



islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

60% Individual Coursework

2025 Spring

Student Name: Aviyaan Shrestha

London Met ID: 24046725

College ID: np01cp4a240080@islingtoncollege.edu.np

Assignment Due Date: Friday, May 16, 2025

Assignment Submission Date: Thursday, May 15, 2025

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

24046725 Aviyaan_Shrestha - Copy.docx

 Islington College, Nepal

Document Details

Submission ID

trn:oid::3618:96171062

Submission Date

May 16, 2025, 10:13 AM GMT+5:45

Download Date

May 16, 2025, 10:14 AM GMT+5:45

File Name

24046725 Aviyaan_Shrestha - Copy.docx

File Size

49.1 KB

64 Pages

5,439 Words

31,054 Characters







Page 2 of 68 - Integrity Overview

Submission ID trn:oid::3618:96171062




14% Overall Similarity

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

Match Groups

-  **45 Not Cited or Quoted 14%**
Matches with neither in-text citation nor quotation marks
-  **0 Missing Quotations 0%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 2%  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 14%
Matches with neither in-text citation nor quotation marks
- 0** Missing Quotations 0%
Matches that are still very similar to source material
- 0** Missing Citation 0%
Matches that have quotation marks, but no in-text citation
- 0** Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 2% 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	9%
2	Submitted works	Runshaw College, Lancashire on 2018-12-13	<1%
3	Submitted works	University of Westminster on 2017-05-01	<1%
4	Submitted works	iacademy on 2025-05-14	<1%
5	Submitted works	Central Queensland University on 2024-06-10	<1%
6	Submitted works	Institute of Business Studies on 2025-03-30	<1%
7	Internet	www.coursehero.com	<1%
8	Submitted works	TAFE Queensland Brisbane on 2023-05-26	<1%
9	Internet	www.um.edu.mt	<1%
10	Internet	www.devx.com	<1%

11	Submitted works	PSB Academy (ACP eSolutions) on 2017-09-11	<1%
12	Submitted works	AlHussein Technical University on 2024-06-09	<1%
13	Submitted works	Colorado Technical University Online on 2014-11-08	<1%
14	Submitted works	Esoft Metro Campus, Sri Lanka on 2025-02-11	<1%
15	Submitted works	King's College on 2016-10-30	<1%

Contents

Introduction.....	5
Aims and Objectives	6
Tools Used:	8
Wireframe	11
Class Diagram.....	12
Pseudocode	15
Method Description	44
RegularMember	48
PremiumMember	52
Testing.....	58
Test 1.....	58
Test 2.1.....	59
Test 2.2.....	62
Test 3.1.....	66
Test 4.1.....	69
Test 4.2.....	72
Test 4.3.....	75
Test 5.1.....	78
Test 5.2.....	80
Error detection and correction	81
Logical error.....	81
Runtime error	81
Conclusion	82
Appendix.....	83

Introduction

This coursework is an exciting opportunity to apply my knowledge of Java and object-oriented programming to a real-world problem. I'll be creating a gym management system that keeps track of members' details, attendance, loyalty points, and membership plans. The system will include a parent class, Gym Member, and two subclasses, Regular Member and Premium Member, to represent different types of memberships. To make the system user-friendly, I'll also design a Graphical User Interface (GUI) that allows users to interact with the system seamlessly.

Java is the perfect language for this project because of its versatility, platform independence, and robustness. It's also a great way to deepen my understanding of object-oriented concepts like inheritance, encapsulation, and polymorphism. By building this system, I'll not only improve my programming skills but also learn how to translate theoretical knowledge into practical solutions.

To bring this project to life, I'll be using two main tools: BlueJ and Balsamiq. BlueJ is a beginner-friendly IDE that will help me write and test my Java code efficiently. Balsamiq, on the other hand, will help me design the GUI by creating wireframes that outline the system's layout and functionality. This combination of tools will ensure that the final product is both functional and user-friendly.

Overall, this coursework is a great way to challenge myself, enhance my skills, and create something that could potentially be used in a real-world setting. I'm looking forward to the journey and the learning experience it will bring!

Java



Java is a widely-used, versatile programming language that was first released in 1995 and is currently owned by Oracle. It is known for its platform independence, allowing programs to run on various operating systems like Windows, macOS, and Linux. Java is open-source, free to use, and is renowned for its robustness, security, and strong memory management. The latest version, JDK 23, was released on September 17, 2024 (W3Schools, n.d.). Java's portability, achieved through its bytecode, enables developers to write code once and run it anywhere. These features make Java an ideal choice for developing scalable and reliable applications, such as the gym management system in this coursework.

Aims and Objectives

The primary aims and objectives of this coursework are as follows:

1. **Develop a Gym Member Management System:** Create a system that efficiently manages gym members, tracks their activities, and stores their personal details.
2. **Understand Java and OOP Concepts:** Gain a deeper understanding of Java's object-oriented principles, such as inheritance, encapsulation, and polymorphism, by applying them in a real-world scenario.
3. **Convert Theoretical Knowledge into Practice:** Implement theoretical concepts learned in class to solve practical problems.
4. **Enhance Programming Skills:** Improve my skills as an undergraduate student by building a functional application from scratch.
5. **Develop Logical Thinking:** Strengthen my ability to design and implement logical solutions for various aspects of the system, such as membership upgrades, attendance tracking, and loyalty point management.

Tools Used:
BlueJ



BlueJ is an Integrated Development Environment (IDE) specifically designed for Java programming. It is particularly suited for beginners due to its simple and intuitive interface. BlueJ allows users to interact with objects, inspect their values, and call methods, making it an excellent tool for learning and teaching Java. It supports all major platforms, including Windows, macOS, and Linux, and is widely used in educational settings to help students quickly grasp programming concepts (BlueJ, n.d.). For this coursework, BlueJ will be used to write, test, and debug the Java code.

Balsamiq



Balsamiq is a wireframing tool used to design the layout of applications and websites. It provides a user-friendly platform for creating early-stage designs, enabling developers to visualize the structure and flow of their applications. Balsamiq encourages collaboration and feedback, which helps refine the design before moving on to the coding phase (Makhani, 2024). For this project, Balsamiq will be used to design the GUI for the gym management system, ensuring a clear and intuitive user experience.

Microsoft Word



Microsoft Word is a word processing program that allows for the creation of both simple and complex documents. It is used in making documents, writing letters, making reports etc. Microsoft Word is a word processor developed by Microsoft. It has advanced features which allow us to perform format and edit our files and documents in the best possible way.

Wireframe

GUI Wireframe

ID:

Name:

Location:

Phone:

Gender:

Email:

DOB:

MembershipStartDate:

Paid Amount:

Trainer's Name:

Referral Source:

Removal Reason:

Plans:

Activate Membership

Deactivate Membership



Revert Membership

Upgrade Plan

Calcuete Discount

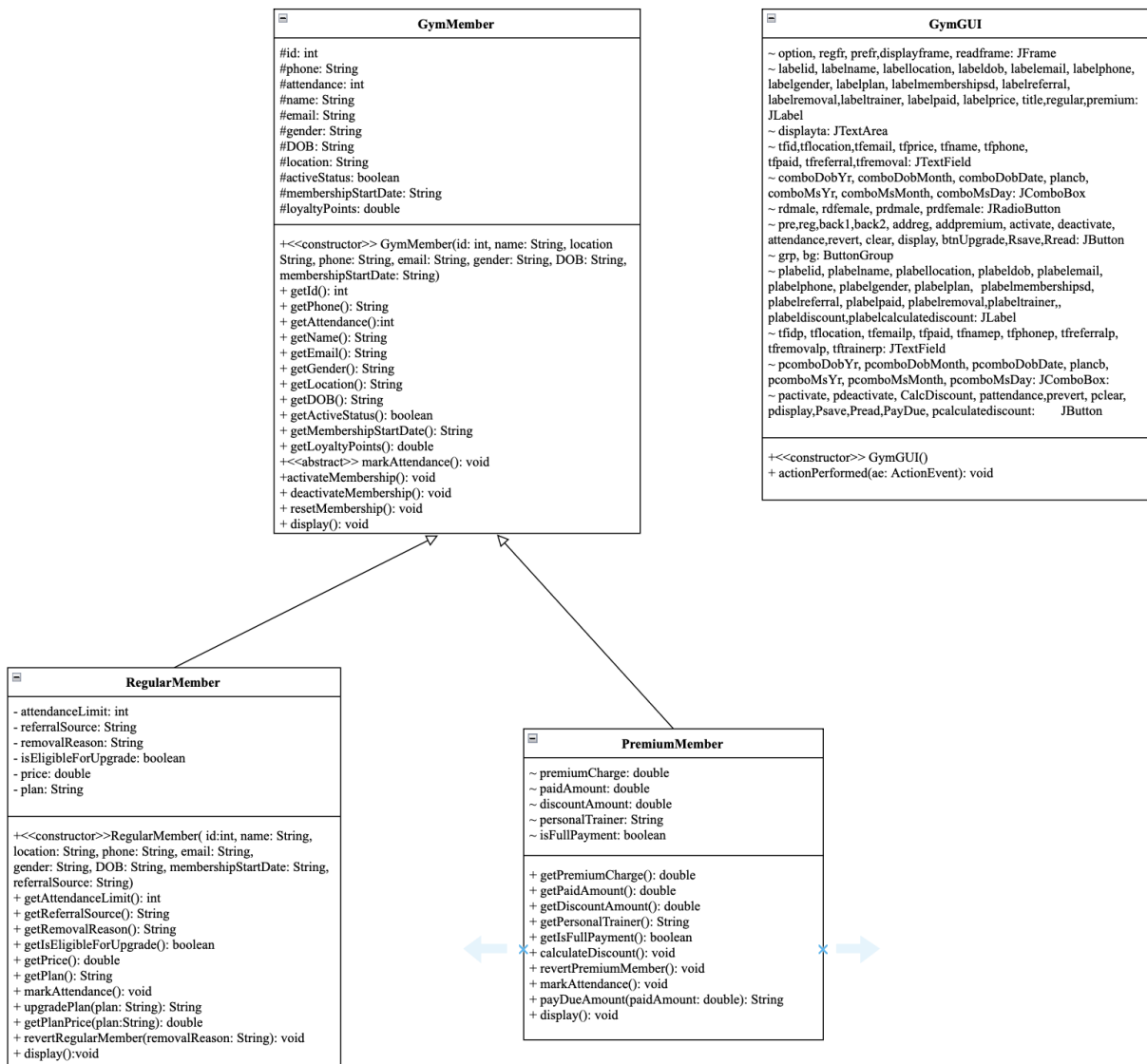
Pay Due Amount

Class Diagram

	PremiumMember
~ premiumCharge: double ~ paidAmount: double ~ discountAmount: double ~ personalTrainer: String ~ isFullPayment: boolean	
+ getPremiumCharge(): double + getPaidAmount(): double + getDiscountAmount(): double + getPersonalTrainer(): String + getIsFullPayment(): boolean + calculateDiscount(): void + revertPremiumMember(): void + markAttendance(): void + payDueAmount(paidAmount: double): String + display(): void	
	RegularMember
- attendanceLimit: int - referralSource: String - removalReason: String - isEligibleForUpgrade: boolean - price: double - plan: String	
+ <<constructor>>RegularMember(id:int, name: String, location: String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String, referralSource: String) + getAttendanceLimit(): int + getReferralSource(): String + getRemovalReason(): String + getIsEligibleForUpgrade(): boolean + getPrice(): double + getPlan(): String + markAttendance(): void + upgradePlan(plan: String): String + getPlanPrice(plan:String): double + revertRegularMember(removalReason: String): void + display():void	

GymGUI
~ option, regfr, prefr,displayframe, readframe: JFrame ~ labelid, labelname, labellocation, labeldob, lalelemail, labelphone, labelgender, labelplan, labelmemberspsd, labelreferral, labelremoval,labeltrainer, labelpaid, labelprice, title,regular,premium: JLabel ~ displayta: JTextArea ~ tfid,tflocation,tfemail, tfprice, tfname, tfphone, tfpaid, tfreferral,tfremoval: JTextField ~ comboDobYr, comboDobMonth, comboDobDate, plancb, comboMsYr, comboMsMonth, comboMsDay: JComboBox ~ rdmale, rdfemale, prdmale, prdfemale: JRadioButton ~ pre,reg,back1,back2, addreg, addpremium, activate, deactivate, attendance,revert, clear, display, btnUpgrade,Rsave,Rread: JButton ~ grp, bg: ButtonGroup ~ labelid, labelname, labellocation, labeldob, plabelemail, labelphone, labelgender, labelplan, labelmemberspsd, labelreferral, labelpaid, labelremoval,labeltrainer,, labeldiscount,labelcalculateddiscount: JLabel ~ tfidp, tflocation, tfemailp, tfpaid, tfnamep, tfphonep, tfreferralp, tfremovalp, tftrainerp: JTextField ~ pcomboDobYr, pcomboDobMonth, pcomboDobDate, plancb, pcomboMsYr, pcomboMsMonth, pcomboMsDay: JComboBox: ~ pactivate, pdeactivate, CalcDiscount, pattendance,prevert, pclear, pdisplay,Psave,Pread,PayDue, pcalculateddiscount: JButton
+<<constructor>> GymGUI() + actionPerformed(ae: ActionEvent): void

GymMember
#id: int #phone: String #attendance: int #name: String #email: String #gender: String #DOB: String #location: String #activeStatus: boolean #membershipStartDate: String #loyaltyPoints: double
+<<constructor>> GymMember(id: int, name: String, location String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String) + getId(): int + getPhone(): String + getAttendance():int + getName(): String + getEmail(): String + getGender(): String + getLocation(): String + getDOB(): String + getActiveStatus(): boolean + getMembershipStartDate(): String + getLoyaltyPoints(): double +<<abstract>> markAttendance(): void +activateMembership(): void + deactivateMembership(): void + resetMembership(): void + display(): void



Pseudocode:

GymMember Abstract Class

ABSTRACT CLASS GymMember

DECLARE id AS INTEGER

DECLARE name AS STRING

DECLARE location AS STRING

DECLARE phone AS STRING

DECLARE email AS STRING

DECLARE gender AS STRING

DECLARE DOB AS STRING

DECLARE membershipStartDate AS STRING

DECLARE attendance AS INTEGER

DECLARE loyaltyPoints AS DOUBLE

DECLARE activeStatus AS BOOLEAN

METHOD GymMember(id, name, location, phone, email, gender, DOB, membershipStartDate)

SET this.id = id

SET this.name = name

SET this.location = location

SET this.phone = phone

SET this.email = email

SET this.gender = gender

SET this.DOB = DOB

SET this.membershipStartDate = membershipStartDate

INITIALIZE attendance = 0

```
    INITIALIZE loyaltyPoints = 0.0
    INITIALIZE activeStatus = FALSE
END METHOD
```

```
METHOD getId()
    RETURN id
END METHOD
```

```
METHOD getName()
    RETURN name
END METHOD
```

```
METHOD getLocation()
    RETURN location
END METHOD
```

```
METHOD getPhone()
    RETURN phone
END METHOD
```

```
METHOD getEmail()
    RETURN email
END METHOD
```

```
METHOD getGender()
    RETURN gender
END METHOD
```


METHOD getDOB()

RETURN DOB

END METHOD

METHOD getMembershipStartDate()

RETURN membershipStartDate

END METHOD

METHOD getAttendance()

RETURN attendance

END METHOD

METHOD getLoyaltyPoints()

RETURN loyaltyPoints

END METHOD

METHOD getActiveStatus()

RETURN activeStatus

END METHOD

ABSTRACT METHOD markAttendance()

METHOD activateMembership()

SET activeStatus = TRUE

END METHOD

METHOD deactivateMembership()

IF activeStatus **IS** TRUE **THEN**

```
        SET activeStatus = FALSE
    ELSE
        OUTPUT "Membership is already deactivated."
    END IF
END METHOD
```

```
METHOD resetMembership()
    SET activeStatus = FALSE
    SET attendance = 0
    SET loyaltyPoints = 0.0
END METHOD
```

```
METHOD display()
    OUTPUT "ID: " + id
    OUTPUT "Name: " + name
    OUTPUT "Location: " + location
    OUTPUT "Phone: " + phone
    OUTPUT "Email: " + email
    OUTPUT "Gender: " + gender
    OUTPUT "DOB: " + DOB
    OUTPUT "Membership Start Date: " + membershipStartDate
    OUTPUT "Attendance: " + attendance
    OUTPUT "Loyalty Points: " + loyaltyPoints
    OUTPUT "Active Status: " + activeStatus
END METHOD
```

```
END CLASS
```

RegularMember Class (Extends GymMember)

CLASS RegularMember **EXTENDS** GymMember

DECLARE attendanceLimit **AS** **CONSTANT** **INTEGER** = 30

DECLARE isEligibleForUpgrade **AS** **BOOLEAN**

DECLARE removalReason **AS** **STRING**

DECLARE referralSource **AS** **STRING**

DECLARE plan **AS** **STRING**

DECLARE price **AS** **DOUBLE**

METHOD RegularMember(id, name, location, phone, email, gender, DOB, membershipStartDate, referralSource)

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

SET this.referralSource = referralSource

INITIALIZE isEligibleForUpgrade = **FALSE**

INITIALIZE plan = "basic"

INITIALIZE price = 6500

INITIALIZE removalReason = ""

END METHOD

METHOD getAttendanceLimit()

RETURN attendanceLimit

END METHOD

METHOD getIsEligibleForUpgrade()

RETURN isEligibleForUpgrade

END METHOD

METHOD getRemovalReason()

RETURN removalReason

END METHOD

METHOD getReferralSource()

RETURN referralSource

END METHOD

METHOD getPlan()

RETURN plan

END METHOD

METHOD getPrice()

RETURN price

END METHOD

METHOD markAttendance()

INCREMENT attendance BY 1

INCREMENT loyaltyPoints BY 5.0

END METHOD

METHOD getPlanPrice(plan)

CONVERT plan TO LOWERCASE

SWITCH plan

CASE "basic": **RETURN** 6500

CASE "standard": **RETURN** 12500

CASE "deluxe": **RETURN** 18500

DEFAULT: **RETURN** -1

END SWITCH

END METHOD

METHOD upgradePlan(newPlan)

IF newPlan **EQUALS** (IGNORE CASE) plan **THEN**

RETURN "You are already subscribed to this plan."

END IF

SET newPrice = **CALL** getPlanPrice(newPlan)

IF newPrice **EQUALS** -1 **THEN**

RETURN "Invalid plan selected."

END IF

IF attendance **GREATER THAN OR EQUAL TO** attendanceLimit **THEN**

SET plan = newPlan

SET price = newPrice

RETURN "Plan upgraded successfully to " + newPlan + "."

END IF

RETURN "You are not eligible for an upgrade."

END METHOD

METHOD revertRegularMember(removalReason)

CALL super.resetMembership()

SET isEligibleForUpgrade = FALSE

SET plan = "basic"

SET price = 6500

SET this.removalReason = removalReason

END METHOD

METHOD display()

CALL super.display()

OUTPUT "Plan: " + plan

OUTPUT "Price: " + price

IF removalReason IS NOT EMPTY **THEN**

OUTPUT "Removal Reason: " + removalReason

END IF

END METHOD

END CLASS

PremiumMember Class (Extends GymMember)

CLASS PremiumMember **EXTENDS** GymMember

DECLARE premiumCharge **AS** **CONSTANT** **DOUBLE** = 50000

DECLARE personalTrainer **AS** **STRING**

DECLARE isFullPayment **AS** **BOOLEAN**

DECLARE paidAmount **AS** **DOUBLE**

DECLARE discountAmount AS DOUBLE

METHOD PremiumMember(id, name, location, phone, email, gender, DOB, membershipStartDate, personalTrainer)

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

SET this.personalTrainer = personalTrainer

INITIALIZE isFullPayment = FALSE

INITIALIZE paidAmount = 0

INITIALIZE discountAmount = 0

END METHOD

METHOD getPremiumCharge()

RETURN premiumCharge

END METHOD

METHOD getPersonalTrainer()

RETURN personalTrainer

END METHOD

METHOD getIsFullPayment()

RETURN isFullPayment

END METHOD

METHOD getPaidAmount()

RETURN paidAmount

END METHOD

METHOD getDiscountAmount()

RETURN discountAmount

END METHOD

METHOD markAttendance()

INCREMENT attendance BY 1

INCREMENT loyaltyPoints BY 10.0

END METHOD

METHOD payDueAmount(amount)

IF isFullPayment **IS TRUE THEN**

RETURN "Payment is already complete."

END IF

IF paidAmount + amount > premiumCharge **THEN**

RETURN "Payment exceeds the premium charge."

END IF

INCREMENT paidAmount BY amount

IF paidAmount **EQUALS** premiumCharge **THEN**

SET isFullPayment = **TRUE**

END IF

SET remainingAmount = premiumCharge - paidAmount


```
    RETURN "Payment successful. Remaining amount: " +  
remainingAmount
```

```
END METHOD
```

```
METHOD calculateDiscount()
```

```
    IF isFullPayment IS TRUE THEN
```

```
        SET discountAmount = premiumCharge * 0.10
```

```
        OUTPUT "Discount calculated: " + discountAmount
```

```
    ELSE
```

```
        OUTPUT "No discount available."
```

```
    END IF
```

```
END METHOD
```

```
METHOD revertPremiumMember()
```

```
    CALL super.resetMembership()
```

```
    SET personalTrainer = ""
```

```
    SET isFullPayment = FALSE
```

```
    SET paidAmount = 0
```

```
    SET discountAmount = 0
```

```
END METHOD
```

```
METHOD display()
```

```
    CALL super.display()
```

```
    OUTPUT "Personal Trainer: " + personalTrainer
```

```
    OUTPUT "Paid Amount: " + paidAmount
```

```
    OUTPUT "Full Payment: " + isFullPayment
```

```
    OUTPUT "Remaining Amount: " + (premiumCharge - paidAmount)
```

```
    IF isFullPayment IS TRUE THEN
        OUTPUT "Discount Amount: " + discountAmount
    END IF
END METHOD
```

```
END CLASS
```

GymGUI

```
BEGIN PROGRAM GymGUI
```

```
    DECLARE array AS ArrayList of GymMember
```

```
    DECLARE all GUI components (JFrames, JLabels, JTextFields,  
    JComboBoxes, JButtons, JRadioButtons, ButtonGroups, JTextAreas)
```

```
    CALL new GymGUI()
```

```
BEGIN GymGUI
```

```
    INITIALIZE option AS JFrame
```

```
    DECLARE title, reg, pre AS GUI components
```

```
    SET title text to "Are you regular or premium member"
```

```
    SET bounds for title, reg, pre buttons
```

```
    ADD title, reg, pre TO option frame
```

```
    SET layout of option TO null
```

```
    SET visibility of option TO true
```

```
    SET size of option
```

INITIALIZE regfr AS JFrame

DECLARE and **SET** bounds for labels: regular, labelid, labelname, labellocation, etc.

DECLARE and **INITIALIZE** JTextFields: tfid, tfname, tfemail, etc.

DECLARE and **INITIALIZE** JComboBoxes: comboDobYr, comboDobMonth, comboDobDate, etc.

DECLARE and **INITIALIZE** JRadioButtons: malerd, femalerd

ADD radio buttons to ButtonGroup grp

DECLARE and **SET** bounds for buttons: addreg, activate, deactivate, etc.

ADD all components TO regfr frame

SET layout of regfr TO null

SET visibility of regfr TO false

SET size of regfr

INITIALIZE prefr AS JFrame

DECLARE and **SET** bounds for labels: premium, plabelid, plabelname, etc.

DECLARE and **INITIALIZE** JTextFields: tfidp, tfnamep, tfemailp, etc.

DECLARE and **INITIALIZE** JComboBoxes: pcomboDobYr, pcomboDobMonth, etc.

DECLARE and **INITIALIZE** JRadioButtons: pmalerd, pfemalerd

ADD radio buttons to ButtonGroup bg

DECLARE and **SET** bounds for buttons: addpre, pactivate, pdeactivate, etc.

ADD all components TO prefr frame

SET layout of prefr TO null

SET visibility of prefr TO false

SET size of prefr

INITIALIZE displayframe AS JFrame

INITIALIZE displayta AS JTextArea

SET bounds for displayta

ADD displayta TO displayframe

SET layout, visibility, and size for displayframe

CALL addActionListener ON reg, pre, addreg, addpre, activate, deactivate, attendance, revert, clear, display, etc.

Method actionPerformed(event):

IF event source is reg **THEN**

CALL set regfr visible to true

CALL set prefr visible to false

ELSE IF event source is pre **THEN**

CALL set prefr visible to true

CALL set regfr visible to false

ELSE IF event source is addreg **THEN**

IF any required registration fields are empty **THEN**

CALL show message "please fill in the application"

ELSE

TRY

DECLARE regid as Integer

SET regid to parsed integer from tfid

DECLARE regLocation, regEmail, regName, regPhone,
regReferral as String

SET each from respective text fields

DECLARE regGender as String

IF male radio is selected **THEN**

SET regGender to "male"

ELSE IF female radio is selected **THEN**

SET regGender to "female"

DECLARE year, month, date as String

SET each from DOB combo boxes

SET regDob to date + "-" + month + "-" + year

DECLARE myear, mmonth, mdate as String

SET each from Membership Start Date combo boxes

SET regMemberships to mdate + "-" + mmonth + "-" + myear

IF array is empty **THEN**

INITIALIZE regobj as new RegularMember with collected data

CALL add regobj to array

CALL show message "Regular Member added"

ELSE

DECLARE idexists as boolean = false

FOR each member in array **DO**

IF member ID equals regid **THEN**

```
        SET idexists to true
        BREAK loop
    IF idexists THEN
        CALL show message "please choose another ID."
    ELSE
        INITIALIZE regobj as new RegularMember with collected
data
        CALL add regobj to array
        CALL show message "Regular Member added"
    CATCH NumberFormatException THEN
        CALL show message "Please input number for ID"

ELSE IF event source is attendance THEN
    IF tfid is empty THEN
        CALL show message "ID found empty"
    ELSE
        TRY
            DECLARE regid as Integer from tfid
            DECLARE idexists as boolean = false
            DECLARE memberExists as GymMember = null

            FOR each member in array DO
                IF member ID equals regid THEN
                    SET idexists to true
                    SET memberExists to member
                    BREAK loop
```

```
IF idexists THEN
    IF member is active THEN
        CALL member.markAttendance()
        CALL show message "Attendance marked"
    ELSE
        CALL show message "member is not activate"
    ELSE
        CALL show message "ID not found"
CATCH NumberFormatException THEN
    CALL show message "Please input number for ID"

ELSE IF event source is activate THEN
    IF tfid is empty THEN
        CALL show message "ID found empty"
    ELSE
        TRY
            DECLARE regid as Integer from tfid
            DECLARE idexists as boolean = false
            DECLARE memberExists as GymMember = null

            FOR each member in array DO
                IF member ID equals regid THEN
                    SET idexists to true
                    SET memberExists to member
                    BREAK loop

            IF idexists THEN
```

```
    IF member is already active THEN
        CALL show message "Membership is already activated"
    ELSE
        CALL member.activateMembership()
        CALL show message "Membership activated"
    ELSE
        CALL show message "ID not found"
CATCH NumberFormatException THEN
    CALL show message "Please input number for ID"

ELSE IF event source is deactivate THEN
    IF tfid is empty THEN
        CALL show message "ID found empty"
    ELSE
        TRY
            DECLARE regid as Integer from tfid
            DECLARE idexists as boolean = false
            DECLARE memberExists as GymMember = null

            FOR each member in array DO
                IF member ID equals regid THEN
                    SET idexists to true
                    SET memberExists to member
                    BREAK loop

            IF idexists THEN
                IF member is already inactive THEN
```



```

        CALL show message "Membership is already deactivated"
    ELSE
        CALL member.deactivateMembership()
        CALL show message "Membership deactivated"
    ELSE
        CALL show message "ID not found"
    CATCH NumberFormatException THEN
        CALL show message "Please input number for ID"
IF event source is revert THEN
    IF tfid is empty THEN
        CALL show message "Please input value for Member ID"
    ELSE IF tfremoval is empty THEN
        CALL show message "Please input a valid removal reason!"
    ELSE
        TRY
            DECLARE memId as Integer from tfid
            DECLARE removalReason as String from tfremoval
            DECLARE isMemberFound as boolean = false
            DECLARE matchedMember as RegularMember = null

            FOR each member in array DO
                IF member is instance of RegularMember THEN
                    CAST member to RegularMember
                    IF member ID equals memId THEN
                        SET isMemberFound to true
                        SET matchedMember to this member
                        BREAK loop

```

IF isMemberFound **THEN**

CALL matchedMember.revertRegularMember(removalReason)

CALL show message "Successfully Reverted Member Status"

ELSE

CALL show message "Member with this ID does not exist"

CATCH NumberFormatException **THEN**

CALL show message "ID can only have number values!"

ELSE IF event source is display **THEN**

CALL set displayframe visible to true

CALL clear displayta text area

IF array is empty **THEN**

CALL append "NO MEMBER REGISTERED!!!" to displayta

ELSE

CALL append "REGULAR MEMBERS:\n\n" to displayta

DECLARE memberExistence as boolean = false

FOR each member in array **DO**

IF member is instance of RegularMember **THEN**

SET memberExistence to true

BREAK loop

IF memberExistence **THEN**

FOR each member in array **DO**

IF member is instance of RegularMember **THEN**

CAST member to RegularMember

CALL append member details to displayta:

- ID
- Name
- Location
- Email
- Phone
- DOB
- Gender
- Plan
- Price
- Membership Start Date
- Attendance
- Loyalty Points
- Active Status
- (IF exists) Removal Reason

CALL append separator line

ELSE

CALL append "NO REGULAR MEMBER REGISTERED!!!" to
display

IF event source is clear **OR** event source is pclear **THEN**

CALL set text "" for each input field: tfid, tflocation, tfemail, tfpaid, tfname,
tfphone, tfreferral, tfremoval

CALL set text "" for each premium input field: tfidp, tflocationp, tfemailp, tfpaidp, tfnamep, tfphonep, tfreferralp, tfremovalp, tftrainerp

CALL set selected item for combo boxes:

 comboDobMonth to "January"

 comboDobYr to "2006"

 comboDobDate to "1"

 comboMsDay to "1"

 comboMsMonth to "January"

 comboMsYr to "2006"

CALL set selected item plancb to "basic"

ELSE IF event source is addpre **THEN**

TRY

IF any required premium input fields are empty **THEN**

CALL show message "please fill in the application"

ELSE

DECLARE preid as Integer from tfidp

DECLARE preLocation, preEmail, preName, prePhone, preTrainer

as Strings from respective inputs

DECLARE preGender as String = ""

IF pmalerd selected **THEN SET** preGender = "male"

ELSE IF pfemalerd selected **THEN SET** preGender = "female"

DECLARE year, month, date from pcomboDobYr,

pcomboDobMonth, pcomboDobDate

DECLARE preDob as String = date + "-" + month + "-" + year

DECLARE myear, mmonth, mdate from pcomboMsYr,
pcomboMsMonth, pcomboMsDay

DECLARE preMemberships as String = mdate + "-" + mmonth + "-"
+ myear

IF array is empty **THEN**

CREATE PremiumMember preobj with above details

ADD preobj to array

CALL show message "Premium Member added"

ELSE

DECLARE idexits as boolean = false

FOR each mem in array **DO**

IF mem.getId() equals preid **THEN**

SET idexits to true

BREAK loop

IF idexits **THEN**

CALL show error message "please choose another ID."

ELSE

CREATE PremiumMember preobj with above details

ADD preobj to array

CALL show message "Premium Member added"

CATCH NumberFormatException **THEN**

CALL show warning message "Please input number for ID"

ELSE IF event source is pattendance **THEN**

IF tfidp is empty **THEN**

```

    CALL show message "ID found empty"
ELSE
    TRY
        DECLARE preid as Integer from tfidp
        DECLARE idexits as boolean = false
        DECLARE memberExists as GymMember = null

        FOR each member in array DO
            IF member.getId() equals preid THEN
                SET idexits to true
                SET memberExists to member
                BREAK loop

            IF idexits THEN
                IF memberExists.getActiveStatus() is true THEN
                    CALL memberExists.markAttendance()
                    CALL show info message "Attendance marked"
                ELSE
                    CALL show error message "member is not active"
                ELSE
                    CALL show error message "ID not found"
            CATCH NumberFormatException THEN
                CALL show warning message "Please input number for ID"

ELSE IF event source is prevert THEN
    IF tfidp is empty THEN
        CALL show message "ID found empty"

```

ELSE IF tfremovalp is empty **THEN**

CALL show message "please fill in the removal reason"

ELSE

TRY

DECLARE preid as Integer from tfidp

DECLARE preRemoval as String from tfremovalp

DECLARE idexits as boolean = false

DECLARE memberExists as GymMember = null

FOR each member in array **DO**

IF member.getId() equals preid **THEN**

SET idexits to true

SET memberExists to member

BREAK loop

IF idexits **THEN**

CALL memberExists.resetMembership()

CALL show info message "Membership Reset"

ELSE

CALL show error message "ID not found"

CATCH NumberFormatException **THEN**

CALL show warning message "Please input number for ID"

ELSE IF event source is pactivate **THEN**

IF tfidp is empty **THEN**

CALL show message "ID found empty"

ELSE

TRY

DECLARE preid as Integer from tfidp

DECLARE idexits as boolean = false

DECLARE memberExists as GymMember = null

FOR each member in array **DO**

IF member.getId() equals preid **THEN**

SET idexits to true

SET memberExists to member

BREAK loop

IF idexits **THEN**

IF memberExists.getActiveStatus() is true **THEN**

CALL show info message "Membership is already activated"

ELSE

CALL memberExists.activateMembership()

CALL show info message "Membership activated"

ELSE

CALL show error message "ID not found"

CATCH NumberFormatException **THEN**

CALL show warning message "Please input number for ID"

ELSE IF event source is pdeactivate **THEN**

IF tfidp is empty **THEN**

CALL show message "ID found empty"

ELSE

TRY


```
DECLARE preid as Integer from tfidp
DECLARE idexits as boolean = false
DECLARE memberExists as GymMember = null
```

```
FOR each member in array DO
    IF member.getId() equals preid THEN
        SET idexits to true
        SET memberExists to member
        BREAK loop
```

```
IF idexits THEN
    IF memberExists.getActiveStatus() is false THEN
        CALL show info message "Membership is already deactivated"
    ELSE
        CALL memberExists.deactivateMembership()
        CALL show info message "Membership deactivated"
ELSE
    CALL show error message "ID not found"
CATCH NumberFormatException THEN
    CALL show warning message "Please input number for ID"
```

```
ELSE IF event source is pdisplay THEN
    CALL set displayframe visible to true
    CALL clear displayta text area
```

```
IF array is empty THEN
    CALL append "NO MEMBER REGISTERED!!!" to displayta
```

ELSE

CALL append "\nPREMIUM MEMBERS:\n\n" to displayta

DECLARE memberExistence as boolean = false

FOR each member in array **DO**

IF member is instance of PremiumMember **THEN**

SET memberExistence to true

BREAK loop

IF memberExistence **THEN**

FOR each member in array **DO**

IF member is instance of PremiumMember **THEN**

CAST to PremiumMember

CALL append member details including:

ID, Name, Location, Email, Phone, DOB, Gender,
Membership Start Date, Attendance, Loyalty Points,
Personal Trainer, Paid Amount, Active Status,
Payment status and discount info

CALL append separator line

ELSE

CALL append "NO PREMIUM MEMBER REGISTERED!!!" to
displayta

ELSE IF event source is pcalculateddiscount **THEN**

IF tfidp is empty **THEN**

CALL show message "Please input value for Member ID"

ELSE

TRY

DECLARE memId as Integer from tfidp

DECLARE isMemberFound as boolean = false

DECLARE matchedMember as PremiumMember = null

FOR each member in array **DO**

IF member is instance of PremiumMember **THEN**

IF member.getId() equals memId **THEN**

SET isMemberFound to true

SET matchedMember to member

BREAK loop

IF isMemberFound **THEN**

IF matchedMember.getActiveStatus() is true **THEN**

IF matchedMember.getIsFullPayment() is true **THEN**

CALL matchedMember.calculateDiscount()

CALL show info message "Member has successfully
received 10% Discount"

ELSE

CALL show message "Member not eligible for discount. Full
Payment has not been made."

ELSE

CALL show message "Activate membership first to mark
attendance"

ELSE

CALL show error message "Member with this ID does not exist"

CATCH NumberFormatException **THEN**

CALL show message "ID can only have number values"

Method Description

This abstract class signifies a general gym member and establishes common attributes and actions that various kinds of gym members share. It includes data fields like ID, name, contact information, attendance records, loyalty points, and membership status, along with a range of methods for handling membership and showcasing member details.

Methods:

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

Constructor: Creates a new gym member instance using the given personal information, while initializing attendance, loyalty points, and active status to their default settings (0 or false).

Accessor (getter) methods:

getId(): returns the member's unique ID. getName(): Returns the member's name. getLoaction(): Returns the member's phone number. getEmail():

Returns the member's email address. getGender(): Returns the member's gender. getDOB(): Returns the member's date of birth.

getMembershipStartDate(): Returns the start date of the membership.

getAttendance(): Returns the number of times the member has attended

the gym. getLoyaltyPoints(): Returns the accumulated loyalty points of the

member. getActiveStatus(): Returns whether the membership is currently active (true or false).

Public abstract void markAttendance():

Abstract method: It should be implemented by subclasses. This method specifies the process for marking attendance for a member, usually increasing the attendance count and possibly adjusting loyalty points.

activateMembership():

Enables the membership by changing activeStatus to true.

deactive

deactivateMembership():

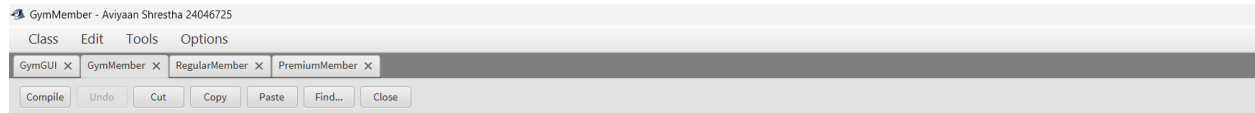
Withdraws the membership by changing activeStatus to false **IF** it is currently active; **IF** not, it displays a message stating that the membership is already deactivated.

resetMembership():

Deactivates the membership status and sets attendance and loyalty points back to zero.

Display():

Displays the details of the member, which consist of personal information, attendance records, loyalty points, and their current membership status on the console.



```
public abstract class GymMember
{
    protected int id;
    protected String name;
    protected String location;
    protected String phone;
    protected String email;
    protected String gender;
    protected String DOB;
    protected String membershipStartDate;
    protected int attendance;
    protected double loyaltyPoints;
    protected boolean activeStatus;

    // Constructor
    public GymMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate)
    {
        this.id = id;
        this.name = name;
        this.location = location;
        this.phone = phone;
        this.email = email;
        this.gender = gender;
        this.DOB = DOB;
        this.membershipStartDate = membershipStartDate;
        this.attendance = 0;
        this.loyaltyPoints = 0;
        this.activeStatus = false;
    }
}
```

```
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;
}

// Abstract method
public abstract void markAttendance();

```

```

public void activateMembership()
{
    this.activeStatus = true;
}

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

// Reset member
public void resetMembership()
{
    this.activeStatus = false;
    this.attendance = 0;
    this.loyaltyPoints = 0;
}

```

```

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

```

RegularMember

Constructor RegularMember()

Sets up a regular member with the default plan categorized as "basic" and a cost of 6500. Additionally, it establishes the referral source and initializes the remaining fields.

getAttendanceLimit()

Provides the highest attendance permitted before a member is eligible for an upgrade (30).

getIsEligibleForUpgrade()

Indicates **IF** the member qualifies to enhance their plan. (Please note: This field is not automatically refreshed based on current attendance in the existing code.)

getRemovalReason()

Provides the explanation for why a member was switched back to the basic plan.

getReferralSource()

Provides the source of referral by which the member became a part of the organization.

getPlan()

Provides the current membership option (for example, "basic" or "standard").

getPrice()

Provides the cost of the existing plan.

markAttendance()

Boosts attendance and awards 5 loyalty points for each visit.

getPlanPrice (String plan)

Provides the cost for a specified plan: basic (6500), standard (12500), deluxe (18500). Returns -1 IF the selected plan is not valid.

upgradePlan(String newPlan)

Attempts to switch to a different plan. IF the new plan is legitimate and attendance exceeds the threshold (30), the upgrade occurs, and the price is adjusted. Otherwise, an error message is issued.

revertRegularMember(String removalReason)

Reverts the member back to the standard "basic" plan, sets the price to 6500, removes upgrade eligibility, and records the reason for the change.

display() (Overridden)

Displays all details of members, including their plan and the associated price. IF there is a specified reason for removal, that will also be shown.

```
RegularMember - Aviyaan Shrestha 24046725
Class Edit Tools Options
GymGUI X GymMember X RegularMember X PremiumMember X
Compile Undo Cut Copy Paste Find... Close Source Code

public class RegularMember extends GymMember
{
    private final int attendanceLimit = 30;
    private boolean isEligibleForUpgrade;
    private String removalReason;
    private String referralSource;
    private String plan;
    private double price;

    // Constructor
    public RegularMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate, String referralSource)
    {
        super(id, name, location, phone, email, gender, DOB, membershipStartDate);
        this.referralSource = referralSource;
        this.isEligibleForUpgrade = false;
        this.plan = "basic";
        this.price = 6500;
        this.removalReason = "";
    }

    // Accessor methods
    public int getAttendanceLimit()
    {
        return attendanceLimit;
    }

    public boolean getIsEligibleForUpgrade()
    {
        return isEligibleForUpgrade;
    }

    public String getRemovalReason()
    {
        return removalReason;
    }

    public String getReferralSource()
    {
        return referralSource;
    }

    public String getPlan()
    {
        return plan;
    }

    public double getPrice()
    {
        return price;
    }

    // Implement abstract method
    @Override
    public void markAttendance()
    {
        this.attendance++;
        this.loyaltyPoints += 5.0;
    }
}
```

```

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

// Upgrade plan
public String upgradePlan(String newPlan)
{
    if (newPlan.equalsIgnoreCase(this.plan))
    {
        return "You are already subscribed to this plan.";
    }
    double newPrice = getPlanPrice(newPlan);
    if (newPrice == -1)
    {
        return "Invalid plan selected.";
    }

    if (this.attendance >= attendanceLimit)
    {
        this.plan = newPlan;
        this.price = newPrice;
        return "Plan upgraded successfully to " + newPlan + ".";
    }
    return "You are not eligible for an upgrade.";
}

```

```

public void revertRegularMember(String removalReason)
{
    super.resetMembership();
    this.isEligibleForUpgrade = false;
    this.plan = "basic";
    this.price = 6500;
    this.removalReason = removalReason;
}

// Display details
@Override
public void display()
{
    super.display();
    System.out.println("Plan: " + plan);
    System.out.println("Price: " + price);
    if (!removalReason.isEmpty())
    {
        System.out.println("Removal Reason: " + removalReason);
    }
}
}

```

PremiumMember

This class builds upon GymMember and signifies a premium-tier gym member with extra features and functions, including payment management, a personal trainer, and special discounts.

Constructor

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

It Constructs a PremiumMember instance by invoking the constructor of the superclass and assigning the personal trainer. Establishes default values: isFullPayment to false, and both paidAmount and discountAmount to 0.

getPremiumCharge()

```
public double getPremiumCharge()
```

Provides a constant premium membership fee of ₹50,000.

getPersonalTrainer()

```
public String getPersonalTrainer()
```

Provides the name of the assigned personal trainer for the member.

getIsFullPayment()

```
public boolean getIsFullPayment()
```

Returns true **IF** the complete payment has been made; otherwise, it returns false.

getPaidAmount()

```
public double getPaidAmount()
```

Calculates the total sum that the member has contributed up to this point.

getDiscountAmount()

```
public double getDiscountAmount()
```

Provides the discount value applied, **IF** the criteria for eligibility are met.

markAttendance()

@Override

```
public void markAttendance()
```

Increments the attendance count and adds 10 loyalty points for each check-in. Overrides the method from GymMember.

payDueAmount(double amount)

```
public String payDueAmount(double amount)
```

Enables the member to submit a payment for the premium fee. Prevents excessive payments and sets isFullPayment to true **IF** the total payment equals the premiumCharge. Provides a message regarding the payment status and any outstanding balance.

calculateDiscount()

```
public void calculateDiscount()
```

IF the total payment has been made, applies a 10% reduction on the premium fee. Shows the discount amount or a notification **IF** not eligible.

revertPremiumMember()

```
public void revertPremiumMember()
```

Resets the premium member's data to default: Calls resetMembership() from GymMember. Restores the premium member's data to its original state: Invokes resetMembership() from GymMember. Removes personal

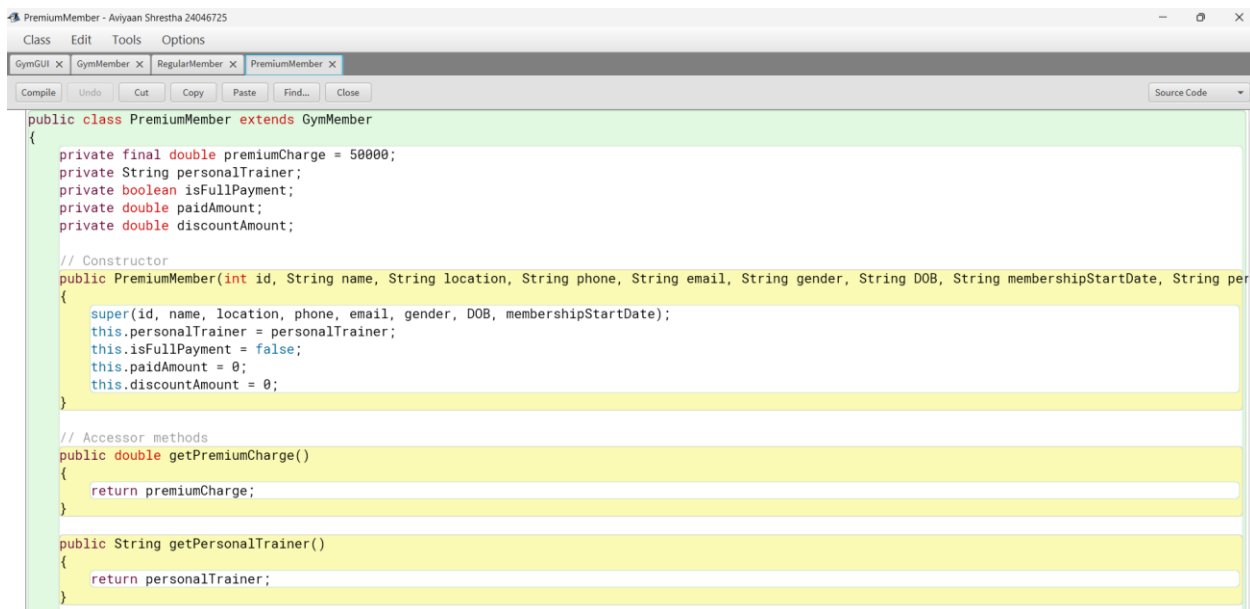
trainer, payment status, and discount. personal trainer, payment status, and discount.

display()

@Override

public void display()

Presents details about the member such as their personal trainer, payment condition, remaining balance, and any applicable discounts. Enhances the display() method from the base class.



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

    // Constructor
    public PremiumMember(int id, String name, String location, String phone, String email, String gender, String DOB, String membershipStartDate, String per
    {
        super(id, name, location, phone, email, gender, DOB, membershipStartDate);
        this.personalTrainer = personalTrainer;
        this.isFullPayment = false;
        this.paidAmount = 0;
        this.discountAmount = 0;
    }

    // Accessor methods
    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  
public void markAttendance()  
{  
    this.attendance++;  
    this.loyaltyPoints += 10.0;  
}
```

```
public String payDueAmount(double amount)
{
    if (isFullPayment)
    {
        return "Payment is already complete.";
    }
    if (paidAmount + amount > premiumCharge)
    {
        return "Payment exceeds the premium charge.";
    }
    paidAmount += amount;
    if (paidAmount == premiumCharge)
    {
        isFullPayment = true;
    }
    double remainingAmount = premiumCharge - paidAmount;
    return "Payment successful. Remaining amount: " + remainingAmount;
}

// Calculate discount
public void calculateDiscount()
{
    if (isFullPayment)
    {
        discountAmount = premiumCharge * 0.10;
        System.out.println("Discount calculated: " + discountAmount);
    } else
    {
        System.out.println("No discount available.");
    }
}
```



```
public void revertPremiumMember()  
{  
    super.resetMembership();  
    this.personalTrainer = "";  
    this.isFullPayment = false;  
    this.paidAmount = 0;  
    this.discountAmount = 0;  
}
```

```
// Display details
```

```
@Override
```

```
public void display()  
{
```

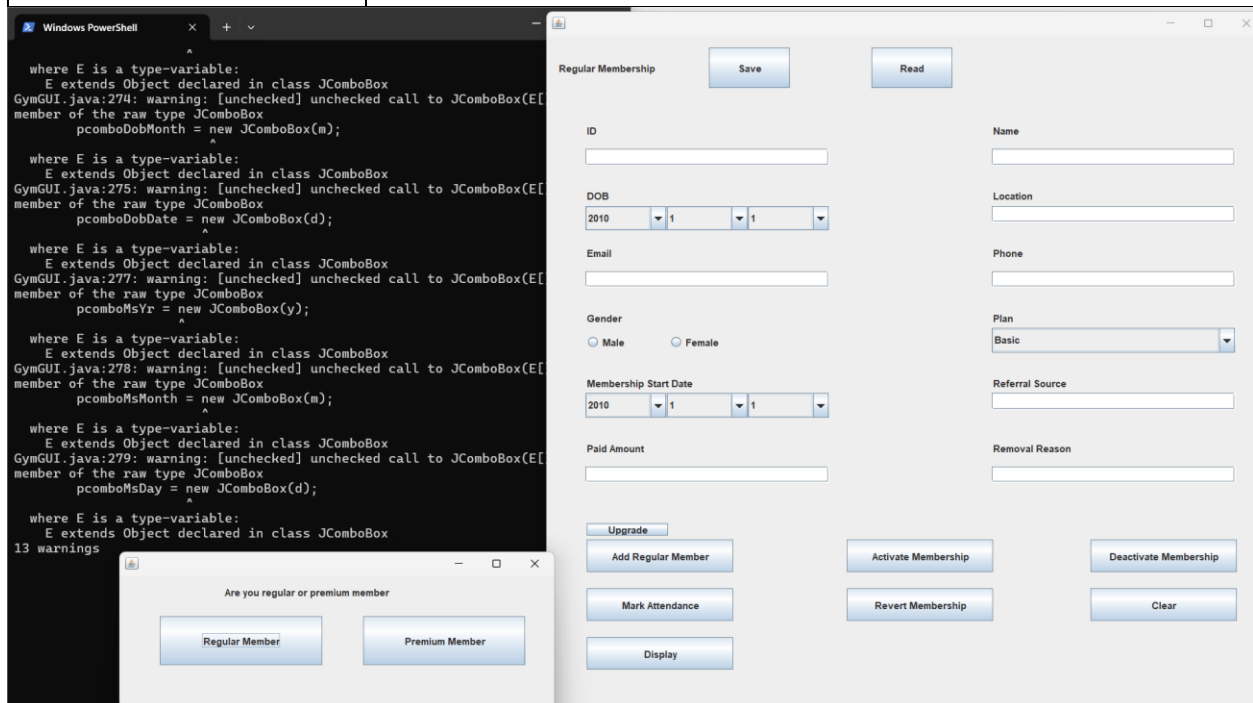
```
    super.display();  
    System.out.println("Personal Trainer: " + personalTrainer);  
    System.out.println("Paid Amount: " + paidAmount);  
    System.out.println("Full Payment: " + isFullPayment);  
    System.out.println("Remaining Amount: " + (premiumCharge - paidAmount));  
    if (isFullPayment)  
    {  
        System.out.println("Discount Amount: " + discountAmount);  
    }  
}
```

```
}
```

Testing

Test 1

Objective	To compile and run the program in terminal.
Action	Write “javac GymGUI” and “java GymGUI.java” in terminal.
Expected output	The option to choose for either regular membership or premium membership appears and when clicked on their respective GUI opens.
Actual output	Option to chose appeared and when chosen one, it’s respective frame appeared.
Result	Test successful.



Test 2.1

Objective	To add regular member.
Action	1. Fill in the application In correct manner and click “add regular member” button.
Expected output	The regular member details should be added and a dialog box showcasing the message with appear.
Actual output	The details were added and the dialog box appeared with a success message.
Result	Test successful.

The screenshot shows a web application window titled 'Regular Membership'. The form contains the following fields and controls:

- ID**: Text input field.
- Name**: Text input field.
- DOB**: Date of Birth field with dropdowns for Year (2010), Month (1), and Day (1).
- Location**: Text input field.
- Email**: Text input field.
- Phone**: Text input field.
- Gender**: Radio buttons for Male and Female.
- Membership Start Date**: Date field with dropdowns for Year (2010), Month (1), and Day (1).
- Referral Source**: Text input field.
- Paid Amount**: Text input field.
- Removal Reason**: Text input field.

Buttons at the top: **Save** and **Read**.

Buttons at the bottom: **Upgrade**, **Add Regular Member**, **Activate Membership**, **Deactivate Membership**, **Mark Attendance**, **Revert Membership**, **Clear**, and **Display**.

A modal dialog box titled 'Message' is displayed in the center, containing an information icon and the text 'please fill in the application', with an **OK** button.

Regular Membership

Save

Read

ID

a

DOB

2010

▼

1

▼

1

▼

Email

a

Gender

☐ Male

☒ Female

Membership Start Date

2010

▼

1

▼

1

▼

Paid Amount

Upgrade

Add Regular Member

Mark Attendance

Display

Name

a

Location

a

Phone

a

Plan

Basic

▼

Referral Source

Removal Reason


Activate Membership

Revert Membership

Deactivate Membership

Clear

Error

 Please input number for ID

OK

Regular Membership

Save

Read

ID

1

DOB

2010

▼

1

▼

1

▼

Email

a

Gender

☐ Male

☒ Female

Membership Start Date

2010

▼

1

▼

1

▼

Paid Amount

Name

a

Location

a

Phone

a

Referral Source

Removal Reason

Upgrade

Add Regular Member

Mark Attendance

Display

Activate Membership

Revert Membership

Deactivate Membership

Clear

Message

i

Regular Member added

OK

Test 2.2

Objective	To add premium member.
Action	1. Fill in the application in the correct manner and click “add premium member button”
Expected output	The premium member details will be added and a dialog box showcasing the message will appear.
Actual output	The details were added and the dialog box appeared with a success message.
Result	Test successful.

The screenshot shows a web application window titled "Premium Membership". At the top, there are "Save" and "Read" buttons. The form contains several input fields: "ID", "Name", "DOB" (with dropdowns for year, month, and day), "Email", "Location", "Phone", "Gender" (with radio buttons for Male and Female), "Membership Start Date" (with dropdowns for year, month, and day), "Trainer's Name", "Referral Source", "Paid Amount", and "Removal Reason". A modal dialog box with the title "Message" and an information icon is displayed in the center, containing the text "please fill in the application" and an "OK" button. At the bottom of the form, there are nine buttons arranged in a 3x3 grid: "Add Premium Member", "Activate Membership", "Deactivate Membership", "Mark Attendance", "Revert Membership", "Clear", "Display", "Calculate Discount", and "Pay Due Amount".

Premium Membership

SaveRead

ID

a

DOB

2010

▼

1

▼

1

▼

Email

a

Gender

☐ Male

☐ Female

Membership Start Date

2010

▼

1

▼

1

▼

Paid Amount

Name

aa

Location

a

Phone


a

Trainer's Name

Referral Source

Removal Reason

Error

 Please input number for ID

OK

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Premium Membership

SaveRead

ID

1

Name

aa

DOB

2010

▼

1

▼

1

▼

Location

a

Email

a

Phone

a

Gender

Male

Female

Trainer's Name

Membership Start Date

2010

▼

1

▼

1

▼

Referral Source

Paid Amount

Removal Reason

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Duplication

✖

please choose another ID.

OK

Premium Membership

Save

Read

ID

2

DOB

2010

▼

1

▼

1

▼

Email

a

Gender

☐ Male

☐ Female

Membership Start Date

2010

▼

1

▼

1

▼

Paid Amount

Name

aa

Location

a

Phone

a

Trainer's Name

Referral Source

Removal Reason

Message

Premium Member added

OK

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Test 3.1

Objective	To increase mark attendance.
Action	1. Activate membership. 2. Click “mark attendance button again”.
Expected output	The premium member details will be added and a dialog box showcasing the message will appear.
Actual output	The details were added and the dialog box appeared with a success message.
Result	Test successful.

The screenshot shows a web application for managing premium memberships. The main form is titled "Premium Membership" and includes fields for ID, Name, DOB, Location, Email, Phone, Gender, Membership Start Date, Paid Amount, Removal Reason, Trainer's Name, and Referral Source. There are "Save" and "Read" buttons at the top. At the bottom, there are several action buttons: "Add Premium Member", "Activate Membership", "Deactivate Membership", "Mark Attendance", "Revert Membership", "Clear", "Display", "Calculate Discount", and "Pay Due Amount". An "Attendance" dialog box is open in the center, displaying a red "X" icon and the message "member is not activate", with an "OK" button.

Premium Membership

Save Read

ID: 2

Name: aa

DOB: 2010 1 1

Location: a

Email: a

Phone: a

Gender: ☐ Male ☐ Female

Membership Start Date: 2010 1 1

Paid Amount:

Removal Reason:

Trainer's Name:

Referral Source:

Attendance

member is not activate

OK

Add Premium Member Activate Membership Deactivate Membership

Mark Attendance Revert Membership Clear

Display Calculate Discount Pay Due Amount

Premium Membership

Save

Read

ID

2

Name

aa

DOB

2010

▼

1

▼

1

▼

Location

a

Email

a

Phone

a

Gender

Male

Female

Trainer's Name

Membership Start Date

2010

▼

1

▼

1

▼

Referral Source

Paid Amount

Removal Reason

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Activated

i

Membership activated

OK

Premium Membership

Save

Read

ID

2

DOB

2010

▼

1

▼

1

▼

Email

a

Gender

☐ Male

☐ Female

Membership Start Date

2010

▼

1

▼

1

▼

Paid Amount

Name

aa

Location

a

Phone

a

Trainer's Name

Referral Source

Removal Reason

Attendance

i

Attendance marked

OK

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Test 4.1

Objective	To calculate the discount.
Action	1. Pay full amount. 2. Click” calculate discount” button.
Expected output	A message dialog box stating that member has received 10% discount should be displayed.
Actual output	A message dialog box stating that member has received 10% discount is displayed.
Result	Test successful.

The screenshot displays a web application for managing premium memberships. At the top, there are 'Save' and 'Read' buttons. The main form contains fields for ID (1), Name (a), DOB (2010, 1, 1), Location (a), Email (a), Phone (a), Gender (Male selected), Membership Start Date (2010, 1), Paid Amount, and Removal Reason. A modal message box is open in the center, displaying an information icon and the text: 'Member not eligible for discount. Full Payment has not been made.' with an 'OK' button. At the bottom, there are several action buttons: 'Add Premium Member', 'Activate Membership', 'Deactivate Membership', 'Mark Attendance', 'Revert Membership', 'Clear', 'Display', 'Calculate Discount', and 'Pay Due Amount'.

Premium Membership

Save

Read

ID

1

Name

a

DOB

2010

▼

1

▼

1

▼

Location

a

Email

a

Phone

a

Gender

☒ Male

☐ Female

Trainer's Name

Membership Start Date

2010

▼

1

▼

1

▼

Source

Paid Amount

40000

Removal Reason

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Message

Payment successful. Remaining amount: 0.0

OK

Premium Membership
Save
Read

ID
1
Name
a

DOB
2010 1 1
Location
a

Email
a
Phone
a

Gender
☒ Male ☐ Female

Membership Start Date
2010 1

Paid Amount
40000
Removal Reason

Add Premium Member
Activate Membership
Deactivate Membership

Mark Attendance
Revert Membership
Clear

Display
Calculate Discount
Pay Due Amount

Message
The member a has successfully received 10% Discount
OK

PREMIUM MEMBERS:

Member ID: 1
Name: a
Location: a
Email: a
Phone: a
Date of Birth: 1-1-2010
Gender: male
Membership Start Date: 1-1-2010
Attendance: 0
Loyalty Points: 0.0
Personal Trainer: a
Payment of Rs.50000.0 has been made
Active Status: true
Full payment has been done
Discount: Rs.5000.0
=====

Test 4.2

Objective	To pay due amount.
Action	Fill the id, enter paid amount and click “pay”.
Expected output	A message dialog box stating the remaining amount which needs to be paid should be displayed.
Actual output	A message dialog box stating the remaining amount which needs to be paid is displayed.
Result	Test successful.

The screenshot displays a web application for managing premium memberships. At the top, there are 'Save' and 'Read' buttons. The form contains several input fields: ID (with value '2'), Name (with value 'aa'), DOB (with year '2010' and month/day '1'), Location (with value 'a'), Email (with value 'a'), Phone (with value 'a'), Gender (radio buttons for Male and Female), Membership Start Date (with year '2010' and month/day '1'), Trainer's Name, Referral Source, Paid Amount, and Removal Reason. A modal message box is centered on the screen, displaying an information icon, the text 'Premium Member added', and an 'OK' button. At the bottom, there are nine buttons arranged in a 3x3 grid: 'Add Premium Member', 'Activate Membership', 'Deactivate Membership', 'Mark Attendance', 'Revert Membership', 'Clear', 'Display', 'Calculate Discount', and 'Pay Due Amount'.

Premium Membership

SaveRead

ID

2

Name

aa

DOB

201011

Location

a

Email

a

Phone

a

Gender

☐ Male☐ Female

Trainer's Name

Membership Start Date

201011

Referral Source

Paid Amount

Removal Reason

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Attendance

member is not activate

OK

Premium Membership

SaveRead

ID

1

Name

a

DOB

201011

Location

a

Email

a

Phone

a

Gender

☒ Male☐ Female

Trainer's Name

Membership Start Date

201011

Referral Source

Paid Amount

10000

Removal Reason

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Activated

Membership activated

OK

Premium Membership

Save

Read

ID

1

Name

a

DOB

2010

1

1

Location

a

Email

a

Phone

a

Gender

Male

Female

Membership Start Date

2010

1

1

Trainer's Name

Source

Paid Amount

10000

Removal Reason

Message

Payment successful. Remaining amount: 40000.0

OK

Add Premium Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Calculate Discount

Pay Due Amount

Test 4.3

Objective	To revert membership.
Action	Fill in the id, removal reason and click revert membership button
Expected output	A message dialog box stating the membership is reverted should be displayed.
Actual output	A message dialog box stating the membership is reverted is displayed.
Result	Test successful.

The screenshot displays a web application interface for managing memberships. At the top, there's a 'Regular Membership' section with 'Save' and 'Read' buttons. Below this, the form is organized into two columns. The left column contains fields for 'ID' (text input with '1'), 'DOB' (date picker set to 2010-1-1), 'Email' (text input with 'a'), 'Gender' (radio buttons for 'Male' and 'Female'), 'Membership Start Date' (date picker set to 2010-1-1), 'Paid Amount' (text input), and 'Upgrade' button. The right column contains fields for 'Name' (text input with 'a'), 'Location' (text input with 'a'), 'Phone' (text input with 'a'), 'Referral Source' (text input), and 'Removal Reason' (text input). At the bottom, there are several buttons: 'Add Regular Member', 'Activate Membership', 'Deactivate Membership', 'Mark Attendance', 'Revert Membership', 'Clear', and 'Display'. A 'Message' dialog box is overlaid in the center, displaying an information icon, the text 'Regular Member added', and an 'OK' button.

REGULAR MEMBERS:

Member ID: 1

Name: a

Location: a

Email: a

Phone: a

Date of Birth: 1-1-2010

Gender:

Plan: basic

Price: 6500.0

Membership Start Date: 1-1-2010

Attendance: 8

Loyalty Points: 40.0

Active Status: true

Regular Membership

SaveRead

ID

1

Name

a

DOB

201011

Location

a

Email

a

Phone

a

Gender

☐ Male☐ Female

Plan

basic

Membership Start Date

201011

Referral Source

Paid Amount

Removal Reason

Upgrade

Add Regular Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Message

please fill in the removal reason

OK

Regular Membership
Save
Read

ID

1

Name

a

DOB

2010

1

1

Location

a

Email

a

Phone

a

Gender

☐ Male
☐ Female

Membership Start Date

2010

1

1

Referral Source

Paid Amount

Removal Reason

shdyu

Upgrade

Add Regular Member

Activate Membership

Deactivate Membership

Mark Attendance

Revert Membership

Clear

Display

Message
X

i
Membership Reset

OK

REGULAR MEMBERS:

Member ID: 1

Name: a

Location: a

Email: a

Phone: a

Date of Birth: 1-1-2010

Gender:

Plan: basic

Price: 6500.0

Membership Start Date: 1-1-2010

Attendance: 0

Loyalty Points: 0.0

Active Status: false

=====

Test 5.1

Objective	To save the file.
Action	1. Fill in the application, add premium member 2. Click “save” button.
Expected output	A message dialog stating the file is saved should be displayed.
Actual output	A message dialog stating the file is saved is displayed.
Result	Test successful.

The screenshot shows a web application window titled "Premium Membership". At the top, there are "Save" and "Read" buttons. The form contains several input fields and dropdown menus: ID (text box with "2"), Name (text box with "aa"), DOB (three dropdowns for year, month, and day, all set to "1"), Location (text box with "a"), Email (text box with "a"), Phone (text box with "a"), Gender (radio buttons for "Male" and "Female"), Membership Start Date (three dropdowns for year, month, and day, all set to "1"), Trainer's Name (text box), Referral Source (text box), Paid Amount (text box), and Removal Reason (text box). A modal message dialog is displayed in the center, titled "Message", with an information icon and the text "Premium Member added". Below the form, there are nine buttons arranged in a 3x3 grid: "Add Premium Member", "Activate Membership", "Deactivate Membership", "Mark Attendance", "Revert Membership", "Clear", "Display", "Calculate Discount", and "Pay Due Amount".

Premium Membership
Save
Read

ID
1

DOB
2010
1
1

Email
a

Gender
☒ Male
☐ Female

Membership Start Date
2010
1
1

Paid Amount
40000

Name
a

Location
a

Phone
a

Trainer's Name

Referral Source

Removal Reason

Add Premium Member
Activate Membership
Deactivate Membership
Mark Attendance
Revert Membership
Clear
Display
Calculate Discount
Pay Due Amount

Saved
X

i
Details Saved To File Successfully!!

OK

Class

Edit

Tools

Options

GymGUI X

MemberDetails.txt X

Compile

Undo

Cut

Copy

Paste

Find...

Close

ID	Attendance	Name	Location	Phone	Email	Gender	DOB	Membership Start	Loyalty Pts	Status	Plan	Price	Trainer	Paid Amt
1	0	a	a	a	a	male	1-1-2010	1-1-2010	0.00	Active	NA	Rs. 50000.0	a	Rs. 50000.0

Test 5.2

Objective	To read from file.
Action	1. Fill in the application and save. 2. Click” read” button.
Expected output	A frame should be opened where information will be stored.
Actual output	A frame should be opened where information will be stored.
Result	Test successful.

PREMIUM MEMBERS:

Member ID: 1
Name: a
Location: a
Email: a
Phone: a
Date of Birth: 1-1-2010
Gender: male
Membership Start Date: 1-1-2010
Attendance: 0
Loyalty Points: 0.0
Personal Trainer: a
Payment of Rs.50000.0 has been made
Active Status: true
Full payment has been done
Discount: Rs.5000.0

-----Regular and Premium Member Details

ID	Attendance	Name	Location	Phone	Email	Gender	DOB	Membership Start	Loyalty Pts	Status	Plan	Price	Trainer	Paid Amt
1	0	a	a	a	a	male	1-1-2010	0.00	Active	NA	Rs.50000.0	a	Rs.50000.0	

Error detection and correction

Errors are the bugs which halts the program and doesn't let it execute as per the need of the user. These bugs can be of any type but these are mainly divided into 3 types, syntax error, logical error and runtime error.

Syntax error

The programming language has grammar the same way our speaking language does. Errors in these grammar as known as syntax errors. These errors can occur by missing a semicolon or a bracket. It can also occur when the spelling of keywords is wrong. These errors are found out at compile time. These errors do not let the program to be executed.

Logical error

These errors occur mainly due to human error. It does not generate the expected output, but it doesn't necessarily halt the program. This type of error is the hardest to figure out and correct. These errors do not halt the program, it just does not generate needed output. One of the logical errors in this coursework is in premium member class where the calculate discount function doesn't return correct discount as one of the part of equations should be float. It was solved by adding .0 behind 10 in 10/100.

Runtime error

These errors are found out during execution. It is not detected during compile time. These errors causes the program to crash during execution. These have correct syntax however the program crashes during execution. In this coursework, these types of errors are handled through **TRY CATCH** statements like file not found, number format exception and many more.

Conclusion

This coursework utilizes the java concepts to form a gym system where user can add member either regular or premium. They can mark attendance, activate membership, deactivate membership, revert membership, upgrade plan **IF** they are regular member, pay and calculate discount **IF** they are premium member. This coursework contains the use of GUI components too. It has used various components such as JLabel, JFrame, JTextField, JTextArea, and many more which is arranged in a way to develop a user friendly interface. It even save the member information in text file and even displays the information by reading from the text file.

This coursework contained many difficulties ranging from syntax error to button not working properly as per the need. These difficulties created a lot of stress during the coursework submission. The errors were logical or runtime. It was difficult to utilize the concept studied theoretically in a program. However, with great difficulties, these programs were completed. These errors were solved by researching about it or BlueJ itself suggesting fixes. These errors once handled ran the program smoothly as per the need.

I gained knowledge on various topics and concepts, which I only knew a very little to non about. This coursework gave me an experience about how a real-world concept and tasks would actually feel like. It was a not so fun time doing this coursework, as many difficulties were found and it only increased along the project. Many runtime and some logical errors were discovered along the way. But regardless of the problems that were presented throughout the time period this projected is completed and I gained a very immense and good knowledge about various concepts in java.

Appendix

```
public abstract class GymMember
{
    protected int id;
    protected String name;
    protected String location;
    protected String phone;
    protected String email;
    protected String gender;
    protected String DOB;
    protected String membershipStartDate;
    protected int attendance;
    protected double loyaltyPoints;
    protected boolean activeStatus;

    public GymMember(int id, String name, String location, String phone,
String email, String gender, String DOB, String membershipStartDate)
    {
        this.id = id;
        this.name = name;
        this.location = location;
        this.phone = phone;
        this.email = email;
        this.gender = gender;
        this.DOB = DOB;
        this.membershipStartDate = membershipStartDate;
```

```
    this.attendance = 0;  
    this.loyaltyPoints = 0;  
    this.activeStatus = false;  
}
```

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

```
public void activateMembership()
{
    this.activeStatus = true;
}
```

```
public void deactivateMembership()
{
    if (this.activeStatus)
    {
        this.activeStatus = false;
    }
    else
    {
        System.out.println("Membership is already deactivated.");
    }
}
```

```
public void resetMembership()
```

```

{
    this.activeStatus = false;
    this.attendance = 0;
    this.loyaltyPoints = 0;
}

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

public class RegularMember extends GymMember
{
    private final int attendanceLimit = 30;
    private boolean isEligibleForUpgrade;
    private String removalReason;

```

```
private String referralSource;  
private String plan;  
private double price;
```

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

```
{  
    super(id, name, location, phone, email, gender, DOB,  
membershipStartDate);  
    this.referralSource = referralSource;  
    this.isEligibleForUpgrade = false;  
    this.plan = "basic";  
    this.price = 6500;  
    this.removalReason = "";  
}
```

```
public int getAttendanceLimit()  
{  
    return attendanceLimit;  
}
```

```
public boolean getIsEligibleForUpgrade()  
{  
    return isEligibleForUpgrade;  
}
```



```
public String getRemovalReason()
{
    return removalReason;
}
```

```
public String getReferralSource()
{
    return referralSource;
}
```

```
public String getPlan()
{
    return plan;
}
```

```
public double getPrice()
{
    return price;
}
```

@Override

```
public void markAttendance()
{
    this.attendance++;
    this.loyaltyPoints += 5.0;
}
```

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

```
public String upgradePlan(String newPlan)
{
    if (newPlan.equalsIgnoreCase(this.plan))
    {
        return "You are already subscribed to this plan.";
    }
    double newPrice = getPlanPrice(newPlan);
    if (newPrice == -1)
    {
        return "Invalid plan selected.";
    }
}
```

```
if (this.attendance >= attendanceLimit)
{
    this.plan = newPlan;
    this.price = newPrice;
}
```

```
        return "Plan upgraded successfully to " + newPlan + ".";
    }
    return "You are not eligible for an upgrade.";
}
```

```
public void revertRegularMember(String removalReason)
{
    super.resetMembership();
    this.isEligibleForUpgrade = false;
    this.plan = "basic";
    this.price = 6500;
    this.removalReason = removalReason;
}
```

```
@Override
public void display()
{
    super.display();
    System.out.println("Plan: " + plan);
    System.out.println("Price: " + price);
    if (!removalReason.isEmpty())
    {
        System.out.println("Removal Reason: " + removalReason);
    }
}
}

public class PremiumMember extends GymMember
```

```
{
    private final double premiumCharge = 50000;
    private String personalTrainer;
    private boolean isFullPayment;
    private double paidAmount;
    private double discountAmount;

    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;
    }

    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
```

```
public void markAttendance()
```

```
{
```

```
    this.attendance++;
```

```
    this.loyaltyPoints += 10.0;
```

```
}
```

```
public String payDueAmount(double amount)
```

```
{
```

```
    if (isFullPayment)
```

```
{
    return "Payment is already complete.";
}
if (paidAmount + amount > premiumCharge)
{
    return "Payment exceeds the premium charge.";
}
paidAmount += amount;
if (paidAmount == premiumCharge)
{
    isFullPayment = true;
}
double remainingAmount = premiumCharge - paidAmount;
return "Payment successful. Remaining amount: " +
remainingAmount;
}

public void calculateDiscount()
{
    if (isFullPayment)
    {
        discountAmount = premiumCharge * 0.10;
        System.out.println("Discount calculated: " + discountAmount);
    } else
    {
        System.out.println("No discount available.");
    }
}
```

```
}
```

```
public void revertPremiumMember()
```

```
{
```

```
    super.resetMembership();
```

```
    this.personalTrainer = "";
```

```
    this.isFullPayment = false;
```

```
    this.paidAmount = 0;
```

```
    this.discountAmount = 0;
```

```
}
```

```
@Override
```

```
public void display()
```

```
{
```

```
    super.display();
```

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

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

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

```
    System.out.println("Remaining Amount: " + (premiumCharge -  
paidAmount));
```

```
    if (isFullPayment)
```

```
    {
```

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

```
    }
```

```
}
```

```
}
```

```
import java.util.ArrayList;
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JComboBox;
import javax.swing.JRadioButton;
import javax.swing.JButton;
import javax.swing.ButtonGroup;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JOptionPane;
import javax.swing.JTextArea;
import java.io.FileWriter;
import java.io.File;
import java.io.FileReader;
import javax.swing.JScrollPane;
```

```
public class GymGUI implements ActionListener
```

```
{
```

```
    ArrayList<GymMember> array = new ArrayList<GymMember>();
```

```
    public static void main(String[] args)
```

```
    {
```

```
        new GymGUI();
```

```
    }
```

```
    JFrame option, regfr, prefr;
```

```
    JLabel labelid, labelname, labellocation, labeldob, labelemail,
```

```
    labelphone, labelgender, labelplan, labelmemberships,
```

```
    labelreferral, labelpaid, labelremoval, labeltrainer, labelprice,
```


labeldiscount,title,regular,premium;

JLabel plabelid, plabelname, plabellocation, plabeldob, plabelemail,
plabelphone, plabelgender, plabelplan, plabelmemberships,
plabelreferral, plabelpaid, plabelremoval,plabeltrainer, plabelprice,
plabeldiscount, plablecalculateddiscount;

JTextField tfid,tflocation,tfemail, tfpaid, tfname, tfphone,
tfreferral,tfremoval;

JTextField tfidp,tflocationp,tfemailp, tfpaidp, tfnamep, tfphonep,
tfreferralp,tfremovalp, tftrainerp;

JComboBox comboDobYr,comboDobMonth,comboDobDate, plancb,
comboMsYr,comboMsMonth, comboMsDay;

JRadioButton malerd, femalerd;

ButtonGroup grp, bg;

JComboBox pcomboDobYr,pcomboDobMonth,pcomboDobDate,
pplancb, pcomboMsYr,pcomboMsMonth, pcomboMsDay;

JRadioButton pmalerd, pfemalerd;

JButton pre,reg;

JButton addreg, addpre, activate, deactivate, attendance,revert, clear,
display;

JButton pactivate, pdeactivate, pattendance,prevert, pclear, pdisplay,
pcalculateddiscount, btnUpgrade;

```
JButton RSave, RRead;
JButton PSave, PRead;
JButton CalcDiscount, PayDue;
JFrame displayframe;
JTextArea displayta;
public GymGUI()
{
    //option gui
    option = new JFrame("");
    title = new JLabel("Are you regular or premium member");
    reg = new JButton("Regular Member");
    pre = new JButton("Premium Member");

    title.setBounds(130,5,250,30);
    reg.setBounds(50,50,200,60);
    pre.setBounds(300,50,200,60);

    option.add(title);
    option.add(reg);
    option.add(pre);

    option.setLayout(null);
    option.setVisible(true);
    option.setSize(550,200);

    //application member gui
```

```
regfr = new JFrame("");
```

```
regular = new JLabel("Regular Membership");
```

```
RSave = new JButton("Save");
```

```
RSave.setBounds(200,15,100,50);
```

```
RRead = new JButton("Read");
```

```
RRead.setBounds(400,15,100,50);
```

```
regfr.add(RSave);
```

```
regfr.add(RRead);
```

```
labelid = new JLabel("ID");
```

```
labelname = new JLabel("Name");
```

```
labellocation = new JLabel("Location");
```

```
labeldob = new JLabel("DOB");
```

```
labelemail = new JLabel("Email");
```

```
labelphone = new JLabel("Phone");
```

```
labelgender = new JLabel("Gender");
```

```
labelplan = new JLabel("Plan");
```

```
labelmembersd = new JLabel("Membership Start Date");
```

```
labelreferral = new JLabel("Referral Source");
```

```
labelpaid = new JLabel("Paid Amount");
```

```
labelremoval = new JLabel("Removal Reason");
```

```
labeltrainer = new JLabel("Trainer's Name");
```

```
labelprice = new JLabel("Plan Price");
```

```
labeldiscount = new JLabel("Discount Amount");
```

```
tfid = new JTextField();
```

```
tflocation = new JTextField();
tfemail = new JTextField();
tfpaid = new JTextField();
tfname = new JTextField();
tfphone = new JTextField();
tfreferral = new JTextField();
tfremoval = new JTextField();
```

```
String y[]={"2010","2011","2012","2013","2014","2015","2016"};
String m[]={"1","2","3","4","5","6","7","8","9","10","11","12"};
String d[]={"1","2","3","4","5","6","7","8","9","10","11","12",
            "13","14","15","16","17","18","19","20","21","22","23","24",
            "25","26","27","28","29","30","31"};
String p[]= {"Basic","Standard","Deluxe"};
comboDobYr = new JComboBox(y);
comboDobMonth = new JComboBox(m);
comboDobDate = new JComboBox(d);
planCb = new JComboBox(p);
comboMsYr = new JComboBox(y);
comboMsMonth = new JComboBox(m);
comboMsDay = new JComboBox(d);
```

```
malerd = new JRadioButton("Male");
femalerd = new JRadioButton("Female");
grp = new ButtonGroup();
```

```
addreg = new JButton("Add Regular Member");
```

```
addpre = new JButton("Add Premium Member");
activate = new JButton("Activate Membership");
deactivate = new JButton("Deactivate Membership");
attendance = new JButton("Mark Attendance");
revert = new JButton("Revert Membership");
clear = new JButton("Clear");
display = new JButton("Display");
btnUpgrade = new JButton("Upgrade");
btnUpgrade.setBounds(50,600,100,15);
```

```
regular.setBounds(15,15,250,50);
```

```
labelid.setBounds(50,110,100,15);
labeldob.setBounds(50,190,100,15);
labelemail.setBounds(50,260,100,15);
labelgender.setBounds(50,340,100,15);
labelmembersd.setBounds(50,420,150,15);
labelpaid.setBounds(50,500,100,15);
```

```
labelname.setBounds(550,110,100,15);
labellocation.setBounds(550,190,100,15);
labelphone.setBounds(550,260,100,15);
labelplan.setBounds(550,340,100,15);
labelreferral.setBounds(550,420,100,15);
labelremoval.setBounds(550,500,150,15);
```

```
addreg.setBounds(50,620,180,40);
```

```
activate.setBounds(370,620,180,40);  
deactivate.setBounds(670,620,180,40);  
attendance.setBounds(50,680,180,40);  
revert.setBounds(370,680,180,40);  
clear.setBounds(670,680,180,40);  
display.setBounds(50,740,180,40);
```

```
tfid.setBounds(48,140,300,20);
```

```
tfemail.setBounds(48,290,300,20);  
tfpaid.setBounds(48,530,300,20);
```

```
tfname.setBounds(548,140,300,20);  
tflocation.setBounds(548,210,300,20);  
tfphone.setBounds(548,290,300,20);  
tfreferral.setBounds(548,440,300,20);  
tfremoval.setBounds(548,530,300,20);
```

```
malerd.setBounds(48,370,100,15);  
femalerd.setBounds(150,370,100,15);
```

```
plancb.setBounds(548,360,300,30);  
comboDobYr.setBounds(48,210,100,30);  
comboDobMonth.setBounds(148,210,100,30);  
comboDobDate.setBounds(248,210,100,30);  
comboMsYr.setBounds(48,440,100,30);  
comboMsMonth.setBounds(148,440,100,30);
```

```
comboMsDay.setBounds(248,440,100,30);
```

```
regfr.add(regular);  
regfr.add(labelid);  
regfr.add(labelname);  
regfr.add(labellocation);  
regfr.add(labelemail);  
regfr.add(labelphone);  
regfr.add(labelgender);  
regfr.add(labelplan);  
regfr.add(labeldob);  
regfr.add(labelmembershpsd);  
regfr.add(labelreferral);  
regfr.add(labelpaid);  
regfr.add(labelremoval);
```

```
regfr.add(labelprice);  
regfr.add(labeldiscount);  
regfr.add(tfid);  
regfr.add(tflocation);  
regfr.add(tfemail);  
regfr.add(tfpaid);  
regfr.add(tfname);  
regfr.add(tfphone);  
regfr.add(tfreferral);  
regfr.add(tfremoval);  
regfr.add(btnUpgrade);
```

```
regfr.add(malerd);
regfr.add(femalerd);
regfr.add(comboDobYr);
regfr.add(comboDobMonth);
regfr.add(comboDobDate);
regfr.add(plancb);
regfr.add(comboMsYr);
regfr.add(comboMsMonth);
regfr.add(comboMsDay);
regfr.add(addrég);
regfr.add(attendance);
regfr.add(activate);
regfr.add(deactivate);
regfr.add(clear);
regfr.add(display);
regfr.add(revert);
grp.add(malerd);
grp.add(femalerd);
```

```
regfr.setLayout(null);
regfr.setVisible(false);
regfr.setSize(900,1000);
```

```
//premium frame
```

```
prefr = new JFrame("");
```



```
premium = new JLabel("Premium Membership");
plabelid = new JLabel("ID");
plabelname = new JLabel("Name");
plabellocation = new JLabel("Location");
plabeldob = new JLabel("DOB");
plabelemail = new JLabel("Email");
plabelphone = new JLabel("Phone");
plabelgender = new JLabel("Gender");
plabelplan = new JLabel("Plan");
plabelmembershipsdate = new JLabel("Membership Start Date");
plabelreferral = new JLabel("Referral Source");
plabelpaid = new JLabel("Paid Amount");
plabelremoval = new JLabel("Removal Reason");
plabeltrainer = new JLabel("Trainer's Name");
plabelprice = new JLabel("Plan Price");
plabeldiscount = new JLabel("Discount Amount");
```

```
pcalculateddiscount = new JButton("Calculate Discount");
PayDue = new JButton("Pay Due Amount");
PayDue.setBounds(670,740,180,40);
prefr.add(PayDue);
```

```
tfidp = new JTextField();
tflocationp = new JTextField();
tfemailp = new JTextField();
tfpaidp = new JTextField();
tfnamep = new JTextField();
```

```
tfphonep = new JTextField();  
tfreferralp = new JTextField();  
tfremovalp = new JTextField();  
tftrainerp = new JTextField();
```

```
pcomboDobYr = new JComboBox(y);  
pcomboDobMonth = new JComboBox(m);  
pcomboDobDate = new JComboBox(d);
```

```
pcomboMsYr = new JComboBox(y);  
pcomboMsMonth = new JComboBox(m);  
pcomboMsDay = new JComboBox(d);
```

```
pmalerd = new JRadioButton("Male");  
pfemalerd = new JRadioButton("Female");  
bg = new ButtonGroup();
```

```
addpre = new JButton("Add Premium Member");  
pactivate = new JButton("Activate Membership");  
pdeactivate = new JButton("Deactivate Membership");  
pattendance = new JButton("Mark Attendance");  
prevert = new JButton("Revert Membership");  
pclear = new JButton("Clear");  
pdisplay = new JButton("Display");  
pcalculatediscount = new JButton("Calculate Discount");
```

```
premium.setBounds(15,15,250,50);
```

plabelid.setBounds(50,110,100,15);
plabellocation.setBounds(550,190,100,15);
plabelemail.setBounds(50,260,100,15);
plabelgender.setBounds(50,340,100,15);
plabelmemberships.setBounds(50,420,150,15);
plabelpaid.setBounds(50,500,100,15);

plabelname.setBounds(550,110,100,15);
plabeldob.setBounds(50,190,100,15);
plabelphone.setBounds(550,260,100,15);
labeltrainer.setBounds(550,340,150,15);
plabelreferral.setBounds(550,420,100,15);
plabelremoval.setBounds(550,500,150,15);

addpre.setBounds(50,620,180,40);
pactivate.setBounds(370,620,180,40);
pdeactivate.setBounds(670,620,180,40);
pattendance.setBounds(50,680,180,40);
prevert.setBounds(370,680,180,40);
pclear.setBounds(670,680,180,40);
pdisplay.setBounds(50,740,180,40);

tfidp.setBounds(48,140,300,20);
tfphonep.setBounds(548,290,300,20);
tfemailp.setBounds(48,300,300,20);
tfpaidp.setBounds(48,530,300,20);

```
tfnamep.setBounds(548,140,300,20);  
tflocationp.setBounds(548,210,300,20);  
tfreferralp.setBounds(548,440,300,20);  
tfremovalp.setBounds(548,530,300,20);
```

```
tftrainerp.setBounds(548,360,300,20);
```

```
pmalerd.setBounds(48,370,100,15);  
pfemalerd.setBounds(150,370,100,15);
```

```
pcomboDobYr.setBounds(48,210,100,30);  
pcomboDobMonth.setBounds(148,210,100,30);  
pcomboDobDate.setBounds(248,210,100,30);  
pcomboMsYr.setBounds(48,440,100,30);  
pcomboMsMonth.setBounds(148,440,100,30);  
pcomboMsDay.setBounds(248,440,100,30);
```

```
PSave = new JButton("Save");  
PSave.setBounds(200,15,100,50);  
PRead = new JButton("Read");  
PRead.setBounds(400,15,100,50);  
pcalculatediscount.setBounds(370,740,180,40);
```

```
prefr.add(pcalculatediscount);  
prefr.add(premium);  
prefr.add(PSave);  
prefr.add(PRead);
```

```
prefr.add(plabelid);  
prefr.add(plabelname);  
prefr.add(plabellocation);  
prefr.add(plabelemail);  
prefr.add(plabelgender);  
prefr.add(plabeldob);  
prefr.add(plabelphone);  
prefr.add(plabelplan);  
prefr.add(plabelmemberships);  
prefr.add(plabelreferral);  
prefr.add(plabelpaid);  
prefr.add(plabelremoval);  
prefr.add(plabeltrainer);  
prefr.add(plabelprice);  
prefr.add(plabeldiscount);  
prefr.add(tfidp);  
prefr.add(tflocationp);  
prefr.add(tfemailp);  
prefr.add(tfpaidp);
```

```
prefr.add(tfnamep);  
prefr.add(tfphonep);  
prefr.add(tfreferralp);  
prefr.add(tfremovalp);
```

```
prefr.add(tftrainerp);  
prefr.add(pmalerd);
```

```
prefr.add(pfemalerd);  
prefr.add(pcomboDobYr);  
prefr.add(pcomboDobMonth);  
prefr.add(pcomboDobDate);
```

```
prefr.add(pcomboMsYr);  
prefr.add(pcomboMsMonth);  
prefr.add(pcomboMsDay);  
prefr.add(addpre);  
prefr.add(pattendance);  
prefr.add(pactivate);  
prefr.add(pdeactivate);  
prefr.add(pclear);  
prefr.add(pdisplay);  
prefr.add(prevert);
```

```
bg.add(pmalerd);  
bg.add(pfemalerd);
```

```
reg.addActionListener(this);  
pre.addActionListener(this);  
addreg.addActionListener(this);  
addpre.addActionListener(this);  
activate.addActionListener(this);  
deactivate.addActionListener(this);  
attendance.addActionListener(this);  
revert.addActionListener(this);
```

```
clear.addActionListener(this);
display.addActionListener(this);
pcalculatediscount.addActionListener(this);
PSave.addActionListener(this);
RSave.addActionListener(this);
PRead.addActionListener(this);
RRead.addActionListener(this);
PayDue.addActionListener(this);
pactivate.addActionListener(this);
pdeactivate.addActionListener(this);
pattendance.addActionListener(this);
prevert.addActionListener(this);
pclear.addActionListener(this);
pdisplay.addActionListener(this);
```

```
prefr.setLayout(null);
prefr.setVisible(false);
prefr.setSize(900,1000);
```

```
displayframe = new JFrame();
displayta = new JTextArea();
JScrollPane scrollDisplay = new JScrollPane(displayta);
```

```
scrollDisplay.setBounds(50,15,500,500);
displayframe.add(scrollDisplay);
displayframe.setLayout(null);
displayframe.setVisible(false);
```

```

displayframe.setSize(600,600);

}

@Override
public void actionPerformed(ActionEvent ae)
{

    if(ae.getSource()==reg)
    {
        regfr.setVisible(true);
        prefr.setVisible(false);
    }
    else if(ae.getSource()==pre)
    {
        prefr.setVisible(true);
        regfr.setVisible(false);
    }
    else if(ae.getSource()==addreg)
    {
        if(tfid.getText().isEmpty()|| tflocation.getText().isEmpty()||
tfemail.getText().isEmpty()||

tfname.getText().isEmpty()||tfphone.getText().isEmpty()||tfreferral.getText().
isEmpty())
        {

```



```

        JOptionPane.showMessageDialog(regfr,"please fill in the
application");
    }
    else
    {
        try{
            int regid = Integer.parseInt(tfid.getText());
            String regLocation = tflocation.getText();
            String regEmail = tfemail.getText();
            String regName = tfname.getText();
            String regPhone = tfphone.getText();
            String regReferral = tfreferral.getText();
            String regGender="";
            if(malerd.isSelected())
            {
                regGender="male";
            }
            else if(femalerd.isSelected())
            {
                regGender="female";
            }
            String year= (String) comboDobYr.getSelectedItemAt();
            String month= (String) comboDobMonth.getSelectedItemAt();
            String date= (String) comboDobDate.getSelectedItemAt();
            String regDob= date+"-"+month+"-"+year;
            String myear= (String) comboMsYr.getSelectedItemAt();
            String mmonth= (String) comboMsMonth.getSelectedItemAt();

```

```

String mdate= (String) comboMsDay.getSelectedItem();
String regMemberships= mdate+"-"+mmonth+"-"+myear;

if(array.isEmpty())
{
    RegularMember regobj = new
RegularMember(regid,regName,regLocation,regPhone,regEmail,
                regGender,regDob, regMemberships, regReferral);
    array.add(regobj);
    JOptionPane.showMessageDialog(regfr,"Regular Member
added");
}
else
{
    boolean idexists=false;
    for(GymMember mem : array)
    {

        if(mem.getId()==regid)
        {
            idexists=true;
            break;
        }
    }
    if(idexists)
    {

```

```

        JOptionPane.showMessageDialog(regfr,"please choose
another ID.");
    }
    else
    {
        RegularMember regobj = new
RegularMember(regid,regName,regLocation,regPhone,regEmail,
        regGender,regDob, regMemberships, regReferral);
        array.add(regobj);
        JOptionPane.showMessageDialog(regfr,"Regular
Member added");
    }
}
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(regfr,"Please input number
for ID","Error",JOptionPane.WARNING_MESSAGE);
}
}
}

else if(ae.getSource()==attendance)
{
    if(tfid.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(regfr,"ID found empty");
    }
}
}

```

```

    }
    else
    {
        try{
            int regid=Integer.parseInt(tfid.getText());
            boolean idexits=false;
            GymMember memberExists = null;
            for(GymMember member : array)
            {

                if(member.getId()==regid)
                {
                    idexits=true;
                    memberExists = member;
                    break;
                }
            }
            if(idexits)
            {
                if(memberExists.getActiveStatus()){
                    memberExists.markAttendance();
                    JOptionPane.showMessageDialog(regfr,"Attendance
marked");
                }
            }
            else
            {

```

```

        JOptionPane.showMessageDialog(regfr,"member is not
activate");
    }
}
else
{
    JOptionPane.showMessageDialog(regfr, "ID not
found","Error", JOptionPane.ERROR_MESSAGE);
}
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(regfr,"Please input number
for ID");
}
}

else if(ae.getSource()==activate)
{
    if(tfid.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(regfr,"ID found empty");
    }
    else
    {
        try{

```

```

int regid=Integer.parseInt(tfid.getText());
boolean idexits=false;
GymMember memberExists = null;
for(GymMember member : array)
{
    if(member.getId()==regid)
    {
        idexits=true;
        memberExists = member;
        break;
    }
}
if(idexits)
{
    if(memberExists.getActiveStatus()){
        JOptionPane.showMessageDialog(regfr,"Membership is
already activated","Activated",JOptionPane.INFORMATION_MESSAGE);
    }
    else
    {
        memberExists.activateMembership();
        JOptionPane.showMessageDialog(regfr,"Membership
activated");
    }
}
else

```

```

        {
            JOptionPane.showMessageDialog(regfr, "ID not found");
        }
    }
    catch(NumberFormatException ex)
    {
        JOptionPane.showMessageDialog(regfr,"Please input number
for ID");
    }
}
}

```

```

else if(ae.getSource()==deactivate)
{
    if(tfid.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(regfr,"ID found empty");
    }
    else
    {
        try{
            int regid=Integer.parseInt(tfid.getText());
            boolean idexits=false;
            GymMember memberExists = null;
            for(GymMember member : array)
            {

```

```

        if(member.getId()==regid)
        {
            idexits=true;
            memberExists = member;
            break;
        }
    }
    if(idexits)
    {
        if(memberExists.getActiveStatus()==false){
            JOptionPane.showMessageDialog(regfr,"Membership is
already deactivated");
        }
        else
        {
            memberExists.deactivateMembership();
            JOptionPane.showMessageDialog(regfr,"Membership
deactivated");
        }
    }
    else
    {
        JOptionPane.showMessageDialog(regfr, "ID not found");
    }
}
catch(NumberFormatException ex)

```



```

        {
            JOptionPane.showMessageDialog(regfr,"Please input number
for ID");
        }
    }
}

```

```

else if(ae.getSource()==revert)
{
    if(tfid.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(regfr,"ID found empty");
    }
    else if(tfremoval.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(regfr,"please fill in the
removal reason");
    }
    else
    {
        try{
            int regid=Integer.parseInt(tfid.getText());
            String regRemoval=tfremoval.getText();
            boolean idexits=false;
            GymMember memberExists = null;
            for(GymMember member : array)
            {

```

```

        if(member.getId()==regid)
        {
            idexits=true;
            memberExists = member;
            break;
        }
    }
    if(idexits)
    {
        memberExists.resetMembership();
        JOptionPane.showMessageDialog(regfr,"Membership
Reset");
    }
    else
    {
        JOptionPane.showMessageDialog(regfr, "ID not found");
    }
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(regfr,"Please input number
for ID");
}
}
}

```

```

else if(ae.getSource()==display)
{
    displayframe.setVisible(true);
    displayta.setText("");
    if(array.isEmpty())
    {

        displayta.append("NO MEMBER REGISTERED!!!");
    }
    else
    {
        displayta.append("\nREGULAR MEMBERS: \n\n ");
        boolean memberExistence=false;

        for(GymMember member : array)
        {
            if(member instanceof RegularMember)
            {
                memberExistence= true;
                break;
            }
        }
        if(memberExistence)
        {
            for(GymMember member:array)
            {
                if(member instanceof RegularMember)

```

```

{
    RegularMember regular = (RegularMember) member;
    displayta.append("\nMember ID: "+regular.getId()+"\n");
    displayta.append("Name: "+regular.getName()+"\n");
    displayta.append("Location: "+regular.getLocation()+"\n");
    displayta.append("Email: "+regular.getEmail()+"\n");
    displayta.append("Phone: "+regular.getPhone()+"\n");
    displayta.append("Date of Birth:
"+regular.getDOB()+"\n");
    displayta.append("Gender: "+regular.getGender()+"\n");
    displayta.append("Plan: "+regular.getPlan()+"\n");
    displayta.append("Price: "+regular.getPrice()+"\n");
    displayta.append("Membership Start Date:
"+regular.getMembershipStartDate()+"\n");
    displayta.append("Attendance:
"+regular.getAttendance()+"\n");
    displayta.append("Loyalty Points:
"+regular.getLoyaltyPoints()+"\n");
    displayta.append("Active Status:
"+regular.getActiveStatus()+"\n");
    if(regular.getRemovalReason()!="")
    {
        displayta.append("Removal Reason:
"+regular.getRemovalReason()+"\n");
    }
    displayta.append("-----
-----");
}

```

```

        }
    }
}
else
{
    displayta.append("NO REGULAR MEMBER
REGISTERED!!!");
}
}
}
}

```

```

else if (ae.getSource()==revert){
    if(tfid.getText().isEmpty()){
        JOptionPane.showMessageDialog(regfr,"Please input value for
Member ID");
    }
}

```

```

else if (tfremoval.getText().isEmpty()){
    JOptionPane.showMessageDialog(regfr, "Please input a valid
removal reason!");
}
}

```

```

else {
    try{
        int memId = Integer.parseInt(tfid.getText());
        String removalReason = tfremoval.getText();
        boolean isMemberFound = false;
    }
}

```

```

RegularMember matchedMember = null;
for(GymMember member : array) {
    if(member instanceof RegularMember) {
        RegularMember regular = (RegularMember) member;
        if (regular.getId() == memId) {
            isMemberFound = true;
            matchedMember = regular;
            break;
        }
    }
}
if (isMemberFound) {
    matchedMember.revertRegularMember(removalReason);
    JOptionPane.showMessageDialog(regfr,"Successfully
Reverted Member Status for: "+matchedMember.getName(),"Success",
JOptionPane.INFORMATION_MESSAGE);
}
else {
    JOptionPane.showMessageDialog(regfr,"Member with this
ID does not exist!","ERROR",JOptionPane.ERROR_MESSAGE);

}
}
catch (NumberFormatException ex) {
    JOptionPane.showMessageDialog(regfr,"ID can only have
number values!","ERROR", JOptionPane.WARNING_MESSAGE);
}

```

```
}  
}
```

```
else if(ae.getSource()==clear || ae.getSource()==pclear)
```

```
{  
    tfid.setText("");  
    tflocation.setText("");  
    tfemail.setText("");  
    tfpaid.setText("");  
    tfname.setText("");  
    tfphone.setText("");  
    tfreferral.setText("");  
    tfremoval.setText("");  
  
    tfidp.setText("");  
    tflocationp.setText("");  
    tfemailp.setText("");  
    tfpaidp.setText("");  
  
    tfnamep.setText("");  
    tfphonep.setText("");  
    tfreferralp.setText("");  
    tfremovalp.setText("");  
  
    tftrainerp.setText("");
```

```

        comboDobMonth.setSelectedItem("January");
        comboDobYr.setSelectedItem("2006");
        comboDobDate.setSelectedItem("1");
        comboMsDay.setSelectedItem("1");
        comboMsMonth.setSelectedItem("January");
        comboMsYr.setSelectedItem("2006");

        planCb.setSelectedItem("basic");
    }

    else if(ae.getSource()==addpre)
    {
        try{
            if(tfidp.getText().isEmpty()|| tflocationp.getText().isEmpty()||
tfemailp.getText().isEmpty()||

tfnamep.getText().isEmpty()||tfphonep.getText().isEmpty()||tfreferralp.getTe
xt().isEmpty())
            {
                JOptionPane.showMessageDialog(prefr,"please fill in the
application");
            }
        }
        else
        {
            int preid = Integer.parseInt(tfidp.getText());
            String preLocation = tflocationp.getText();

```



```

String preEmail = tfemailp.getText();
String preName = tfnamep.getText();
String prePhone = tfphonep.getText();
String preTrainer = tftrainerp.getText();
String preGender="";
if(pmalerd.isSelected())
{
    preGender="male";
}
else if(pfemalerd.isSelected())
{
    preGender="female";
}
String year= (String) pcomboDobYr.getSelectedItemAt();
String month= (String) pcomboDobMonth.getSelectedItemAt();
String date= (String) pcomboDobDate.getSelectedItemAt();
String preDob= date+"-"+month+"-"+year;
String myear= (String) pcomboMsYr.getSelectedItemAt();
String mmonth= (String) pcomboMsMonth.getSelectedItemAt();
String mdate= (String) pcomboMsDay.getSelectedItemAt();
String preMemberships= mdate+"-"+mmonth+"-"+myear;

if(array.isEmpty())
{
    //creating object
    PremiumMember preobj = new
PremiumMember(preid,preName,preLocation,prePhone,preEmail,

```

```

        preGender,preDob, preMemberships, preTrainer);
array.add(preobj);
JOptionPane.showMessageDialog(prefr,"Premium Member
added");
    }
    else
    {
        boolean idexists=false;
        for(GymMember mem : array)
        {

            if(mem.getId()==preid)
            {
                idexists=true;
                break;
            }
        }
        if(idexists)
        {
            JOptionPane.showMessageDialog(prefr,"please choose
another ID.", "Duplication",JOptionPane.ERROR_MESSAGE);
        }
        else
        {
            PremiumMember preobj = new
PremiumMember(preid,preName,preLocation,prePhone,preEmail,
                preGender,preDob, preMemberships, preTrainer);

```

```

        array.add(preobj);
        JOptionPane.showMessageDialog(prefr,"Premium
Member added");
    }
}
}
}
catch(NumberFormatException ex){
    JOptionPane.showMessageDialog(prefr,"Please input number for
ID","Error",JOptionPane.WARNING_MESSAGE);
}
}
else if (ae.getSource()==pattendance)
{
    if(tfidp.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(prefr,"ID found empty");
    }
    else
    {
        try{
            int preid=Integer.parseInt(tfidp.getText());
            boolean idexits=false;
            GymMember memberExists = null;
            for(GymMember member : array)
            {
                if(member.getId()==preid)

```

```

        {
            idexits=true;
            memberExists = member;
            break;
        }
    }
    if(idexits)
    {
        if(memberExists.getActiveStatus()){
            memberExists.markAttendance();
            JOptionPane.showMessageDialog(prefr,"Attendance
marked","Attendance",JOptionPane.INFORMATION_MESSAGE);
        }
        else
        {
            JOptionPane.showMessageDialog(prefr,"member is not
activate","Attendance",JOptionPane.ERROR_MESSAGE);
        }
    }
    else
    {
        JOptionPane.showMessageDialog(prefr, "ID not
found","Error", JOptionPane.ERROR_MESSAGE);
    }
}
catch(NumberFormatException ex){

```

```

        JOptionPane.showMessageDialog(prefr,"Please input number
for ID","Error",JOptionPane.WARNING_MESSAGE);
    }
}
}
else if(ae.getSource()==prevert)
{
    if(tfidp.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(prefr,"ID found empty");
    }
    else if(tfremovalp.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(prefr,"please fill in the
removal reason");
    }
    else
    {
        try{
            int preid=Integer.parseInt(tfidp.getText());
            String preRemoval=tfremovalp.getText();
            boolean idexits=false;
            GymMember memberExists = null;
            for(GymMember member : array)
            {
                if(member.getId()==preid)
                {

```

```

        idexits=true;
        memberExists = member;
        break;
    }
}
if(idexits)
{
    memberExists.resetMembership();
    JOptionPane.showMessageDialog(prefr,"Membership
Reset","Reset",JOptionPane.INFORMATION_MESSAGE);
}
else
{
    JOptionPane.showMessageDialog(prefr, "ID not
found","Error", JOptionPane.ERROR_MESSAGE);
}
}
catch(NumberFormatException ex){
    JOptionPane.showMessageDialog(prefr,"Please input number
for ID","Error",JOptionPane.WARNING_MESSAGE);
}
}
}
else if(ae.getSource()==pactivate)
{
    if(tfidp.getText().isEmpty())
    {

```

```

        JOptionPane.showMessageDialog(prefr,"ID found empty");
    }
    else
    {
        try{
            int preid=Integer.parseInt(tfidp.getText());
            boolean idexits=false;
            GymMember memberExists = null;
            for(GymMember member : array)
            {

                if(member.getId()==preid)
                {
                    idexits=true;
                    memberExists = member;
                    break;
                }
            }
            if(idexits)
            {
                if(memberExists.getActiveStatus()){
                    JOptionPane.showMessageDialog(prefr,"Membership is
already activated","Activated",JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {
                    memberExists.activateMembership();
                }
            }
        }
        catch (Exception e)
        {
            JOptionPane.showMessageDialog(prefr,"Invalid ID",JOptionPane.ERROR_MESSAGE);
        }
    }
}

```

```

        JOptionPane.showMessageDialog(prefr,"Membership
activated","Activated",JOptionPane.INFORMATION_MESSAGE);
    }
}
else
{
    JOptionPane.showMessageDialog(prefr, "ID not
found","Error", JOptionPane.ERROR_MESSAGE);
}
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(prefr,"Please input number
for ID","Error",JOptionPane.WARNING_MESSAGE);
}
}
else if(ae.getSource()==pdeactivate)
{
    if(tfidp.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(prefr,"ID found empty");
    }
    else
    {
        try{
            int preid = Integer.parseInt(tfidp.getText());

```



```

boolean idexits=false;
GymMember memberExists = null;
for(GymMember member : array)
{

    if(member.getId()==preid)
    {
        idexits=true;
        memberExists = member;
        break;
    }
}
if(idexits)
{
    if(memberExists.getActiveStatus()==false){
        JOptionPane.showMessageDialog(prefr,"Membership is
already
deactivated","Deactivated",JOptionPane.INFORMATION_MESSAGE);
    }
    else
    {
        memberExists.deactivateMembership();
        JOptionPane.showMessageDialog(prefr,"Membership
deactivated","Deactivated",JOptionPane.INFORMATION_MESSAGE);
    }
}
else

```

```

        {
            JOptionPane.showMessageDialog(prefr, "ID not
found", "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
    catch(NumberFormatException ex){
        JOptionPane.showMessageDialog(prefr, "Please input number
for ID", "Error", JOptionPane.WARNING_MESSAGE);
    }
}
else if(ae.getSource()==pdisplay)
{
    displayframe.setVisible(true);
    displayta.setText("");
    if(array.isEmpty())
    {
        displayta.append("NO MEMBER REGISTERED!!!");
    }
    else
    {
        displayta.append("\nPREMIUM MEMBERS: \n\n ");
        boolean memberExistence=false;
        for(GymMember member : array)
        {
            if(member instanceof PremiumMember)
            {

```

```

        memberExistence= true;
        break;
    }
}
if(memberExistence)
{
    for(GymMember member:array)
    {
        if(member instanceof PremiumMember)
        {
            PremiumMember premium = (PremiumMember) member;
            displayta.append("\nMember ID: "+premium.getId()+"\n");
            displayta.append("Name: "+premium.getName()+"\n");
            displayta.append("Location:
"+premium.getLocation()+"\n");
            displayta.append("Email: "+premium.getEmail()+"\n");
            displayta.append("Phone: "+premium.getPhone()+"\n");
            displayta.append("Date of Birth:
"+premium.getDOB()+"\n");
            displayta.append("Gender: "+premium.getGender()+"\n");
            displayta.append("Membership Start Date:
"+premium.getMembershipStartDate()+"\n");
            displayta.append("Attendance:
"+premium.getAttendance()+"\n");
            displayta.append("Loyalty Points:
"+premium.getLoyaltyPoints()+"\n");

```

```

        displayta.append("Personal Trainer:
"+premium.getPersonalTrainer()+"\n");
        displayta.append("Payment of
Rs."+premium.getPaidAmount()+" has been made\n");
        displayta.append("Active Status:
"+premium.getActiveStatus()+"\n");
        displayta.append(premium.getIsFullPayment()?"Full
payment has been done":"Full payment has not been done"+"\\n");
        if(premium.getIsFullPayment()==false)
        {
            double
remainingAmount=premium.getPremiumCharge()-
premium.getPaidAmount();
            displayta.append("Payment of
Rs."+remainingAmount+" is left to be paid.\\n");
        }
        else
        {
            displayta.append("\\nDiscount:
Rs."+premium.getDiscountAmount()+"\\n");
        }

displayta.append("=====
=====");
    }
}
}

```

```

        else
        {
            displayta.append("NO PREMIUM MEMBER
REGISTERED!!!");
        }
    }
}

```

```

else if (ae.getSource()==pcalculatediscount){
    if(tfidp.getText().isEmpty()){
        JOptionPane.showMessageDialog(prefr, "Please input value for
Member ID");
    }
}

```

```

try{
    int memId = Integer.parseInt(tfidp.getText());
    boolean isMemberFound = false;
    PremiumMember matchedMember = null;

    for (GymMember member : array) {
        if (member instanceof PremiumMember){
            PremiumMember premiummember = (PremiumMember)
member;
            if (member.getId() == memId) {
                isMemberFound = true;
                matchedMember = premiummember;
                break;
            }
        }
    }
}

```

```
    }  
  }  
}
```

```
if(isMemberFound){  
    if(matchedMember.getActiveStatus()) {  
        if(matchedMember.getIsFullPayment())  
        {  
  
            matchedMember.calculateDiscount();  
            JOptionPane.showMessageDialog(prefr, "The member  
"+matchedMember.getName()+" has successfully received 10% Discount");  
        }  
        else{  
            JOptionPane.showMessageDialog(prefr, "Member not  
eligible for discount. Full Payment has not been made.");  
        }  
    }  
    else{  
        JOptionPane.showMessageDialog(prefr,"Activate  
membership first for: "+matchedMember.getName()+" to mark  
attendance","Success",JOptionPane.INFORMATION_MESSAGE);  
  
    }  
}  
else{
```

```

        JOptionPane.showMessageDialog(prefr,"Member with this ID
does not exist!","Error", JOptionPane.ERROR_MESSAGE);
    }
}
catch (NumberFormatException ex){
    JOptionPane.showMessageDialog(prefr,"ID can only have
number values");
}

}
else if (ae.getSource()==PayDue){
    if(tfidp.getText().isEmpty()){
        JOptionPane.showMessageDialog(prefr, "Please input value for
Member ID");

    }
    try{
        int preid = Integer.parseInt(tfidp.getText());
        double paidAmount = Double.parseDouble(tfpaidp.getText());
        boolean isMemberFound = false;
        PremiumMember matchedMember = null;

        for (GymMember member : array) {
            if (member instanceof PremiumMember){
                PremiumMember premiummember = (PremiumMember)
member;
                if (member.getId() == preid) {

```

```

        isMemberFound = true;
        matchedMember = premiummember;
        break;

    }
}

if(isMemberFound){
    if(matchedMember.getActiveStatus()) {
        String message =
matchedMember.payDueAmount(paidAmount);
        JOptionPane.showMessageDialog(prefr,message);
    }
    else{
        JOptionPane.showMessageDialog(prefr,"Activate
membership first ");
    }
}
else{
    JOptionPane.showMessageDialog(prefr,"Member with this ID
does not exist!","Error", JOptionPane.ERROR_MESSAGE);
}
}

catch (NumberFormatException e){
    JOptionPane.showMessageDialog(prefr,"ID can only have
number values");
}

```



```

    }

}

else if (ae.getSource() == PSave || ae.getSource() == RSave){

    try {
        File file = new File("MemberDetails.txt");
        FileWriter writer = new FileWriter(file);

        writer.write("Regular and Premium Member Details\n\n");

        writer.write(String.format(
            "%-5s %-12s %-15s %-15s %-20s %-25s %-8s %-18s %-20s %-15s %-10s %-7s %-10s %-10s %-10s\n",
            "ID", "Attendance", "Name", "Location", "Phone", "Email",
            "Gender", "DOB", "Membership Start", "Loyalty Pts",
            "Status",
            "Plan", "Price", "Trainer", "Paid Amt", "Discount"
        ));

        writer.write("-----\n");

        // Loop to save member data

```

```

    for (GymMember member : array) {
        String plan = "NA", trainer = "NA", paidAmt = "NA", discount =
"NA", price = "NA";

        if (member instanceof RegularMember) {
            RegularMember rm = (RegularMember) member;
            plan = rm.getPlan();
            price = "Rs." + rm.getPrice();
        } else if (member instanceof PremiumMember) {
            PremiumMember pm = (PremiumMember) member;
            trainer = pm.getPersonalTrainer();
            paidAmt = "Rs." + pm.getPaidAmount();
            discount = pm.getIsFullPayment() ? "Rs." +
pm.getDiscountAmount() : "NA";
            price = "Rs." + pm.getPremiumCharge();
        }

        writer.write(String.format(
            "%-5d %-12d %-15s %-15s %-20s %-25s %-8s %-18s %-
20s %-15.2f %-10s %-7s %-10s %-10s %-10s\n",
            member.getId(),
            member.getAttendance(),
            member.getName(),
            member.getLocation(),
            member.getPhone(),
            member.getEmail(),
            member.getGender(),

```

```

        member.getDOB(),
        member.getMembershipStartDate(),
        member.getLoyaltyPoints(),
        member.getActiveStatus() ? "Active" : "Inactive",
        plan,
        price,
        trainer,
        paidAmt,
        discount
    ));
}

writer.close();
JOptionPane.showMessageDialog(prefr, "Details Saved To File
Successfully!!", "Saved", JOptionPane.INFORMATION_MESSAGE);
} catch (Exception ex) {
    System.out.println("Exception Occurred : " + ex);
    JOptionPane.showMessageDialog(prefr, "Error saving member
details!", "Error", JOptionPane.ERROR_MESSAGE);
}
}
else if (ae.getSource()==PRead || ae.getSource()==RRead){
    displayframe.setVisible(true);
    try {
        FileReader fileReader = new
FileReader("MemberDetails.txt");
        int character;

```

```

        while ((character = fileReader.read()) != -1) {
            char ch = (char) character;
            displayta.append(String.valueOf(ch));
        }

        fileReader.close();

    } catch (Exception ex) {
        System.out.println("Exception Occurred: " + ex);
        JOptionPane.showMessageDialog(prefr, "Error reading
member details!", "Error", JOptionPane.ERROR_MESSAGE);
    }
}
}
}

```

