                - CALL premiumTextArea.append(member.display() + "\n\n")
            END
      END

    - CREATE scrollRegular AS NEW JScrollPane(regularTextArea)
    - CREATE scrollPremium AS NEW JScrollPane(premiumTextArea)
    - CALL frame.add(scrollRegular)
    - CALL frame.add(scrollPremium)
    - SET frame.visible = true
END METHOD

// Open regular member frame
CREATE PRIVATE METHOD openRegularFrame()
DO
```

```
    - CREATE rframe AS NEW JFrame("Regular Member")

    - SET rframe.size = (800, 600)

    - SET rframe.defaultCloseOperation = JFrame.DISPOSE_ON_CLOSE

    - SET rframe.layout = null



    - CALL addRegularFormComponents(rframe)

    - CALL addRegularActionButtons(rframe)



    - SET rframe.visible = true

END METHOD



// Add form components for regular member

CREATE PRIVATE METHOD addRegularFormComponents(frame: JFrame)

DO

    - CREATE idLabel AS NEW JLabel("ID:")

    - SET idLabel.bounds = (30, 100, 100, 30)

    - CREATE idField AS NEW JTextField(20)

    - SET idField.bounds = (120, 100, 150, 30)
    - CALL frame.add(idLabel)

    - CALL frame.add(idField)



    [...] (add all other form components similarly)

END METHOD



// Add action buttons for regular member

CREATE PRIVATE METHOD addRegularActionButtons(frame: JFrame)

DO

    - CREATE saveButton AS NEW JButton("SAVE")

    - SET saveButton.bounds = (450, 320, 150, 25)

    - SET saveButton.action = CALL saveMembersToFile()

    - CALL frame.add(saveButton)



    [...] (add all other action buttons similarly)

END METHOD



// Save members to file
```

```
CREATE PRIVATE METHOD saveMembersToFile()
DO
   - TRY
      DO
         - CREATE writer AS NEW BufferedWriter(NEW
FileWriter("MemberDetails.txt", true))
         - CALL writer.write(memberData)
         - CALL writer.close()
         - SHOW "Members data saved" message
      END
   - CATCH IOException
      DO
         - SHOW "Error saving to file" message
      END
END METHOD


// Find member by ID
CREATE PRIVATE METHOD findMemberById(id: int) RETURNS GymMember
DO
```

```
        - FOR EACH member IN members
           DO
              - IF member.getId() == id
                 DO
                    - RETURN member
                 END
           END
        - RETURN null
     END METHOD
  END CLASS
```

Method Description

- addMember()

  This method is used to add new members(either regular or premium) to the system. It collects input from the GUI from form fields. It also checks duplicate IDs to avoid the same member twice.
- activateMember(int id)

  This methodis used to activate a member by their ID. It searches the list of members and updates activeStatus to true.If the member is already active or not found, the system shows a relevant message.
- DeactivateMember(int id)

  This method is used to deactivate members by their ID. . It searches the list of members and updates activeStatus to false. It manages the availability of members without deleting their data.
- markAttendance(int id)

  This method is used when the attendance button is clicked. It finds the members with the given ID and updates their loyalty points.
- upgradePlan(int id)
- This method allows a RegularMember to be upgraded to a premium member. It finds the selected regular member by ID and collects additional information such as personal trainer name. Then it creates a new PremiumMember object with the same data and replaces the existing object in the list.
- revertToRegular(int id)

  This method does the reverse of upgradePaln. It converts a premium member back to a regular one by creating a new RegualrMember object using the original details and removing the premium version.
- makePayment(int id)

  This method processes payment for a membr. It calculates the final payable amount after applying discount and displays it on the screen.
- displayAllMembers()

  This method displays all members(both regular and premium) in a new frame or panel. IT loops through the ArrayList and formats each member's details beatly so that users can view all current data.

- saveToFile()

  This method writes the member data into a text file. It uses BufferedWriter and FileWriter to format and store each fiekd in customs. Making the file human-readable.
- readFromFile()

  This method reads and displays the contents of the text files saved by rhe program. It uses BufferedReader to each line and prints it on the console or a frame.
- clearFormFields()

  This method clears all the input fields in the form.
- getMemberById(int id)

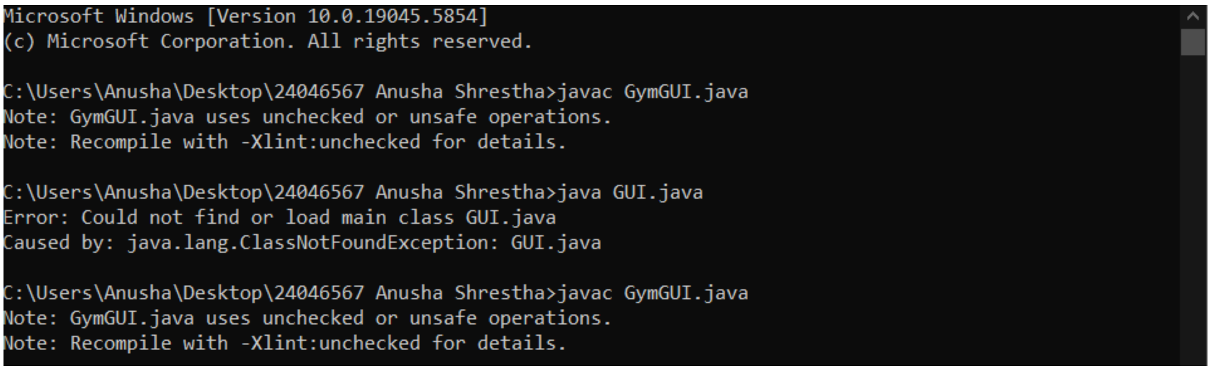
  This method is used to fetch a member from the list using their unique ID.It helps to avoid repetitive code.
- toString()

  Each member class(Regular or Premium) can override the toString() method to return a formatted string of the member’s details.

Testing

Test 1: Compile and run the program using command prompt/terminal

Objective	Compile and run the program using command prompt/terminal
Action	javac GymGUI.java command was typed in the command prompt terminal.
Expected Output	The program should be compiled and run. The GUI should be displayed.
Actual Output	The GUI frame popped up..
Result	The test was successful.



```
Microsoft Windows [Version 10.0.19045.5854]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Anusha\Desktop\24046567 Anusha Shrestha>javac GymGUI.java
Note: GymGUI.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\Anusha\Desktop\24046567 Anusha Shrestha>java GUI.java
Error: Could not find or load main class GUI.java
Caused by: java.lang.ClassNotFoundException: GUI.java

C:\Users\Anusha\Desktop\24046567 Anusha Shrestha>javac GymGUI.java
Note: GymGUI.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
```

Figure 1 command prompt terminal to compile

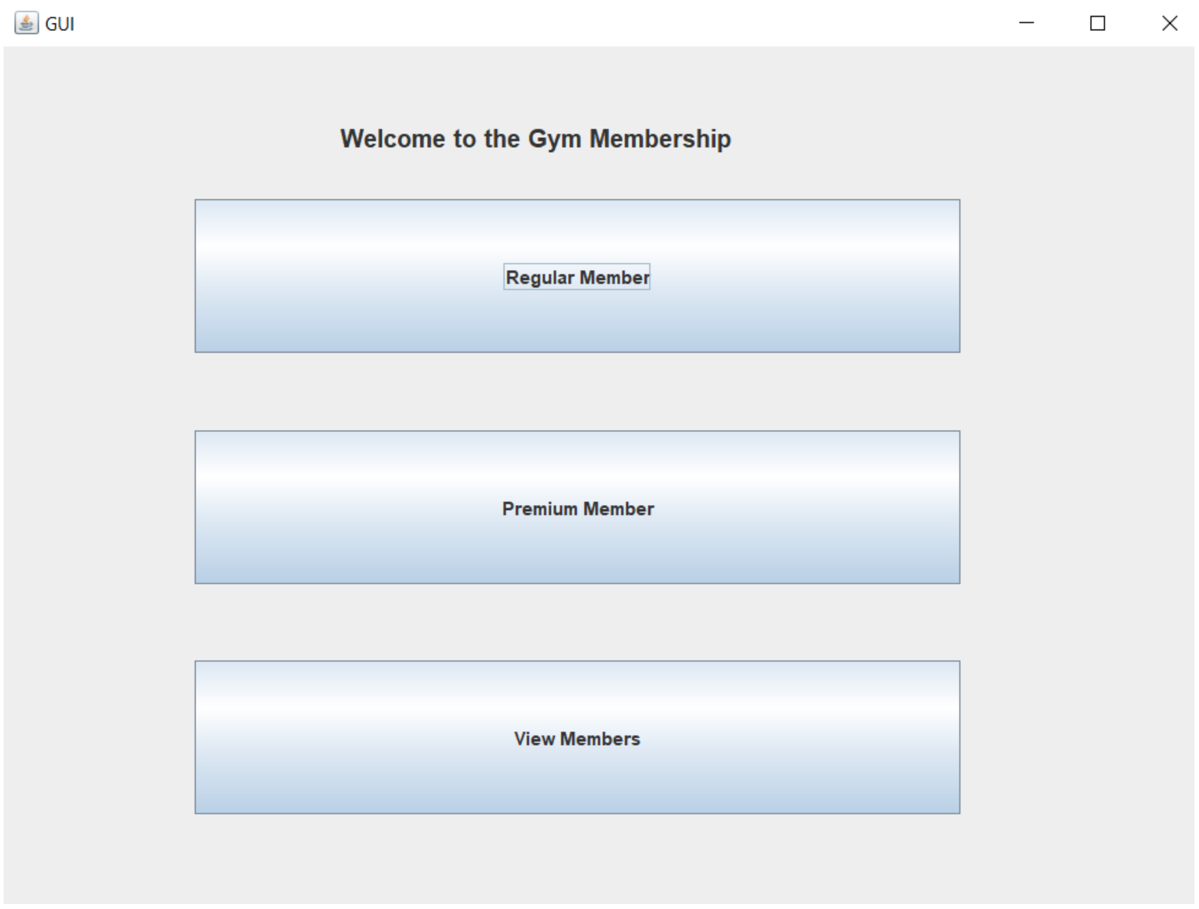
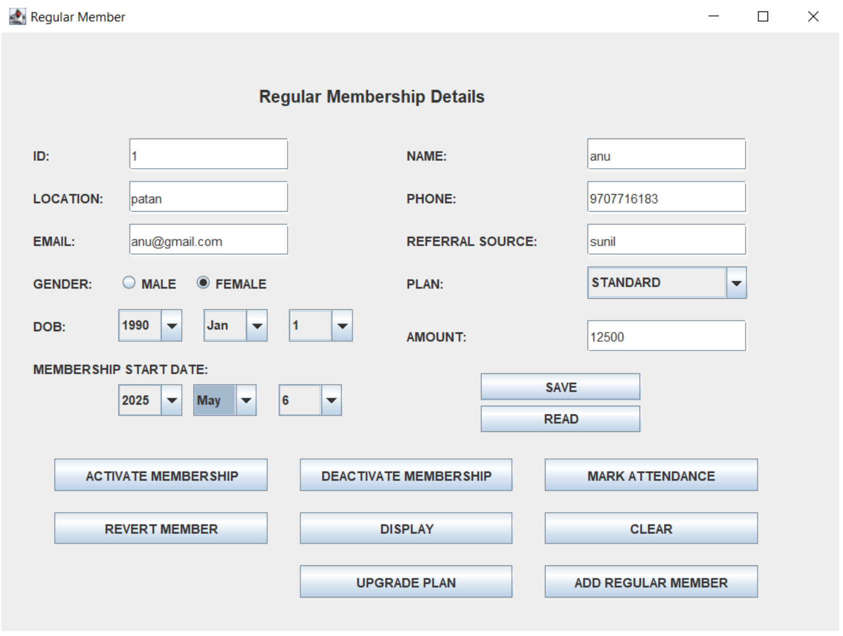


Figure 2 output after compiling in command prompt

Test 2: Adding regular and premium members respectively

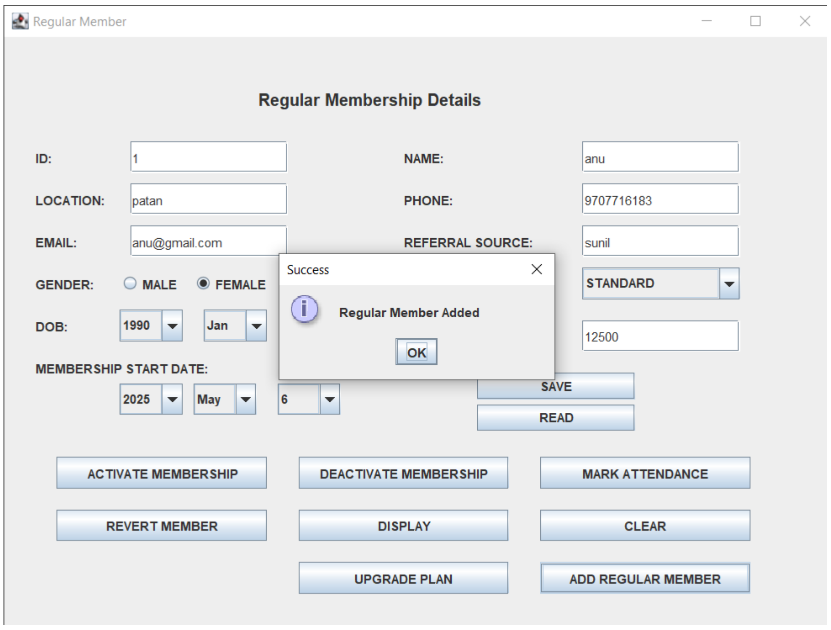
Objective	Adding regular and premium members respectively
Action	All the fields were filled and “Add Regular Member/Add Premium Member “ button was clicked.
Expected Output	A dialog box should pop up with “Regular Member Added/ Premium Member added” message.
Actual Output	A dialog box popped up with “Regular Member Added/ Premium Member Added” message.
Result	The test was successful.


Regular Member

Regular Membership Details

ID: 1
NAME: anu
LOCATION: patan
PHONE: 9707716183
EMAIL: anu@gmail.com
REFERRAL SOURCE: sunil
GENDER: ○ MALE ● FEMALE
PLAN: STANDARD
DOB: 1990 Jan 1
AMOUNT: 12500
MEMBERSHIP START DATE: 2025 May 6

Figure 3 fillings details of regular member

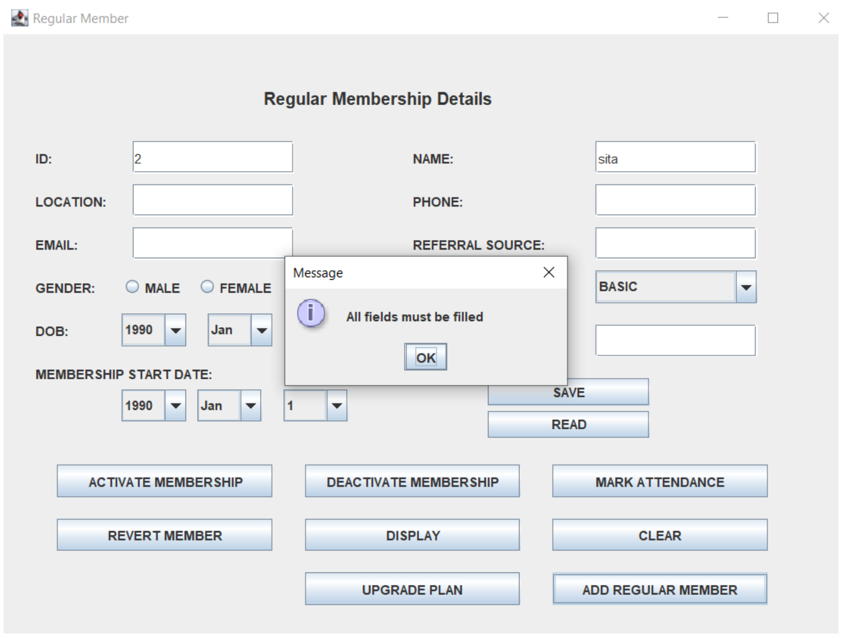

Regular Member

Regular Membership Details

ID: 1
NAME: anu
LOCATION: patan
PHONE: 9707716183
EMAIL: anu@gmail.com
REFERRAL SOURCE: sunil
GENDER: MALE [✓ FEMALE]
STANDARD
DOB: 1990 Jan
12500
MEMBERSHIP START DATE: 2025 May 6

Success
Regular Member Added
OK

Figure 4 adding regular member

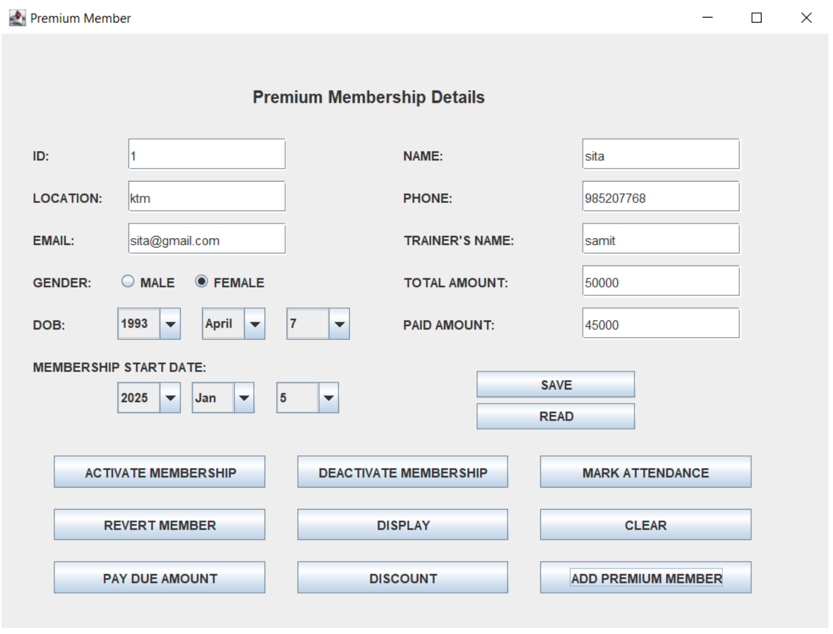

Regular Member

Regular Membership Details

ID: 2
NAME: sita
LOCATION:
PHONE:
EMAIL:
REFERRAL SOURCE:
GENDER: MALE FEMALE
BASIC
DOB: 1990 Jan
MEMBERSHIP START DATE: 1990 Jan 1

Message
All fields must be filled
OK

Figure 5 filling incomplete data in regular membership



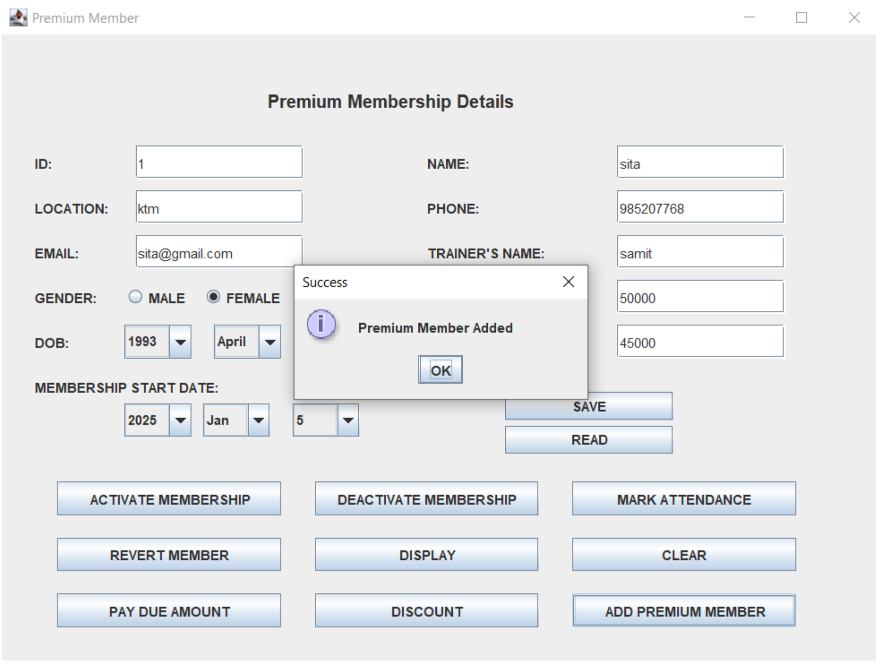
Premium Member

Premium Membership Details

ID:	1	NAME:	sita
LOCATION:	ktm	PHONE:	985207768
EMAIL:	sita@gmail.com	TRAINER'S NAME:	samit
GENDER:	○ MALE [✓ FEMALE]	TOTAL AMOUNT:	50000
DOB:	1993 April 7	PAID AMOUNT:	45000

MEMBERSHIP START DATE: 2025 Jan 5

Figure 6 filling details of premium member



Premium Member

Premium Membership Details

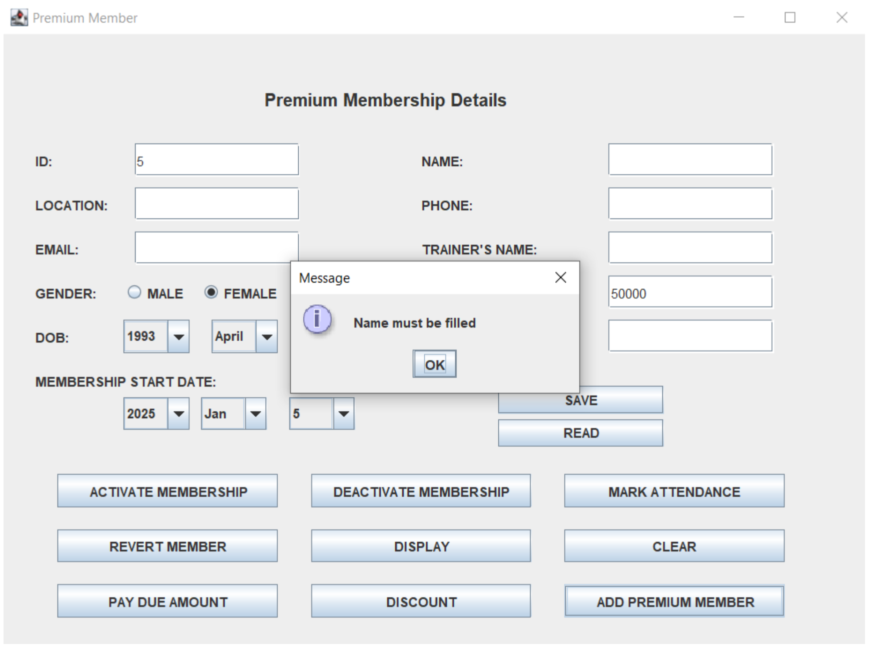
ID:	1	NAME:	sita
LOCATION:	ktm	PHONE:	985207768
EMAIL:	sita@gmail.com	TRAINER'S NAME:	samit
GENDER:	○ MALE [✓ FEMALE]		50000
DOB:	1993 April		45000

MEMBERSHIP START DATE: 2025 Jan 5

Success

Premium Member Added

Figure 7 adding premium member


Premium Member

Premium Membership Details

ID:	5	NAME:	
LOCATION:		PHONE:	
EMAIL:		TRAINER'S NAME:	
GENDER:	○ MALE [✓ FEMALE]		50000
DOB:	1993, April		
MEMBERSHIP START DATE:	2025, Jan, 5		

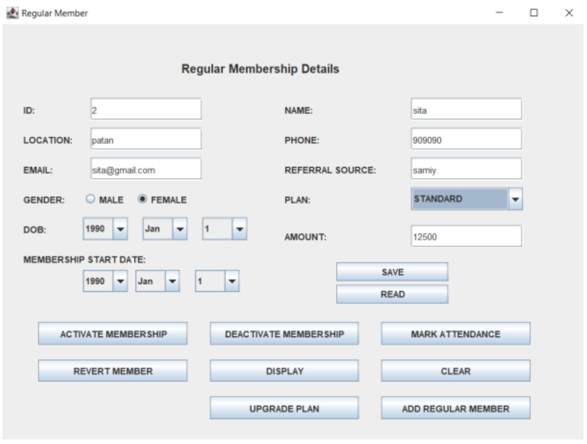
Message

Name must be filled

Figure 8 filling incomplete details in premium membership

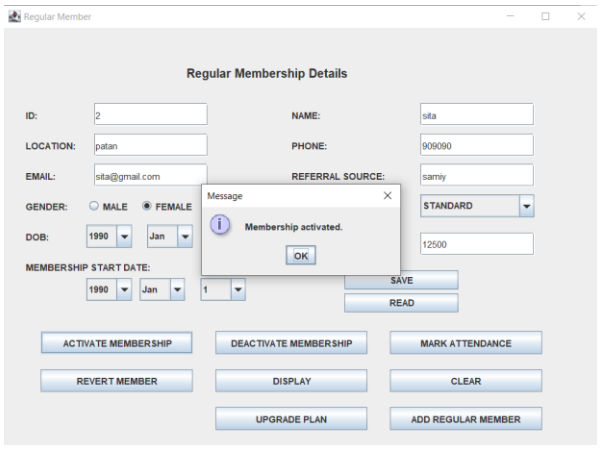
Test 3: Test case for mark attendance

Objective	Test case for mark attendance
Action	All the required fields were filled up "Attendance marked" button was clicked.
Expected Output	The program should be compiled and show a dialog box with a message" Attendance marked.".
Actual Output	A dialog box popped up when the button was clicked.
Result	The test was successful.


Regular Member

Regular Membership Details

ID: 2
NAME: sita
LOCATION: patan
PHONE: 909090
EMAIL: sita@gmail.com
REFERRAL SOURCE: samiy
GENDER: MALE ● FEMALE
PLAN: STANDARD
DOB: 1990 Jan 1
AMOUNT: 12500
MEMBERSHIP START DATE: 1990 Jan 1

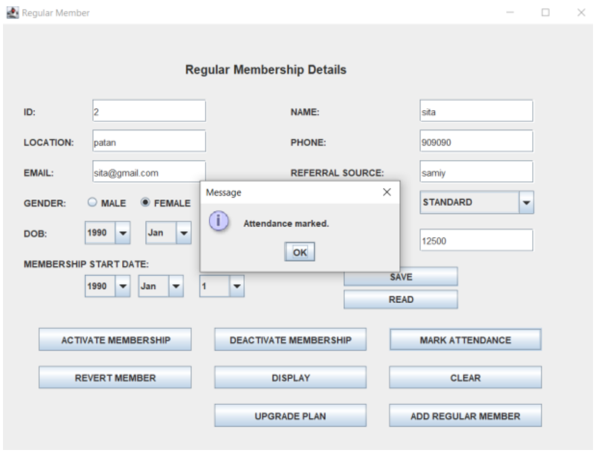

Regular Member

Regular Membership Details

ID: 2
NAME: sita
LOCATION: patan
PHONE: 909090
EMAIL: sita@gmail.com
REFERRAL SOURCE: samiy
GENDER: MALE ● FEMALE
STANDARD
DOB: 1990 Jan
12500
MEMBERSHIP START DATE: 1990 Jan 1

Message
Membership activated.

Figure 9 activating membership in regular membership[

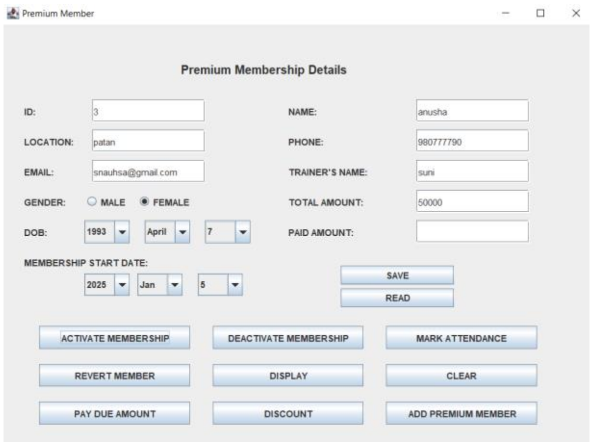

Regular Member

Regular Membership Details

ID: 2
NAME: sita
LOCATION: patan
PHONE: 909090
EMAIL: sita@gmail.com
REFERRAL SOURCE: samiy
GENDER: MALE ● FEMALE
STANDARD
DOB: 1990 Jan
12500
MEMBERSHIP START DATE: 1990 Jan 1

Message
Attendance marked.

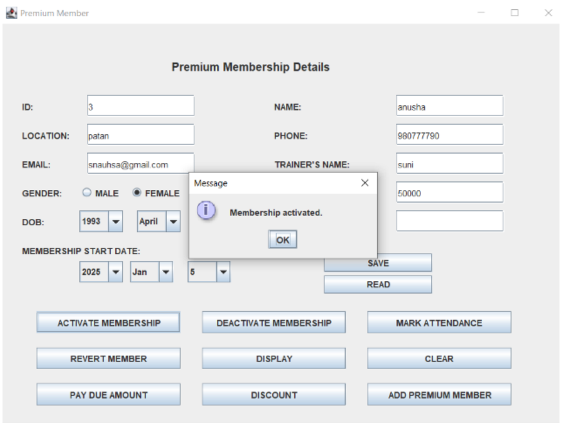
Figure 10 attendance marked for regular membership


Premium Member

Premium Membership Details

ID:	3	NAME:	anusha
LOCATION:	patan	PHONE:	980777790
EMAIL:	snauhsa@gmail.com	TRAINER'S NAME:	suni
GENDER:	○ MALE ● FEMALE	TOTAL AMOUNT:	50000
DOB:	1993 April 7	PAID AMOUNT:	

MEMBERSHIP START DATE: 2025 Jan 5


Premium Member

Premium Membership Details

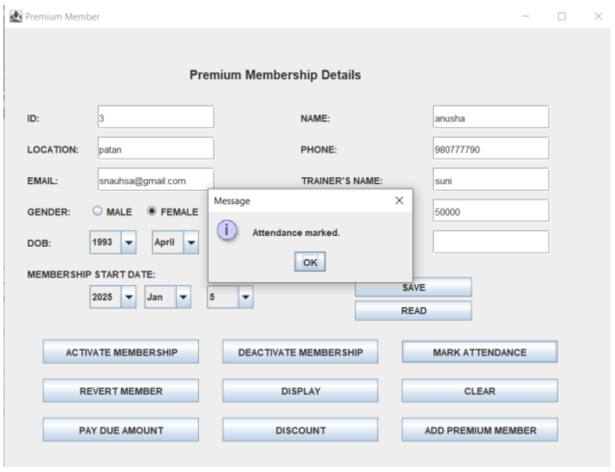
ID:	3	NAME:	anusha
LOCATION:	patan	PHONE:	980777790
EMAIL:	snauhsa@gmail.com	TRAINER'S NAME:	suni
GENDER:	○ MALE ● FEMALE		50000
DOB:	1993 April		

MEMBERSHIP START DATE: 2025 Jan 5

Message

Membership activated.

Figure 11 activating membership in premium membership


Premium Member

Premium Membership Details

ID:	3	NAME:	anusha
LOCATION:	patan	PHONE:	980777790
EMAIL:	snauhsa@gmail.com	TRAINER'S NAME:	suni
GENDER:	○ MALE ● FEMALE		50000
DOB:	1993 April		

MEMBERSHIP START DATE: 2025 Jan 5

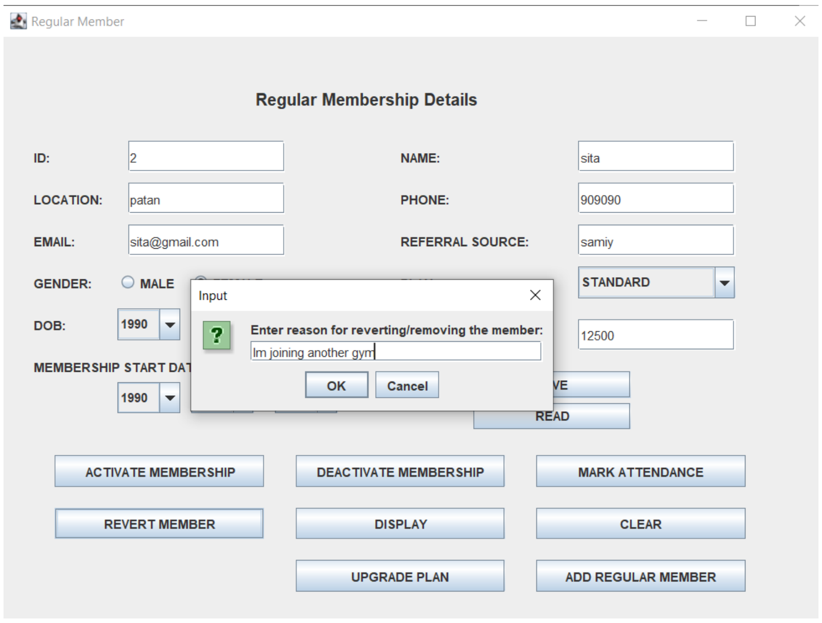
Message

Attendance marked.

Figure 12 marking attendance in premium membership

Test 4: Test case to calculate discount, pay due, revert members

Objective	Test case to calculate discount, pay due, revert members
Action	The necessary fields were filled, and a discount was applied if the user has full done full payment, if full payment is not made then the due amount is shown and in revert the members the data of the user is deleted. The button Discount, Pay Due Amount and Revert button need to be clicked in order to work.
Expected Output	The discount of 10% should be given, the amount needed to be paid will be shown and the users' data will be deleted.The pop-up box should also be popped up.
Actual Output	Discount was should of 10% when the discount button was pressed , the due amount was shown when the pay due amount button was pressed, and the data of the user was removed when revert button was clicked. The pop-up box popped up.
Result	The test was successful.


Regular Member

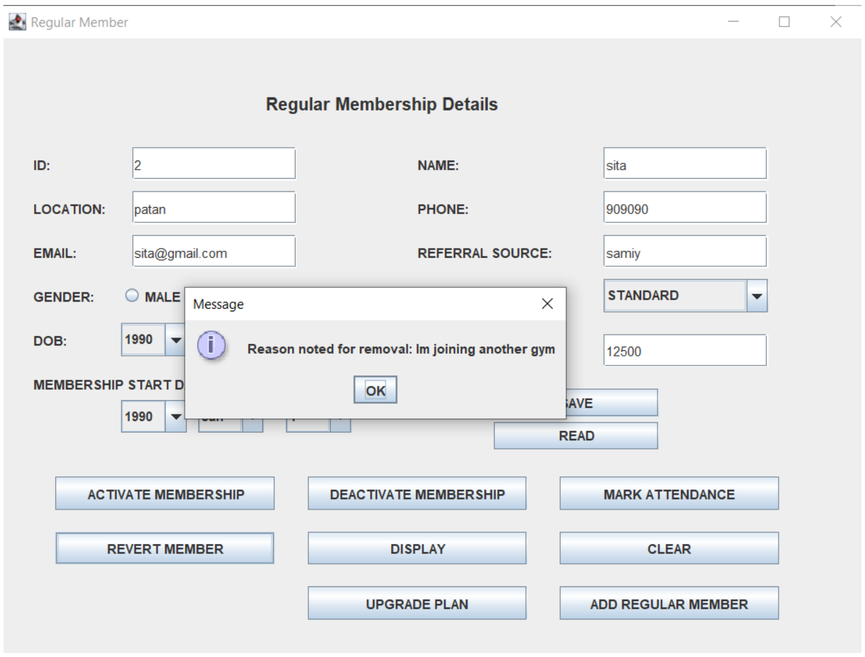
Regular Membership Details

ID: 2
NAME: sita
LOCATION: patan
PHONE: 909090
EMAIL: sita@gmail.com
REFERRAL SOURCE: samiy
GENDER: MALE
STANDARD
DOB: 1990
12500
MEMBERSHIP START DAT
1990

Input

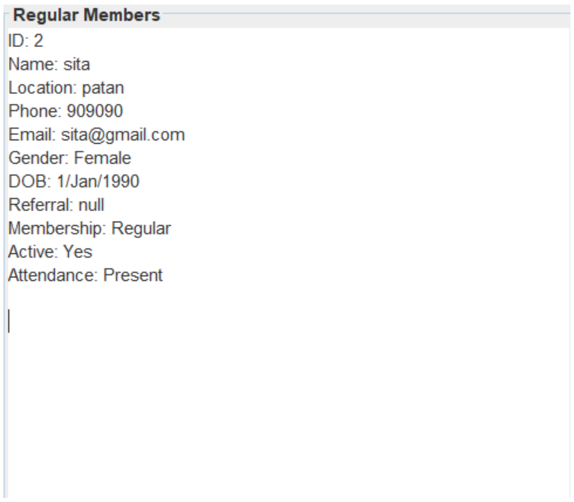
Enter reason for reverting/removing the member:
Im joining another gym

Figure 13 entering reason to revert for regular member



Reason noted for removal: Im joining another gym

Figure 14 reason for revert was done for regular member



```
Regular Members
ID: 2
Name: sita
Location: patan
Phone: 909090
Email: sita@gmail.com
Gender: Female
DOB: 1/Jan/1990
Referral: null
Membership: Regular
Active: Yes
Attendance: Present
```

Figure 15 before the removal of regular member

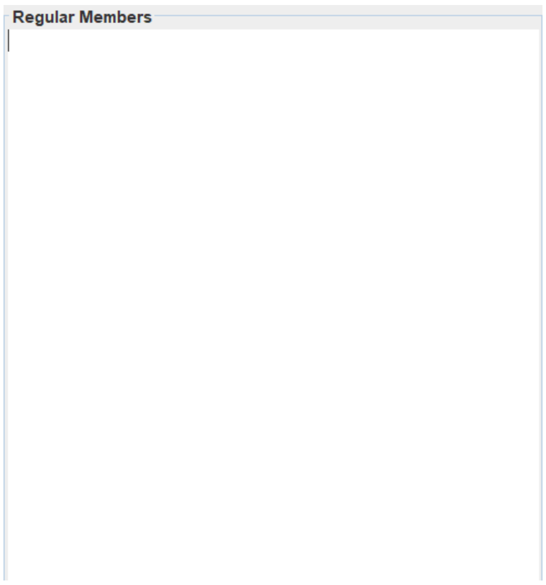
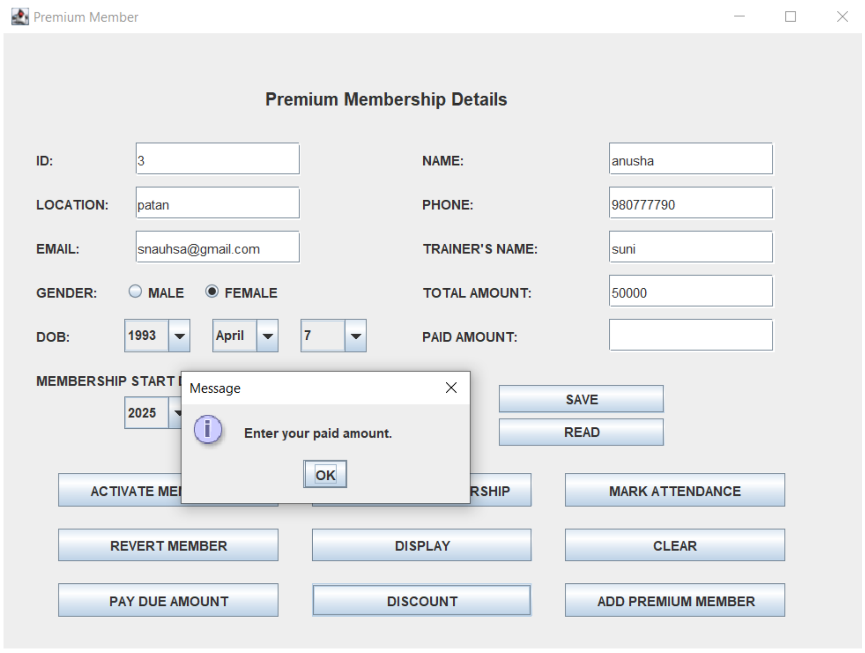


Figure 16 after reverting regular member

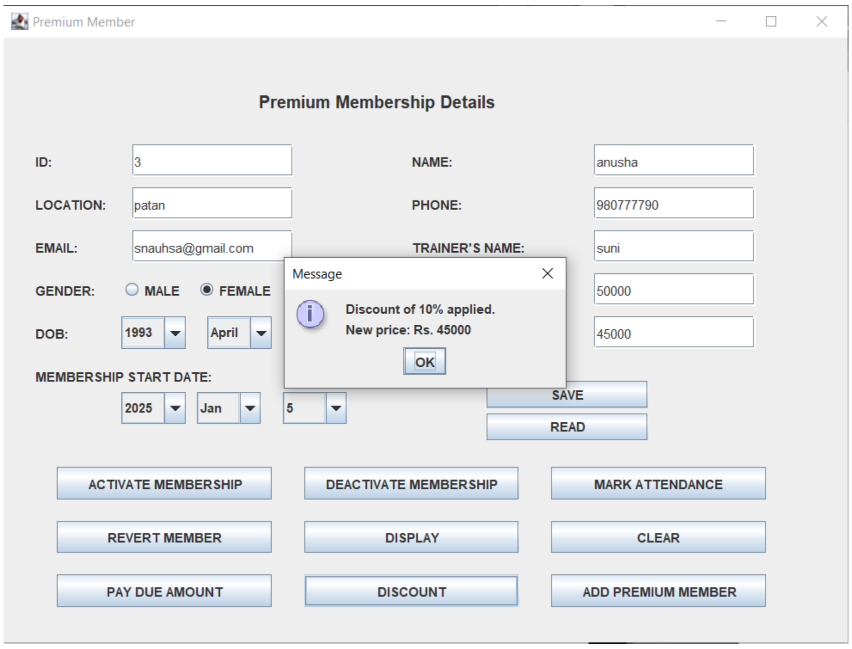


ID:	3	NAME:	anusha
LOCATION:	patan	PHONE:	980777790
EMAIL:	snauhsa@gmail.com	TRAINER'S NAME:	suni
GENDER:	MALE [✓ FEMALE]	TOTAL AMOUNT:	50000
DOB:	1993 April 7	PAID AMOUNT:	
MEMBERSHIP START	2025		

Message

Enter your paid amount.

Figure 17 paid amount need to be entered for discount

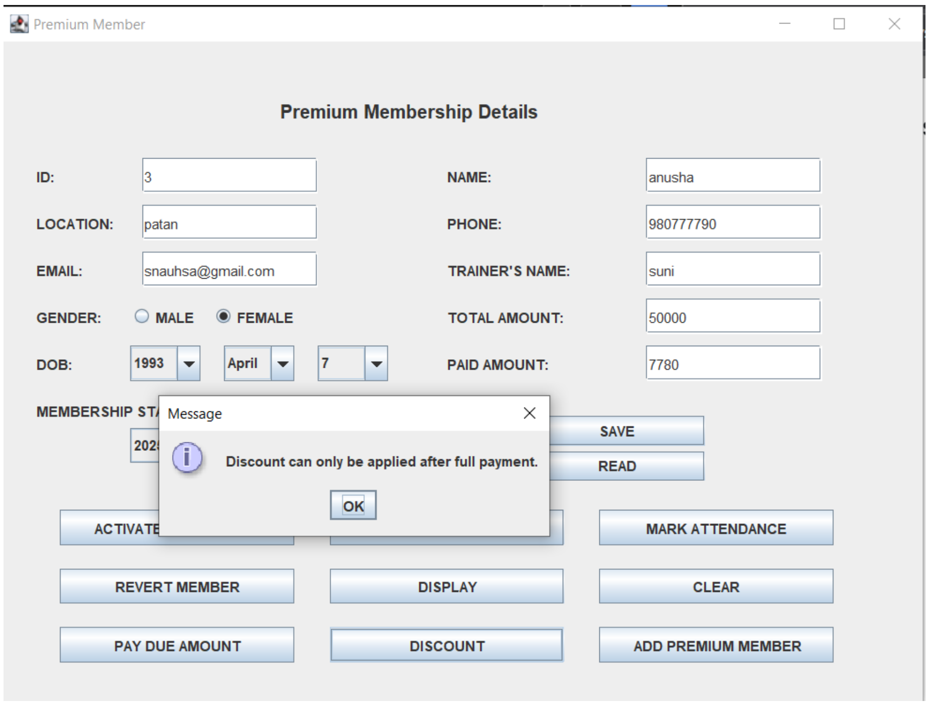

Premium Member

Premium Membership Details

ID: 3
NAME: anusha
LOCATION: patan
PHONE: 980777790
EMAIL: snauhsa@gmail.com
TRAINER'S NAME: suni
GENDER: MALE [✓ FEMALE]
50000
DOB: 1993 April
45000
MEMBERSHIP START DATE: 2025 Jan 5

Message
Discount of 10% applied.
New price: Rs. 45000
OK

Figure 18 discount applied when full amount is paid

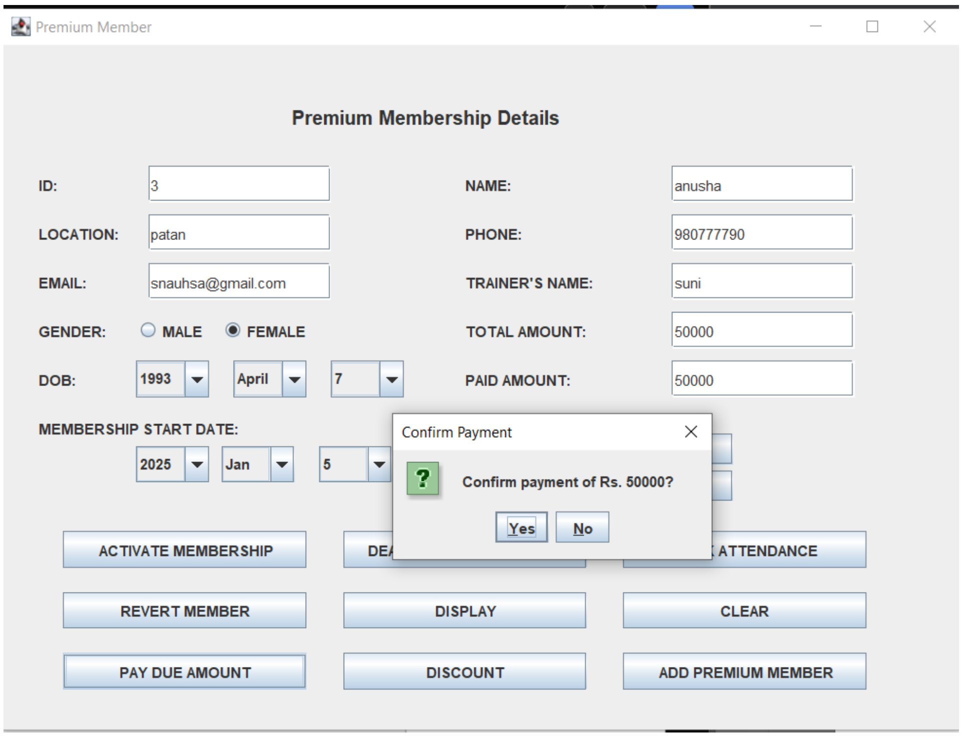

Premium Member

Premium Membership Details

ID: 3
NAME: anusha
LOCATION: patan
PHONE: 980777790
EMAIL: snauhsa@gmail.com
TRAINER'S NAME: suni
GENDER: MALE [✓ FEMALE]
TOTAL AMOUNT: 50000
DOB: 1993 April 7
PAID AMOUNT: 7780
MEMBERSHIP ST
202

Message
Discount can only be applied after full payment.
OK

Figure 19 discount not applied for partial amount


Premium Member

Premium Membership Details

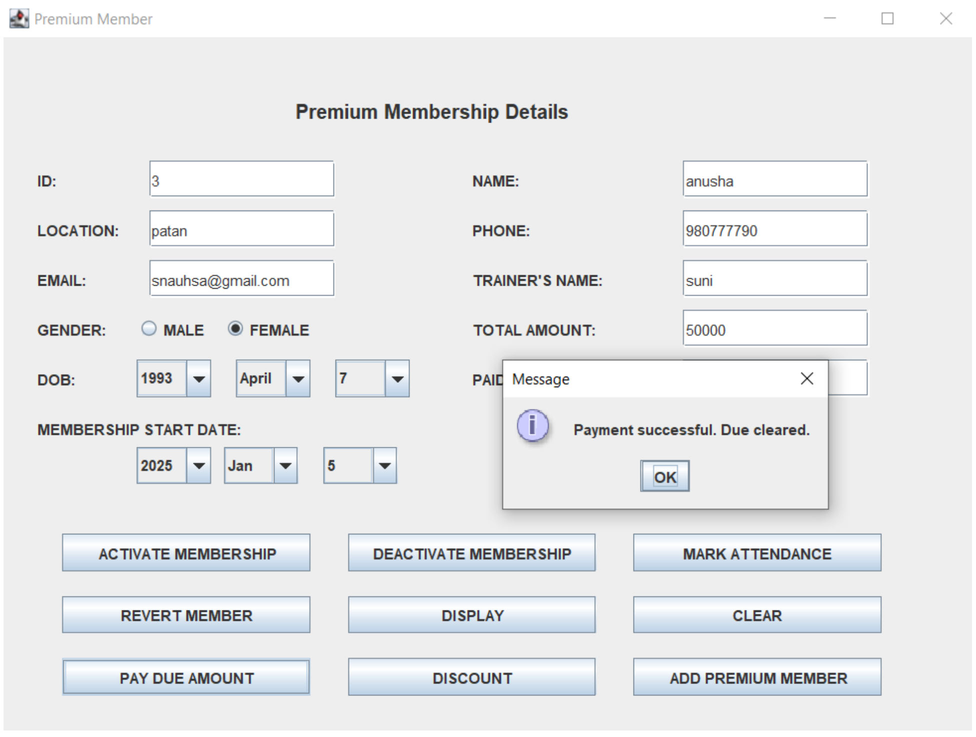
ID: 3
NAME: anusha
LOCATION: patan
PHONE: 980777790
EMAIL: snauhsa@gmail.com
TRAINER'S NAME: suni
GENDER: MALE [✓ FEMALE]
TOTAL AMOUNT: 50000
DOB: 1993 April 7
PAID AMOUNT: 50000
MEMBERSHIP START DATE: 2025 Jan 5

Confirm Payment

Confirm payment of Rs. 50000?

Yes No

Figure 20 due amount


Premium Member

Premium Membership Details

ID: 3
NAME: anusha
LOCATION: patan
PHONE: 980777790
EMAIL: snauhsa@gmail.com
TRAINER'S NAME: suni
GENDER: MALE [✓ FEMALE]
TOTAL AMOUNT: 50000
DOB: 1993 April 7
PAID
MEMBERSHIP START DATE: 2025 Jan 5

Message

Payment successful. Due cleared.

OK

Figure 21 due cleared successfully

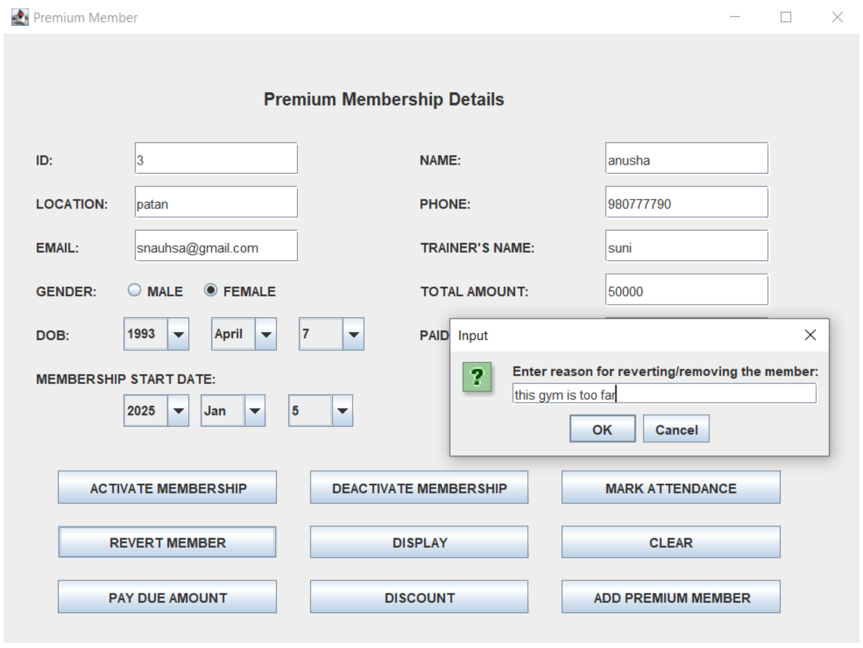


Figure 22 removal reason for premium members

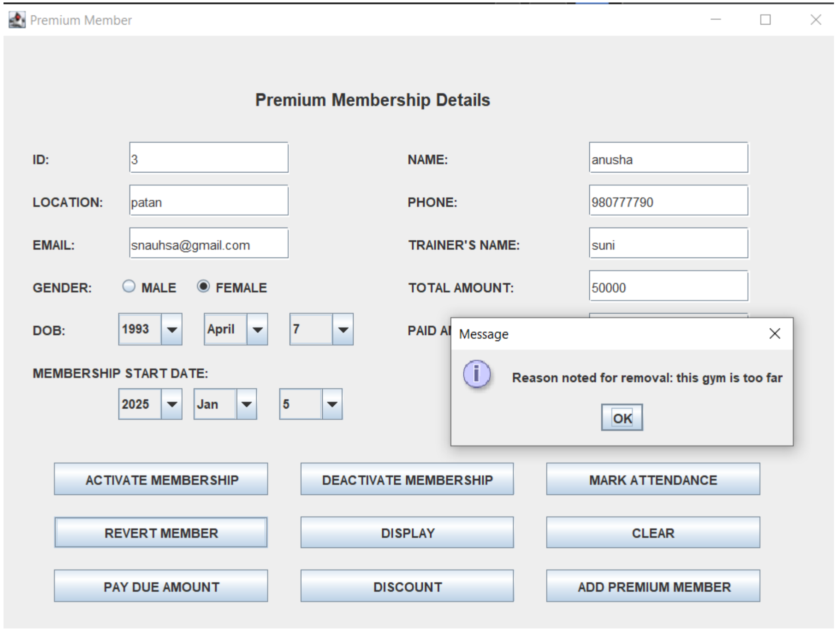
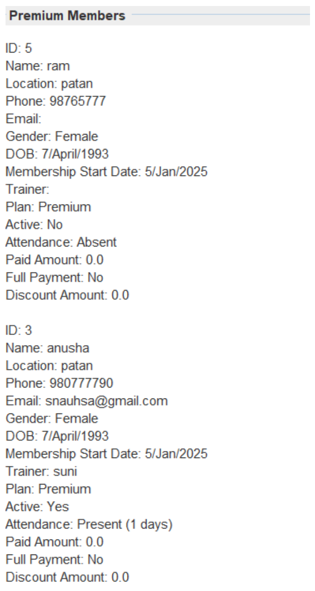


Figure 23 removal reason noted

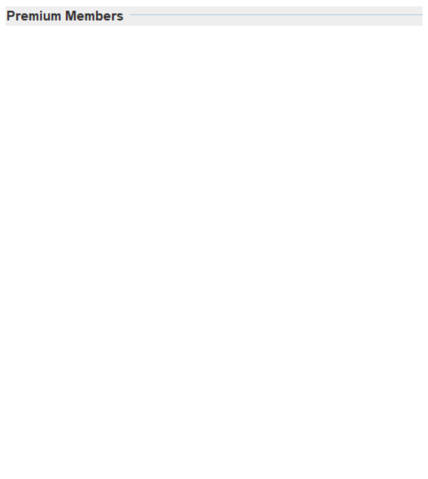


```
Premium Members

ID: 5
Name: ram
Location: patan
Phone: 98765777
Email:
Gender: Female
DOB: 7/April/1993
Membership Start Date: 5/Jan/2025
Trainer:
Plan: Premium
Active: No
Attendance: Absent
Paid Amount: 0.0
Full Payment: No
Discount Amount: 0.0

ID: 3
Name: anusha
Location: patan
Phone: 980777790
Email: snauhsa@gmail.com
Gender: Female
DOB: 7/April/1993
Membership Start Date: 5/Jan/2025
Trainer: suni
Plan: Premium
Active: Yes
Attendance: Present (1 days)
Paid Amount: 0.0
Full Payment: No
Discount Amount: 0.0
```

Figure 24 before removal of premium member

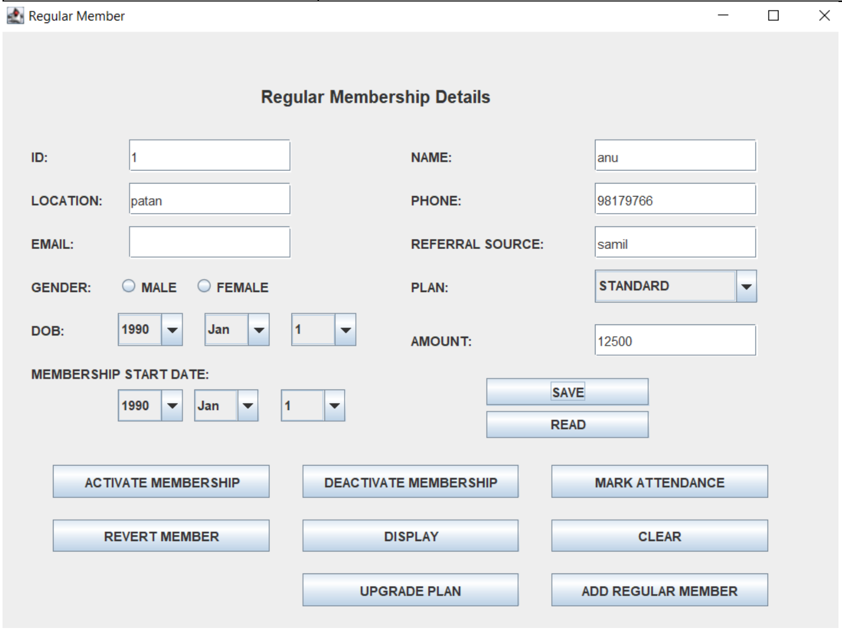


```
Premium Members
```

Figure 25 after reverting premium member

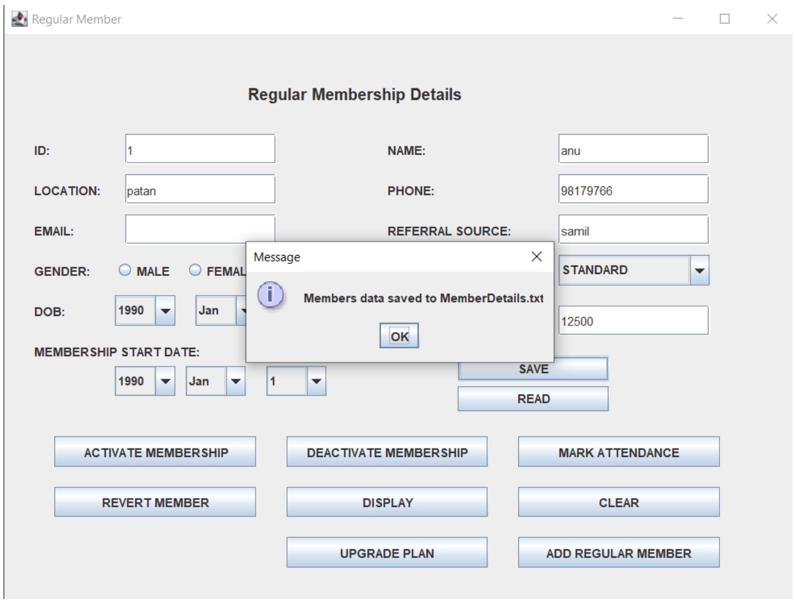
Test 5: Test case for save to and read from file

Objective	Test case for save to and read from file
Action	To fill all the fields and click the save button.
Expected Output	The data should be saved in “MemberDetail.txt” file. And the file should be able to be read.
Actual Output	The data was saved in “MemberDetail.txt” and can the data was able to read.
Result	The test was successful.



ID: 1
NAME: anu
LOCATION: patan
PHONE: 98179766
EMAIL:
REFERRAL SOURCE: samil
GENDER: MALE FEMALE
PLAN: STANDARD
DOB: 1990 Jan 1
AMOUNT: 12500
MEMBERSHIP START DATE: 1990 Jan 1

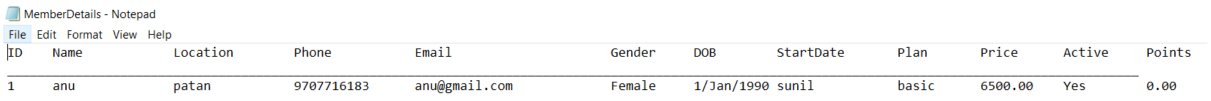
Figure 26 filling details of regular member to save



ID: 1
NAME: anu
LOCATION: patan
PHONE: 98179766
EMAIL:
REFERRAL SOURCE: samil
GENDER: MALE FEMAL
STANDARD
DOB: 1990 Jan
12500
MEMBERSHIP START DATE: 1990 Jan 1

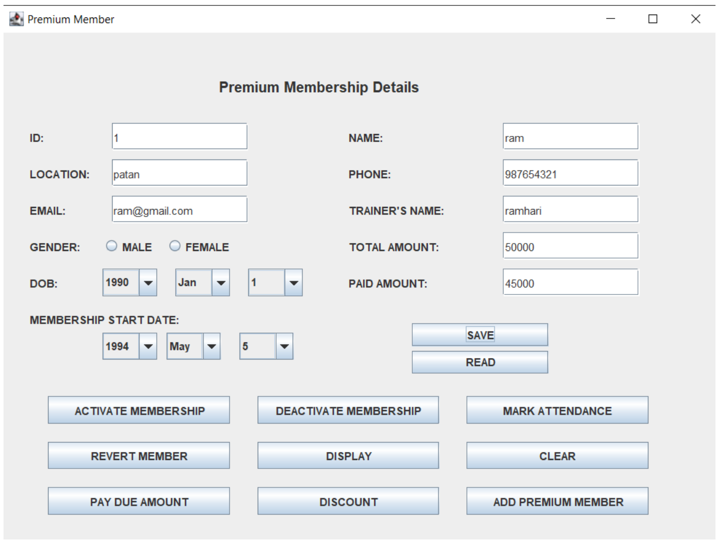
Members data saved to MemberDetails.txt

Figure 27 details sved successfully



```
ID   Name       Location    Phone       Email               Gender  DOB        StartDate  Plan   Price    Active  Points
1    anu        patan       9707716183  anu@gmail.com       Female  1/Jan/1990 sunil      basic  6500.00  Yes     0.00
```

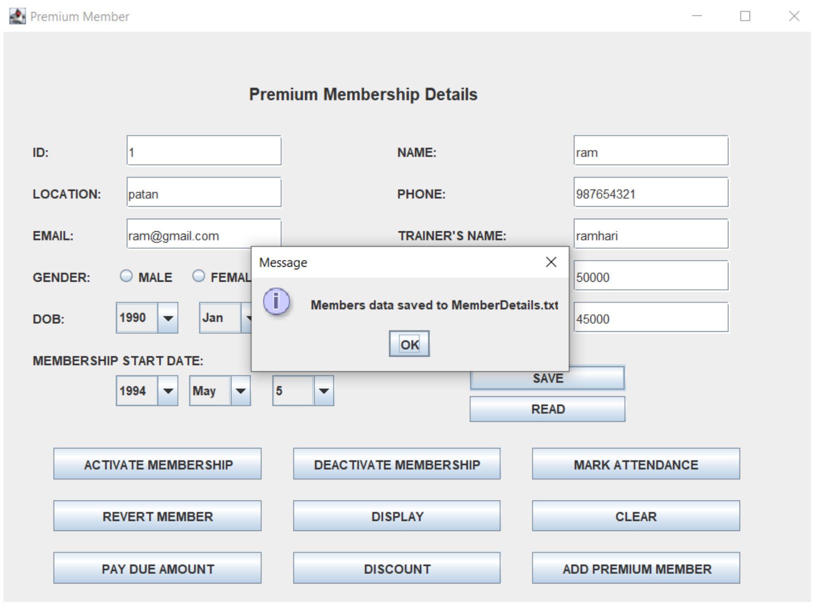
Figure 28 data being saved and read of regular member



Premium Membership Details

ID:	1	NAME:	ram
LOCATION:	patan	PHONE:	987654321
EMAIL:	ram@gmail.com	TRAINER'S NAME:	ramhari
GENDER:	◯ MALE ◯ FEMALE	TOTAL AMOUNT:	50000
DOB:	1990 Jan 1	PAID AMOUNT:	45000
MEMBERSHIP START DATE:	1994 May 5		

Figure 29 filling details of premium member



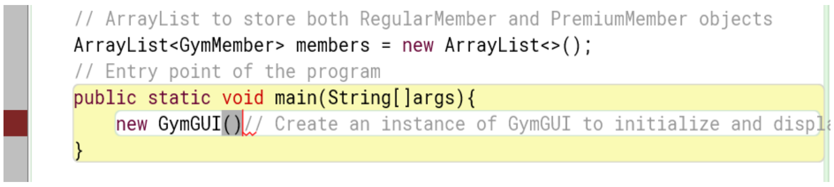
Premium Membership Details

Message: Members data saved to MemberDetails.txt

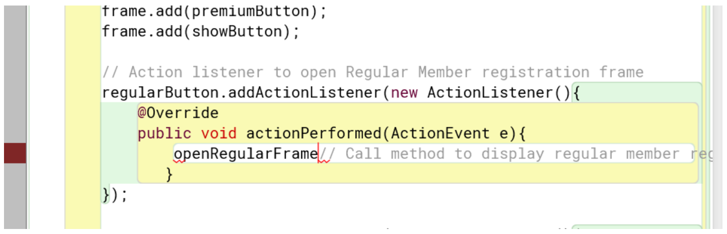
Figure 30 details saved successfully of premium member

Error Detection and Correction

Syntax Error



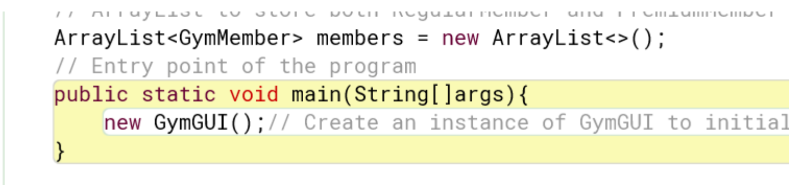
```
// ArrayList to store both RegularMember and PremiumMember objects
ArrayList<GymMember> members = new ArrayList<>();
// Entry point of the program
public static void main(String[]args){
    new GymGUI()// Create an instance of GymGUI to initialize and displ
}
```



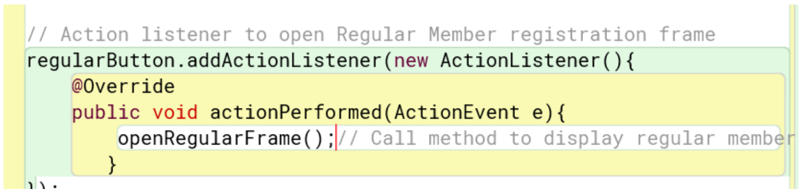
```
frame.add(premiumButton);
frame.add(showButton);

// Action listener to open Regular Member registration frame
regularButton.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e){
        openRegularFrame// Call method to display regular member reg
    }
});
```

During development , a syntax error occurred where a semi colon and brackets were missing. This error was caught by BlueJ compiler , which highlighted the exact lines causing the error.



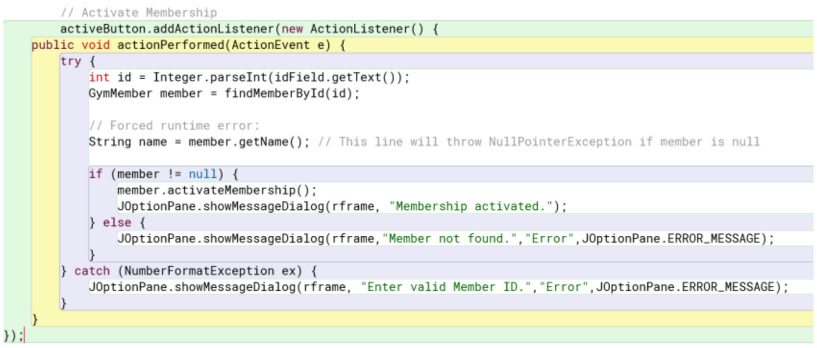
```
ArrayList<GymMember> members = new ArrayList<>();
// Entry point of the program
public static void main(String[]args){
    new GymGUI();// Create an instance of GymGUI to initial
}
```



```
// Action listener to open Regular Member registration frame
regularButton.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent e){
        openRegularFrame();// Call method to display regular member
    }
```

Thr error was found only when compiling . So the code was not compiled and the code coudnt run. Then after adding the brackets and semi colon. The error was corrected then the program ran.

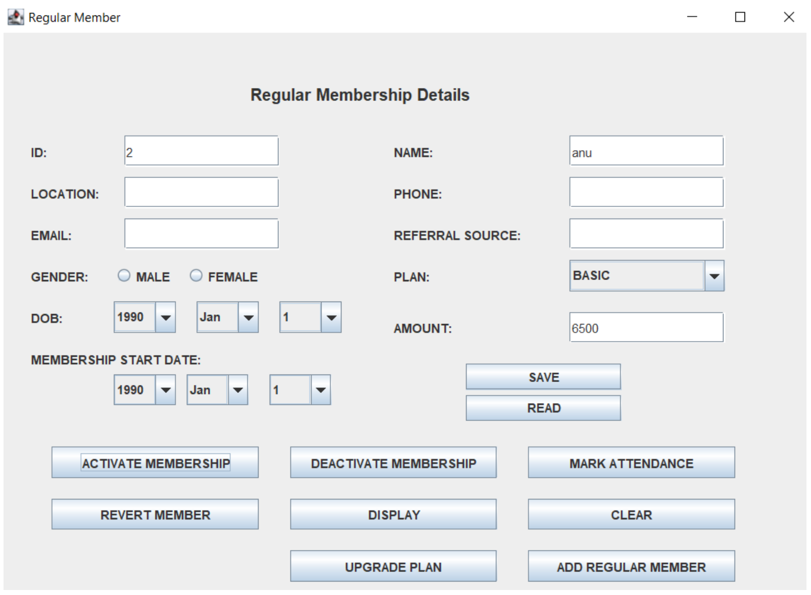
Runtime Error



```
        // Activate Membership
        activeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            int id = Integer.parseInt(idField.getText());
            GymMember member = findMemberById(id);

            // Forced runtime error:
            String name = member.getName(); // This line will throw NullPointerException if member is null

            if (member != null) {
                member.activateMembership();
                JOptionPane.showMessageDialog(rframe, "Membership activated.");
            } else {
                JOptionPane.showMessageDialog(rframe,"Member not found.","Error",JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(rframe, "Enter valid Member ID.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
});
```

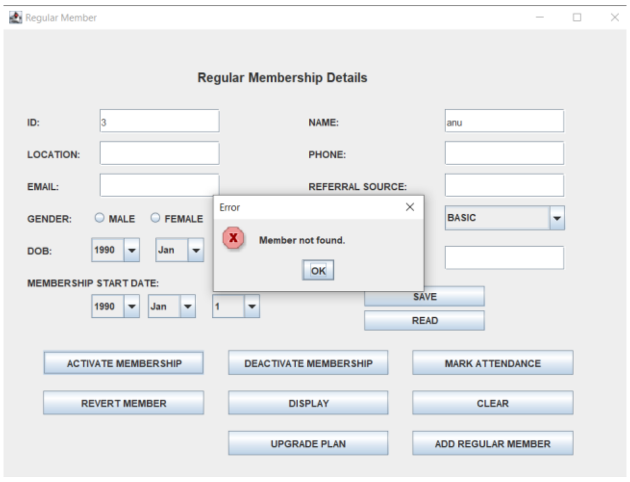


Regular Member

Regular Membership Details

ID: 2
NAME: anu
LOCATION:
PHONE:
EMAIL:
REFERRAL SOURCE:
GENDER: MALE FEMALE
PLAN: BASIC
DOB: 1990 Jan 1
AMOUNT: 6500
MEMBERSHIP START DATE: 1990 Jan 1

While running the program, there was an error where if the member is null, this will cause NullPointerException.due to which any dialog box didn’t appear.



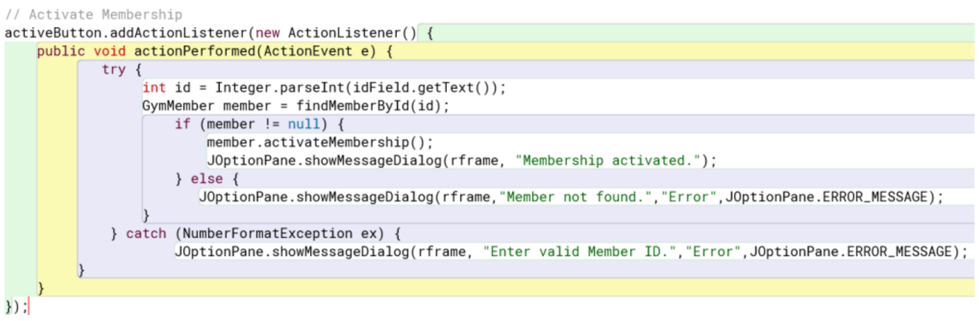
Regular Member

Regular Membership Details

ID: 3
NAME: anu
LOCATION:
PHONE:
EMAIL:
REFERRAL SOURCE:
GENDER: MALE FEMALE
BASIC
DOB: 1990 Jan
MEMBERSHIP START DATE: 1990 Jan 1

Error

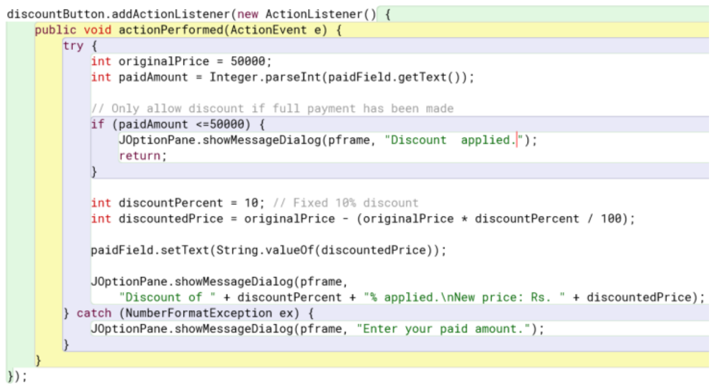
Member not found.



```
// Activate Membership
activeButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            int id = Integer.parseInt(idField.getText());
            GymMember member = findMemberById(id);
            if (member != null) {
                member.activateMembership();
                JOptionPane.showMessageDialog(rframe, "Membership activated.");
            } else {
               JOptionPane.showMessageDialog(rframe,"Member not found.","Error",JOptionPane.ERROR_MESSAGE);
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(rframe, "Enter valid Member ID.","Error",JOptionPane.ERROR_MESSAGE);
        }
    }
});
```

After editing the code, the dialog box appeared and the program ran smoothly.

Logical Error



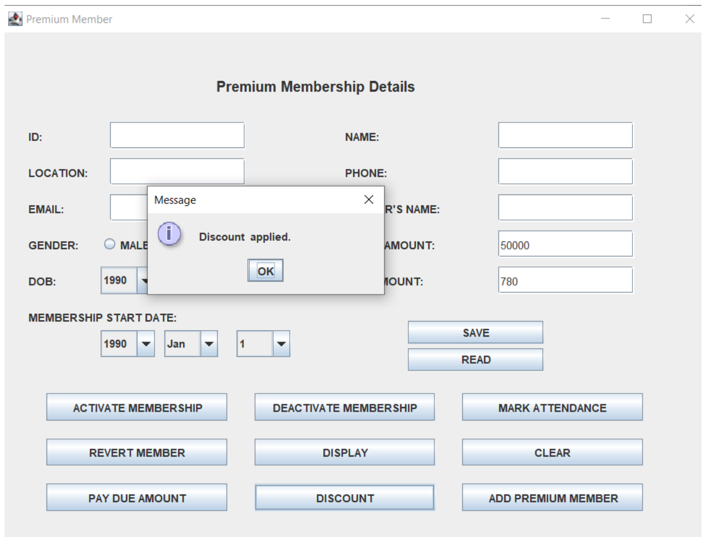
```
discountButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            int originalPrice = 50000;
            int paidAmount = Integer.parseInt(paidField.getText());

            // Only allow discount if full payment has been made
            if (paidAmount <=50000) {
                JOptionPane.showMessageDialog(pframe, "Discount  applied.");
                return;
            }

            int discountPercent = 10; // Fixed 10% discount
            int discountedPrice = originalPrice - (originalPrice * discountPercent / 100);

            paidField.setText(String.valueOf(discountedPrice));

            JOptionPane.showMessageDialog(pframe,
                "Discount of " + discountPercent + "% applied.\nNew price: Rs. " + discountedPrice);
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(pframe, "Enter your paid amount.");
        }
    }
});
```



Premium Member

Premium Membership Details

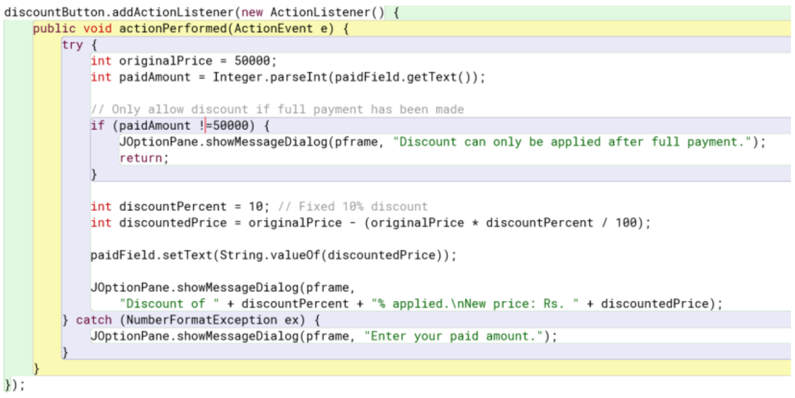
ID:
NAME:
LOCATION:
PHONE:
EMAIL:
R'S NAME:
GENDER: MALE
AMOUNT: 50000
DOB: 1990
OUNT: 780
MEMBERSHIP START DATE: 1990 Jan 1

Message

Discount applied.

OK

During the discount part, the discount was given to person who gave certain amount. Even when the user was not paying full payment , the user was getting discount.



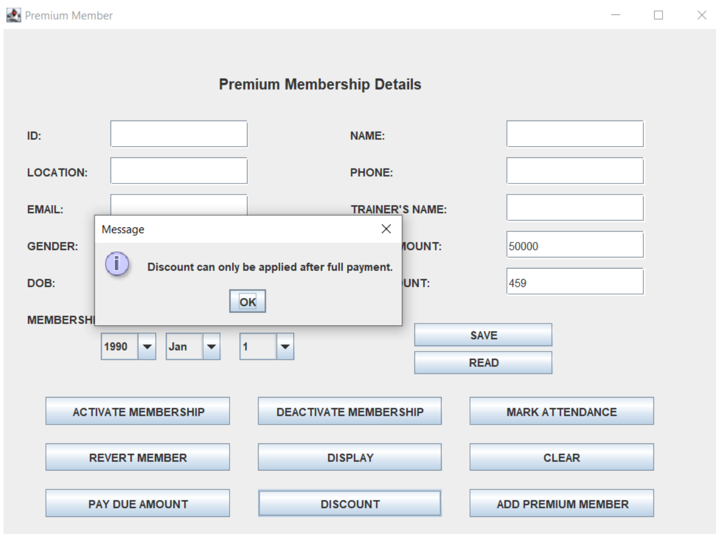
```java
discountButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        try {
            int originalPrice = 50000;
            int paidAmount = Integer.parseInt(paidField.getText());

            // Only allow discount if full payment has been made
            if (paidAmount !=50000) {
                JOptionPane.showMessageDialog(pframe, "Discount can only be applied after full payment.");
                return;
            }

            int discountPercent = 10; // Fixed 10% discount
            int discountedPrice = originalPrice - (originalPrice * discountPercent / 100);

            paidField.setText(String.valueOf(discountedPrice));

            JOptionPane.showMessageDialog(pframe,
                "Discount of " + discountPercent + "% applied.\nNew price: Rs. " + discountedPrice);
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(pframe, "Enter your paid amount.");
        }
    }
});
```



50000

459

Discount can only be applied after full payment.

Since, the error occurred due to a logical error in the discount. After solving the error the program ran successfully. And the discount was given to the user who gave full payment only.

Conclusion

This coursework involved creating a fully functional Java application using OOP principles, file handling, and GUI components. The system successfully allows for adding , updating, activating, deactivating, and tracking gym members. The features like loyalty point management and plan upgrades added real-world relevance. Overall, the implementation met the intended goals and requirements as per the coursework.

Through this coursework, I gained a deeper understanding of Java programming, especially OOP concepts like inheritance, abstraction and polymorphism. I also learned how to work with Swing to build user interfaces and use file I/O data persistence. Additionally , I improved my problem -solving and debugging skills, especially skills, especially when dealing with runtime and logical issues.

One of the biggest challenges was organizing the logic in such a way that different buttons handled member data independently without interfering with each other . Managing the dynamic data using ArrayList and reflecting changes on the GUI also posed a challenge . Additionally, understanding how to structure event driven programming using ActionListener took some time to grasp.

I overcame these issues by breaking down the problem into smaller parts and solving them one at a time. I referred to official Java documentation and online forums. I also frequently tested each functionality after implementation to make sure it worked as expected.

Appendix

RegularMember.java

```
public class RegularMember extends GymMember
{
    private final int attendanceLimit;
    private boolean isEligibleforUpgrade;
    private String removalReason;
    private String referralSource;
    private String plan;
    private double price;
    private boolean attendanceMarked = false;

    //constructor to initialize a RegularMember
    public RegularMember(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.attendanceLimit = 30;
       this.isEligibleforUpgrade = false; //initial isEligibleforUpgrade set to false
       this.removalReason = removalReason;
       this.plan = "basic"; //inital plan set to basic
       this.price = 6500; //inital price set to 6500
    }

    //getter methods for accessing private attributes
    public String getReferralSource(){
       return referralSource;
    }
    public boolean getisEligibleforUpgrade(){
```

```
    return isEligibleforUpgrade;
}
public String getRemovalReason(){
    return removalReason;
}
public String getPlan(){
    return plan;
}
public double getPrice(){
    return price;
}
public String getAttendanceStatus() {
    return attendanceMarked ? "Present" : "Absent";
}

public void markAttendance() {
    attendanceMarked = true;
}

public void LoyalityPoints()
{
    attendance++;
    loyaltyPoints += 5;
    if(attendance >= attendanceLimit){
        isEligibleforUpgrade = true;
    }
}

//method to get Plan Price using switch case
```

```
public double getPlanPrice(String plan){
    switch (plan.toLowerCase()){
        case "basic":
            return 6500;
        case "standard":
            return 12500;
        case "deluxe":
            return 18500;
            default:
        return -1;
    }
}

//method to upgrade Plan
public String upgradePlan(String newPlan)
{
    double newPrice =getPlanPrice(newPlan);
    if(newPrice == -1){
        return "invalid plan";
    }
    if(isEligibleforUpgrade == false){
        return "You cannot have an upgrade.";
    }
    if(this.plan.equalsIgnoreCase(newPlan)){
        return "You have been subscribed to the plan.";
    }
    this.plan = newPlan;
    this.price = newPrice;
    return "Plan has been upgraded."+newPlan;
```

```
        }

        //method to revert Regularmember
        public void revertRegularmember(String removalReason){
            resetMember();
            this.isEligibleforUpgrade = false;
            this.plan = "Basic";
            this.price = 6500;
            this.removalReason = removalReason;
        }

        /**
         * Overide the display() method to include additional details
         */
        @Override
        public String display() {
        return "ID: " + getId() +
               "\nName: " + getName() +
               "\nLocation: " + location +
               "\nPhone: " + phone +
               "\nEmail: " + email +
               "\nGender: " + gender +
               "\nDOB: " + DOB +
               "\nReferral: " + referralSource +
               "\nMembership: Regular" +
               "\nActive: " + (getActiveStatus() ? "Yes" : "No") +
               "\nAttendance: " + getAttendanceStatus();
    }
}
```

PremumMember.java

```
public class PremiumMember extends GymMember
{
   private final double premiumCharge = 50000;//premiumCharge set to 50000
   private String personalTrainer;
   private boolean isFullPayment;
   private double paidAmount;
   private double discountAmount;

   //constructor to initialize PremiumMember
   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.personalTrainer = personalTrainer;
      this.isFullPayment = false;
      this.paidAmount = 0;
      this.discountAmount = 0;
   }

   //getter method
   public double getPremiumCharge(){
      return premiumCharge;
   }
   public String getPersonalTrainer(){
      return personalTrainer;
   }
   public boolean getIsFullPayment(){
    return isFullPayment;
}
public double getPaidAmount(){
    return paidAmount;
}
public double getDiscountAmount(){
    return discountAmount;
}

/**
 * Override markAttendance to increase attendance and loyalty points
 */
@Override
public void markAttendance(){
    this.attendance++;
    this.loyaltyPoints +=10;
}

//method to payDueAmount usinf if loop
public String payDueAmount(double paidAmount){
    if(isFullPayment){
        return "Payment done.";
    }

    double totalPaid = this.paidAmount + paidAmount;
    if(totalPaid > premiumCharge){
        return "Error : Total paid amount exceeds the required charge.";
    }
    this.paidAmount  = totalPaid;
    if(this.paidAmount == premiumCharge){
        this.isFullPayment = true;
    }

    double remainingAmount = premiumCharge - this.paidAmount;
    return "Payment done.Remaining payment:"+remainingAmount;
}

//method to calculate discount if full payment is done
public void calculatediscount(){
    if(isFullPayment){
        this.discountAmount = 0.1*premiumCharge;
        System.out.println("Discount:" + discountAmount);
    }
    else{
        System.out.println("No discount.Full Payment is to done.");
    }
}

//method to reset the PremiumMember details
public void revertPremiumMember(){
    super.resetMember();
    this.personalTrainer = "";
    this.isFullPayment = false;
    this.paidAmount = 0;
    this.discountAmount = 0;
}
```

```
    /**
     * Overide display() method to show the details
     */
    @Override
    public String display() {
    return "ID: " + getId() +
           "\nName: " + getName() +
           "\nLocation: " + getLocation() +
           "\nPhone: " + getPhone() +
           "\nEmail: " + getEmail() +
           "\nGender: " + getGender() +
           "\nDOB: " + getDOB() +
           "\nMembership Start Date: " + getMembershipStartDate() +
           "\nTrainer: " + getPersonalTrainer() +
           "\nPlan: Premium" +
           "\nActive: " + (getActiveStatus() ? "Yes" : "No") +
           "\nAttendance: " + getAttendanceStatus() +
           "\nPaid Amount: " + paidAmount +
           "\nFull Payment: " + (isFullPayment ? "Yes" : "No") +
           "\nDiscount Amount: " + discountAmount;
    }
}
```

GymMember.java

```
//abstract class Gym member
public abstract class GymMember
{
    //protected attributes used by child class
    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 = 0; //initial loyality point set to 0
    protected boolean activeStatus = false;//membership  inactice by default

    //constructor to initialize GymMember details
    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;
    }

   public void markAttendance() {
        if (activeStatus) {
            attendance++;
            loyaltyPoints += 10;
            System.out.println("Attendance marked.");
        } else {
            System.out.println("Cannot mark attendance. Membership not active.");
        }
    }

    //method to activate membership
    public void activateMembership(){
        activeStatus = true;
    }

    //method to deactivate membership
    public void deactivateMembership(){
        if(activeStatus){
            activeStatus = false;
        }
        else{
            System.out.println("Membership is inactive");
        }
    }

    //method to reset member details
    public void resetMember(){
        activeStatus = false;
        attendance = 0;
        loyaltyPoints = 0;
    }

    //getter methods for accessing private attributes
    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 getGender(){
        return gender;
    }
    public String getDOB(){
        return DOB;
    }
    public String getMembershipStartDate(){
        return membershipStartDate;
    }
    public int getAttendance(){
        return attendance;
    }
    public double getLoyaltyPoints(){
        return loyaltyPoints;
    }
```

```
    public boolean getActiveStatus(){
        return activeStatus;
    }
    public String getAttendanceStatus() {
        return attendance > 0 ? "Present (" + attendance + " days)" : "Absent";
    }
    //method to display member details
    public String display() {
        return "ID: " + getId() +
                "\nName: " + getName() +
                "\nLocation: " + getLocation() +
                "\nPhone: " + getPhone() +
                "\nEmail: " + getEmail() +
                "\nGender: " + getGender() +
                "\nDOB: " + getDOB() +
                "\nAttendance: " + attendance +
                "\nLoyalty Points: " + loyaltyPoints +
                "\nActive: " + activeStatus+
                "\nMembership Start Date: "  + getMembershipStartDate();
    }
}
```

GymGUI.java

```
//importing ArrayList package for storing member objects
import java.util.ArrayList;
import java.awt.*;
import javax.swing.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.File;

public class GymGUI{
   // Declare all input fields and components used in the GUI
   private JTextField idField, nameField, locationField, phoneField;
   private JTextField emailField, referralField,  trainerField, paidField;
   private JRadioButton maleButton, femaleButton;
   private JComboBox dobyearCombo, dobmonthCombo, dobdayCombo;
   private JComboBox msyearCombo, msmonthCombo, msdayCombo;
   private JComboBox plansCombo;
   // ArrayList to store both RegularMember and PremiumMember objects
   ArrayList<GymMember> members = new ArrayList<>();
   // Entry point of the program
   public static void main(String[]args){
       new GymGUI();// Create an instance of GymGUI to initialize and display the
GUI
   }

   // Constructor to build the main frame of the Gym Membership GUI
   public  GymGUI(){
       // Create and configure the main application window
       JFrame frame = new JFrame("GUI");
       frame.setSize(800,600);
```

```
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLayout(null);// Use absolute positioning for layout

        // Title label for the window
        JLabel titleLabel = new JLabel("Welcome to the Gym Membership ",
JLabel.CENTER);
        titleLabel.setBounds(100,10,500,100);
        titleLabel.setFont(new Font("Arial", Font.BOLD,16));
        frame.add(titleLabel);

        // Buttons for navigating to different membership forms and viewing members
        JButton regularButton = new JButton("Regular Member");
        regularButton.setBounds(125,100,500,100);
        JButton premiumButton = new JButton("Premium Member");
        premiumButton.setBounds(125,250,500,100);
        JButton showButton = new JButton("View Members");
        showButton.setBounds(125,400,500,100);

        // Add all buttons to the main frame
        frame.add(regularButton);
        frame.add(premiumButton);
        frame.add(showButton);

        // Action listener to open Regular Member registration frame
        regularButton.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                openRegularFrame();// Call method to display regular member registration
form
            }
```

```
    });

    premiumButton.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent e){
            openPremiumFrame();
        }
    });

    showButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
            showMembers();
        }
    });



    // Display the main frame
    frame.setVisible(true);
    }

private void showMembers() {
// Create a new JFrame to display both Regular and Premium Members
JFrame frame = new JFrame("All Members");
frame.setSize(800, 600);
frame.setLayout(new GridLayout(1, 2));// Two columns for regular and premium

 // Text areas to display member details
JTextArea regularTextArea = new JTextArea();
JTextArea premiumTextArea = new JTextArea();
```

```
    regularTextArea.setEditable(false);
    premiumTextArea.setEditable(false);

     // Add scroll functionality to the text areas
    JScrollPane scrollRegular = new JScrollPane(regularTextArea);
    JScrollPane scrollPremium = new JScrollPane(premiumTextArea);

    // Set titled borders for better visual separation
    scrollRegular.setBorder(BorderFactory.createTitledBorder("Regular Members"));
    scrollPremium.setBorder(BorderFactory.createTitledBorder("Premium Members"));

     // Loop through the members list and categorize based on their type
    for (GymMember member : members) {
        if (member instanceof RegularMember) {
            regularTextArea.append(member.display() + "\n\n");
        } else if (member instanceof PremiumMember) {
            premiumTextArea.append(member.display() + "\n\n");
        }
    }

    // Add both scroll panes to the main frame and make it visible
    frame.add(scrollRegular);
    frame.add(scrollPremium);
    frame.setVisible(true);
}

        public void openRegularFrame(){
            // Create a new frame for Regular Member registration
        JFrame rframe = new JFrame("Regular Member");
        rframe.setSize(800,600);
        rframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        rframe.setLayout(null);// Using absolute layout

        // Title label
        JLabel titleLabel = new JLabel("Regular Membership Details ",
JLabel.CENTER);
        titleLabel.setBounds(100,10,500,100);
        titleLabel.setFont(new Font("Arial", Font.BOLD,16));
        rframe.add(titleLabel);

        // Input fields and labels for personal details
        JLabel idLabel = new JLabel("ID:");
        idLabel.setBounds(30,100,100,30);
        JTextField idField = new JTextField(20);
        idField.setBounds(120,100,150,30);

        JLabel nameLabel = new JLabel("NAME:");
        nameLabel.setBounds(380,100,100,30);
        JTextField nameField = new JTextField(20);
        nameField.setBounds(550,100,150,30);

        JLabel locationLabel = new JLabel("LOCATION:");
        locationLabel.setBounds(30,140,100,30);
        JTextField locationField = new JTextField(20);
        locationField.setBounds(120,140,150,30);

        JLabel phoneLabel = new JLabel("PHONE:");
```

```
phoneLabel.setBounds(380,140,100,30);
JTextField phoneField = new JTextField(20);
phoneField.setBounds(550,140,150,30);

JLabel emailLabel = new JLabel("EMAIL:");
emailLabel.setBounds(30,180,100,30);
JTextField emailField = new JTextField(20);
emailField.setBounds(120,180,150,30);

JLabel referralLabel = new JLabel("REFERRAL SOURCE:");
referralLabel.setBounds(380,180,150,30);
JTextField referralField = new JTextField(20);
referralField.setBounds(550,180,150,30);

JLabel genderLabel = new JLabel("GENDER:");
genderLabel.setBounds(30,220,100,30);
JRadioButton maleButton= new JRadioButton("MALE");
maleButton.setBounds(110,220,60,30);
JRadioButton femaleButton= new JRadioButton("FEMALE");
femaleButton.setBounds(180,220,160,30);

ButtonGroup genderGroup = new ButtonGroup();
genderGroup.add(maleButton);
genderGroup.add(femaleButton);

 // Plan selection ComboBox
JLabel planLabel = new JLabel("PLAN:");
planLabel.setBounds(380,220,150,30);
String [] plan = {"BASIC","STANDARD","DELUXE"};
```

```
            JComboBox plansCombo = new JComboBox(plan);
            plansCombo.setBounds(550,220,150,30);
            plansCombo.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String selectedPlan = (String) plansCombo.getSelectedItem();
                String price = switch (selectedPlan) {
                    case "BASIC" -> "29.99";
                    case "STANDARD" -> "49.99";
                    case "DELUXE" -> "69.99";
                    default -> "0.00";
        };
        paidField.setText(price);
        }
    });


            // ComboBoxes for Date of Birth
            JLabel dobLabel = new JLabel("DOB:");
            dobLabel.setBounds(30,260,100,30);
            String [] years =
{"1990","1991","1992","1993","1994","1995","1996","1996","1997","1998","1999","20
00","2001","2018","2019","2020","2021","2022","2023","2024","2025"};
            String [] months =
{"Jan","Feb","Mar","April","May","June","July","Aug","Sept","Oct","Nov","Dec"};
            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","30","31","32"};


            JComboBox<String> dobyearCombo = new JComboBox(years);
            dobyearCombo.setBounds(110,260,60,30);
            JComboBox dobmonthCombo = new JComboBox(months);
            dobmonthCombo.setBounds(190,260,60,30);
            JComboBox dobdayCombo = new JComboBox(days);
            dobdayCombo.setBounds(270,260,60,30);

            JLabel paidLabel = new JLabel("AMOUNT:");
            paidLabel.setBounds(380,270,150,30);
            JTextField paidField = new JTextField(20);
            paidField.setBounds(550,270,150,30);

            // ComboBoxes for Membership Start Date
            JLabel memberLabel = new JLabel("MEMBERSHIP START DATE:");
            memberLabel.setBounds(30,300,200,30);
            String [] msdyears =
{"1990","1991","1992","1993","1994","1995","1996","1996","1997","1998","1999","20
00","2001","2018","2019","2020","2021","2022","2023","2024","2025"};
            String [] msdmonths =
{"Jan","Feb","Mar","April","May","June","July","Aug","Sept","Oct","Nov","Dec"};
            String [] msddays =
{"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","32"};

            JComboBox<String> msyearCombo = new JComboBox(msdyears);
            msyearCombo.setBounds(110,330,60,30);
            JComboBox msmonthCombo = new JComboBox(msdmonths);
            msmonthCombo.setBounds(180,330,60,30);
            JComboBox msdayCombo = new JComboBox(msddays);
            msdayCombo.setBounds(260,330,60,30);

            //save button
            JButton saveButton = new JButton("SAVE");
            saveButton.setBounds(450,320,150,25);
            saveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try (BufferedWriter writer = new BufferedWriter(new
FileWriter("MemberDetails.txt",true))) {
                writer.write(String.format("%-5s %-15s %-15s %-15s %-25s %-10s %-
10s %-15s %-10s %-10s %-10s %-10s\n",
                    "ID", "Name", "Location", "Phone", "Email", "Gender", "DOB",
"StartDate", "Plan", "Price", "Active", "Points"));
                writer.write("_".repeat(150));
                writer.newLine();
                for (GymMember member : members) {
                    if (member instanceof RegularMember) {
                        RegularMember r = (RegularMember) member;
                        writer.write(String.format("%-5d %-15s %-15s %-15s %-25s %-10s
%-10s %-15s %-10s %-10.2f %-10s %-10.2f\n",
                            r.getId(), r.getName(), r.getLocation(), r.getPhone(), r.getEmail(),
r.getGender(),r.getDOB(), r.getMembershipStartDate(), r.getPlan(), r.getPrice(),
                            r.getActiveStatus() ? "Yes" : "No", r.getLoyaltyPoints()));
                    }
                }
                writer.write("_".repeat(150));
                writer.newLine();

                JOptionPane.showMessageDialog(rframe, "Members data saved to
MemberDetails.txt");
                } catch (IOException ex) {
                JOptionPane.showMessageDialog(rframe, "Error saving to file: " +
ex.getMessage());
                }
            }
        });

            //read button
            JButton readButton = new JButton("READ");
            readButton.setBounds(450,350,150,25);
            readButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                try (BufferedReader reader = new BufferedReader(new
FileReader("MemberDetails.txt"))) {
                    String line;
                    while ((line = reader.readLine()) != null) {
                        System.out.println(line);
                    }
                } catch (IOException ex) {
                    System.out.println("Error reading file: " + ex.getMessage());
                }
                }
            });

            // Buttons for membership actions
            JButton activeButton = new JButton("ACTIVATE MEMBERSHIP");
            activeButton.setBounds(50,400,200,30);

            JButton deactiveButton = new JButton("DEACTIVATE MEMBERSHIP");
            deactiveButton.setBounds(280,400,200,30);

            JButton markButton = new JButton("MARK ATTENDANCE");
        markButton.setBounds(510,400,200,30);


        JButton revertButton = new JButton("REVERT MEMBER");
        revertButton.setBounds(50,450,200,30);


        revertButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            try {
                if (idField.getText().equals("")) {
                    JOptionPane.showMessageDialog(null, "Enter Member ID to revert.");
                    return;
                }


                int id = Integer.parseInt(idField.getText());
                GymMember member = findMemberById(id);


                if (member == null) {
                    JOptionPane.showMessageDialog(null, "Member not found.");
                    return;
                }


                // Ask for removal reason
                String reason = JOptionPane.showInputDialog(null, "Enter reason for
reverting/removing the member:");


                if (reason == null || reason.equals("")) {
                    JOptionPane.showMessageDialog(null, "Removal reason is
required.");
                    return;
                }
                // Log or show the reason
                JOptionPane.showMessageDialog(null, "Reason noted for removal: " +
reason);

                // You can optionally log this to the console or a file
                System.out.println("Removal reason for Member ID " + id + ": " +
reason);

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



        JButton displayButton = new JButton("DISPLAY");
        displayButton.setBounds(280,450,200,30);

    displayButton.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e){
                showMembers();
            }
    });
        JButton clearButton = new JButton("CLEAR");
        clearButton.setBounds(510,450,200,30);

        JButton upgradeButton = new JButton("UPGRADE PLAN");
        upgradeButton.setBounds(280,500,200,30);
    upgradeButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
                try {
                if (idField.getText().isEmpty()) {
                JOptionPane.showMessageDialog(null, "Enter Member ID");
                return;
                }

                int id = Integer.parseInt(idField.getText());
                GymMember member = findMemberById(id);

                if (member == null) {
                JOptionPane.showMessageDialog(null, "Member not found.");
                return;
                }

                if (member instanceof PremiumMember) {
                JOptionPane.showMessageDialog(null, "This member is already a
Premium member.");
                return;
                }

                // Prompt for Trainer Name (Premium-specific detail)
                String trainerName = JOptionPane.showInputDialog(null, "Enter Trainer
Name for Premium Membership:");

                if (trainerName == null || trainerName.trim().isEmpty()) {
                JOptionPane.showMessageDialog(null, "Trainer name is required.");
                return;
```

```
                }

                // Remove regular member
                members.remove(member);

                // Create PremiumMember using existing details from input fields
                String name = nameField.getText();
                String location = locationField.getText();
                String phone = phoneField.getText();
                String email = emailField.getText();
                String gender = maleButton.isSelected() ? "Male" :
(femaleButton.isSelected() ? "Female" : "Others");
                String dob = dobdayCombo.getSelectedItem() + "/" +
dobmonthCombo.getSelectedItem() + "/" + dobyearCombo.getSelectedItem();

                // Create and add PremiumMember
                PremiumMember upgradedMember = new PremiumMember(id, name,
location, phone, email, gender, dob, "Today", trainerName);
                members.add(upgradedMember);

                JOptionPane.showMessageDialog(null, "Regular member upgraded to
Premium!");

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

```
JButton addrButton = new JButton("ADD REGULAR MEMBER");
addrButton.setBounds(510,500,200,30);

// Add all components to the frame
rframe.add(idLabel);
rframe.add(idField);
rframe.add(nameLabel);
rframe.add(nameField);
rframe.add(locationLabel);
rframe.add(locationField);
rframe.add(phoneLabel);
rframe.add(phoneField);
rframe.add(emailLabel);
rframe.add(emailField);
rframe.add(referralLabel);
rframe.add(referralField);
rframe.add(genderLabel);
rframe.add(maleButton);
rframe.add(femaleButton);
rframe.add(dobLabel);
rframe.add(dobyearCombo);
rframe.add(dobmonthCombo);
rframe.add(dobdayCombo);
rframe.add(paidLabel);
rframe.add(paidField);
rframe.add(memberLabel);
rframe.add(msyearCombo);
rframe.add(msmonthCombo);
rframe.add(msdayCombo);
rframe.add(planLabel);
rframe.add(plansCombo);
rframe.add(saveButton);
rframe.add(readButton);
rframe.add(activeButton);
rframe.add(deactiveButton);
rframe.add(markButton);
rframe.add(revertButton);
rframe.add(displayButton);
rframe.add(clearButton);
rframe.add(upgradeButton);
rframe.add(addrButton);

// Action for adding a regular member
addrButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e){
try {
    // Validate ID field
    if (idField.getText().isEmpty()) {
        JOptionPane.showMessageDialog(rframe, "ID must be filled");
        return;
    }
    int id = Integer.parseInt(idField.getText());
    GymMember member = findMemberById(id);
    if(member != null){
        JOptionPane.showMessageDialog(rframe, "Member already exists.");
        return;
    }
     // Validate name field
```

```
            if (nameField.getText().isEmpty()) {
                JOptionPane.showMessageDialog(rframe, "Name must be filled");
                return;
            }
            if(phoneField.getText().isEmpty()){
                    JOptionPane.showMessageDialog(rframe,"Phone number msut be
filled.");
            }
            if (locationField.getText().isEmpty() || emailField.getText().isEmpty() ||
referralField.getText().isEmpty()) {
                JOptionPane.showMessageDialog(rframe, "All fields must be filled");
                return;
            }



            // Collect other inputs
            String name = nameField.getText();
            String location = locationField.getText();
            String phone = phoneField.getText();
            String email = emailField.getText();
            String gender = maleButton.isSelected() ? "Male" :
femaleButton.isSelected() ? "Female" : "Others";
            String dob = dobdayCombo.getSelectedItem() + "/" +
dobmonthCombo.getSelectedItem() + "/" + dobyearCombo.getSelectedItem();
            String referral = referralField.getText();
            String planSelected = (String) plansCombo.getSelectedItem();


             // Create new member and add to list
            RegularMember newMember = new RegularMember(id, name, location,
phone, email, gender, dob, referral);
            members.add(newMember);
```

```
                // Confirmation message
                JOptionPane.showMessageDialog(rframe, "Regular Member Added",
"Success", JOptionPane.INFORMATION_MESSAGE);

            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(rframe, "ID must be a number");
                JOptionPane.showMessageDialog(rframe, "Phone Number must be a
number");
            }
        }
        });


        // Activate Membership
        activeButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                    try {
                        int id = Integer.parseInt(idField.getText());
                        GymMember member = findMemberById(id);
                            if (member != null) {
                                member.activateMembership();
                                JOptionPane.showMessageDialog(rframe, "Membership
activated.");
                            } else {
                               JOptionPane.showMessageDialog(rframe,"Member not
found.","Error",JOptionPane.ERROR_MESSAGE);
                        }
                    } catch (NumberFormatException ex) {
                            JOptionPane.showMessageDialog(rframe, "Enter valid Member
ID.","Error",JOptionPane.ERROR_MESSAGE);
```

```
                }
            }
        });



        // Deactivate Membership
            deactiveButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    try {
                        int id = Integer.parseInt(idField.getText());
                        GymMember member = findMemberById(id);
                            if (member != null) {
                                member.deactivateMembership();
                                JOptionPane.showMessageDialog(rframe, "Membership
deactivated.");
                            } else {
                                JOptionPane.showMessageDialog(rframe, "Member not
found,","Error",JOptionPane.ERROR_MESSAGE);
                        }
                    } catch (NumberFormatException ex) {
                            JOptionPane.showMessageDialog(rframe, "Enter valid Member
ID.","Error",JOptionPane.ERROR_MESSAGE);
                }
            }
        });


        // Mark Attendance
            markButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    try {
```

```
                    int id = Integer.parseInt(idField.getText());
                    GymMember member = findMemberById(id);
                        if (member != null) {
                            if (member.getActiveStatus()) {
                                member.markAttendance();
                                JOptionPane.showMessageDialog(rframe, "Attendance
marked.");
                            } else {
                                JOptionPane.showMessageDialog(rframe, "Membership is not
active.","Error",JOptionPane.ERROR_MESSAGE);
                            }
                        } else {
                            JOptionPane.showMessageDialog(rframe, "Member not
found.","Error",JOptionPane.ERROR_MESSAGE);
                    }
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(rframe, "Enter valid Member
ID.","Error",JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    // Clear Button - Resets all form fields
        clearButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            idField.setText("");
            nameField.setText("");
            locationField.setText("");
            phoneField.setText("");
            emailField.setText("");
            referralField.setText("");
            paidField.setText("");
            maleButton.setSelected(false);
            femaleButton.setSelected(false);
            dobyearCombo.setSelectedIndex(0);
            dobmonthCombo.setSelectedIndex(0);
            dobdayCombo.setSelectedIndex(0);
            msyearCombo.setSelectedIndex(0);
            msmonthCombo.setSelectedIndex(0);
            msdayCombo.setSelectedIndex(0);
            plansCombo.setSelectedIndex(0);
        }
    });

    //plan price change
        plansCombo.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            String selectedPlan = (String) plansCombo.getSelectedItem();

            if (selectedPlan.equals("BASIC")) {
                paidField.setText("6500");
            } else if (selectedPlan.equals("STANDARD")) {
                paidField.setText("12500");
            } else if (selectedPlan.equals("DELUXE")) {
                paidField.setText("18500");
            }
        }
    });
// Make the Regular Member form visible
```

```
    rframe.setVisible(true);

}

    // Utility method to find a member from the ArrayList by their ID
    private GymMember findMemberById(int id) {
        for (GymMember member : members) {
            if (member.getId() == id) {// Compare IDs
                return member; // Return member if match found
            }
        }
            return null;// Return null if no match is found
    }


    // Method to open the Premium Member registration frame
    public void openPremiumFrame(){
        JFrame pframe = new JFrame("Premium Member");
        pframe.setSize(800,600);
        pframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        pframe.setLayout(null);// Use absolute layout for manual positioning

        // Title label
        JLabel titleLabel = new JLabel("Premium Membership Details ",
JLabel.CENTER);
        titleLabel.setBounds(100,10,500,100);
        titleLabel.setFont(new Font("Arial", Font.BOLD,16));
        pframe.add(titleLabel);
```

```
// Input fields and labels for premium member details
JLabel idLabel = new JLabel("ID:");
idLabel.setBounds(30,100,100,30);
JTextField idField = new JTextField(20);
idField.setBounds(120,100,150,30);

JLabel nameLabel = new JLabel("NAME:");
nameLabel.setBounds(380,100,100,30);
JTextField nameField = new JTextField(20);
nameField.setBounds(550,100,150,30);

JLabel locationLabel = new JLabel("LOCATION:");
locationLabel.setBounds(30,140,100,30);
JTextField locationField = new JTextField(20);
locationField.setBounds(120,140,150,30);

JLabel phoneLabel = new JLabel("PHONE:");
phoneLabel.setBounds(380,140,100,30);
JTextField phoneField = new JTextField(20);
phoneField.setBounds(550,140,150,30);

JLabel emailLabel = new JLabel("EMAIL:");
emailLabel.setBounds(30,180,100,30);
JTextField emailField = new JTextField(20);
emailField.setBounds(120,180,150,30);

JLabel trainerLabel = new JLabel("TRAINER'S NAME:");
trainerLabel.setBounds(380,180,150,30);
JTextField trainerField = new JTextField(20);
```

```
        trainerField.setBounds(550,180,150,30);

         // Gender selection using radio buttons
        JLabel genderLabel = new JLabel("GENDER:");
        genderLabel.setBounds(30,220,100,30);
        JRadioButton maleButton= new JRadioButton("MALE");
        maleButton.setBounds(110,220,60,30);
        JRadioButton femaleButton= new JRadioButton("FEMALE");
        femaleButton.setBounds(180,220,160,30);

        ButtonGroup genderGroup = new ButtonGroup();
        genderGroup.add(maleButton);
        genderGroup.add(femaleButton);

        JLabel amountLabel = new JLabel("TOTAL AMOUNT:");
        amountLabel.setBounds(380,220,150,30);
        JTextField amountField = new JTextField(20);
        amountField.setBounds(550,220,150,30);
        amountField.setText("50000");// Fixed premium plan charge

         // ComboBoxes for Date of Birth
        JLabel dobLabel = new JLabel("DOB:");
        dobLabel.setBounds(30,260,100,30);
        String [] years =
{"1990","1991","1992","1993","1994","1995","1996","1996","1997","1998","1999","20
00","2001","2018","2019","2020","2021","2022","2023","2024","2025"};
        String [] months =
{"Jan","Feb","Mar","April","May","June","July","Aug","Sept","Oct","Nov","Dec"};
        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","30","31","32"};
```

```
        JComboBox<String> dobyearCombo = new JComboBox(years);
        dobyearCombo.setBounds(110,260,60,30);
        JComboBox dobmonthCombo = new JComboBox(months);
        dobmonthCombo.setBounds(190,260,60,30);
        JComboBox dobdayCombo = new JComboBox(days);
        dobdayCombo.setBounds(270,260,60,30);

        JLabel paidLabel = new JLabel("PAID AMOUNT:");
        paidLabel.setBounds(380,260,150,30);
        JTextField paidField = new JTextField(20);
        paidField.setBounds(550,260,150,30);

        JButton saveButton = new JButton("SAVE");
        saveButton.setBounds(450,320,150,25);
        saveButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try (BufferedWriter writer = new BufferedWriter(new
FileWriter("MemberDetails.txt",true))) {
                    writer.write(String.format("%-5s %-15s %-15s %-15s %-25s %-10s %-
10s %-15s %-10s %-10s %-10s %-10s\n",
                        "ID", "Name", "Location", "Phone", "Email", "Gender", "DOB",
"StartDate", "Plan", "Price", "Active", "Points"));
                    writer.write("_".repeat(150));
                    writer.newLine();
                    for (GymMember member : members) {
                        if (member instanceof RegularMember) {
                            RegularMember r = (RegularMember) member;
                            writer.write(String.format("%-5d %-15s %-15s %-15s %-25s %-10s
%-10s %-15s %-10s %-10.2f %-10s %-10.2f\n",
                        r.getId(), r.getName(), r.getLocation(), r.getPhone(), r.getEmail(),
r.getGender(),r.getDOB(), r.getMembershipStartDate(), r.getPlan(), r.getPrice(),
                        r.getActiveStatus() ? "Yes" : "No", r.getLoyaltyPoints()));
                        }
                    }
                    writer.write("_".repeat(150));
                    writer.newLine();

                    JOptionPane.showMessageDialog(pframe, "Members data saved to
MemberDetails.txt");
                } catch (IOException ex) {
                    JOptionPane.showMessageDialog(pframe, "Error saving to file: " +
ex.getMessage());
                }

            }
        });

            //read button
            JButton readButton = new JButton("READ");
            readButton.setBounds(450,350,150,25);
            readButton.addActionListener(new ActionListener() {
                @Override
                public void actionPerformed(ActionEvent e) {
                    try (BufferedReader reader = new BufferedReader(new
FileReader("MemberDetails.txt"))) {
                        String line;
                        while ((line = reader.readLine()) != null) {
                            System.out.println(line);
                }
            } catch (IOException ex) {
                System.out.println("Error reading file: " + ex.getMessage());
            }
        }
    });



        // Membership start date ComboBoxes
        JLabel memberLabel = new JLabel("MEMBERSHIP START DATE:");
        memberLabel.setBounds(30,300,200,30);
        String [] msdyears =
{"1990","1991","1992","1993","1994","1995","1996","1996","1997","1998","1999","20
00","2001","2018","2019","2020","2021","2022","2023","2024","2025"};
        String [] msdmonths =
{"Jan","Feb","Mar","April","May","June","July","Aug","Sept","Oct","Nov","Dec"};
        String [] msddays =
{"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","32"};

        JComboBox<String> msyearCombo = new JComboBox(msdyears);
        msyearCombo.setBounds(110,330,60,30);
        JComboBox msmonthCombo = new JComboBox(msdmonths);
        msmonthCombo.setBounds(180,330,60,30);
        JComboBox msdayCombo = new JComboBox(msddays);
        msdayCombo.setBounds(260,330,60,30);

        // Membership-related buttons
        JButton activeButton = new JButton("ACTIVATE MEMBERSHIP");
        activeButton.setBounds(50,400,200,30);
```

```
JButton deactiveButton = new JButton("DEACTIVATE MEMBERSHIP");
deactiveButton.setBounds(280,400,200,30);

JButton markButton = new JButton("MARK ATTENDANCE");
markButton.setBounds(510,400,200,30);

JButton revertButton = new JButton("REVERT MEMBER");
revertButton.setBounds(50,450,200,30);
revertButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
    try {
       if (idField.getText().equals("")) {
          JOptionPane.showMessageDialog(null, "Enter Member ID to revert.");
          return;
       }

       int id = Integer.parseInt(idField.getText());
       GymMember member = findMemberById(id);

       if (member == null) {
          JOptionPane.showMessageDialog(null, "Member not found.");
          return;
       }

       // Ask for removal reason
       String reason = JOptionPane.showInputDialog(null, "Enter reason for
reverting/removing the member:");
```

```
                if (reason == null || reason.equals("")) {
                    JOptionPane.showMessageDialog(null, "Removal reason is 
required.");
                    return;
                }

                // Log or show the reason
                JOptionPane.showMessageDialog(null, "Reason noted for removal: " + 
reason);

                // You can optionally log this to the console or a file
                System.out.println("Removal reason for Member ID " + id + ": " + 
reason);

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



    JButton displayButton = new JButton("DISPLAY");
    displayButton.setBounds(280,450,200,30);

    displayButton.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e){
                showMembers();
            }
    });
    JButton clearButton = new JButton("CLEAR");
    clearButton.setBounds(510,450,200,30);

    JButton upgradeButton = new JButton("PAY DUE AMOUNT");
    upgradeButton.setBounds(50,500,200,30);
    upgradeButton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
    try {
        if (idField.getText().equals("")) {
            JOptionPane.showMessageDialog(null, "Enter Member ID.");
            return;
        }

        int id = Integer.parseInt(idField.getText());
        GymMember member = findMemberById(id);

        if (member == null) {
            JOptionPane.showMessageDialog(null, "Member not found.");
            return;
        }

        String amount = paidField.getText();
        if (amount.equals("") || amount.equals("0")) {
            JOptionPane.showMessageDialog(null, "No due amount found.");
            return;
        }
```

```
            int confirm = JOptionPane.showConfirmDialog(null, "Confirm payment of Rs.
" + amount + "?", "Confirm Payment", JOptionPane.YES_NO_OPTION);
            if (confirm == JOptionPane.YES_OPTION) {
                paidField.setText("0");
                JOptionPane.showMessageDialog(null, "Payment successful. Due
cleared.");
            }

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

        JButton discountButton = new JButton("DISCOUNT");
        discountButton.setBounds(280,500,200,30);

        discountButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                try {
                    int originalPrice = 50000;
                    int paidAmount = Integer.parseInt(paidField.getText());

                    // Only allow discount if full payment has been made
                    if (paidAmount !=50000) {
                        JOptionPane.showMessageDialog(pframe, "Discount can only be
applied after full payment.");
                        return;
                    }
                    int discountPercent = 10; // Fixed 10% discount
                    int discountedPrice = originalPrice - (originalPrice * discountPercent / 100);

                    paidField.setText(String.valueOf(discountedPrice));

                    JOptionPane.showMessageDialog(pframe,
                        "Discount of " + discountPercent + "% applied.\nNew price: Rs. " + discountedPrice);
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(pframe, "Enter your paid amount.");
                }
            }
        });



        JButton addpButton = new JButton("ADD PREMIUM MEMBER");
        addpButton.setBounds(510,500,200,30);

        // Action for adding a premium member
        addpButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e){
                try {
                     // Check if ID is provided
                    if (idField.getText().isEmpty()) {
                        JOptionPane.showMessageDialog(pframe, "ID must be filled");
                        return;
                    }
                    int id = Integer.parseInt(idField.getText());
                    // Check if name is provided
                    if (nameField.getText().isEmpty()) {
                        JOptionPane.showMessageDialog(pframe, "Name must be filled");
                        return;
                    }

                    if(phoneField.getText().isEmpty()){
                        JOptionPane.showMessageDialog(pframe,"Phone number msut be filled.");
                        return;
                    }

                     // Create new PremiumMember object using form data
                    PremiumMember newMember = new PremiumMember(id, nameField.getText(), locationField.getText(), phoneField.getText(),emailField.getText(), maleButton.isSelected() ? "Male" : femaleButton.isSelected() ? "Female" : "Others",dobdayCombo.getSelectedItem() + "/" + dobmonthCombo.getSelectedItem() + "/" + dobyearCombo.getSelectedItem(),msdayCombo.getSelectedItem() + "/" + msmonthCombo.getSelectedItem() + "/" + msyearCombo.getSelectedItem(), trainerField.getText());
                    members.add(newMember);// Add to main list
                    JOptionPane.showMessageDialog(pframe, "Premium Member Added", "Success", JOptionPane.INFORMATION_MESSAGE);



                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(pframe, "ID must be a number");
                    JOptionPane.showMessageDialog(pframe, "Phone Number must be a number");
                }
```

```
    }
});
//adding the fields and labels in the panel
    pframe.add(idLabel);
    pframe.add(idField);
    pframe.add(nameLabel);
    pframe.add(nameField);
    pframe.add(locationLabel);
    pframe.add(locationField);
    pframe.add(phoneLabel);
    pframe.add(phoneField);
    pframe.add(emailLabel);
    pframe.add(emailField);
    pframe.add(trainerLabel);
    pframe.add(trainerField);
    pframe.add(amountField);
    pframe.add(amountLabel);
    pframe.add(genderLabel);
    pframe.add(maleButton);
    pframe.add(femaleButton);
    pframe.add(dobLabel);
    pframe.add(dobyearCombo);
    pframe.add(dobmonthCombo);
    pframe.add(dobdayCombo);
    pframe.add(paidLabel);
    pframe.add(paidField);
    pframe.add(memberLabel);
    pframe.add(msyearCombo);
    pframe.add(msmonthCombo);
```

```
        pframe.add(msdayCombo);

        pframe.add(saveButton);

        pframe.add(readButton);

        pframe.add(activeButton);

        pframe.add(deactiveButton);

        pframe.add(markButton);

        pframe.add(revertButton);

        pframe.add(displayButton);

        pframe.add(clearButton);

        pframe.add(upgradeButton);

        pframe.add(discountButton);

        pframe.add(addpButton);


    // Activate Membership

      activeButton.addActionListener(new ActionListener() {

          public void actionPerformed(ActionEvent e) {

            try {

              int id = Integer.parseInt(idField.getText());

              GymMember member = findMemberById(id);

                if (member != null) {

                  member.activateMembership();

                  JOptionPane.showMessageDialog(pframe,"Membership
activated.");

                } else {

                  JOptionPane.showMessageDialog(pframe, "Member not
found.","Error",JOptionPane.ERROR_MESSAGE);

              }

            } catch (NumberFormatException ex) {

                JOptionPane.showMessageDialog(pframe, "Enter valid Member
ID.","Error",JOptionPane.ERROR_MESSAGE);
```

```
            }
        }
    });

    // Deactivate Membership
    deactiveButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            try {
                int id = Integer.parseInt(idField.getText());
                GymMember member = findMemberById(id);
                    if (member != null) {
                        member.deactivateMembership();
                        JOptionPane.showMessageDialog(pframe, "Membership
deactivated.");
                    } else {
                        JOptionPane.showMessageDialog(pframe, "Member not
found.","Error",JOptionPane.ERROR_MESSAGE);
                }
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(pframe, "Enter valid Member
ID.","Error",JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    // Mark Attendance
        markButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            try {
            int id = Integer.parseInt(idField.getText());
```

```
                GymMember member = findMemberById(id);
                if (member != null) {
                    if (member.getActiveStatus()) {
                        member.markAttendance();
                        JOptionPane.showMessageDialog(pframe, "Attendance marked.");
                    } else {
                        JOptionPane.showMessageDialog(pframe, "Membership is not
active.","Error",JOptionPane.ERROR_MESSAGE);
                    }
                } else {
                    JOptionPane.showMessageDialog(pframe, "Member not
found.","Error",JOptionPane.ERROR_MESSAGE);
                }
            } catch (NumberFormatException ex) {
                JOptionPane.showMessageDialog(pframe, "Enter valid Member
ID.","Error",JOptionPane.ERROR_MESSAGE);
            }
        }
    });

    //clear button
    clearButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        idField.setText("");
        nameField.setText("");
        locationField.setText("");
        phoneField.setText("");
        emailField.setText("");
        trainerField.setText("");
        paidField.setText("");
```

```
                maleButton.setSelected(false);
                femaleButton.setSelected(false);
                amountField.setText("50000");
            }
        });

        //making the frame visible
        pframe.setVisible(true);
    }
    }
```