



CS4001NI Programming

60% Individual Coursework

2025 Spring

Student Name: Aaki Prajapati

London Met ID: 24046683

College ID: NP01CP4A240116

Group: C10

Assignment Due Date: Friday, May 16, 2025

Assignment Submission Date: Friday, May 16, 2025

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

Table of Contents

1. Introduction:	1
1.1 Aim:	2
1.2 Objectives:	2
1.3 Tools Used:	2
2. GUI Wireframe:	6
2.1 Add a Member:	6
2.1.1 Regular Member:	6
2.1.2 Premium Member:	7
2.2 Membership Details:	8
2.2.1 Regular Member:	8
2.2.2 Premium Member:	9
2.3 Member Information:	10
2.3.1 Regular Member:	10
2.3.2 Premium Member:	11
3. Class Diagram:	12
4. Pseudocode:	22
4.1 GymMember:	22
4.2 RegularMember:	25
4.3 PremiumMember:	29
4.4 GymGUI:	32
4.5 InvalidIdException:	62
4.6 InvalidPhoneException:	63
5. Method Description:	63
5.1 GymMember:	63
5.2 RegularMember:	65
5.3 PremiumMember:	66
5.4 GymGUI:	67
6. Testing:	77
6.1 Test 1:	77
6.2 Test 2:	79
6.2.1 To add a regular member:	79
6.2.2 To add a premium member:	85

6.3	Test 3:	91
6.3.1	Regular Member:.....	91
6.3.2	Premium Member:	97
6.4	Test 4:	100
6.5	Test 5:	108
7.	Error Detection and Correction:.....	113
8.	Conclusion:.....	121
9.	References:	123
	Bibliography	123
10.	Appendix:.....	124
10.1	GymMember Class:.....	124
10.2	RegularMember Class:.....	129
10.3	PremiumMember Class:.....	135
10.4	GymGUI Class:	141
10.5	InvalidIdException Class:	249
10.6	InvalidPhoneException Class:.....	249

Table of Figures:

Figure 1: BlueJ (BlueJ, n.d.)	3
Figure 2: MS-Word (Microsoft, n.d.)	3
Figure 3: Notepad (Microsoft, n.d.)	4
Figure 4: Balsamiq (Peldi, 2017)	5
Figure 5: Add a Member- Regular Member	6
Figure 6: Add a Member- Premium Member	7
Figure 7: Membership Details- Regular Member	8
Figure 8: Membership Details- Premium Member	9
Figure 9: Member Information- Regular Member.....	10
Figure 10: Member Information- Premium Member.....	11
Figure 11: Screenshot to show inheritance between the classes	22
Figure 12: Screenshot for Test 1	78
Figure 13: Trying to add a Regular member with invalid ID	80
Figure 14: Trying to add a Regular member with invalid ID range	81
Figure 15: Trying to add a Regular member with valid ID but invalid phone number length...	82
Figure 16: Trying to add a Regular Member with empty field(s)	83
Figure 17: Successful addition of Regular Member	84
Figure 18: Trying to add a Premium member with invalid ID	86

Figure 19: Trying to add a Premium member with invalid ID range	87
Figure 20: Trying to add a Premium member with valid ID but invalid phone number length ..	88
Figure 21: Trying to add a Premium Member with empty field(s)	89
Figure 22: Successful addition of Premium Member	90
Figure 23: Trying to mark attendance without activating membership (Regular)	92
Figure 24: Attendance after membership is activated (Regular)	93
Figure 25: Trying to upgrade membership without enough attendance	94
Figure 26: Successful upgrade with enough attendance	95
Figure 27: Unsuccessful upgrade from Standard to Basic.....	96
Figure 28: Trying to mark attendance without activating membership (Premium)	98
Figure 29: Attendance after membership is activated (Premium)	99
Figure 30: Allowing the user to pay amount.....	101
Figure 31: Showing remaining amount	102
Figure 32: Trying to calculate discount without full payment.....	102
Figure 33: Paying remaining amount	103
Figure 34: Dialog box to show full payment	104
Figure 35: Calculated discount amount after full payment	104
Figure 36: GUI before reverting membership	105
Figure 37: Confirmation message to show successful reversion	106
Figure 38: GUI after reverting membership	107
Figure 39: User information to be stored (Regular)	109
Figure 40: User data saved to file (Regular)	110
Figure 41: User data read from file (Regular)	110
Figure 42: User information to be stored (Premium).....	111
Figure 43: User data saved to file (Premium)	112
Figure 44: User data read from file (Premium)	112
Figure 45: Syntax error detection	113
Figure 46: Syntax error correction	114
Figure 47: Runtime error detection	115
Figure 48: Runtime error correction	115
Figure 49: Logical error detection	116
Figure 50: Error message even if the plan selected is logically incorrect	117
Figure 51: Price returned as -1 due to logical error	118
Figure 52: Logical error correction	119
Figure 53: Success message of plan upgrade.....	119
Figure 54: Correct value of price according to plan selected	120

Table of Tables:

Table 1: Class Diagram- Gym Member	12
Table 2: Class Diagram- Regular Member	13
Table 3: Class Diagram- Premium Member.....	14
Table 4: Class Diagram- GymGUI	15

Table 5: InvalidIdException Class.....	21
Table 6: InvalidPhoneException Class	21
Table 7: To compile and run the program using command prompt/ terminal.....	77
Table 8: To add a regular member	79
Table 9: To add a premium member	85
Table 10: To check mark attendance and Upgrade Plan button : Regular Member	91
Table 11: To check mark attendance button : Premium Member	97
Table 12: To check functionality of discount, payment and revert buttons	100
Table 13: To save to and read from file	108

24046683 Aaki Prajapati- javafinalturnitin.docx

 Islington College,Nepal

Document Details

Submission ID

trn:oid::3618:96172045

Submission Date

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

Download Date

May 16, 2025, 10:22 AM GMT+5:45

File Name

24046683 Aaki Prajapati- javafinalturnitin.docx

File Size

85.5 KB

97 Pages

12,864 Words





72,540 Characters






6% Overall Similarity

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

Match Groups

-  **78 Not Cited or Quoted 6%**
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

- 1%  Internet sources
- 2%  Publications
- 5%  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.

1. Introduction:

The growing advancement in science and technology has introduced multiple concepts to overcome a problem. One of such concepts include Programming. A sequence of instructions build a program and these set of programs build a software. To make this work, developers have vigorously worked for a very long period of time to create Programming Language, something that could be a way to link our implementations with the computer instructions. A programming language, refers to a way of writing a program through which the users can instruct the computer to perform particular tasks. This coursework was assigned to us by the Programming module teachers, which is an implementation of the programming language “Java”, that we learnt throughout the first year.

In this coursework, we have been assigned to develop a Gym Application using the concept of Object Oriented Programming (OOP) and its pillars: Encapsulation, Inheritance, Abstraction and Polymorphism. Here, an abstract parent class GymMember was created, consisting of the instance variables, constructor, accessor methods and various other methods: both concrete and abstract. Secondly, a subclass of the parent class was made, named RegularMember. It contained the inherited attributes, its unique attributes, concrete methods and the implementation of abstract method from the parent class. Furthermore, a second subclass, PremiumMember was made which also consisted of the inherited attributes, unique attributes, concrete methods, and implementation of the abstract methods as inherited from the parent class. Finally a class called GymGUI was made where an ArrayList was constructed which stored the objects of both subclasses.

To make all of our logical solutions actually applicable to the users, use of GUI components such as JFrame, JPanel, JLabel, JButtons, etc. were made in the GymGUI class. Furthermore, the use of Event Handling promoted the user responsiveness and effectiveness, allowing smooth flow of the program.

Wireframes were developed using “Balsamiq”, allowing us to have an idea of the interface to develop, before actually developing it.

1.1 Aim:

- The aim of this coursework is to develop a user friendly gym application, implementing concepts of OOP and its pillars, graphical user interface (GUI), event handling and more.

1.2 Objectives:

- The main objective of this coursework was not only to make and implement gym membership, but also to know and understand about new and better concepts of programming with Java.
- To understand GUI, user interactions for smooth program flow
- To provide an idea of how different scenarios could be dealt with in real life.

1.3 Tools Used:**i. BlueJ:**

- BlueJ is an IDE (Integrated Development Environment) for implementing the Java programming language, allowing easy and efficient execution of the programs. It is a relatively small and simple interface, which is a great tool for beginners to understand Java and its importance in programming. Being a portable interface, it provides the freedom to the users to download and use it in any system: Windows OS, MAC OS, Linux and other platforms. By the support of a full-time team, BlueJ is updated time to time, and any technical issue is also solved. All in all, BlueJ is a constantly updating, simple, small, yet efficient platform for implementing programs in Java.

How it was used in the coursework:

- In this coursework, the gym application was executed by using BlueJ. Through its simple and easy to understand interface, a user- friendly, responsive and effective application which registers and manages the gym members, and performs several other operations based on user command was built.



Figure 1: BlueJ (BlueJ, n.d.)

ii. Microsoft Word:

Microsoft Word is a software designed to allow users to create, store, use, share, edit and delete the files. It has been widely known all over the world for its simple and flexible use in creation of any kind of file, along with additional features such as adding images, inserting tables, figures and much more.

How it was used in the coursework:

- In this coursework, MS-Word was used to create the report. After the completion of the requirements of this coursework, report to show evidences of the working of the program through testing, images, and explanations were done in this platform providing an efficient way to create the report.



Figure 2: MS-Word (Microsoft, n.d.)

iii. Notepad:

Notepad is a built in application for Windows, which is a simple text editor to create, edit, format and delete files. The files are stored in .txt format which is very useful for a number of programming languages to retrieve, store and update the data stored. Being a simple and easy to use application, its popularity and use has increased worldwide.

How it was used in the coursework:

- In this coursework, notepad was used to save the member data to the file, and read the same data to display it on the screen. It allowed us to have a platform where we could see the user and their information, making the program flow even more effective.



Figure 3: Notepad (Microsoft, n.d.)

iv. Balsamiq:

- Balsamiq is an application developed for making wireframes to create layout/structures that the user wishes to develop, may be it an application or web design. It is a user-friendly tool, having the drag and drop functionality, speeding the design process. It allow the developers to visualize their system before coding, creating a fixed mindset about what next to do. Furthermore, its speed and simple interface has increased its use over the years.

How it was used in the coursework:

- In this coursework, Balsamiq was used to create a wireframe of the gym membership application. Its easy interface allowed us to visualize the type of application to be made, hence easing the development process. It also saved a lot of time because

we already had an idea about the type of GUI to be developed, before we start to code.



balsamiq® Wireframes

Figure 4: Balsamiq (Peldi, 2017)

2. GUI Wireframe:

2.1 Add a Member:

2.1.1 Regular Member:

	Elevate Gym		
QUICK ACCESS	Regular Member	Premium Member	
	Add a Member		
Membership Details	Name: [text input]	Location: [text input]	ID: [text input]
Member Information	Phone No: [text input]	Email: [text input]	Gender: [text input]
	DOB: [text input]	Start Date: [text input]	Referral Source: [text input]
	<input type="button" value="Add a Regular Member"/>		<input type="button" value="Clear"/>

Figure 5: Add a Member- Regular Member

2.1.2 Premium Member:

Elevate Gym		
QUICK ACCESS	Regular Member	Premium Member
	Add a Member	
	Name: <input type="text"/> Location: <input type="text"/> ID: <input type="text"/>	
	Phone No: <input type="text"/> Email: <input type="text"/> Gender: <input type="text"/>	
Membership Details	DOB: <input type="text"/> Start Date: <input type="text"/> Trainer: <input type="text"/>	
	<input type="button" value="Add a Premium Member"/> <input type="button" value="Clear"/>	
Member Information		

Figure 6: Add a Member- Premium Member

2.2 Membership Details:

2.2.1 Regular Member:

Elevate Gym	
QUICK ACCESS	Regular Member Premium Member
Add a Member	Name: ██████████ ID: ██████████ Location: ██████████ Phone No: ██████████ Email: ██████████ Gender: ██████████ DOB: ██████████ Start Date: ██████████
Membership Details	
Member Information	<div> <div> <input type="button" value="Activate Membership"/> <input type="button" value="Deactivate Membership"/> </div> <div> <input type="button" value="Upgrade Plan"/> <input type="button" value="Mark Attendance"/> </div> <div> <input type="button" value="Revert"/> </div> </div> <div> Plan: ████████ Price: ████████ Removal Reason: ████████ </div> <div> Attendance: ████████ Loyalty Points: ████████ Plan: ████████ Removal Reason: ████████ Referral Source: ████████ </div> <div> <input type="button" value="Clear"/> <input type="button" value="Save"/> <input type="button" value="Save to File"/> <input type="button" value="Read from File"/> <input type="button" value="Display all Members"/> </div>

Figure 7: Membership Details- Regular Member

2.2.2 Premium Member:

Elevate Gym	
QUICK ACCESS	<div>Regular Member</div> <div>Premium Member</div>
Add a Member	Name: ██████████ ID: ██████████ Location: ██████████ Phone No: ██████████ Email: ██████████ Gender: ██████████ DOB: ██████████ Start Date: ██████████
Membership Details	
Member Information	<div> <div>Activate Membership</div> <div>Deactivate Membership</div> <div>Trainer ██████████</div> <div>Discount</div> <div>Mark Attendance</div> <div>Paid Amount: ██████████</div> <div>Payment</div> <div>Revert</div> <div>Full Payment: ██████████</div> </div> <div> Attendance: ████████ Loyalty Points: ████████ Total Charge: ████████ </div> <div> <div>Clear</div> <div>Save</div> <div>Save to File</div> <div>Read from File</div> <div>Display all Members</div> </div>

Figure 8: Membership Details- Premium Member

2.3 Member Information:

2.3.1 Regular Member:

Elevate Gym	
QUICK ACCESS	Regular Member
	Premium Member
	<div>Input Enter your ID: <div>OK Cancel</div></div>

Figure 9: Member Information- Regular Member

2.3.2 Premium Member:

Elevate Gym		
QUICK ACCESS	Regular Member	
	Premium Member	
	Add a Member	
	Membership Details	
Member Information	<div>Input Enter your ID: <div>OKCancel</div></div>	

Figure 10: Member Information- Premium Member

3. Class Diagram:

Table 1: Class Diagram- Gym Member

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

```
+ resetMember(): void
+ display(): void
```

Table 2: Class Diagram- Regular Member

RegularMember
<ul style="list-style-type: none"> - attendanceLimit: int {readOnly} - isEligibleForUpgrade: boolean - removalReason: String - referralSource: String - plan: String - price: double
<pre>+<<constructor>> RegularMember(id: int, name: String, location: String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String, referralSource: String) + getattendanceLimit(): int + getisEligibleForUpgrade(): boolean + getRemovalReason(): String + getReferralSource(): String + getPlan(): String + getPrice(): double + markAttendance(): void + getPlanPrice (plan: String): double + upgradePlan (plan:String): String + revertRegularMember (removalReason: String): void + display(): void</pre>

Table 3: Class Diagram- Premium Member

PremiumMember
<ul style="list-style-type: none"> - premiumCharge: double {readOnly} - personalTrainer: String - isFullPayment: boolean - paidAmount: double - discountAmount: double
<pre> +<<constructor>> PremiumMember(id: int, name: String, location: String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String, personalTrainer: String) + getPremiumCharge(): double + getPersonalTrainer(): String + getIsFullPayment(): boolean + getPaidAmount(): double + getDiscountAmount(): double + markAttendance(): void + payDueAmount (paidAmount: int): String + calculateDiscount (): void + revertPremiumMember (): void + display(): void </pre>

Table 4: Class Diagram- GymGUI

GymGUI
<ul style="list-style-type: none"> - frame: JFrame - addAMemberLeftPanel: JPanel - memberManageLeftPanel: JPanel - allMembersLeftPanel: JPanel - addARegularMemberPanel: JPanel - addAPremiumMemberPanel: JPanel - regularMemberPanel: JPanel - premiumMemberPanel: JPanel - regularInfoPanel: JPanel - premiumInfoPanel: JPanel - allMembersRegularPanel: JPanel - allMembersPremiumPanel: JPanel - displayPanel1: JPanel - displayPanel2: JPanel - readFromFilePanel1: JPanel - readFromFilePanel2: JPanel - quickAccess1: JLabel - quickAccess2: JLabel - quickAccess3: JLabel - gymName1: JLabel - gymName2: JLabel - gymName3: JLabel - gymName4: JLabel - gymName5: JLabel - gymName6: JLabel - nameLabel1: JLabel - nameLabel2: JLabel - nameLabel3: JLabel

- nameLabel4: JLabel
- nameOutputLabel1: JLabel
- nameOutputLabel2: JLabel
- idLabel1: JLabel
- idOutputLabel1: JLabel
- idOutputLabel2: JLabel
- idLabel2: JLabel
- idLabel3: JLabel
- idLabel4: JLabel
- locationLabel1: JLabel
- locationLabel2: JLabel
- locationOutputLabel1: JLabel
- locationOutputLabel2: JLabel
- locationLabel3: JLabel
- locationLabel4: JLabel
- phoneNoLabel1: JLabel
- phoneNoLabel2: JLabel
- phoneNoLabel3: JLabel
- phoneNoLabel4: JLabel
- phoneOutputLabel1: JLabel
- phoneOutputLabel2: JLabel
- emailLabel1: JLabel
- emailLabel2: JLabel
- emailLabel3: JLabel
- emailLabel4: JLabel
- emailOutputLabel1: JLabel
- emailOutputLabel2: JLabel
- genderLabel1: JLabel
- genderLabel2: JLabel
- genderLabel3: JLabel
- genderLabel4: JLabel

- genderOutputLabel1: JLabel
- genderOutputLabel2: JLabel
- dateOfBirthLabel1: JLabel
- dateOfBirthLabel2: JLabel
- dateOfBirthLabel3: JLabel
- dateOfBirthLabel4: JLabel
- dobOutputLabel1: JLabel
- dobOutputLabel2: JLabel
- startDateLabel1: JLabel
- startDateLabel: JLabel
- startDateLabel3: JLabel
- startDateLabel4: JLabel
- startDateOutputLabel1: JLabel
- startDateOutputLabel2: JLabel
- attendanceLabel1: JLabel
- attendanceLabel2: JLabel
- attendanceOutputLabel1: JLabel
- attendanceOutputLabel2: JLabel
- loyaltyPointsLabel1: JLabel
- loyaltyPointsLabel2: JLabel
- loyaltyPointsOutputLabel1: JLabel
- loyaltyPointsOutputLabel2: JLabel
- planLabel1: JLabel
- planLabel2: JLabel
- planOutputLabel: JLabel
- priceLabel2: JLabel
- priceOutputLabel: JLabel
- referralSourceLabel1: JLabel
- referralSourceLabel2: JLabel
- referralSourceOutputLabel: JLabel
- removalReasonLabel: JLabel

- removalReasonOutputLabel: JLabel
- removalReasonLabel2: JLabel
- paidAmountLabel: JLabel
- paidAmountOutputLabel: JLabel
- discountLabel: JLabel
- discountOutputLabel: JLabel
- fullPaymentLabel: JLabel
- fullPaymentOutputLabel: JLabel
- personalTrainerLabel2: JLabel
- personalTrainerLabel: JLabel
- personalTrainerOutputLabel2: JLabel
- totalChargeLabel: JLabel
- totalChargeOutputLabel: JLabel
- addAMemberButton1: JButton
- addAMemberButton2: JButton
- addAMemberButton3: JButton
- membershipDetailsButton1: JButton
- membershipDetailsButton2: JButton
- membershipDetailsButton3: JButton
- allMembersButton1: JButton
- allMembersButton2: JButton
- allMembersButton3: JButton
- allMembersRegularButton1: JButton
- allMembersRegularButton2: JButton
- allMembersPremiumButton: JButton
- allMembersPremiumButton2: JButton
- regularMemberInfoButton1: JButton
- regularMemberInfoButton2: JButton
- premiumMemberInfoButton1: JButton
- premiumMemberInfoButton2: JButton
- addARegularMemberButtonPanel1: JButton

- addAPremiumMemberButtonPanel1: JButton
- addARegularMemberButtonPanel2: JButton
- addAPremiumMemberButtonPanel2: JButton
- addARegularMemberButtonBottom: JButton
- addAPremiumMemberButtonBottom: JButton
- activateButton1: JButton
- activateButton2: JButton
- deactivateButton1: JButton
- deactivateButton2: JButton
- upgradePlanButton: JButton
- attendanceButton1: JButton
- attendanceButton2: JButton
- revertButton1: JButton
- revertButton2: JButton
- displayButton1: JButton
- displayButton2: JButton
- addClearButton1: JButton
- addClearButton2: JButton
- memberClearButton1: JButton
- memberClearButton2: JButton
- discountButton: JButton
- paymentButton: JButton
- saveButton1: JButton
- saveButton2: JButton
- saveToFileButton1: JButton
- saveToFileButton2: JButton
- readFromFileButton1: JButton
- readFromFileButton2: JButton
- backButton1: JButton
- backButton2: JButton
- readBackButton1: JButton

- readBackButton2: JButton
- nameText1: JTextField
- nameText2: JTextField
- locationText1: JTextField
- locationText2: JTextField
- idText1: JTextField
- idText2: JTextField
- phoneNoText1: JTextField
- phoneNoText: JTextField
- emailText1: JTextField
- emailText2: JTextField
- personalTrainerText: JTextField
- referralSourceText: JTextField
- removalReasonText: JTextField
- maleRadioButton1: JRadioButton
- femaleRadioButton1: JRadioButton
- maleRadioButton2: JRadioButton
- femaleRadioButton2: JRadioButton
- DOBYear1: JComboBox
- DOBMonth1: JComboBox
- DOBDay1: JComboBox
- startDateYear1: JComboBox
- startDateMonth1: JComboBox
- startDateDay1: JComboBox
- planComboBox: JComboBox
- DOBYear2: JComboBox
- DOBMonth2: JComboBox
- DOBDay2: JComboBox
- startDateYear2: JComboBox
- startDateMonth2: JComboBox
- startDateDay2: JComboBox

<ul style="list-style-type: none"> - displayAreaRegular: JTextArea - displayAreaPremium: JTextArea - readAreaRegular: JTextArea - readAreaPremium: JTextArea - scrollPane1: JScrollPane - scrollPane2: JScrollPane - scrollRead1: JScrollPane - scrollRead2: JScrollPane - file: File - file2: File - reader: FileReader - writer: FileWriter
<<constructor>> GymGUI() + actionPerformed(e: ActionEvent): void

Table 5: InvalidIdException Class

InvalidIdException
+<<constructor>> InvalidIdException (message: String)

Table 6: InvalidPhoneException Class

InvalidPhoneException
+<<constructor>> InvalidPhoneException (message: String)

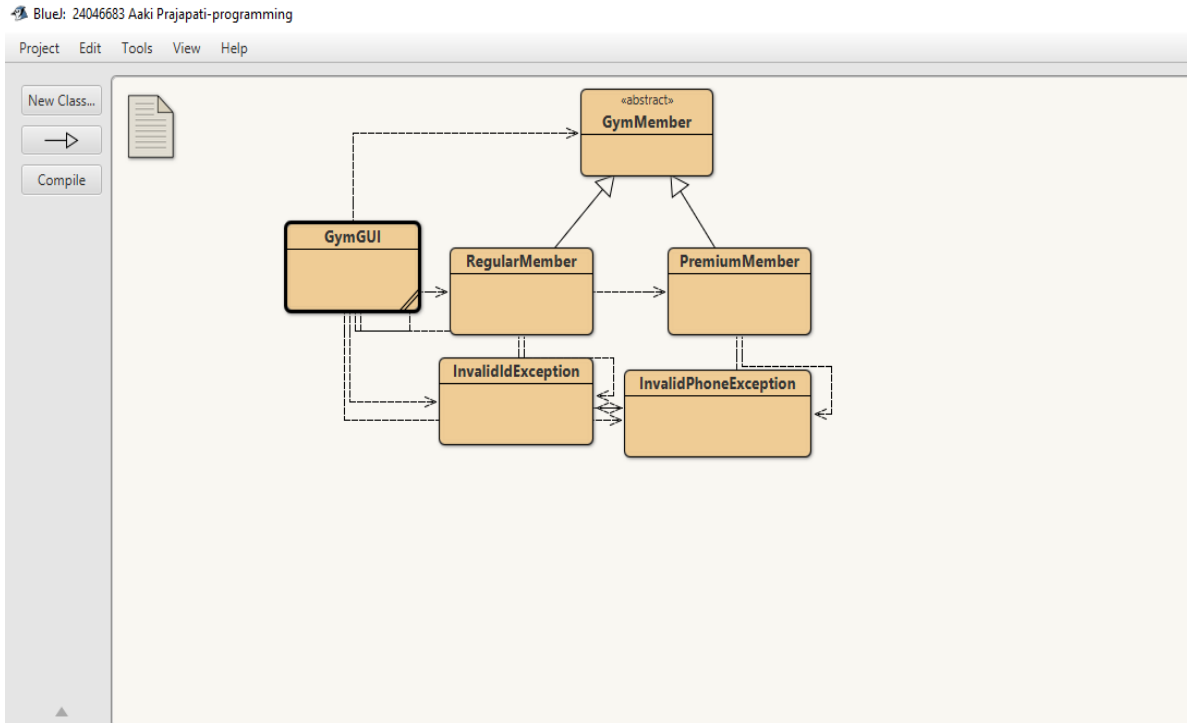


Figure 11: Screenshot to show inheritance between the classes

4. Pseudocode:

4.1 GymMember:

CREATE a parent abstract class **GymMember**

DO

DECLARE instance variable **id** as **int**

DECLARE instance variable **name** as **String**

DECLARE instance variable **location** as **String**

DECLARE instance variable **phone** as **String**

DECLARE instance variable **email** as **String**

DECLARE instance variable **gender** as **String**

DECLARE instance variable **DOB** as **String**

DECLARE instance variable **membershipStartDate** as **String**

DECLARE instance variable attendance as int

DECLARE instance variable loyaltyPoints as double

DECLARE instance variable activeStatus as Boolean

CREATE a constructor GymMember (id as int, name as String, location as String, phone as String, email as String, gender as String, DOB as String, membershipStartDate as String)

DO

SET instance variable id=id

SET instance variable name=name

SET instance variable location=location;

SET instance variable phone=phone;

SET instance variable email=email;

SET instance variable gender=gender;

SET instance variable DOB=DOB;

SET instance variable membershipStartDate=membershipStartDate;

SET instance variable attendance= 0;

SET instance variable loyaltyPoints= 0;

SET instance variable activeStatus= false;

END DO

CREATE an abstract method markAttendance() with return type void

CREATE a method activateMembership() with return type void

DO

SET activeStatus= true

DISPLAY "Your membership has been activated successfully"

END DO

CREATE a method deactivateMembership() with return type void

DO

IF activeStatus is true **THEN**

SET activeStatus= false

DISPLAY "Your membership has been deactivated successfully"

ELSE

DISPLAY "Please activate your membership first"

END IF

END DO

CREATE a method resetMember() with return type void

DO

SET activeStatus=false

SET loyaltyPoints=0

SET attendance=0

END DO

CREATE a method display() with return type void

DO

DISPLAY "The id is "+ id +"."

DISPLAY "The name is "+ name +"."

DISPLAY "The location is "+ location +"."

DISPLAY "The phone number is "+ phone +"."

DISPLAY "The email is "+ email +"."

DISPLAY "The gender is "+ gender +"."

DISPLAY "The DOB is "+ DOB +"."

DISPLAY "The membership start date is "+ membershipStartDate +"."

DISPLAY "The attendance is "+ attendance +"."

DISPLAY "The loyalty points is "+ loyaltyPoints +"."

```
        DISPLAY "The active status is "+ activeStatus +"."  
END DO  
END CLASS
```

4.2 RegularMember

CREATE a subclass RegularMember that **EXTENDS** GymMember

DO

DECLARE instance variable attendanceLimit as a constant int

DECLARE instance variable isEligibleForUpgrade as boolean

DECLARE instance variable removalReason as String

DECLARE instance variable referralSource as String

DECLARE instance variable plan as String

DECLARE instance variable price as double

CREATE a constructor RegularMember (id as int, name as String, location as String, phone as String, email as String, gender as String, DOB as String, membershipStartDate as String, referralSource as String) that throws InvalidIdException, InvalidPhoneException

DO

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

SET instance variable referralSource=referralSource

SET instance variable attendanceLimit=30

SET instance variable isEligibleForUpgrade=false

SET instance variable removalReason=" "

SET instance variable plan="Basic"


```
        SET instance variable price=6500

    END DO

    OVERRIDE method markAttendance()

    DO

        INCREASE attendance by 1

        INCREASE loyaltyPoints by 5

        IF attendance is greater than or equals to attendanceLimit THEN

            SET isEligibleForUpgrade=true

        END IF

    END DO

    DECLARE a method getPlanPrice (plan as String) with return type double

    DO

        CONVERT plan to lowercase

        SWITCH plan

        DO

            case "basic":

                RETURN 6500

            case "standard":

                RETURN 12500

            case "deluxe":

                RETURN 18500

            default:

                RETURN -1
```

```
END DO

END SWITCH

END DO

DECLARE a method upgradePlan (plan as String) with return type String
DO

    IGNORE case of plan entered

    IF isEligibleForUpgrade is false THEN

        RETURN "Sorry! Ineligible for upgrade. Attendance must be
        atleast 30!"

    ELSE

        IF current plan is equal to plan entered THEN

            RETURN "You have already subscribed to this plan."

        ELSE IF current plan is "standard" and the plan entered is "basic"
        THEN

            RETURN "You cannot upgrade from standard to basic"

        ELSE IF (current plan is "deluxe" and the plan entered is "basic")
        OR (current plan is "deluxe" and the plan entered is "standard")
        THEN

            RETURN "You cannot upgrade from deluxe to "+ plan

        ELSE

            DECLARE local variable new Price as double

            SET newPrice= getPlanPrice (plan)

            IF newPrice is -1 THEN

                RETURN "Invalid plan selected!"
```

```
        ELSE

            SET instance variable plan=plan

            SET instance variable price=price

            RETURN "You have successfully upgraded
                to"+current plan+". Thank You!"

        END IF

    END IF

END IF

END DO

CREATE a method revertRegularMember (removalReason as String)
DO

    CALL the method resetMember() from parent class

    SET instance variable removalReason= removalReason

    SET isEligibleForUpgrade= false

    SET plan= "Basic"

    SET price=6500

    DISPLAY "Your plan has successfully been reverted"

END DO

OVERRIDE method display()
DO

    CALL the method display() from parent class

    DISPLAY "The plan is "+ plan +". "

    DISPLAY "The price is "+ price +". "
```

```
        IF removalReason is not equal to (" ") THEN
            DISPLAY "The removal reason is "+ removalReason
        END IF
    END DO
END DO
END CLASS
```

4.3 PremiumMember

```
CREATE a subclass PremiumMember that EXTENDS GymMember
DO
    DECLARE instance variable premiumCharge as a constant double
    DECLARE instance variable personalTrainer as String
    DECLARE instance variable isFullPayment as boolean
    DECLARE instance variable paidAmount as double
    DECLARE instance variable discountAmount as double
    CREATE a constructor PremiumMember (id as int, name as String, location as
String, phone as String, email as String, gender as String, DOB as String,
membershipStartDate as String, personalTrainer as String) that throws
InvalidIdException, InvalidPhoneException
DO
    CALL parent constructor (id, name, location, phone, email, gender, DOB,
membershipStartDate)
    SET instance variable personalTrainer=personalTrainer
    SET instance variable premiumCharge=50000
    SET instance variable isFullPayment =false
```

```
    SET instance variable paidAmount=0

    SET instance variable discountAmount=0

END DO

OVERRIDE method markAttendance()

DO

    INCREASE attendance by 1

    INCREASE loyaltyPoints by 10

END DO

CREATE a method payDueAmount (paidAmount as int) with return type String

DO

    SET current paidAmount= current paidAmount+ paidAmount entered

    IF paidAmount is equal to premiumCharge THEN

        SET isFullPayment=true

    ELSE

        SET isFullPayment=false

    END IF

    IF isFullPayment is true THEN

        RETURN "You have paid the full payment."

    ELSE IF paidAmount> premiumCharge THEN

        DECLARE local variable extraAmount as double

        SET extraAmount= paidAmount- premiumCharge

        RETURN "The payment has exceeded the premium charge. The

        extra amount you have paid is "+extraAmount
```

ELSE

DECLARE local variable remainingAmount as double

SET remainingAmount= premiumCharge- paidAmount

RETURN "The remaining amount is "+remainingAmount

END IF

DECLARE method calculateDiscount() with return type void

DO

IF isFullPayment is true **THEN**

SET discountAmount= premiumCharge/10

DISPLAY "The discount amount is "+ discountAmount

ELSE

DISPLAY "Sorry! Discount not possible. Full payment must be done."

END IF

END DO

DECLARE a method revertPremiumMember() with return type void

DO

CALL the method resetMember() from parent class

SET personalTrainer=""

SET isFullPayment= false

SET paidAmount=0

SET discountAmount=0

DISPLAY "Your plan has successfully been reverted"

```
END DO

OVERRIDE method display()

DO

    CALL the method display() from parent class

    DISPLAY "The personal trainer is "+ personalTrainer + "."

    DISPLAY "The paid amount is "+ paidAmount + "."

    DISPLAY "The full payment is "+ isFullPayment + "."

    DECLARE local variable remainingAmount as double

    SET remainingAmount= premiumCharge- paidAmount

    DISPLAY "The remaining amount is "+remainingAmount

    IF isFullPayment is true THEN

        DISPLAY "The discount amount is "+ discountAmount + "."

    END IF

END DO

END DO

END CLASS
```

4.4 GymGUI:

```
IMPORT java.util.ArrayList
IMPORT javax.swing.*
IMPORT java.awt.*
IMPORT javax.swing.border.MatteBorder
IMPORT java.awt.event.ActionListener
IMPORT java.awt.event.ActionEvent
```

IMPORT java.io.IOException

IMPORT java.io.File

IMPORT java.io.FileWriter

IMPORT java.io.FileReader

CREATE a class GymGUI that implements ActionListener

DO

CREATE a list members that stores the objects of type GymMember

DECLARE instance variable frame as a JFrame

DECLARE instance variables addAMemberLeftPanel,

memberManageLeftPanel, allMembersLeftPanel, addARegularMemberPanel,

addAPremiumMemberPanel, regularMemberPanel, premiumMemberPanel,

regularInfoPanel, premiumInfoPanel, allMembersRegularPanel,

allMembersPremiumPanel, displayPanel1, displayPanel2, readFromFilePanel1,

readFromFilePanel2 as JPanel

DECLARE instance variables quickAccess1, quickAccess2, quickAccess3,

gymName1, gymName2, gymName3, gymName4, gymName5, gymName6,

nameLabel1, nameLabel2, nameLabel3, nameLabel4, nameOutputLabel1,

nameOutputLabel2, idLabel1, idOutputLabel1, idOutputLabel2, idLabel2,

idLabel3, idLabel4, locationLabel1, locationLabel2, locationOutputLabel1,

locationOutputLabel2, locationLabel3, locationLabel4, phoneNoLabel1,

phoneNoLabel2, phoneNoLabel3, phoneNoLabel4, phoneOutputLabel1,

phoneOutputLabel2, emailLabel1, emailLabel2, emailLabel3, emailLabel4,

emailOutputLabel1, emailOutputLabel2, genderLabel1, genderLabel2,

genderLabel3, genderLabel4, genderOutputLabel1, genderOutputLabel2,
 dateOfBirthLabel1, dateOfBirthLabel2, dateOfBirthLabel3, dateOfBirthLabel4,
 dobOutputLabel1, dobOutputLabel2, startDateLabel1, startDateLabel2,
 startDateLabel3, startDateLabel4, startDateOutputLabel1,
 startDateOutputLabel2, attendanceLabel1, attendanceLabel2,
 attendanceOutputLabel1, attendanceOutputLabel2, loyaltyPointsLabel1,
 loyaltyPointsLabel2, loyaltyPointsOutputLabel1, loyaltyPointsOutputLabel2,
 planLabel1, planLabel2, planOutputLabel, priceLabel2, priceOutputLabel,
 referralSourceLabel1, referralSourceLabel2, referralSourceOutputLabel,
 removalReasonLabel, removalReasonOutputLabel, removalReasonLabel2,
 paidAmountLabel, paidAmountOutputLabel, discountLabel, discountOutputLabel,
 fullPaymentLabel, fullPaymentOutputLabel, personalTrainerLabel2,
 personalTrainerLabel, personalTrainerOutputLabel2, totalChargeLabel,
 totalChargeOutputLabel as JLabel

DECLARE instance variables addAMemberButton1, addAMemberButton2,
 addAMemberButton3, membershipDetailsButton1, membershipDetailsButton2,
 membershipDetailsButton3, allMembersButton1, allMembersButton2,
 allMembersButton3, allMembersRegularButton1, allMembersRegularButton2,
 allMembersPremiumButton1, allMembersPremiumButton2,
 regularMemberInfoButton1, regularMemberInfoButton2,
 premiumMemberInfoButton1, premiumMemberInfoButton2,
 addARegularMemberButtonPanel1, addAPremiumMemberButtonPanel1,
 addARegularMemberButtonPanel2, addAPremiumMemberButtonPanel2,

addARegularMemberButtonBottom, addAPremiumMemberButtonBottom,
activateButton1, activateButton2, deactivateButton1, deactivateButton2,
upgradePlanButton, attendanceButton1, attendanceButton2, revertButton1,
revertButton2, displayButton1, displayButton2, addClearButton1,
addClearButton2, memberClearButton1, memberClearButton2, discountButton,
paymentButton, saveButton1, saveButton2, saveToFileButton1,
saveToFileButton2, readFromFileButton1, readFromFileButton2, backButton1,
backbutton2, readBackButton1, readBackButton2 as JButton

DECLARE instance variables nameText1, nameText2, locationText1,
locationText2, idText1, idText2, phoneNoText1, phoneNoText2, emailText1,
emailText2, personalTrainerText, referralSourceText, removalReasonText as
JTextField

DECLARE instance variables maleRadioButton1, femaleRadioButton1,
maleRadioButton2, femaleRadioButton2 as JRadioButton

DECLARE instance variables DOBYear1, DOBMonth1, DOBDay1,
startDateYear1, startDateMonth1, startDateDay1, planComboBox, DOBYear2,
DOBMonth2, DOBDay2, startDateYear2, startDateMonth2, startDateDay2 as
JComboBox

DECLARE instance variables displayAreaRegular, displayAreaPremium,
readAreaRegular, readAreaPremium as JTextArea

DECLARE instance variables scrollPane1, scrollPane2, scrollRead1,
scrollRead2 as JScrollPane

DECLARE instance variables file, file2 as File

DECLARE instance variable reader as FileReader

DECLARE instance variable writer as FileWriter

DECLARE a method labelFonts(JLabel label) with return type void, and that takes a variable label of type JLabel as its parameter

DO

SET the font for label

END DO

DECLARE a method headingButtonFonts(JButton button1) with return type void, and that takes a variable button1 of type JButton as its parameter

DO

SET the font for button1

SET focusPainted property for button1 to false

END DO

DECLARE a method otherButtonFonts(JButton button2) with return type void, and that takes a variable button1 of type JButton as its parameter

DO

SET the font for button2

SET focusPainted property for button2 to false

END DO

DECLARE a method comboBoxColour (JComboBox combo) with return type void, and that takes a variable combo of type JComboBox as its parameter

DO

SET the background color for combo

END DO

CREATE a constructor GymGUI()

DO

CREATE a new JFrame object and assign it to frame

SET the layout of frame as null (no layout manager)

SET the size of frame

CREATE JPanel objects and assign them to the instance variables of type

JPanel

SET the layout of panels as null (no layout manager), background color and border if required.

CREATE JTextArea objects and assign them to the instance variables of type

JTextArea

SET the background color, font and editable status

CREATE JLabel objects and assign them to the instance variables of type
JLabel

SET the text to be displayed on the JLabel component and apply custom styles

CREATE JButton objects and assign them to the instance variables of type
JButton

SET the text to be displayed on the JButton component and apply custom styles

REGISTER the button as an event source using addActionListener.

CREATE JRadioButton objects and assign them to the instance variables of
type JRadioButton

APPLY styles to the radiobutton.

CREATE a ButtonGroup, and add all radio buttons to this group (allowing only one radio button to be selected at once).

CREATE an array to store data items for comboboxes.

CREATE JComboBox objects and assign them to the instance variables of type JComboBox.

ASSIGN the array to each JComboBox object.

APPLY styles to the combobox.

CREATE JTextField objects and assign them to the instance variables of type JTextField.

CREATE JScrollPane objects and assign them to the instance variables of type JScrollPane.

SET the position and size of the addAMemberLeftPanel.

ADD labels and buttons to the addAMemberLeftPanel with custom styling and positions.

ADD the addAMemberLeftPanel to frame

SET the position and size of the addARegularMemberPanel.

ADD labels, buttons, textfields, radiobuttons, combo boxes, to the addARegularMemberPanel, with custom styling and positions.

ADD addARegularMemberPanel to frame.

SET the position and size of the addAPremiumMemberPanel.

ADD labels, buttons, textfields, radiobuttons, combo boxes, to the

addAPremiumMemberPanel, with custom styling and positions.

SET the position and size of the memberManageLeftPanel

styling
ADD labels and buttons to the addAMemberLeftPanel with custom
and positions.

SET the position and size of the regularMemberPanel

to
ADD labels, buttons, textfields, sub-panel which has labels, combo box
the regularMemberPanel with custom styling and positions

SET the position and size of premiumMemberPanel

ADD labels, buttons, sub-panel which has labels to the
premiumMemberPanel with custom styling and positions

SET the position and size of allMembersLeftPanel

ADD labels and buttons to the allMembersLeftPanel with custom styling
and positions.

SET the position and size of allMembersRegularPanel

ADD labels and buttons to the allMembersRegularPanel with custom
styling and positions

SET the position and size of allMembersPremiumPanel

ADD labels and buttons to the allMembersPremiumPanel with custom
styling and positions

SET the position and size of displayPanel1

ADD the required headings to the text area in proper format

ADD text area to the scroll pane

ADD scroll pane and buttons to the displayPanel1 with custom styling
and
positions

SET the position and size of readFromFilePanel1

ADD text area to the scroll pane

ADD scroll pane and buttons to the readFromFilePanel1 with custom
styling and positions

SET the position and size of displayPanel2

ADD the required headings to the text area in proper format

ADD text area to the scroll pane

ADD scroll pane and buttons to the displayPanel2 with custom styling
and
positions

SET the position and size of readFromFilePanel2

ADD text area to the scroll pane

ADD scroll pane and buttons to the readFromFilePanel2 with custom
styling and positions

TERMINATE the execution of the program upon closing the window.

SET the frame visibility to true.

SET the frame's resizable property to false.

END DO

DEFINE the method actionPerformed to handle an action event e whenever it occurs

DO

IF the event source is addAMemberButton1, addAMemberButton2 or addAMemberButton3 **THEN**

ADD addARegularMemberPanel, addAMemberLeftPanel to frame

REMOVE all other panels from frame

UPDATE and refresh the window layout

ELSE IF the event source is addARegularMemberButtonPanel1 or addARegularMemberButtonPanel2 **THEN**

ADD addARegularMemberPanel, addAMemberLeftPanel to frame

REMOVE all other panels from frame

UPDATE and refresh the window layout

ELSE IF the event source is addAPremiumMemberButtonPanel1 or addAPremiumMemberButtonPanel2 **THEN**

ADD addAPremiumMemberPanel, addAMemberLeftPanel to frame

REMOVE all other panels from frame

UPDATE and refresh the window layout

ELSE IF the event source is or membershipDetailsButton1 or membershipDetailsButton2 or membershipDetailsButton3 or regularMemberInfoButton1 or regularMemberInfoButton2 **THEN**

ADD regularMemberPanel, memberManageLeftPanel to frame

REMOVE all other panels from frame

UPDATE and refresh the window layout

TRY

ASK the user to enter their ID and convert it to integer

FOR each member in members arraylist

IF the entered ID is present in the arraylist and if the current member is a RegularMember **THEN**

DISPLAY regular member details on the labels

EXIT the method

END IF

END FOR

DISPLAY error message "Member not found" if entered ID is not present in arraylist

CATCH NumberFormatException ex

DISPLAY error message "ID must be a number" if input is not an integer value

END TRY

ELSE IF the event source is premiumMemberInfoButton1 or premiumMemberInfoButton2 **THEN**

ADD premiumMemberPanel, memberManageLeftPanel to frame

REMOVE all other panels from frame

UPDATE and refresh the window layout

TRY

ASK the user to enter their ID and convert it to integer

```
    FOR each member in members arraylist
        IF the entered ID is present in the arraylist and if the current
            member is a PremiumMember THEN
            DISPLAY premium member details on the labels
            EXIT the method
        END IF
    END FOR

    DISPLAY error message "Member not found" if entered ID is not
        present in arraylist

    CATCH NumberFormatException ex
        DISPLAY error message "ID must be a number" if input ID is not
an
        integer value

    END TRY

ELSE IF the event source is addARegularMemberButtonBottom THEN
    GET the values entered in the text fields, radio buttons and combo box
in
    the form of String.

    IF the text fields are empty THEN
        DISPLAY error message "Please fill out all the fields."
    END IF

    TRY

        ASK the user to enter their ID and convert it to integer
```

IF the entered ID not within the range 1 to 90000 **THEN**

THROW InvalidIdException

END IF

IF the length of phone number is not 10 **THEN**

THROW InvalidPhoneException

END IF

FOR each member in members arraylist

IF the entered ID is present in the arraylist and if the current member is a RegularMember **THEN**

DISPLAY error message "Member already exists"

EXIT the method

END IF

END FOR

CREATE an object of RegularMember and store all input fields in

it

ADD the new regular member to the members arraylist

SET all fields empty

DISPLAY success message "You have been added as a regular member"

CATCH NumberFormatException ex

DISPLAY error message "ID must be a number" if input ID is not

an

integer value

CATCH InvalidIdException i

DISPLAY error message about invalid ID using getMessage()

CATCH InvalidPhoneException p

DISPLAY error message about incorrect phone number length
using getMessage()

END TRY

ELSE IF the event source is addAPremiumMemberButtonBottom **THEN**

GET the values entered in the text fields, radio buttons and combo box

in

the form of String.

IF the text fields are empty **THEN**

DISPLAY error message "Please fill out all the fields."

END IF

TRY

ASK the user to enter their ID and convert it to integer

IF the entered ID not within the range 1 to 90000 **THEN**

THROW InvalidIdException

END IF

IF the length of phone number is not 10 **THEN**

THROW InvalidPhoneException

END IF

FOR each member in members arraylist

IF the entered ID is present in the arraylist and if the current

```
        member is a PremiumMember THEN

            DISPLAY error message "Member already exists"

            EXIT the method

        END IF

    END FOR

    CREATE an object of PremiumMember and store all input fields
in    it

        ADD the new premium member to the members arraylist

        SET all fields empty

        DISPLAY success message "You have been added as a premium
member"

    CATCH NumberFormatException ex

        DISPLAY error message "ID must be a number" if input ID is not
an    integer value

    CATCH InvalidIdException i

        DISPLAY error message about invalid ID using getMessage()

    CATCH InvalidPhoneException p

        DISPLAY error message about incorrect phone number length
using getMessage()

    END TRY

ELSE IF the event source is allMembersButton1 or allMembersButton2 or
```

allMembersButton3 or allMembersRegularButton1 or
allMembersRegularButton2 THEN

ADD allMembersRegularPanel, allMembersLeftPanel to the frame

REMOVE all other panels from frame

UPDATE and refresh the window layout

TRY

ASK the user to enter their ID and convert it to integer

FOR each member in members arraylist

IF the entered ID is present in the arraylist and if the
current member is a RegularMember THEN

CALL display() method of each regular member

EXIT the method

END IF

END FOR

DISPLAY error message "Member not found" if entered ID is not
present in arraylist

CATCH NumberFormatException ex

DISPLAY error message "ID must be a number" if input is
not an integer value

END TRY

ELSE IF the event source is allMembersPremiumButton1,
allMembersPremiumButton2 THEN

ADD allMembersPremiumPanel, allMembersLeftPanel to the frame

REMOVE all other panels from frame

UPDATE and refresh the window layout

TRY

ASK the user to enter their ID and convert it to integer

FOR each member in members arraylist

IF the entered ID is present in the arraylist and if the current
 member is a PremiumMember **THEN**

CALL display() method of each premium member

EXIT the method

END IF

END FOR

DISPLAY error message "Member not found" if entered ID is not
 present in arraylist

CATCH NumberFormatException ex

DISPLAY error message "ID must be a number" if input is not an
 integer value

END TRY

ELSE IF the event source is addClearButton1 or addClearButton2 **THEN**

SET all text fields empty

DISPLAY confirmation message "Welcome! Please enter information
 about the new member"

ELSE IF the event source is memberClearButton1 or memberClearButton2
THEN

```
    SET all labels empty

ELSE IF the event source is attendanceButton1 THEN

    GET the value of regular member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a RegularMember THEN

            IF the active status of the member is true THEN

                CALL the markAttendance() method

                GET the value of updated attendance

                SET the value of updated attendance in the form of
                String in the label

                GET the value of updated loyalty points

                SET the value of updated loyalty points in the form
of
                String in the label

            ELSE

                DISPLAY warning message "Please activate your
                membership first"

            END IF

        END IF

    END FOR

ELSE IF the event source is attendanceButton2 THEN
```


GET the value of premium member's ID from label in the form of String

CONVERT the ID into integer

FOR each member in members arraylist

IF the entered ID is present in the arraylist and if the current member is a PremiumMember THEN

IF the active status of the member is true THEN

CALL the markAttendance() method

GET the value of updated attendance

SET the value of updated attendance in the form of String in the label

GET the value of updated loyalty points

SET the value of updated loyalty points in the form

of

String in the label

ELSE

DISPLAY warning message "Please activate your membership first"

END IF

END IF

END FOR

ELSE IF the event source is activateButton1 THEN

GET the value of regular member's ID from label in the form of String

CONVERT the ID into integer

```
FOR each member in members arraylist

    IF the entered ID is present in the arraylist and if the current
    member is a RegularMember THEN

        CALL the method activateMembership()

        DISPLAY the confirmation message "Your account has
        been
        activated"

    END IF

END FOR

ELSE IF the event source is activateButton2 THEN

    GET the value of premium member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a PremiumMember THEN

            CALL the method activateMembership()

            DISPLAY the confirmation message "Your account has
            been
            activated"

        END IF

    END FOR

ELSE IF the event source is deactivateButton1 THEN

    GET the value of regular member's ID from label in the form of String
```

```
CONVERT the ID into integer

FOR each member in members arraylist

    IF the entered ID is present in the arraylist and if the current
    member is a RegularMember THEN

        CALL the method deactivateMembership()

    END IF

END FOR

ELSE IF the event source is deactivateButton2 THEN

    GET the value of premium member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a PremiumMember THEN

            CALL the method deactivateMembership()

        END IF

    END FOR

ELSE IF the event source is saveButton1 THEN

    GET the value of regular member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a RegularMember THEN

            GET the value of removal reason from text field
```

```
        SET the retrieved value of removal reason in label

    END IF

END FOR

ELSE IF the event source is upgradePlanButton THEN

    GET the value of regular member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a RegularMember THEN

            GET the value of upgraded plan from combo box

            DISPLAY confirmation message of upgraded plan

            DISPLAY upgraded plan and respective price in labels

        END IF

    END FOR

ELSE IF the event source is revertButton1 THEN

    GET the value of regular member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a RegularMember THEN

            SET the textfields and labels empty

        END IF

    END FOR
```

```
ELSE IF the event source is revertButton2 THEN

    GET the value of premium member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a PremiumMember THEN

            SET the textfields and labels empty

        END IF

    END FOR

ELSE IF the event source is discountButton THEN

    GET the value of premium member's ID from label in the form of String

    CONVERT the ID into integer

    FOR each member in members arraylist

        IF the entered ID is present in the arraylist and if the current
        member is a PremiumMember THEN

            CALL the calculateDiscount() method of premium member

        END IF

    END FOR

ELSE IF the event source is paymentButton THEN

    GET the value of premium member's ID from label in the form of String

    CONVERT the ID into integer

    ASK the user to enter the amount they want to pay in the form of String

    CONVERT the amount into integer
```

```
FOR each member in members arraylist

    IF the entered ID is present in the arraylist and if the current
    member is a PremiumMember THEN

        CALL the payDueAmount(int paidAmount)

        DISPLAY message retrieved from the method

        DISPLAY the value of paid amount from the method
        getPaidAmount() in the label

        DISPLAY the Boolean value of whether the full payment
        has
        been done or not from getIsFullPayment() method in the
        label.

    END IF

END FOR

ELSE IF the event source is saveToFileButton1 THEN

    CREATE File object and assign them to the instance variables of type
    File along with the file name

    TRY

        IF the file does not exist THEN

            DISPLAY message "File is created"

        ELSE

            DISPLAY message "File already exists"

        END IF

    CREATE FileWriter object and assign them to the instance
```

variables of type `FileWriter`, along with the file name

"MemberDetailsRegular.txt" and permission to add the data

instead of overriding

SET proper format for the headings such as Id, Name, Location, Phone, Email, Gender, DOB, Membership Start Date, Attendance, Loyalty Points, Active Status, Plan, Price, Eligible for Upgrade, Referral Source using `String.format`

WRITE these headings in the created/ existing file

FOR each member in members arraylist

IF the member is a `RegularMember` **THEN**

GET the data of regular member using getter methods, and format it using `String.format`

WRITE these formatted data to the file

END IF

END FOR

CATCH `IOException f`

DISPLAY an error message "An error has occurred while writing. Please try again."

FINALLY

TRY

CLOSE the writer

CATCH `IOException f`

DISPLAY an error message "Sorry! File couldn't be closed"

properly. Please try again"

END TRY

ELSE IF the event source is readFromFileButton1 **THEN**

ADD readFromFilePanel1

REMOVE all other panels

UPDATE and refresh the window layout

TRY

CREATE FileReader object and assign them to the instance

variables of type FileReader along with the file name

DECLARE a variable ch of type int to store ASCII values

WHILE it is not the end of the file

CONVERT the ASCII value in the form of integer into char

to

get the value stored in it

ADD the characters to the textfield in the form of String

END WHILE

CATCH IOException f

DISPLAY error message "An error has occurred while reading.

Please try again."

FINALLY

TRY

CLOSE the writer

CATCH IOException f

DISPLAY an error message "Sorry! File couldn't be closed properly. Please try again"

END TRY

ELSE IF the event source is saveToFileButton2 **THEN**

CREATE File object and assign them to the instance variables of type

File along with the file name "MemberDetailsPremium.txt"

TRY

IF the file does not exist **THEN**

DISPLAY message "File is created"

ELSE

DISPLAY message "File already exists"

END IF

CREATE FileWriter object and assign them to the instance

variables of type FileWriter, along with the file name and

permission

to add the data instead of overriding

SET proper format for the headings such as Id, Name, Location,

Phone, Email, Gender, DOB, Membership Start Date, Attendance,

Loyalty Points, Active Status, Plan, Price, Eligible for Upgrade,

Referral Source using String.format

WRITE these headings in the created/ existing file

FOR each member in members arraylist

IF the member is a RegularMember **THEN**

GET the data of regular member using getter methods, and format it using String.format

WRITE these formatted data to the file

END IF

END FOR

CATCH IOException f

DISPLAY an error message "An error has occurred while writing.
Please try again."

FINALLY

TRY

CLOSE the writer

CATCH IOException f

DISPLAY an error message "Sorry! File couldn't be closed
properly. Please try again"

END TRY

ELSE IF the event source is readFromFileButton2 **THEN**

ADD readFromFilePanel2

REMOVE all other panels

UPDATE and refresh the window layout

TRY

CREATE FileReader object and assign them to the instance
variables of type FileReader, along with the file name

DECLARE a variable ch of type int to store ASCII values

WHILE it is not the end of the file

CONVERT the ASCII value in the form of integer into char

to

 get the value stored in it

ADD the characters to the textfield in the form of String

END WHILE

CATCH IOException f

DISPLAY error message "An error has occurred while reading.
Please try again."

FINALLY

TRY

CLOSE the writer

CATCH IOException f

DISPLAY an error message "Sorry! File couldn't be closed
properly. Please try again"

END TRY

ELSE IF the event source is displayButton1 **THEN**

ADD displayPanel1 to the frame

REMOVE all other panels from the frame

UPDATE and refresh the window layout

FOR each member in members arraylist

IF the member is a RegularMember **THEN**

GET the data of regular member using getter methods, and

format it using String.format

ADD the obtained values to the text area in proper format

END IF

END FOR

ELSE IF the event source is displayButton2 THEN

ADD displayPanel2 to the frame

REMOVE all other panels from the frame

UPDATE and refresh the window layout

FOR each member in members arraylist

IF the member is a PremiumMember THEN

GET the data of premium member using getter methods,
and

format it using String.format

ADD the obtained values to the text area in proper format

END IF

END FOR

ELSE IF the event source is backButton1 or readBackButton1 THEN

ADD regularMemberPanel, memberManageLeftPanel to the frame

REMOVE all other panels from the frame

UPDATE and refresh the window layout

ELSE IF the event source is backButton2 or readBackButton2 THEN

ADD premiumMemberPanel, memberManageLeftPanel to the frame

REMOVE all other panels from the frame

UPDATE and refresh the window layout

END IF

END DO

CREATE the main method

CREATE an object of the GymGUI() to call the constructor, for all processes to

begin on

END DO

END CLASS

4.5 InvalidIdException:

CREATE CLASS InvalidIdException that extends Exception

DO

CREATE constructor InvalidIdException with message of type String as

parameter

DO

CALL the parent class Exception constructor with the message entered

in

the current class's constructor

END DO

END DO

END CLASS

4.6 InvalidPhoneException

CREATE CLASS InvalidPhoneException that extends Exception

DO

CREATE constructor InvalidPhoneException with message of type String as
parameter

DO

CALL the parent class Exception constructor with the message entered
in
the current class's constructor

END DO

END DO

END CLASS

5. Method Description:

5.1 GymMember:

- i. Constructor:
 - Constructor is a special type of method having the same name as the class, and no return type, and is called automatically when an object is created. Here, it is automatically invoked when an object of GymMember is created. It initializes the attributes of the object and also sets the attendance and loyalty points to 0, and active status to false by default.

ii. Getter Methods:

- The getter methods are used to return the value of a particular private attribute, so that other classes can access those values without the permission to make changes in it.

iii. public abstract void markAttendance():

- This method is an abstract method of return type void. Its access modifier is public, which means this method can be accessed by other classes. Since it is an abstract method, it must be implemented by its sub classes, i.e every sub class has its own way of marking attendance.

iv. public void activateMembership():

- This method is used to set the value of active status as true whenever called. After doing so, a confirmation message to show successful activation is printed.

v. public void deactivateMembership():

- This method checks if the active status is true. If it is true, then the active status is set to false, and a confirmation message to show successful deactivation is printed. If the membership is not activated, it asks the user to activate their membership first.

vi. public void resetMember():

- This method is used to set the values of the attributes active status to false, loyalty points to 0 and attendance to 0.

vii. public void display():

- This method is used to display all common details of both regular and premium members.

5.2 RegularMember:

i. Constructor:

- The constructor is called when an object of RegularMember is made. It firstly calls the constructor of the parent class. It sets the referral source to the one entered by the user, the attendanceLimit to 30, isEligibleForUpgrade to false, removalReason to empty, plan to "Basic" and price to 6500 by default.

ii. markAttendance():

- This method overrides the method from the GymMember (parent) class. It increases the attendance by 1 point and loyalty points by 5. If the attendance is greater or equals to attendanceLimit, then the isEligibleForUpgrade is set to true. All these operations take place when this method is called.

iii. public double getPlanPrice(String plan)

- This method takes plan entered by the user as the parameter and returns a value of double data type. The concept of switch case is used here, on the basis of the plan entered in lower case. If the plan is "basic" then 6500 is returned. Similarly, if the plan is "standard" then price 12500 and if the plan is "deluxe" then price 18500 is returned. Finally, if an invalid plan is entered, -1 is returned. This method is used to return the respective price on the basis of plan.

iv. public String upgradePlan(String plan)

- This method takes the plan entered by the user as the parameter and returns data of String data type. Here, this method is used to compare between the plans, and decide if the user can be upgraded to the plan entered. Firstly, it checks if the user is eligible to upgrade (attendance atleast 30). If not, a sorry message is returned. Whereas, if they are eligible, then further process takes place. If the user tries to upgrade to the plan that they are currently using, then the respective message is returned. Similarly, if the user tries to upgrade from standard to basic or from deluxe to standard/ basic, then the message of not possible upgrade is returned. If the plan they enter has -1 as its price (invalid plan), then this method asks the user to enter a valid plan. Finally,

if the user is suitable to be upgraded, then the success message is returned, and the user's plan and price are upgraded.

v. public void revertRegularMember(String removalReason)

- This method takes the removal reason entered by the user as a parameter and has a void data type. This method calls the parent method resetMember(), performing its actions and also performs its own unique ones. It sets the removal reason as entered by the user, isEligibleForUpgrade to false, plan to "Basic", price to 6500 and a message of successful method reversion.

vi. public void display():

- This method overrides the display() method from the GymMember (parent) class. Along with the statements from the parent class, it is used to display the plan and price. If the removal reason is not empty, then the removal reason is displayed as well.

5.3 PremiumMember:

i. Constructor:

- The constructor is called when an object of PremiumMember is made. It firstly calls the constructor of the parent class. It sets the personal trainer to the one entered by the user, premium charge to 50000, isFullPayment to false, paid amount to 0 and discount amount to 0 default.

ii. public void markAttendance()

- This method overrides the method from the GymMember (parent) class. It increases the attendance by 1 point and loyalty points by 10.

iii. public String payDueAmount(int paidAmount)

- This method takes the paid amount entered by the user as a parameter and has a String data type. Upon calling, it adds the currently paid amount to the previous paid ones. If the paid amount is equal to the premium charge, it sets isFullPayment to true.

If not, it is set to false. If `isFullPayment` is true, a success message of full payment is returned. If the paid amount is more than required, then the extra paid amount is calculated, and with suitable message is returned. Finally, in payment is not complete, then remaining amount to be paid along with suitable message is returned.

iv. `public void calculateDiscount()`

- In this method, if `isFullPayment` is true, discount amount of 10% of premium charge is calculated, and is printed along with suitable message. If not, a sorry message is printed.

v. `public void revertPremiumMember()`

- This method calls the parent method `resetMember()`, performing its actions and also performs its own unique ones. It sets the personal trainer to empty, `isFullPayment` to false, `paidAmount` to 0, `discountAmount` to 0 and prints the success message of member reversion.

vi. `public void display():`

- This method overrides the `display()` method from the `GymMember` (parent) class. Along with the statements from the parent class, it is used to display the personal trainer, paid amount, boolean value of full payment and remaining amount to be paid. If the full payment has been done, then the discount amount is printed.

5.4 GymGUI

i. Constructor:

- The constructor is called when an object of `GymGUI` is made. Within the constructor, the components of the GUI are defined, styled and positioned. The components such as frame, panels, labels, buttons, text fields, text areas, radiobuttons, combo boxes and more are given custom styles such as size, font, background color, visibility and more. Since all the visible sections are created here, it is one of the important parts of the coursework.

ii. public void actionPerformed(ActionEvent e)

- This method is performed to add functionality to the components of GUI, especially buttons. Without handling an event, the GUI developed would not make any sense since the page would be static, and decreasing user functionality. Hence, to make any web page/ application user friendly and effective, any action performed must be handled. It may include changing panels, adding text or displaying dialog boxes. The following operations were performed when action was performed (clicking a button).
1. Clicking the addAMemberButton1, addAMemberButton2, addAMemberButton3, addARegularMemberButtonPanel1 or addARegularMemberButtonPanel2 buttons:
 - When any one of these buttons were clicked, the panels addARegularMemberPanel and addAMemberLeftPanel were added to the frame, whereas all other panels were removed from the frame. The frame was updated and refreshed.
 2. Clicking the addAPremiumMemberButtonPanel1 or addAPremiumMemberButtonPanel2 button:
 - When any one of these buttons were clicked, the panels addAPremiumMemberPanel and addAMemberLeftPanel were added to the frame, whereas all other panels were removed from the frame. The frame was updated and refreshed.
 3. Clicking the membershipDetailsButton1, membershipDetailsButton2, membershipDetailsButton3, regularMemberInfoButton1 or regularMemberInfoButton2
 - When any one of these buttons were clicked, the panels regularMemberPanel and memberManageLeftPanel were added to the frame, whereas all other panels were removed from the frame. The frame was updated and refreshed. Upon clicking any of these buttons, it asked the user to enter their ID. If the ID existed in the arraylist and belonged to the RegularMember class, then all user data was set in the labels. If the user entered an ID that did not belong to the RegularMember class, an error

message was displayed. Similarly, if they entered any other data type instead of an integer while inputting ID, an error message was displayed again. The process continued till a valid and existing ID was entered.

4. Clicking the premiumMemberInfoButton1 or premiumMemberInfoButton2

- When any one of these buttons were clicked, the panels premiumMemberPanel and memberManageLeftPanel were added to the frame, whereas all other panels were removed from the frame. The frame was updated and refreshed. Upon clicking any of these buttons, it asked the user to enter their ID. If the ID existed in the arraylist and belonged to the PremiumMember class, then all user data was set in the labels. If the user entered an ID that did not belong to the PremiumMember class, an error message was displayed. Similarly, if they entered any other data type instead of an integer while inputting ID, an error message was displayed again. The process continued till a valid and existing ID was entered.

5. Clicking the addARegularMemberButtonBottom

- When this button is clicked, the data from the text fields, radio buttons and combo boxes are retrieved. If any of the text fields are left empty by the user, then a message to fill out all the fields was displayed. Then, the ID is converted into an integer. If the entered ID is less than or equal to 0 or if the ID is greater than 90000, then a custom exception InvalidIdException is thrown with the message to provide ID within range. Similarly, if the length of the phone number entered is not equal to 10, then a custom exception, InvalidPhoneException was thrown with the message to provide phone number of length 10.

Next, an object of regular member is made and is added to the arraylist. If the entered ID exists in the arraylist and belongs RegularMember, an error message of duplicate ID is displayed. If there are no more issues, the text fields are set to empty, and combo boxes to their first option. Finally, a success message to show the addition of regular member is displayed.

The exceptions that occurs when user enters another datatype instead of an integer in ID (NumberFormatException), the InvalidIdException and InvalidPhoneException are handling sequentially using catch, hence solving any further issues.

6. Clicking the addAPremiumMemberButtonBottom

- When this button is clicked, the data from the text fields, radio buttons and combo boxes are retrieved. If any of the text fields were left empty by the user, then a message to fill out all the fields is displayed. Then, the ID is converted into an integer. If the entered ID is less than or equal to 0 or if the ID is greater than 90000, then a custom exception InvalidIdException is thrown with the message to provide ID within range. Similarly, if the length of the phone number entered is not equal to 10, then a custom exception, InvalidPhoneException is thrown with the message to provide phone number of length 10.

Next, an object of premium member is made and is added to the arraylist. If the entered ID exists in the arraylist and belongs to PremiumMember, an error message of duplicate ID is displayed. If there are no more issues, the text fields are set to empty, and combo boxes to their first option. Finally, a success message to show the addition of regular member is displayed.

The exceptions that occurs when user enters another datatype instead of an integer in ID (NumberFormatException), the InvalidIdException and InvalidPhoneException were handling sequentially using catch, hence solving any further issues.

7. Clicking allMembersButton1, allMembersButton2, allMembersButton3, allMembersRegularButton1 or allMembersRegularButton2

- When any one of these buttons are clicked, the panels allMembersRegularPanel and allMembersLeftPanel are added to the frame, whereas all other panels are removed from the frame. The frame was updated and refreshed. Upon clicking any of these buttons, it asks the user to enter their ID. If the ID exists in the arraylist and belongs

to the RegularMember class, the member is casted to regular member and display() method is called to display all user details. The method is immediately exited, and finally if the member doesn't meet the condition, a message "Member not found" is displayed. Finally, if ID of invalid data type is entered, NumberFormatException is handled, and an ID of valid type is asked to enter.

8. Clicking allMembersPremiumButton1 or allMembersPremiumButton2

- When any one of these buttons are clicked, the panels allMembersPremiumPanel and allMembersLeftPanel are added to the frame, whereas all other panels are removed from the frame. The frame is updated and refreshed. Upon clicking any of these buttons, it asks the user to enter their ID. If the ID exists in the arraylist and belongs to the PremiumMember class, the member is casted to premium member and display() method is called to display all user details. The method is immediately exited, and finally if the member doesn't meet the condition, a message "Member not found" is displayed. Finally, if ID of invalid data type is entered, NumberFormatException is handled, and an ID of valid type is asked to enter.

9. Clicking addClearButton1

- When this button is clicked, all text fields are set to empty, and combo boxes to their first option. Then a success message is denoted by a welcome message to add a new member.

10. Clicking addClearButton2

- When this button is clicked, all text fields are set to empty, and combo boxes to their first option. Then a success message is denoted by a welcome message to add a new member.

11. Clicking memberClearButton1

- When this button is clicked, all labels where user data is shown are set to empty.

12. Clicking memberClearButton2

- When this button is clicked, all labels where user data is shown are set to empty.

13. Clicking attendanceButton1

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the RegularMember class, the active status is checked. If the active status is true, then the attendance() method is called. Then, the updated attendance is retrieved and is set to the label. Again, the loyalty points is also retrieved and is set to the respective label. If the active status is false, then a warning message to activate the membership first is shown.

14. Clicking attendanceButton2

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the PremiumMember class, the active status is checked. If the active status is true, then the attendance() method is called. Then, the updated attendance is retrieved and is set to the label. Again, the loyalty points is also retrieved and is set to the respective label. If the active status is false, then a warning message to activate the membership first is shown.

15. Clicking activateButton1

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the RegularMember class, then the activateMembership() method is called. Finally, a message of successful account activation is displayed.

16. Clicking activateButton2

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the PremiumMember class, then the activateMembership() method is called. Finally, a message of successful account activation is displayed.

17. Clicking deactivateButton1

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the RegularMember class, then the deactivateMembership() method is called.

18. Clicking deactivateButton2

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the PremiumMember class, then the deactivateMembership() method is called.

19. Clicking saveButton1

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the RegularMember, then the removal reason is retrieved from the text field. Finally, this removal reason is displayed to the label.

20. Clicking upgradePlanButton

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the RegularMember, then the selected plan from combo box is retrieved. The member is casted to RegularMember, and the method upgradePlan(String plan) is called with the new plan. This method's return statement is displayed through dialog box. Finally, the plan and price are displayed through the getter methods.

21. Clicking revertButton1:

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the RegularMember, the removal reason is retrieved. Then, the revertRegularMember(String removalReason) method is called with the retrieved reason as parameter. Finally, all the data items are set to their initial values such as plan to basic, price to 6500, attendance and

loyalty points to empty and other specific labels and textfields to empty too. Then the confirmation message is displayed.

22. Clicking revertButton2:

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the PremiumMember. Then, the revertPremiumMember() method is called. Finally, some labels are set as empty such as attendance, loyalty points, trainer, etc. Then the confirmation message is displayed.

23. Clicking discountButton

- When this button is clicked, the ID is retrieved from the text field and converted to integer. If the ID exists in the arraylist and belongs to the PremiumMember. Then, the calculateDiscount() method is called.

24. Clicking paymentButton

- When this button is clicked, the ID is retrieved from the text field and converted to integer. Then, the user is asked to enter the amount they want to pay. The amount is converted to integer. If the ID exists in the arraylist and belongs to the PremiumMember. Then, the payDueAmount(int paidAmount) method is called with the entered amount as parameter. This methods' statement is displayed through dialog box. Finally, the value of paid amount and IsFullPayment is retrieved and displayed in the form of String.

25. Clicking saveToFileButton1

- When this button, a file "MemberDetailsRegular.txt" is created. Whether the file exists or not, respective message is displayed. Then, to write in the file, FileWriter is used. The headings are formatted using String.format and are written to the file, with the permission to add the data instead of overriding. If the member is a RegularMember, the data of each member is displayed, again by using String.format. If there occurs any issue while writing to the file, it is handled by

IOException. Finally, the `FileWriter` is closed. Again, if there is any issue while closing the file, it is handled by `IOException`.

26. Clicking `saveToFileButton2`

- When this button, a file "`MemberDetailsPremium.txt`" is created. Whether the file exists or not, respective message is displayed. Then, to write in the file, `FileWriter` is used. The headings are formatted using `String.format` and are written to the file, with the permission to add the data instead of overriding. If the member is a `PremiumMember`, the data of each member is displayed, again by using `String.format`. If there occurs any issue while writing to the file, it is handled by `IOException`. Finally, the `FileWriter` is closed. Again, if there is any issue while closing the file, it is handled by `IOException`.

27. Clicking `readFromFileButton1`

- When this button is clicked, the panel `readFromFilePanel1` is added and all other panels are removed. The window is updated and refreshed. Then, to write in the file "`MemberDetailsRegular.txt`", `FileReader` is used. It is used to display the characters in the form of string till it is not the end of the file. If there occurs any issue while reading from file, `IOException` handles it. Finally, `FileReader` is closed. Again, if there occurs any issue while closing the file, it is handled by `IOException`.

28. Clicking `readFromFileButton2`

- When this button is clicked, the panel `readFromFilePanel2` is added and all other panels are removed. The window is updated and refreshed. Then, to write in the file "`MemberDetailsPremium.txt`", `FileReader` is used. It is used to display the characters in the form of string till it is not the end of the file. If there occurs any issue while reading from file, `IOException` handles it. Finally, `FileReader` is closed. Again, if there occurs any issue while closing the file, it is handled by `IOException`.

29. Clicking `displayButton1`

- When this button is clicked, the panel displayPanel1 is added and all other panels are removed. The window is updated and refreshed. . If the member is a RegularMember, the data of each member is displayed, again by using String.format.

30. Clicking displayButton2

- When this button is clicked, the panel displayPanel2 is added and all other panels are removed. The window is updated and refreshed. If the member is a PremiumMember, the data of each member is displayed, again by using String.format.

31. Clicking backButton1 or readBackButton1

- When any one of this button is clicked, the panel regularMemberPanel and memberManageLeftPanel is added and all other panels are removed. The window is updated and refreshed.

32. Clicking backButton2 or readBackButton2

- When any one of this this button is clicked, the panel premiumMemberPanel and memberManageLeftPanel is added and all other panels are removed. The window is updated and refreshed.

iii. main method

- This method is used to create an object of the GymGUI() constructor. Without the creation of this, the constructor would not be invoked, and hence the creation of JComponents along with their styling, fonts, backgrounds, positions, visibility, etc. would not be possible. It is used to display the components used throughout the program.

6. Testing:**6.1 Test 1:***Table 7: To compile and run the program using command prompt/ terminal*

OBJECTIVE	To compile and run the program.
ACTION	The commands: <code>javac GymGUI.java</code> and <code>java GymGUI</code> were performed in the command prompt.
EXPECTED OUTPUT	The command prompt should run the GymGUI program.
ACTUAL OUTPUT	The GymGUI program was successfully executed
RESULT	The test was successful.

Elevate Gym

Quick Access

Add a Member

Membership Details

Member Information

Regular Member | Premium Member

Name: Location: ID:

Phone No: Email: Gender: ☐ Male ☐ Female

DOB: 1995 1 01 Start Date: 2015 1 01 Referral Source:

Add a Regular Member Clear

```
Command Prompt - java GymGUI
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd C:\Users\DELL\Desktop\programming\bluej\24046683 Aaki Prajapati-programming

C:\Users\DELL\Desktop\programming\bluej\24046683 Aaki Prajapati-programming>javac GymGUI.java
Note: GymGUI.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\Users\DELL\Desktop\programming\bluej\24046683 Aaki Prajapati-programming>java GymGUI
```

Figure 12: Screenshot for Test 1

6.2 Test 2:**6.2.1 To add a regular member:***Table 8: To add a regular member*

OBJECTIVE	To add a regular member.
ACTION	All the required fields were filled and “Add a Regular Member” button was clicked.
EXPECTED OUTPUT	The regular gym member details should be successfully added which is confirmed by a dialog box.
ACTUAL OUTPUT	The regular gym member details were successfully added and a confirmation message was shown.
RESULT	The test was successful.

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular Member

Premium Member

Name:

Aaki Prajapati

Location:

KTM

ID:

abc

Phone No:

1234567890

Email:

aakii@gmail.com

Gender:

☐ Male ☒ Female

DOB:

2006

9

09

Start Date:

2022


8

01

Referral Source:

Instagram

Invalid ID

 ID must be a number!

OK

Add a Regular Member

Clear

Figure 13: Trying to add a Regular member with invalid ID

Elevate Gym

Quick Access

- Add a Member
- Membership Details
- Member Information

Regular Member | **Premium Member**

Name: Aaki Prajapati **Location:** KTM **ID:** -8

Phone No: 1234567890 **Email:** aakii@gmail.com **Gender:** ☐ Male ☒ Female

DOB: 2006 9 09 **Start Date:** 2022 8 01 **Referral Source:** Instagram

Add a Regular Member **Clear**

Invalid ID
ID must be within 1 to 90000! Please enter a valid ID.
OK

Figure 14: Trying to add a Regular member with invalid ID range

The screenshot shows a web application titled "Elevate Gym" with a sidebar on the left containing "Quick Access", "Add a Member", "Membership Details", and "Member Information". The main content area has tabs for "Regular Member" and "Premium Member". The "Regular Member" tab is active, showing a form with the following fields:

- Name: Aaki Prajapati
- Location: KTM
- ID: 1
- Phone No: 1234
- Email: aakii@gmail.com
- Gender: ☐ Male ☒ Female
- DOB: 2006 9 09
- Start Date: 2022 8 01
- Referral Source: Instagram

At the bottom right, there are two buttons: "Add a Regular Member" and "Clear". A validation error message is displayed in a dialog box:

Invalid phone number
Phone number must be of 10 digits!
OK

Figure 15: Trying to add a Regular member with valid ID but invalid phone number length

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member


Name: Aaki PrajapatiLocation: KTMID: 1

Phone No: 1234567890Email: Gender: ☐ Male ☒ Female

DOB: 1995 1 01Start Date: 2015 1 01Referral Source: Instagram

Add a Regular MemberClear

Empty field(s)

 Please fill out all the fields.

OK

Figure 16: Trying to add a Regular Member with empty field(s)

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Aaki PrajapatiLocation: KTMID: 1


Phone No: 1234567890Email: aakii@gmail.comGender: ☐ Male ☒ Female

DOB: 2006909Start Date: 2022801Referral Source: Instagram

Add a Regular Member

Clear

Message

 You have been added as a regular member!

OK

Figure 17: Successful addition of Regular Member

6.2.2 To add a premium member:*Table 9: To add a premium member*

OBJECTIVE	To add a premium member.
ACTION	All the required fields were filled and “Add a Premium Member” button was clicked.
EXPECTED OUTPUT	The premium gym member details should be successfully added which is confirmed by a dialog box.
ACTUAL OUTPUT	The premium gym member details were successfully added and a confirmation message was shown.
RESULT	The test was successful.

The screenshot shows a web application titled "Elevate Gym". On the left is a sidebar with a "Quick Access" menu containing "Add a Member", "Membership Details", and "Member Information". The main content area has a header with "Regular Member" and "Premium Member" tabs. The "Premium Member" tab is active. Below the tabs is a form with the following fields:

- Name: Aiko Shrestha
- Location: KTM
- ID: xyz
- Phone No: 9827364516
- Email: aikoo@gmail.com
- Gender: ☐ Male ☒ Female
- DOB: 2005, 1, 25
- Start Date: 2023, 7, 07
- Trainer: Aarav Dangol

At the bottom right of the form are two buttons: "Add a Premium Member" and "Clear". A modal dialog box is open in the center of the screen with the title "Invalid ID" and a close button (X). The dialog contains a red octagon icon with a white "X" and the text "ID must be a number!". There is an "OK" button at the bottom of the dialog.

Figure 18: Trying to add a Premium member with invalid ID

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Aiko Shrestha

Location: KTM

ID: -5

Phone No: 9827364516

Email: aikoo@gmail.com


Gender: ☐ Male ☒ Female

DOB: 2005125

Start Date: 2023707

Trainer: Aarav Dangol

Invalid ID

 ID must be within 1 to 90000! Please enter a valid ID.

OK

Add a Premium Member

Clear

Figure 19: Trying to add a Premium member with invalid ID range

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular Member

Premium Member

Name:

Aiko Shrestha

Location:

KTM

ID:

2

Phone No:

9827

Email:

aikoo@gmail.com

Gender:

☐ Male

☒ Female

DOB:

2005

1

25

Start Date:

2023

7

07

Trainer:

Aarav Dangol

Invalid phone number

X

Phone number must be of 10 digits!

OK

Add a Premium Member

Clear

Figure 20: Trying to add a Premium member with valid ID but invalid phone number length

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Aiko Shrestha

Location: KTM

ID: 2

Phone No: 9827364516

Email:


Gender: ☐ Male ☒ Female

DOB: 2005125

Start Date: 2023707

Trainer: Aarav Dangol

Empty field(s)

 Please fill out all the fields.

OK

Add a Premium Member

Clear

Figure 21: Trying to add a Premium Member with empty field(s)

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Aiko Shrestha

Location: KTM

ID: 2

Phone No: 9827364516

Email: aikoo@gmail.com

Gender: ☐ Male ☒ Female

DOB: 2005125


Start Date: 2023707

Trainer: Aarav Dangol

Add a Premium Member

Clear

Message

 You have been added as a premium member!

OK

Figure 22: Successful addition of Premium Member

6.3 Test 3:**6.3.1 Regular Member:***Table 10: To check mark attendance and Upgrade Plan button : Regular Member*

OBJECTIVE	To check the attendance and upgrade plan button of Regular Member
ACTION	The attendance of the regular member increased everytime “Mark Attendance” button was clicked only if membership was activated. The user was eligible for upgrade only if the attendance was atleast 30.
EXPECTED OUTPUT	The attendance of the regular member should be increased everytime “Mark Attendance” button is clicked, only if the membership is activated. The user should eligible for upgrade only if the attendance is atleast 30.
ACTUAL OUTPUT	The attendance of the regular member increased everytime “Mark Attendance” button is clicked, only if the membership is activated. The user is eligible for upgraded only when the attendance is atleast 30.
RESULT	The test was successful.

The screenshot shows a web application for 'Elevate Gym'. On the left is a sidebar with 'Quick Access' links: 'Add a Member', 'Membership Details', and 'Member Information'. The main content area is titled 'Elevate Gym' and has tabs for 'Regular Member' (selected) and 'Premium Member'. It displays member details for 'Aaki Prajapati' (ID: 1, Location: KTM, Phone No: 1234567890, Email: aakii@gmail.com, Gender: Female, DOB: 2006-9-09, Start Date: 2022-8-01). Below the details are buttons for 'Activate Membership', 'Upgrade Plan', 'Mark Attendance', and 'Revert'. A modal dialog titled 'Membership Activation' is open, displaying a warning icon and the message 'Please activate your membership first to mark attendance!' with an 'OK' button. To the right of the buttons, the member's plan is 'Basic' with a price of '6500.0' and a 'Removal Reason' field. At the bottom, there are fields for 'Attendance', 'Loyalty Points', 'Plan' (a dropdown menu currently showing 'Basic'), 'Removal Reason' (a text input field), and 'Referral Source' (set to 'Instagram'). A footer bar contains buttons for 'Clear', 'Save', 'Save to File', 'Load from File', and 'Display all Members'.

Figure 23: Trying to mark attendance without activating membership (Regular)

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Aaki PrajapatiID: 1

Location: KTMPhone No: 1234567890

Email: aakii@gmail.comGender: Female

DOB: 2006-9-09Start Date: 2022-8-01

Activate MembershipDeactivate Membership

Upgrade PlanMark Attendance

Revert

Plan: Basic

Price: 6500.0

Removal Reason:

Attendance: 1Loyalty Points: 5.0Plan: Basic

Removal Reason:Referral Source: Instagram

CloseSaveSave to FileLoad from FileDisplay all Members

BlueJ: Terminal Window - 24046683 Aaki Prajapati-programming
Options
Your membership has been activated successfully

Figure 24: Attendance after membership is activated (Regular)

Elevate Gym

Quick Access

- Add a Member
- Membership Details
- Member Information

Regular Member | **Premium Member**

Name: Aaki Prajapati ID: 1

Location: KTM Phone No: 1234567890

Email: aakii@gmail.com Gender: Female

DOB: 2006-9-09

Activate Membership Deactivate Membership

Upgrade Plan Mark Attendance

Revert

Plan: Basic

Price: 6500.0

Removal Reason:

Attendance: 1 Loyalty Points: 5.0 Plan: Standard ▼

Removal Reason: Referral Source: Instagram

Message

Sorry! Ineligible for upgrade. Attendance must be atleast 30!

OK

Figure 25: Trying to upgrade membership without enough attendance

The screenshot displays the 'Elevate Gym' web application interface. On the left is a sidebar with navigation links: 'Quick Access', 'Add a Member', 'Membership Details', and 'Member Information'. The main content area is titled 'Elevate Gym' and features a toggle for 'Regular Member' (selected) and 'Premium Member'. Member details for 'Aaki Prajapati' (ID: 1) are shown, including location (KTM), phone number (1234567890), email (aakii@gmail.com), and date of birth (2006-9-09). A modal message box states: 'You have successfully upgraded to Standard. Thank You!' with an 'OK' button. Below the details are buttons for 'Activate Membership', 'Deactivate Membership', 'Upgrade Plan', 'Mark Attendance', and 'Revert'. To the right, a summary box shows 'Plan: Basic', 'Price: 6500.0', and 'Removal Reason:'. At the bottom, 'Attendance: 31', 'Loyalty Points: 155.0', and 'Plan: Standard' (dropdown) are displayed, along with 'Removal Reason' and 'Referral Source: Instagram' fields.

Elevate Gym	
Regular Member	Premium Member
<p>Name: Aaki Prajapati ID: 1</p> <p>Location: KTM Phone No: 1234567890</p> <p>Email: aakii@gmail.com</p> <p>DOB: 2006-9-09</p>	
<p>Activate Membership Deactivate Membership</p> <p>Upgrade Plan Mark Attendance</p> <p>Revert</p>	
<p>Plan: Basic</p> <p>Price: 6500.0</p> <p>Removal Reason:</p>	
<p>Attendance: 31 Loyalty Points: 155.0 Plan: Standard</p> <p>Removal Reason: Referral Source: Instagram</p>	

Figure 26: Successful upgrade with enough attendance

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Aaki PrajapatiID: 1

Location: KTMPhone No: 1234567890

Email: aakii@gmail.com

DOB: 2006-9-09

Activate MembershipDeactivate Membership

Upgrade PlanMark Attendance

Revert

Plan: Standard

Price: 12500.0

Removal Reason:

Attendance: 31Loyalty Points: 155.0Plan: Basic

Removal Reason:Referral Source: Instagram

ClearSaveSave to FileLoad from FileDisplay all Members

Message

You cannot upgrade from Standard to Basic

OK

96

6.3.2 Premium Member:*Table 11: To check mark attendance button : Premium Member*

OBJECTIVE	To check the attendance button of Premium Member
ACTION	The attendance of the premium member increased everytime “Mark Attendance” button was clicked only if membership was activated.
EXPECTED OUTPUT	The attendance of the regular member should be increased everytime “Mark Attendance” button is clicked, only if the membership is activated.
ACTUAL OUTPUT	The attendance of the regular member increased everytime “Mark Attendance” button is clicked, only if the membership is activated
RESULT	The test was successful.

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular Member

Premium Member

Name: Aiko Shrestha

ID: 2

Location: KTM

Phone No: 9827364516

Email: aikoo@gmail.com

DOB: 2005-1-25

Activate Membership

Deactivate Membership

Discount

Mark Attendance

Payment

Revert

Trainer: Aarav Dangol

Paid Amount:

Full Payment:

Attendance:

Loyalty Points:

Total Charge: 50000

Clear


Save

Save to File

Read from File

Display all Members

Membership Activation

 Please activate your membership first to mark attendance!

OK

Figure 28: Trying to mark attendance without activating membership (Premium)

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Aiko ShresthaID: 2

Location: KTMPhone No: 9827364516

Email: aikoo@gmail.comGender: Female

DOB: 2005-1-25Start Date: 2023-7-07

Activate MembershipDeactivate Membership

DiscountMark Attendance

PaymentRevert

Trainer: Aarav Dangol

Paid Amount:

Full Payment:

Attendance: 1Loyalty Points: 10.0Total Charge: 50000

Clear

Save

Save to File

Read from File

Display all Members

Blue: Terminal Window - 24046683 Aaki Prajapati-programming

Options

Your membership has been activated successfully

Your membership has been activated successfully

Figure 29: Attendance after membership is activated (Premium)

6.4 Test 4:*Table 12: To check functionality of discount, payment and revert buttons*

OBJECTIVE	To check the discount, payment and revert buttons of Premium Member
ACTION	The payment button allowed the user to pay the amount they want to. The discount button only provided discount when full payment was done. The revert button reverted the membership details.
EXPECTED OUTPUT	The user should be able to pay the amount they want to. The user should be able to receive a discount only if full payment has been done. The user should be able to revert their membership anytime.
ACTUAL OUTPUT	The user was able to pay the amount they want to. The user was able to receive a discount only if full payment has been done. The user was able to revert their membership anytime.
RESULT	The test was successful.

The screenshot displays the 'Elevate Gym' application window. On the left is a sidebar with 'Quick Access' and buttons for 'Add a Member', 'Membership Details', and 'Member Information'. The main area is titled 'Elevate Gym' and has tabs for 'Regular Member' and 'Premium Member'. The 'Regular Member' tab is active, showing details for member Aiko Shrestha (ID: 2). A modal dialog titled 'Input' is open, prompting the user to 'Enter the amount you want to pay.' with a text box containing '10000' and 'OK'/'Cancel' buttons. Below the member details are buttons for 'Activate Membership', 'Deactivate Membership', 'Discount', 'Mark Attendance', 'Payment', and 'Revert'. To the right of these buttons is a section for 'Aarav Dangol' with 'Paid Amount:' and 'Full Payment:' labels. At the bottom, there are labels for 'Attendance:', 'Loyalty Points:', and 'Total Charge: 50000', followed by a row of buttons: 'Clear', 'Save', 'Save to File', 'Read from File', and 'Display all Members'.

Elevate Gym	
Regular Member	Premium Member
Name: Aiko Shrestha	ID: 2
Location: KTM	Phone No: 9827364516
Email: aikoo@gmail.com	Gender: Female
DOB: 2005-1-25	Start Date: 2022-7-07
Activate Membership	Deactivate Membership
Discount	Mark Attendance
Payment	Revert
Aarav Dangol	
Paid Amount:	
Full Payment:	
Attendance:	Loyalty Points:
Total Charge: 50000	
Clear	Save
Save to File	Read from File
Display all Members	

Figure 30: Allowing the user to pay amount

The screenshot shows a web application titled "Elevate Gym". On the left is a sidebar with "Quick Access" containing links for "Add a Member", "Membership Details", and "Member Information". The main content area is divided into "Regular Member" and "Premium Member" tabs. The "Regular Member" tab is active, displaying details for a member named Aiko Shrestha (ID: 2). The details include Location: KTM, Phone No: 9827364516, Email: aikoo@gmail.com, Gender: Female, DOB: 2005-1-25, and Start Date: 2023-7-07. Below the details are buttons for "Activate Membership", "Discount", "Payment", "Mark Attendance", and "Revert". A message dialog box is open in the center, displaying the text "The remaining amount is 40000.0" with an "OK" button. To the right of the buttons, there are fields for "Paid Amount:" and "Full Payment:". At the bottom, there are fields for "Attendance:", "Loyalty Points:", and "Total Charge: 50000". At the very bottom are buttons for "Clear", "Save", "Save to File", "Read from File", and "Display all Members".

Figure 31: Showing remaining amount

```

Blue: Terminal Window - 24046683 Aaki Prajapati-programming
Options
Your membership has been activated successfully
Sorry! Discount not possible. Full payment must be done.

```

Figure 32: Trying to calculate discount without full payment

The screenshot displays the 'Elevate Gym' application window. On the left is a sidebar with navigation options: 'Quick Access', 'Add a Member', 'Membership Details', and 'Member Information'. The main area is divided into 'Regular Member' and 'Premium Member' tabs. The 'Regular Member' tab is active, showing details for a member named Aiko Shrestha. A modal dialog titled 'Input' is open, prompting the user to 'Enter the amount you want to pay.' with a text input field containing '40000' and 'OK'/'Cancel' buttons. The background shows the member's details (Name, ID, Location, Phone No, Email, Gender, DOB), a grid of action buttons (Activate Membership, Discount, Mark Attendance, Payment, Revert), and a summary section with 'Attendance', 'Loyalty Points', and 'Total Charge' (50000). At the bottom are buttons for 'Clear', 'Save', 'Save to File', 'Read from File', and 'Display all Members'.

Elevate Gym	
Regular Member	Premium Member
Name: Aiko Shrestha	ID: 2
Location: KTM	Phone No: 9827364516
Email: aikoo@gmail.com	Gender: Female
DOB: 2005-1-25	2023-7-07
Activate Membership	Discount
Discount	Mark Attendance
Payment	Revert
Aarav Dangol	
Paid Amount: 10000.0	
Full Payment: false	
Attendance:	Loyalty Points:
Total Charge: 50000	
Clear	Save
Save to File	Read from File
Display all Members	

Figure 33: Paying remaining amount

The screenshot shows a web application for 'Elevate Gym'. On the left is a sidebar with 'Quick Access' links: 'Add a Member', 'Membership Details' (selected), and 'Member Information'. The main content area is divided into 'Regular Member' and 'Premium Member' tabs. The 'Regular Member' tab is active, displaying details for member Aiko Shrestha (ID: 2). The details include Name, Location (KTM), Phone No (9827364516), Email (aikoo@gmail.com), Gender (Female), DOB (2005-1-25), and Start Date (2023-7-07). Below the details are buttons for 'Activate Membership', 'Deactivate Membership', 'Discount', 'Mark Attendance', 'Payment', and 'Revert'. A modal dialog titled 'Message' is open, displaying the message 'You have paid the full payment.' with an 'OK' button. To the right of the modal, the name 'Aarav Dangol' is visible. Below the buttons, there are fields for 'Attendance:', 'Loyalty Points:', and 'Total Charge: 50000'. At the bottom are buttons for 'Clear', 'Save', 'Save to File', 'Read from File', and 'Display all Members'.

Figure 34: Dialog box to show full payment

BlueJ: Terminal Window - 24046683 Aaki Prajapati-programming

Options

```
Your membership has been activated successfully
Sorry! Discount not possible. Full payment must be done.
The discount amount is 5000.0
```

Figure 35: Calculated discount amount after full payment

Elevate Gym	
Quick Access	Regular Member
Add a Member	Premium Member
Membership Details	<p>Name: Aiko Shrestha ID: 2</p> <p>Location: KTM Phone No: 9827364516</p> <p>Email: aikoo@gmail.com Gender: Female</p> <p>DOB: 2005-1-25 Start Date: 2023-7-07</p>
Member Information	<p> <input type="button" value="Activate Membership"/> <input type="button" value="Deactivate Membership"/> </p> <p> <input type="button" value="Discount"/> <input type="button" value="Mark Attendance"/> </p> <p> <input type="button" value="Payment"/> <input type="button" value="Revert"/> </p> <p> Trainer: Aarav Dangol Paid Amount: 50000.0 Full Payment: true </p> <p> Attendance: 8 Loyalty Points: 80.0 Total Charge: 50000 </p> <p> <input type="button" value="Clear"/> <input type="button" value="Save"/> <input type="button" value="Save to File"/> <input type="button" value="Read from File"/> <input type="button" value="Display all Members"/> </p>

Figure 36: GUI before reverting membership

The screenshot displays the 'Elevate Gym' web application interface. On the left is a sidebar with navigation links: 'Quick Access', 'Add a Member', 'Membership Details', and 'Member Information'. The main content area is titled 'Elevate Gym' and features a tabbed interface with 'Regular Member' and 'Premium Member' tabs. The 'Regular Member' tab is active, showing details for a member named Aiko Shrestha. The details include Name, ID (2), Location (KTM), Phone No (9827364516), Email (aikoo@gmail.com), Gender (Female), and DOB (2005-1-25). A modal message box is overlaid on the details, stating 'Your membership has been reset.' with an 'OK' button. Below the details are buttons for 'Activate Membership', 'Deactivate Membership', 'Discount', 'Mark Attendance', 'Payment', and 'Revert'. To the right of these buttons are fields for 'Trainer:', 'Paid Amount:', and 'Full Payment:'. At the bottom, there are fields for 'Attendance:', 'Loyalty Points:', and 'Total Charge: 50000'. A row of buttons at the very bottom includes 'Clear', 'Save', 'Save to File', 'Read from File', and 'Display all Members'.

Elevate Gym	
Regular Member	Premium Member
Name: Aiko Shrestha	ID: 2
Location: KTM	Phone No: 9827364516
Email: aikoo@gmail.com	Female
DOB: 2005-1-25	2023-7-07
Activate Membership	Deactivate Membership
Discount	Mark Attendance
Payment	Revert
Trainer:	
Paid Amount:	
Full Payment:	
Attendance:	Loyalty Points:
Total Charge: 50000	
Clear	Save
Save to File	Read from File
Display all Members	

Figure 37: Confirmation message to show successful reversion

— □ ×

Elevate Gym	
Quick Access	Regular Member
Add a Member	Premium Member
Membership Details	<p>Name: Aiko Shrestha ID: 2</p> <p>Location: KTM Phone No: 9827364516</p> <p>Email: aikoo@gmail.com Gender: Female</p> <p>DOB: 2005-1-25 Start Date: 2023-7-07</p>
Member Information	<div> <div>Activate Membership</div> <div>Deactivate Membership</div> <div>Discount</div> <div>Mark Attendance</div> <div>Payment</div> <div>Revert</div> </div> <div> Trainer: Paid Amount: Full Payment: </div>
	<p>Attendance: Loyalty Points: Total Charge: 50000</p> <div> <div>Clear</div> <div>Save</div> <div>Save to File</div> <div>Read from File</div> <div>Display all Members</div> </div>

Figure 38: GUI after reverting membership

6.5 Test 5:*Table 13: To save to and read from file*

OBJECTIVE	To save user information to file and read the data to display it
ACTION	The save to file button allowed the user data to be saved in a .txt file. The read from file button allowed the data stored in the .txt file to be read and presented well.
EXPECTED OUTPUT	A new file should be created for the first time, and should be used again everytime save to file button is clicked. The file should contain all of the data of regular or premium. This data should be read and displayed in a panel.
ACTUAL OUTPUT	A new file was created for the first time, and was used again everytime save to file button is clicked. The file contained all of the data of regular or premium. This data should be read and displayed in a panel.
RESULT	The test was successful.

Elevate Gym

Quick Access

- Add a Member
- Membership Details
- Member Information

Regular Member

Name: Asta Laxmi ID: 2

Location: KTM Phone No: 9813181741

Email: asta@gmail.com Gender: Female

DOB: 1999-11-04 Start Date: 2021-7-07

Activate Membership Deactivate Membership
 Upgrade Plan Mark Attendance
 Revert

Plan: Standard

Price: 12500.0

Removal Reason:

Attendance: 32 Loyalty Points: 160.0 Plan: Standard

Removal Reason: Referral Source: Instagram

Clear Save Save to File Read from File Display all Members

Figure 39: User information to be stored (Regular)

ID	Name	Location	Phone	Email	Gender	DOB
2	Asta Laxmi	KTM	9813181741	asta@gmail.com	Female	1999-11-04

B	Membership Start Date	Attendance	Loyalty Points	ActiveStatus	Plan	Price
99-11-04	2021-7-07	32	160.0	true	Standard	12500.0

MemberDetailsRegular - Notepad

File Edit Format View Help

Loyalty Points	ActiveStatus	Plan	Price	Eligible for Upgrade	Referral Source
160.0	true	Standard	12500.0	true	Instagram

Figure 40: User data saved to file (Regular)

Back

ID	Name	Location	Phone	Email	Gender	DOB
2	Asta Laxmi	KTM	9813181741	asta@gmail.com	Female	1999-

Back

DOB	Membership Start Date	Attendance	Loyalty Points	ActiveStatus	Plan	Price
1999-11-04	2021-7-07	32	160.0	true	Standard	12500.0

Back

Loyalty Points	ActiveStatus	Plan	Price	Eligible for Upgrade	Referral Source
160.0	true	Standard	12500.0	true	Instagram

Figure 41: User data read from file (Regular)

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular Member

Premium Member

Name: Naresh Prajapati

ID: 3

Location: KTM

Phone No: 9803546301

Email: naresh@gmail.com

Gender: Male

DOB: 2001-4-01

Start Date: 2019-6-01

Activate Membership

Deactivate Membership

Discount

Mark Attendance

Payment

Revert

Trainer: Anil Dangol

Paid Amount: 45000.0

Full Payment: false

Attendance: 10

Loyalty Points: 100.0

Total Charge: 50000

Clear

Save

Save to File

Read from File

Display all Members

Figure 42: User information to be stored (Premium)

ID	Name	Location	Phone	Email	Gender	DOB
3	Naresh Prajapati	KTM	9803546301	naresh@gmail.com	Male	2001-4-01

DOB	Membership Start Date	Attendance	Loyalty Points	ActiveStatus	Full Payment	Paid Amount
2001-4-01	2019-6-01	10	100.0	true	false	45000.0

MemberDetailsPremium - Notepad

File Edit Format View Help

tendance	Loyalty Points	ActiveStatus	Full Payment	Paid Amount	Discount Amount	Premium Charge
	100.0	true	false	45000.0	0.0	50000.0

Figure 43: User data saved to file (Premium)

Back

ID	Name	Location	Phone	Email	Gender	DOB
3	Naresh Prajapati	KTM	9803546301	naresh@gmail.com	Male	2001-4-01

Back

DOB	Membership Start Date	Attendance	Loyalty Points	ActiveStatus	Full Payment	Paid
2001-4-01	2019-6-01	10	100.0	true	false	45000.0

Back

ID	Loyalty Points	ActiveStatus	Full Payment	Paid Amount	Discount Amount	Premium Charge
3	100.0	true	false	45000.0	0.0	50000.0

Figure 44: User data read from file (Premium)

7. Error Detection and Correction:

i. Syntax Error:

- Syntax Errors occur when there is violation of the rules that a language should follow. It is similar to following wrong grammatical rules in real life. These types of errors include forgetting semi-colon at the end of a statement, wrong spellings of key words, etc. The syntax error that I came across multiple times in this coursework was unintentionally closing the class bracket before defining the main method, eventually leading to compilation error. The presence of many loops and conditions, created confusion making it easy for me to misplace the bracket.

- Error Detection:

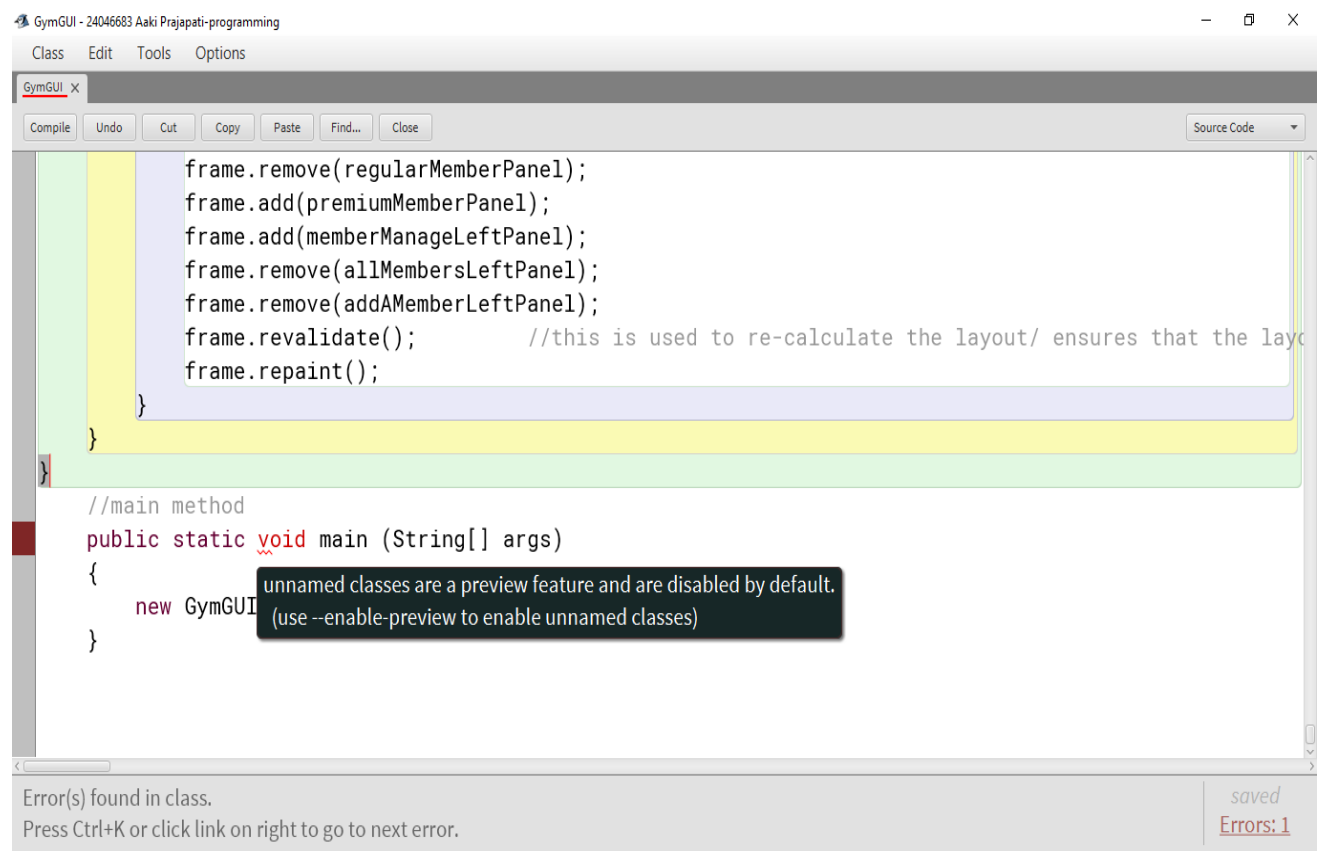


Figure 45: Syntax error detection

Upon unintentionally closing the class bracket before the GymGUI() object was made in the GymGUI class, the interface detected the error and highlighted it in red. Upon hovering over the highlighted area, the error saying unnamed class was shown. It means that the main method was not declared within any class since the class closing bracket was declared before the creation of the main method. Therefore, after learning about the error, I could successfully identify my mistake and could carefully position the closing bracket.

- Error Correction:

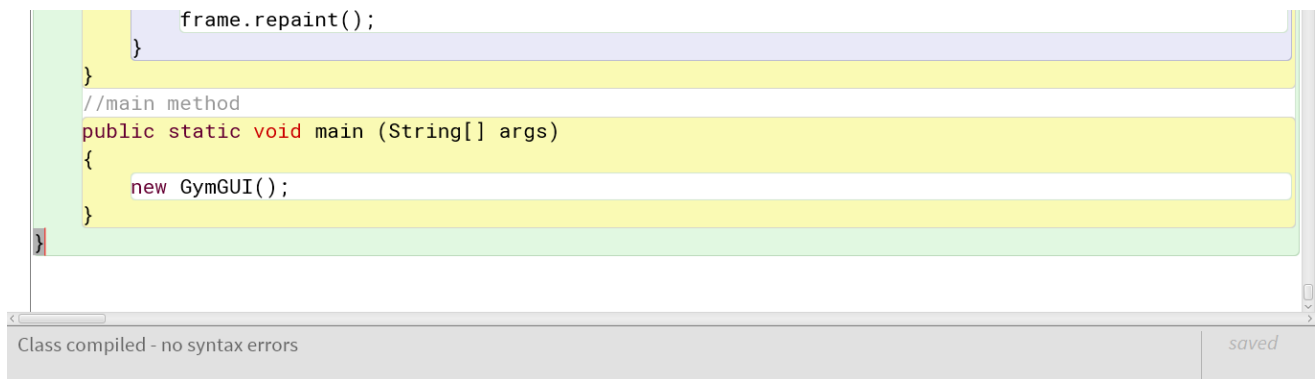


Figure 46: Syntax error correction

ii. Run-time Error:

- This type of error occurs when the program compiles successfully, but there is an issue during the run-time or execution. Some run-time errors can include dividing a number by 0, ArrayOutOfBoundsException, NumberFormatException and much more. During the execution of the GymGUI class, the run-time error that I came across was the NullPointerException. This occurred when I declared a button, but forgot to create the component using the 'new' keyword. Now, upon calling methods on this uninitialized button, the program compiled successfully but a NullPointerException was thrown.
- Error Detection:

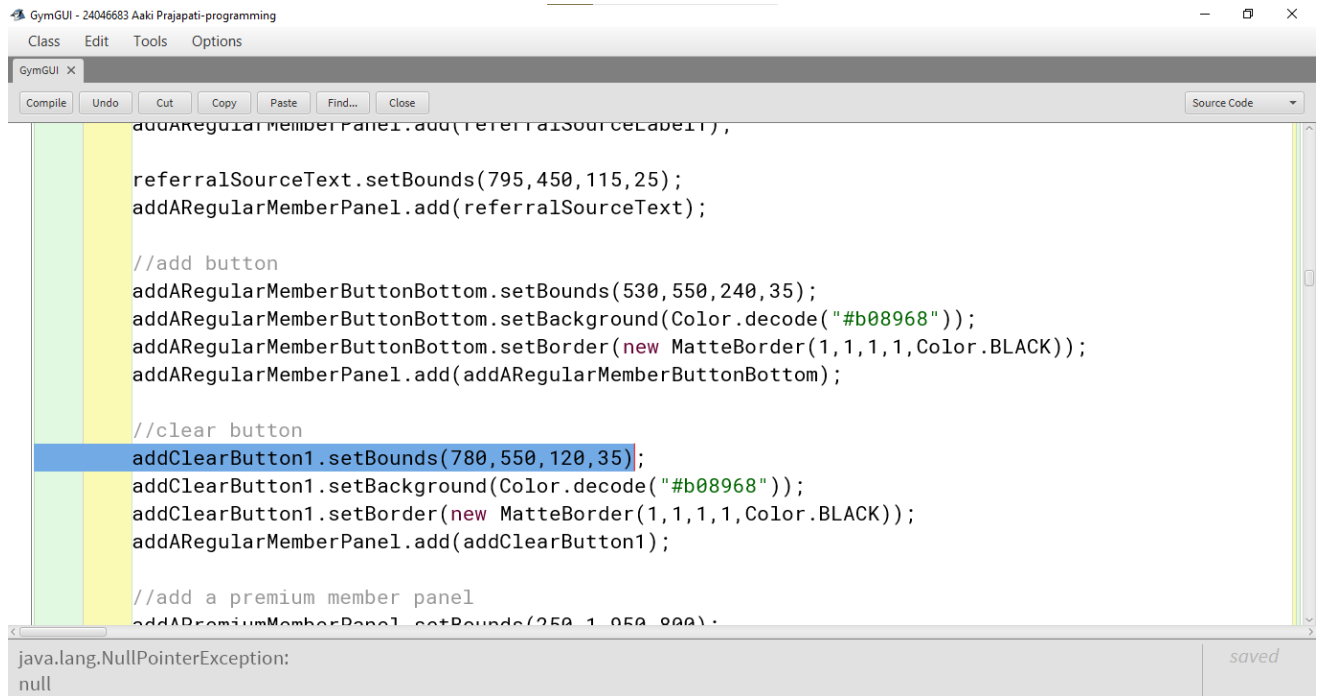


Figure 47: Runtime error detection

Here, in the GymGUI class, the addClearButton1 had not been initialized but only declared. When the method setBounds() was called upon this button, the compilation was successful. But, during program execution, the place where error might've occurred was highlighted as above, and the type of exception was mentioned below. At last the required code to create this button was written, and the program execution was successful.

- Error correction:

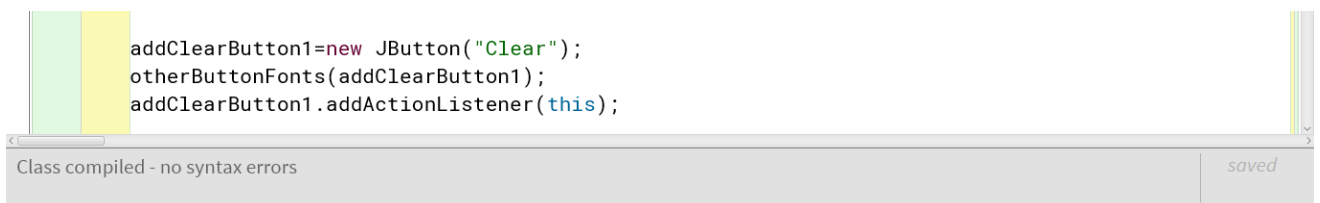
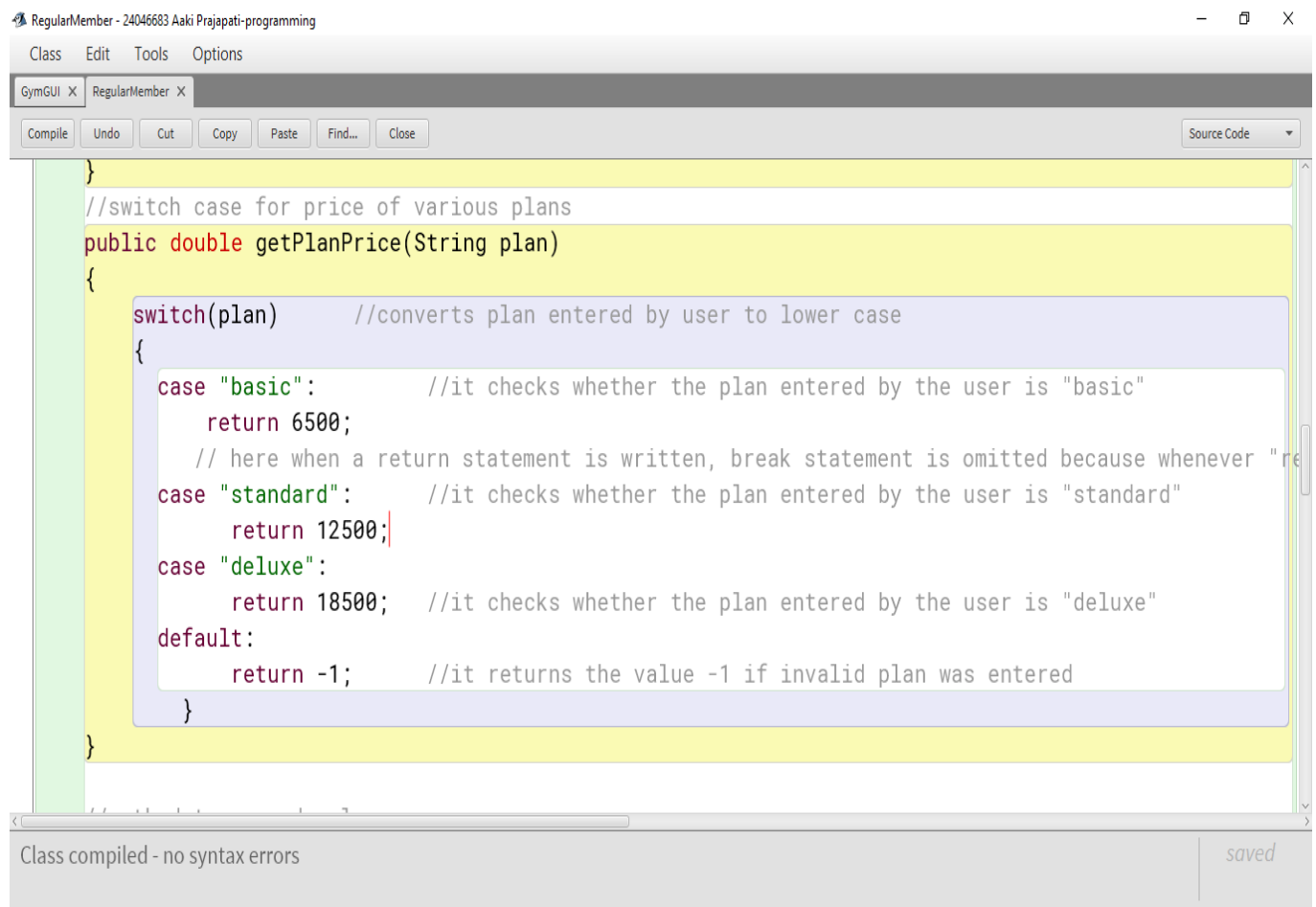


Figure 48: Runtime error correction

iii. Logical errors:

- Logical errors are the errors that have successful compilation or execution, but behave incorrectly, or have error in the logic. Such errors might occur during calculations, comparisons, or due to incorrect assumptions. During the coursework, the logical error that I faced was during the comparison between strings. Since java is a case- sensitive language, for instance the word “Basic” and “basic” are two different things. So, in the method `getPlanPrice(String plan)` due to the absence of `“toLowerCase()”` a small logical error occurred, resulting the price to be returned as - 1 even if the user has chose the correct plan “Standard” because Java understands that “Standard” is a different plan than “standard” even though it is logically correct.
- Error Detection:



The screenshot shows an IDE window titled "RegularMember - 24046683 Aaki Prajapati-programming". The code editor displays the following Java code:

```

}

//switch case for price of various plans
public double getPlanPrice(String plan)
{
    switch(plan)           //converts plan entered by user to lower case
    {
        case "basic":       //it checks whether the plan entered by the user is "basic"
            return 6500;
            // here when a return statement is written, break statement is omitted because whenever "re
        case "standard":    //it checks whether the plan entered by the user is "standard"
            return 12500;
        case "deluxe":      //it checks whether the plan entered by the user is "deluxe"
            return 18500;
        default:
            return -1;       //it returns the value -1 if invalid plan was entered
    }
}

```

The code is syntactically correct and compiles without errors, as indicated by the status bar at the bottom: "Class compiled - no syntax errors". However, there is a logical error: the `switch` statement does not convert the input string to lowercase. This means that if a user enters "Standard", the `switch` statement will not match any case and will execute the `default` case, returning -1.

Figure 49: Logical error detection

The screenshot shows a web application titled "Elevate Gym". On the left is a sidebar with "Quick Access" links: "Add a Member", "Membership Details", and "Member Information". The main content area is divided into "Regular Member" and "Premium Member" tabs. The "Regular Member" tab is active, showing details for a member named Kia Shrestha (ID: 5). The details include Location: KTM, Phone No: 9837456172, Email: kiaa@gmail.com, Gender: Female, and DOB: 2002-1-01. Below the details are buttons for "Activate Membership", "Deactivate Membership", "Upgrade Plan", "Mark Attendance", and "Revert". A modal message box is open, displaying the error "Invalid plan selected!". To the right of the buttons, there is a section for "Basic" plan with "Price: 6500.0" and a "Removal Reason:" field. At the bottom, there are fields for "Attendance: 31", "Loyalty Points: 155.0", "Plan: Standard" (dropdown), "Removal Reason:", and "Referral Source: Facebook".

Elevate Gym	
Regular Member	Premium Member
Member Details	
Name: Kia Shrestha	ID: 5
Location: KTM	Phone No: 9837456172
Email: kiaa@gmail.com	Gender: Female
DOB: 2002-1-01	Start Date: 2022-1-01
Actions	
Activate Membership	Deactivate Membership
Upgrade Plan	Mark Attendance
Revert	
Plan Details	
Basic	
Price: 6500.0	
Removal Reason:	
Attendance & Loyalty	
Attendance: 31	Loyalty Points: 155.0
Plan: Standard	
Removal Reason:	Referral Source: Facebook

Figure 50: Error message even if the plan selected is logically incorrect

Elevate Gym	
Regular Member	Premium Member
Name: Kia Shrestha	ID: 5
Location: KTM	Phone No: 9837456172
Email: kiaa@gmail.com	Gender: Female
DOB: 2002-1-01	Start Date: 2022-1-01
Activate Membership Upgrade Plan Revert	Deactivate Membership Mark Attendance Plan: Basic Price: -1.0 Removal Reason:
Attendance: 31	Loyalty Points: 155.0 Plan: Standard
Removal Reason: <input type="text"/>	Referral Source: Facebook

Figure 51: Price returned as -1 due to logical error

Here, in the RegularMember class, during the comparison between the strings, due to the absence of `.toLowerCase()` the two same words with different casing wouldn't be treated as the same. Hence the word "Standard" and "standard" would never be equal, program logic wouldn't flow as required. When the user tries to upgrade to the Standard, a message saying that the plan is invalid is displayed even though it is valid logically, and hence the price -1 is returned instead of 12500.

Now to solve this, I used the `.toLowerCase()` method causing the program convert the case of the plan selected by the user ensuring the successful comparison.

- Error Correction:

```

switch(plan.toLowerCase()) //converts plan entered by user to lower case
{
    case "basic": //it checks whether the plan entered by the user is "basic"
        return 6500;
        // here when a return statement is written, break statement is omitted because whenever "re
    case "standard": //it checks whether the plan entered by the user is "standard"
        return 12500;
    case "deluxe": //it checks whether the plan entered by the user is "deluxe"
        return 18500;
    default:
        return -1; //it returns the value -1 if invalid plan was entered
}

//method to upgrade plan
public String upgradePlan(String plan)
{

```

Class compiled - no syntax errors

Figure 52: Logical error correction

Quick Access

Add a Member

Membership Details

Member Information

Elevate Gym

Regular MemberPremium Member

Name: Kia ShresthaID: 5

Location: KTMPhone No: 9837456172

Email: kiaa@gmail.comGender: Female

DOB: 2002-1-01Start Date: 2022-1-01

Activate MembershipDe

Upgrade PlanMark Attendance

Revert

Basic

Price: 6500.0

Removal Reason:

Attendance: 31Loyalty Points: 155.0Plan: Standard

Removal Reason: Referral Source: Facebook

Figure 53: Success message of plan upgrade

The screenshot displays the 'Elevate Gym' web application interface. On the left is a sidebar with navigation links: 'Quick Access', 'Add a Member', 'Membership Details', and 'Member Information'. The main content area is titled 'Elevate Gym' and features a tabbed interface with 'Regular Member' and 'Premium Member' tabs. The 'Regular Member' tab is active, showing details for a member named 'Kia Shrestha' with ID '5'. The details include Location: 'KTM', Phone No: '9837456172', Email: 'kiaa@gmail.com', Gender: 'Female', DOB: '2002-1-01', and Start Date: '2022-1-01'. Below these details are several action buttons: 'Activate Membership', 'Deactivate Membership', 'Upgrade Plan', 'Mark Attendance', and 'Revert'. To the right of these buttons is a box containing 'Plan: Standard', 'Price: 12500.0', and 'Removal Reason:'. At the bottom, there are fields for 'Attendance: 31', 'Loyalty Points: 155.0', 'Plan: Standard' (with a dropdown arrow), 'Removal Reason: ' (with an input field), and 'Referral Source: Facebook'. At the very bottom, there are buttons for 'Clear', 'Save', 'Save to File', 'Load from File', and 'Display all Members'.

Figure 54: Correct value of price according to plan selected

Therefore, as shown in the screenshots above, the `.toLowerCase()` converted “Standard” to “standard” for its own understanding, and provided success message with required price.

8. Conclusion:

This coursework assigned to us by the Programming faculty is a simple yet functional implementation of Java programming language to develop a user-friendly and responsive application of gym membership. It was developed using the concept of OOP and its pillars, and the concepts of GUI. Being a user-friendly and responsive application, it allowed the user to be added as a Regular or Premium member, mark their attendance and hence the loyalty points and much more functionality. The system works well with buttons, and enhances the user experience. Furthermore, the coursework also validates the inputs such as ID to develop a logically sensible application. In short, the gym application is focused on increasing the user experience by maintaining data accuracy and preventing data duplication.

Through this project we not only got a chance to showcase our knowledge, but also got a chance to surf new information and learn to apply it. Apart from the basic and most important topic taught to us in college, this coursework required more research and learning for its completion. This coursework expanded our knowledge to a wide extent. From knowing what the concepts of OOP are and how they could be implemented, to developing a fully functional GUI the level of understanding and knowledge can be seen and felt. This coursework taught us as a beginner to implement the concepts of Java taught throughout the year, and allowed us to have an idea of how real life applications are dealt with, providing limitations to users to enter specific type of data, and display the output as the user wishes. Moreover, it also enhanced our debugging skills, and error detection skills, especially the logical ones. In short, this coursework was really helpful as well as informative, which expanded our knowledge to a greater extent and enhanced our skills to deal with big or small problems.

The coursework was full of difficulties to be overcome for the required implementation of the program. Having to develop a number of panels, labels, buttons and other components, it was difficult to keep up with all the components. There were also quite a few logical errors, where the program was compiled and executes successfully, but didn't show the intended output. Logical errors, unlike other errors are

not detected, and hence a lot of time was given to find and solve such errors. As a beginner in Java programming language, the successful execution of this coursework is a huge accomplishment for us. Starting with only basic knowledge about GUI to having to develop a functional application was a difficult step in itself. Hence difficulties were indeed faced, but the determination to develop a functional application help us overcome all such difficulties.

The coursework when first assigned had a lot of tasks to be performed, and a lot of difficulties to be overcome. However, with research, consultation and learning, these difficulties were solved too. We learnt to detect and solve all kinds of errors, and handle event sources. In conclusion a fully functional development of gym membership system was completed.

In conclusion, the coursework has been a great experience to implement our knowledge. Getting to see how the things we learn could be implemented, increased our motivation to learn and grow even more. All in all, it made us have if not a lot, but a little information about real life implementation.

9. References:

Bibliography

BlueJ, n.d. *BlueJ*. [Online]

Available at: <https://www.bluej.org/index.html>

[Accessed 09 05 2025].

Microsoft, n.d. *Microsoft Learn*. [Online]

Available at: <https://learn.microsoft.com/en-us/microsoft-365/cloud-storage-partner-program/online/branding>

[Accessed 09 05 2025].

Microsoft, n.d. *Microsoft Store*. [Online]

Available at: <https://apps.microsoft.com/detail/9msmlrh6lzf3?hl=en-US&gl=US>

[Accessed 13 05 2025].

Peldi, 2017. *Balsamiq*. [Online]

Available at: <https://balsamiq.com/company/news/balsamiq-wireframes/>

[Accessed 15 05 2025].

10. Appendix:

10.1 GymMember Class:

```
public abstract class GymMember
{
    //instance variables
    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;
}
```

```
//accessor methods
```

```
public int getId()
{
    return this.id;
}
```

```
public String getName()
{
    return this.name;
}
```

```
public String getLocation()
{
    return this.location;
}
```

```
public String getPhone()
{
    return this.phone;
}
```

```
public String getEmail()
```

```
{  
    return this.email;  
}
```

```
public String getGender()  
{  
    return this.gender;  
}
```

```
public String getDOB()  
{  
    return this.DOB;  
}
```

```
public String getMembershipStartDate()  
{  
    return this.membershipStartDate;  
}
```

```
public int getAttendance()  
{  
    return this.attendance;  
}
```

```
public double getLoyaltyPoints()  
{  
    return this.loyaltyPoints;  
}
```

```
public boolean getActiveStatus()
{
    return this.activeStatus;
}

//methods

public abstract void markAttendance(); //creates an abstract method which is
implemented in the subclasses

public void activateMembership()
{
    this.activeStatus= true;           //active status is set to true when the membership
is activated
    System.out.println("Your membership has been activated successfully");
}

public void deactivateMembership()
{
    if(this.activeStatus==true)        //sets the condition that deactivation is only
possible when active status is initially true
    {
        this.activeStatus= false;      //active status is set to false when the
membership is deactivated
        System.out.println("Your membership has been deactivated successfully");
    }
    else
    {
        System.out.println("Please activate your membership first");
    }
}
```

```
}

public void resetMember()
{
    this.activeStatus= false;           // active status is set to false when the membership
is reset
    this.loyaltyPoints=0;               // loyalty points status is set to zero when the
membership is reset
    this.attendance=0;                 // attendance is set to zero when the membership
is reset
}

//display method
public void display()
{
    System.out.println("The id is "+this.id +".");           //it displays the
id of the member
    System.out.println("The name is "+this.name +".");       //it displays
the name of the member
    System.out.println("The location is "+this.location +"."); //it displays
the location of the member
    System.out.println("The phone number is "+this.phone +"."); //it
displays the phone number of the member
    System.out.println("The email is "+this.email +".");     //it displays
the email of the member
    System.out.println("The gender is "+this.gender +".");   //it displays
the gender of the member
    System.out.println("The DOB is "+this.DOB +".");         //it displays
the dob of the member
    System.out.println("The membership start date is "+this.membershipStartDate +".");
//it displays the start date of the member
```

```

        System.out.println("The attendance is "+this.attendance +".");           //it
displays the of the member

        System.out.println("The loyalty points is "+this.loyaltyPoints +".");     //it displays
the of the member

        System.out.println("The active status is "+this.activeStatus +".");     //it
displays the of the member
    }
}

```

10.2 RegularMember Class:

```

public class RegularMember extends GymMember
{
    //instance variables
    private final int attendanceLimit;
    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)
throws InvalidIdException, InvalidPhoneException
    {
        super(id, name, location, phone, email, gender, DOB, membershipStartDate);
//calls the instance variables from the parent class
        this.referralSource=referralSource;
        this.attendanceLimit=30;
        this.isEligibleForUpgrade=false;
        this.removalReason="";
    }
}

```



```
        this.plan="Basic";
        this.price=6500;
    }

    //accessor methods
    public int getattendanceLimit()
    {
        return this.attendanceLimit;
    }

    public boolean getisEligibleForUpgrade()
    {
        return this.isEligibleForUpgrade;
    }

    public String getRemovalReason()
    {
        return this.removalReason;
    }

    public String getReferralSource()
    {
        return this.referralSource;
    }

    public String getPlan()
    {
        return this.plan;
    }

    public double getPrice()
```

```

{
    return this.price;
}

@Override
public void markAttendance()    //overriding the abstract method from parent class
{
    this.attendance++;        //increases attendance by 1
    this.loyaltyPoints+=5;    //increases loyalty points by 5
    if(this.attendance>=this.attendanceLimit)
    {
        this.isEligibleForUpgrade=true;        //the member is eligible for upgrade
        only if their attendance is equals to or greater than 30
    }
}

//switch case for price of various plans
public double getPlanPrice(String plan)
{
    switch(plan.toLowerCase())    //converts plan entered by user to lower case
    {
        case "basic":        //it checks whether the plan entered by the user is "basic"
            return 6500;

            // here when a return statement is written, break statement is omitted because
            whenever "return" is written, it immediately exits the case, effectively acting like
            "break"

        case "standard":    //it checks whether the plan entered by the user is "standard"
            return 12500;

        case "deluxe":
            return 18500;    //it checks whether the plan entered by the user is "deluxe"

        default:
            return -1;    //it returns the value -1 if invalid plan was entered
    }
}

```

```

    }
}

//method to upgrade plan
public String upgradePlan(String plan)
{
    if(this.isEligibleForUpgrade==false)
    {
        //if the user is ineligible
        return "Sorry! Ineligible for upgrade. Attendance must be atleast 30!";
    }
    else
    {
        if(this.plan.equalsIgnoreCase(plan))           //equalsIgnoreCase ignores the
case of the plan entered by the user to compare with the previous plans
        {
            //if the user enters the same plan
            return "You have already subscribed to this plan.";
        }

        else                                     if(this.plan.equalsIgnoreCase("standard")&&
plan.equalsIgnoreCase("basic"))
        {
            /*this is done to compare if the user is trying to upgrade from standard to
basic which is
            not possible*/
            return "You cannot upgrade from "+this.plan + " to "+plan;
        }
    }
}

```

```

        else if((this.plan.equalsIgnoreCase("deluxe")&&
plan.equalsIgnoreCase("basic"))||(this.plan.equalsIgnoreCase("deluxe")&&
plan.equalsIgnoreCase("standard")))
    /*this is done to compare if the user is trying to upgrade from deluxe to basic or
standard which is
    not possible*/
    {
        return "You cannot upgrade from "+this.plan + " to "+plan;
    }

    else{
        double newPrice=this.getPlanPrice(plan); /*this.getPlanPrice(plan) is used
to input plan from user, and get the respective price through the method calling,
        whose value is stored in the local variable
newPrice*/
        if(newPrice==-1){
            return "Invalid plan selected!";
        }
        else{
            this.plan=plan; //this.plan represents the instance variable whereas
plan represents the method's local variable
            this.price = newPrice; //this.price is updated with the new price according
to the user's updated plan
            return "You have successfully upgraded to "+this.plan+". Thank You!";
        }
    }
}

public void revertRegularMember(String removalReason) //asks the user to
enter removal reason

```

```
{
    super.resetMember();           //calls the resetMember method from the
GymMember class
    this.removalReason=removalReason;
    this.isEligibleForUpgrade=false;
    this.plan="Basic";
    this.price=6500;
    System.out.println("Your plan has successfully been reverted");
}

@Override
public void display()
{
    super.display();               //calls the display method from the
GymMember class
    System.out.println("The plan is "+this.plan +".");    //it displays the plan of the
member
    System.out.println("The price is "+this.price +".");  //it displays the price of the
respective plan
    if(!this.removalReason.equals(""))
    {
        System.out.println("The removal reason is "+this.removalReason);    //it
displays the removal reason only if it isn't empty
    }
}
}
```

10.3 PremiumMember Class:

```
public class PremiumMember extends GymMember
{
    private final double premiumCharge;

    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
personalTrainer) throws InvalidIdException, InvalidPhoneException
    {
        super(id, name, location, phone, email, gender, DOB, membershipStartDate);

        this.personalTrainer=personalTrainer;

        this.premiumCharge=50000;

        this.isFullPayment=false;

        this.paidAmount=0;

        this.discountAmount=0;

    }
```

```
//accessor methods

public double getPremiumCharge()
{
    return this.premiumCharge;
}

public String getPersonalTrainer()
{
    return this.personalTrainer;
}

public boolean getIsFullPayment()
{
    return this.isFullPayment;
}

public double getPaidAmount()
{
    return this.paidAmount;
}

public double getDiscountAmount()
{

```

```
    return this.discountAmount;
}
```

@Override

```
public void markAttendance()    //overriding the abstract method from parent class
{
    this.attendance++;        //increases attendance by 1
    this.loyaltyPoints+=10;    //increases loyalty points by 10
}
```

//method to determine the paid amount

```
public String payDueAmount(int paidAmount)
{
    this.paidAmount+=paidAmount;        //it adds the new paid amount to the already
paid amount
    if(this.paidAmount==this.premiumCharge)
    {
        //checking if the user has already done the full payment
        this.isFullPayment=true;
    }
    else{
        this.isFullPayment=false;
    }
}
```



```
if(this.isFullPayment==true)
{
    //when full payment is done
    return "You have paid the full payment.";
}
else if(this.paidAmount>this.premiumCharge)
{
    //when the customer pays extra amount than required
    double extraAmount=this.paidAmount-this.premiumCharge;
    return "The payment has exceeded the premium charge. The extra amount you
have paid is "+extraAmount;
}
else{
    //returning the remaining amount to be paid by the user
    double remainingAmount= this.premiumCharge-this.paidAmount;
    return "The remaining amount is "+remainingAmount;
}
}

//method to calculate discount
public void calculateDiscount()
{
```

```
if(isFullPayment==true)

{

    //showing the discount amount to the premium customers if the full payment
has been done

    this.discountAmount= this.premiumCharge/10;           // 10% of premium
charge=(10/100)*premium charge

    System.out.println("The discount amount is "+this.discountAmount);

}

else{

    System.out.println("Sorry! Discount not possible. Full payment must be done.");

}

}

public void revertPremiumMember()

{

    super.resetMember();           //it calls the resetMember method from parent class
and performs the required functions

    this.personalTrainer="";       //it sets the personal trainer to empty

    this.isFullPayment=false;      //it sets the full payment value to false

    this.paidAmount=0;             //it sets the paid amount to 0

    this.discountAmount=0;         //it sets the discount amount to 0

    System.out.println("Your plan has successfully been reverted");

}
```

```
@Override

public void display()

{

    super.display();           //it displays all the attributes of parent class

    System.out.println("The personal trainer is "+this.personalTrainer +".");

    System.out.println("The paid amount is "+this.paidAmount +".");

    System.out.println("The full payment is "+this.isFullPayment +".");

    double remainingAmount= this.premiumCharge-this.paidAmount;

    System.out.println("The remaining amount is "+remainingAmount);

    if(isFullPayment==true)

    {

        System.out.println("The discount amount is "+this.discountAmount +".");

        //it displays the discount amount only if the customer has completed the
payment

    }

}

}
```

10.4 GymGUI Class:

```

import java.util.ArrayList;    //importing arraylist from util package
import javax.swing.*;          //importing all components from swing package
import java.awt.*;             //importing all components from awt package
import javax.swing.border.MatteBorder;    //importing the border components
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.IOException;
import java.io.File;
import java.io.FileWriter;
import java.io.FileReader;

public class GymGUI implements ActionListener
{
    ArrayList<GymMember> members= new ArrayList<GymMember>();
    //creates an arraylist of the class GymMember which stores the object of the class
    GymMember and its subclasses

    //instance variables
    private JFrame frame;
    private      JPanel      addAMemberLeftPanel,      memberManageLeftPanel,
    allMembersLeftPanel,  addARegularMemberPanel,  addAPremiumMemberPanel,
    regularMemberPanel, premiumMemberPanel, regularInfoPanel, premiumInfoPanel,
    allMembersRegularPanel, allMembersPremiumPanel, displayPanel1, displayPanel2,
    readFromFilePanel1, readFromFilePanel2;
    private  JLabel  quickAccess1, quickAccess2, quickAccess3, gymName1,
    gymName2, gymName3, gymName4, gymName5, gymName6, nameLabel1,
    nameLabel2, nameLabel3, nameLabel4, nameOutputLabel1, nameOutputLabel2,
    idLabel1, idOutputLabel1, idOutputLabel2, idLabel2, idLabel3, idLabel4,
    locationLabel1, locationLabel2, locationOutputLabel1, locationOutputLabel2,
    locationLabel3, locationLabel4, phoneNoLabel1, phoneNoLabel2, phoneNoLabel3,

```

phoneNoLabel4, phoneOutputLabel1, phoneOutputLabel2, emailLabel1,
 emailLabel2, emailLabel3, emailLabel4, emailOutputLabel1, emailOutputLabel2,
 genderLabel1, genderLabel2, genderLabel3, genderLabel4, genderOutputLabel1,
 genderOutputLabel2, dateOfBirthLabel1, dateOfBirthLabel2, dateOfBirthLabel3,
 dateOfBirthLabel4, dobOutputLabel1, dobOutputLabel2, startDateLabel1,
 startDateLabel2, startDateLabel3, startDateLabel4, startDateOutputLabel1,
 startDateOutputLabel2, attendanceLabel1, attendanceLabel2,
 attendanceOutputLabel1, attendanceOutputLabel2, loyaltyPointsLabel1,
 loyaltyPointsLabel2, loyaltyPointsOutputLabel1, loyaltyPointsOutputLabel2,
 planLabel1, planLabel2, planOutputLabel, priceLabel2, priceOutputLabel,
 referralSourceLabel1, referralSourceLabel2, referralSourceOutputLabel,
 removalReasonLabel, removalReasonOutputLabel, removalReasonLabel2,
 paidAmountLabel, paidAmountOutputLabel, discountLabel, discountOutputLabel,
 fullPaymentLabel, fullPaymentOutputLabel, personalTrainerLabel2,
 personalTrainerLabel, personalTrainerOutputLabel2, totalChargeLabel,
 totalChargeOutputLabel;

private JButton addAMemberButton1, addAMemberButton2, addAMemberButton3,
 membershipDetailsButton1, membershipDetailsButton2, membershipDetailsButton3,
 allMembersButton1, allMembersButton2, allMembersButton3,
 allMembersRegularButton1, allMembersRegularButton2,
 allMembersPremiumButton1, allMembersPremiumButton2,
 regularMemberInfoButton1, regularMemberInfoButton2,
 premiumMemberInfoButton1, premiumMemberInfoButton2,
 addARegularMemberButtonPanel1, addAPremiumMemberButtonPanel1,
 addARegularMemberButtonPanel2, addAPremiumMemberButtonPanel2,
 addARegularMemberButtonBottom, addAPremiumMemberButtonBottom,
 activateButton1, activateButton2, deactivateButton1, deactivateButton2,
 upgradePlanButton, attendanceButton1, attendanceButton2, revertButton1,
 revertButton2, displayButton1, displayButton2, addClearButton1, addClearButton2,
 memberClearButton1, memberClearButton2, discountButton, paymentButton,
 saveButton1, saveButton2, saveToFileButton1, saveToFileButton2,

```

readFromFileButton1,    readFromFileButton2,    backButton1,    backButton2,
readBackButton1, readBackButton2;

    private JTextField nameText1, nameText2, locationText1, locationText2, idText1,
idText2,    phoneNoText1,    phoneNoText2,    emailText1,    emailText2,
personalTrainerText, referralSourceText, removalReasonText;

    private JRadioButton maleRadioButton1, femaleRadioButton1, maleRadioButton2,
femaleRadioButton2;

    private JComboBox DOBYear1, DOBMonth1, DOBDay1, startDateYear1,
startDateMonth1, startDateDay1, planComboBox, DOBYear2, DOBMonth2,
DOBDay2, startDateYear2, startDateMonth2, startDateDay2;

    private JTextArea displayAreaRegular, displayAreaPremium, readAreaRegular,
readAreaPremium;

    private JScrollPane scrollPane1, scrollPane2, scrollRead1, scrollRead2;
    private File file, file2;
    private FileReader reader;
    private FileWriter writer;

    //creating a method for storing font styles
    public void labelFonts(JLabel label)           //creating a reference variable "label"
for component JLabel
    {
        label.setFont(new Font("Century Gothic",Font.BOLD,18));
    }

    public void headingButtonFonts(JButton button1)
    {
        button1.setFont(new Font("Century Gothic",Font.BOLD,20));
        button1.setFocusPainted(false);           //to remove the border around text
    }

    public void otherButtonFonts(JButton button2)

```

```
{
    button2.setFont(new Font("Century Gothic",Font.BOLD,18));
    button2.setFocusPainted(false);
}

public void comboBoxColour(JComboBox combo)
{
    combo.setBackground(Color.WHITE);
}

//constructors
public GymGUI()
{
    //frame
    frame=new JFrame();
    frame.setLayout(null);           //setting layout null to use setbounds
    frame.setSize(1200,800);         //set size of frame: width, height

    //panels
    addAMemberLeftPanel=new JPanel();
    addAMemberLeftPanel.setLayout(null);
    addAMemberLeftPanel.setBackground(Color.decode("#F2E3BC"));
    //setting the background color using hexcodes
    addAMemberLeftPanel.setBorder(new MatteBorder(0,0,0,3,Color.BLACK));
    //top, left, bottom, right, color

    memberManageLeftPanel=new JPanel();
    memberManageLeftPanel.setLayout(null);
    memberManageLeftPanel.setBackground(Color.decode("#F2E3BC"));
    memberManageLeftPanel.setBorder(new MatteBorder(0,0,0,3,Color.BLACK));

    allMembersLeftPanel=new JPanel();
```

```
allMembersLeftPanel.setLayout(null);
allMembersLeftPanel.setBackground(Color.decode("#F2E3BC"));
allMembersLeftPanel.setBorder(new MatteBorder(0,0,0,3,Color.BLACK));

addARegularMemberPanel=new JPanel();
addARegularMemberPanel.setLayout(null);
addARegularMemberPanel.setBackground(Color.decode("#F2E3BC"));

addAPremiumMemberPanel=new JPanel();
addAPremiumMemberPanel.setLayout(null);
addAPremiumMemberPanel.setBackground(Color.decode("#F2E3BC"));

regularMemberPanel=new JPanel();
regularMemberPanel.setLayout(null);
regularMemberPanel.setBackground(Color.decode("#F2E3BC"));

premiumMemberPanel=new JPanel();
premiumMemberPanel.setLayout(null);
premiumMemberPanel.setBackground(Color.decode("#F2E3BC"));

allMembersRegularPanel=new JPanel();
allMembersRegularPanel.setLayout(null);
allMembersRegularPanel.setBackground(Color.decode("#F2E3BC"));

allMembersPremiumPanel=new JPanel();
allMembersPremiumPanel.setLayout(null);
allMembersPremiumPanel.setBackground(Color.decode("#F2E3BC"));

regularInfoPanel=new JPanel();
regularInfoPanel.setLayout(null);
regularInfoPanel.setBackground(Color.decode("#b08968"));
```



```
premiumInfoPanel=new JPanel();
premiumInfoPanel.setLayout(null);
premiumInfoPanel.setBackground(Color.decode("#b08968"));
```

```
displayPanel1=new JPanel();
displayPanel1.setLayout(null);
displayPanel1.setBackground(Color.decode("#F2E3BC"));
```

```
displayPanel2=new JPanel();
displayPanel2.setLayout(null);
displayPanel2.setBackground(Color.decode("#F2E3BC"));
```

```
readFromFilePanel1=new JPanel();
readFromFilePanel1.setLayout(null);
readFromFilePanel1.setBackground(Color.decode("#F2E3BC"));
```

```
readFromFilePanel2=new JPanel();
readFromFilePanel2.setLayout(null);
readFromFilePanel2.setBackground(Color.decode("#F2E3BC"));
```

```
//text area
displayAreaRegular=new JTextArea();
displayAreaRegular.setBackground(Color.decode("#F2E3BC"));
displayAreaRegular.setFont(new Font("Consolas",Font.BOLD,18));
displayAreaRegular.setEditable(false);
```

```
displayAreaPremium=new JTextArea();
displayAreaPremium.setBackground(Color.decode("#F2E3BC"));
displayAreaPremium.setFont(new Font("Consolas",Font.BOLD,18));
displayAreaPremium.setEditable(false);
```

```
readAreaRegular=new JTextArea();
readAreaRegular.setBackground(Color.decode("#F2E3BC"));
readAreaRegular.setFont(new Font("Consolas",Font.BOLD,18));
readAreaRegular.setEditable(false);

readAreaPremium=new JTextArea();
readAreaPremium.setBackground(Color.decode("#F2E3BC"));
readAreaPremium.setFont(new Font("Consolas",Font.BOLD,18));
readAreaPremium.setEditable(false);

//labels
quickAccess1=new JLabel("Quick Access");
quickAccess1.setFont(new Font("Bernard MT Condensed", Font.PLAIN,29));

quickAccess2=new JLabel("Quick Access");
quickAccess2.setFont(new Font("Bernard MT Condensed", Font.PLAIN,29));

quickAccess3=new JLabel("Quick Access");
quickAccess3.setFont(new Font("Bernard MT Condensed", Font.PLAIN,29));

//gym name
gymName1=new JLabel("Elevate Gym");
gymName1.setFont(new Font("Bernard MT Condensed", Font.PLAIN,32));

gymName2=new JLabel("Elevate Gym");
gymName2.setFont(new Font("Bernard MT Condensed", Font.PLAIN,32));

gymName3=new JLabel("Elevate Gym");
gymName3.setFont(new Font("Bernard MT Condensed", Font.PLAIN,32));
```

```
gymName4=new JLabel("Elevate Gym");  
gymName4.setFont(new Font("Bernard MT Condensed", Font.PLAIN,32));
```

```
gymName5=new JLabel("Elevate Gym");  
gymName5.setFont(new Font("Bernard MT Condensed", Font.PLAIN,32));
```

```
gymName6=new JLabel("Elevate Gym");  
gymName6.setFont(new Font("Bernard MT Condensed", Font.PLAIN,32));
```

```
//member names
```

```
nameLabel1=new JLabel("Name:");  
labelFonts(nameLabel1);      //calling the method consisting of fonts
```

```
nameLabel2=new JLabel("Name:");  
labelFonts(nameLabel2);
```

```
nameLabel3=new JLabel("Name:");  
labelFonts(nameLabel3);
```

```
nameLabel4=new JLabel("Name:");  
labelFonts(nameLabel4);
```

```
nameOutputLabel1=new JLabel("");  
labelFonts(nameOutputLabel1);
```

```
nameOutputLabel2=new JLabel("");  
labelFonts(nameOutputLabel2);
```

```
//member id
```

```
idLabel1=new JLabel("ID:");  
labelFonts(idLabel1);
```

```
idLabel2=new JLabel("ID:");  
labelFonts(idLabel2);
```

```
idLabel3=new JLabel("ID:");  
labelFonts(idLabel3);
```

```
idLabel4=new JLabel("ID:");  
labelFonts(idLabel4);
```

```
idOutputLabel1=new JLabel("");  
labelFonts(idOutputLabel1);
```

```
idOutputLabel2=new JLabel("");  
labelFonts(idOutputLabel2);
```

```
//location  
locationLabel1=new JLabel("Location:");  
labelFonts(locationLabel1);
```

```
locationLabel2=new JLabel("Location:");  
labelFonts(locationLabel2);
```

```
locationLabel3=new JLabel("Location:");  
labelFonts(locationLabel3);
```

```
locationLabel4=new JLabel("Location:");  
labelFonts(locationLabel4);
```

```
locationOutputLabel1=new JLabel("");  
labelFonts(locationOutputLabel1);
```

```
locationOutputLabel2=new JLabel("");
labelFonts(locationOutputLabel2);

//phone number
phoneNoLabel1=new JLabel("Phone No:");
labelFonts(phoneNoLabel1);

phoneNoLabel2=new JLabel("Phone No:");
labelFonts(phoneNoLabel2);

phoneNoLabel3=new JLabel("Phone No:");
labelFonts(phoneNoLabel3);

phoneNoLabel4=new JLabel("Phone No:");
labelFonts(phoneNoLabel4);

phoneOutputLabel1=new JLabel("");
labelFonts(phoneOutputLabel1);

phoneOutputLabel2=new JLabel("");
labelFonts(phoneOutputLabel2);

//email
emailLabel1=new JLabel("Email:");
labelFonts(emailLabel1);

emailLabel2=new JLabel("Email:");
labelFonts(emailLabel2);

emailLabel3=new JLabel("Email:");
```

```
labelFonts(emailLabel3);

emailLabel4=new JLabel("Email:");
labelFonts(emailLabel4);

emailOutputLabel1=new JLabel("");
labelFonts(emailOutputLabel1);

emailOutputLabel2=new JLabel("");
labelFonts(emailOutputLabel2);

//gender
genderLabel1=new JLabel("Gender:");
labelFonts(genderLabel1);

genderLabel2=new JLabel("Gender:");
labelFonts(genderLabel2);

genderLabel3=new JLabel("Gender:");
labelFonts(genderLabel3);

genderLabel4=new JLabel("Gender:");
labelFonts(genderLabel4);

genderOutputLabel1=new JLabel("");
labelFonts(genderOutputLabel1);

genderOutputLabel2=new JLabel("");
labelFonts(genderOutputLabel2);

//date of birth
```

```
dateOfBirthLabel1=new JLabel("DOB:");  
labelFonts(dateOfBirthLabel1);
```

```
dateOfBirthLabel2=new JLabel("DOB:");  
labelFonts(dateOfBirthLabel2);
```

```
dateOfBirthLabel3=new JLabel("DOB:");  
labelFonts(dateOfBirthLabel3);
```

```
dateOfBirthLabel4=new JLabel("DOB:");  
labelFonts(dateOfBirthLabel4);
```

```
dobOutputLabel1=new JLabel("");  
labelFonts(dobOutputLabel1);
```

```
dobOutputLabel2=new JLabel("");  
labelFonts(dobOutputLabel2);
```

```
//start date  
startDateLabel1=new JLabel("Start Date:");  
labelFonts(startDateLabel1);
```

```
startDateLabel2=new JLabel("Start Date:");  
labelFonts(startDateLabel2);
```

```
startDateLabel3=new JLabel("Start Date:");  
labelFonts(startDateLabel3);
```

```
startDateLabel4=new JLabel("Start Date:");  
labelFonts(startDateLabel4);
```

```
startDateOutputLabel1=new JLabel("");
labelFonts(startDateOutputLabel1);

startDateOutputLabel2=new JLabel("");
labelFonts(startDateOutputLabel2);

//attendance
attendanceLabel1=new JLabel("Attendance:");
labelFonts(attendanceLabel1);

attendanceLabel2=new JLabel("Attendance:");
labelFonts(attendanceLabel2);

attendanceOutputLabel1=new JLabel("");
labelFonts(attendanceOutputLabel1);

attendanceOutputLabel2=new JLabel("");
labelFonts(attendanceOutputLabel2);

//loyalty points
loyaltyPointsLabel1=new JLabel("Loyalty Points:");
labelFonts(loyaltyPointsLabel1);

loyaltyPointsLabel2=new JLabel("Loyalty Points:");
labelFonts(loyaltyPointsLabel2);

loyaltyPointsOutputLabel1=new JLabel("");
labelFonts(loyaltyPointsOutputLabel1);

loyaltyPointsOutputLabel2=new JLabel("");
labelFonts(loyaltyPointsOutputLabel2);
```



```
//plan
planLabel1=new JLabel("Plan:");
labelFonts(planLabel1);

planLabel2=new JLabel("Plan:");
labelFonts(planLabel2);

planOutputLabel=new JLabel("");
labelFonts(planOutputLabel);

//price
priceLabel2=new JLabel("Price:");
labelFonts(priceLabel2);

priceOutputLabel=new JLabel("");
labelFonts(priceOutputLabel);

//referral source
referralSourceLabel1=new JLabel("Referral Source:");
labelFonts(referralSourceLabel1);

referralSourceLabel2=new JLabel("Referral Source:");
labelFonts(referralSourceLabel2);

referralSourceOutputLabel=new JLabel("");
labelFonts(referralSourceOutputLabel);

//removal reason
removalReasonLabel=new JLabel("Removal Reason:");
labelFonts(removalReasonLabel);
```

```
removalReasonLabel2=new JLabel("Removal Reason:");  
labelFonts(removalReasonLabel2);
```

```
removalReasonOutputLabel=new JLabel("");  
labelFonts(removalReasonOutputLabel);
```

```
//paid amount  
paidAmountLabel=new JLabel("Paid Amount:");  
labelFonts(paidAmountLabel);
```

```
paidAmountOutputLabel=new JLabel("");  
labelFonts(paidAmountOutputLabel);
```

```
//discount  
discountLabel=new JLabel("Discount:");  
labelFonts(discountLabel);
```

```
discountOutputLabel=new JLabel("");  
labelFonts(discountOutputLabel);
```

```
//full payment  
fullPaymentLabel=new JLabel("Full Payment:");  
labelFonts(fullPaymentLabel);
```

```
fullPaymentOutputLabel=new JLabel("");  
labelFonts(fullPaymentOutputLabel);
```

```
//personal trainer  
personalTrainerLabel=new JLabel("Trainer:");  
labelFonts(personalTrainerLabel);
```

```
personalTrainerLabel2=new JLabel("Trainer:");  
labelFonts(personalTrainerLabel2);
```

```
personalTrainerOutputLabel2=new JLabel("");  
labelFonts(personalTrainerOutputLabel2);
```

```
//total charge  
totalChargeLabel=new JLabel("Total Charge:");  
labelFonts(totalChargeLabel);
```

```
totalChargeOutputLabel=new JLabel("50000");  
labelFonts(totalChargeOutputLabel);
```

```
//buttons
```

```
addAMemberButton1=new JButton("Add a Member");  
headingButtonFonts(addAMemberButton1);
```

```
addAMemberButton1.addActionListener(this);           //registering the source to  
the listener, and creating he object of the present class
```

```
addAMemberButton2=new JButton("Add a Member");  
headingButtonFonts(addAMemberButton2);  
addAMemberButton2.addActionListener(this);
```

```
addAMemberButton3=new JButton("Add a Member");  
headingButtonFonts(addAMemberButton3);  
addAMemberButton3.addActionListener(this);
```

```
membershipDetailsButton1=new JButton("Membership Details");  
headingButtonFonts(membershipDetailsButton1);  
membershipDetailsButton1.addActionListener(this);
```

```
membershipDetailsButton2=new JButton("Membership Details");  
headingButtonFonts(membershipDetailsButton2);  
membershipDetailsButton2.addActionListener(this);
```

```
membershipDetailsButton3=new JButton("Membership Details");  
headingButtonFonts(membershipDetailsButton3);  
membershipDetailsButton3.addActionListener(this);
```

```
allMembersButton1=new JButton("Member Information");  
headingButtonFonts(allMembersButton1);  
allMembersButton1.addActionListener(this);
```

```
allMembersButton2=new JButton("Member Information");  
headingButtonFonts(allMembersButton2);  
allMembersButton2.addActionListener(this);
```

```
allMembersButton3=new JButton("Member Information");  
headingButtonFonts(allMembersButton3);  
allMembersButton3.addActionListener(this);
```

```
regularMemberInfoButton1=new JButton("Regular Member");  
headingButtonFonts(regularMemberInfoButton1);  
regularMemberInfoButton1.addActionListener(this);
```

```
regularMemberInfoButton2=new JButton("Regular Member");  
headingButtonFonts(regularMemberInfoButton2);  
regularMemberInfoButton2.addActionListener(this);
```

```
premiumMemberInfoButton1=new JButton("Premium Member");  
headingButtonFonts(premiumMemberInfoButton1);
```

```
premiumMemberInfoButton1.addActionListener(this);

premiumMemberInfoButton2=new JButton("Premium Member");
headingButtonFonts(premiumMemberInfoButton2);
premiumMemberInfoButton2.addActionListener(this);

addARegularMemberButtonPanel1=new JButton("Regular Member");
headingButtonFonts(addARegularMemberButtonPanel1);
addARegularMemberButtonPanel1.addActionListener(this);

addAPremiumMemberButtonPanel1=new JButton("Premium Member");
headingButtonFonts(addAPremiumMemberButtonPanel1);
addAPremiumMemberButtonPanel1.addActionListener(this);

addARegularMemberButtonPanel2=new JButton("Regular Member");
headingButtonFonts(addARegularMemberButtonPanel2);
addARegularMemberButtonPanel2.addActionListener(this);

addAPremiumMemberButtonPanel2=new JButton("Premium Member");
headingButtonFonts(addAPremiumMemberButtonPanel2);
addAPremiumMemberButtonPanel2.addActionListener(this);

allMembersRegularButton1=new JButton("Regular Member");
headingButtonFonts(allMembersRegularButton1);
allMembersRegularButton1.addActionListener(this);

allMembersRegularButton2=new JButton("Regular Member");
headingButtonFonts(allMembersRegularButton2);
allMembersRegularButton2.addActionListener(this);

allMembersPremiumButton1=new JButton("Premium Member");
```

```
headingButtonFonts(allMembersPremiumButton1);  
allMembersPremiumButton1.addActionListener(this);
```

```
allMembersPremiumButton2=new JButton("Premium Member");  
headingButtonFonts(allMembersPremiumButton2);  
allMembersRegularButton2.addActionListener(this);
```

```
addClearButton1=new JButton("Clear");  
otherButtonFonts(addClearButton1);  
addClearButton1.addActionListener(this);
```

```
addClearButton2=new JButton("Clear");  
otherButtonFonts(addClearButton2);  
addClearButton2.addActionListener(this);
```

```
memberClearButton1=new JButton("Clear");  
otherButtonFonts(memberClearButton1);  
memberClearButton1.addActionListener(this);
```

```
memberClearButton2=new JButton("Clear");  
otherButtonFonts(memberClearButton2);  
memberClearButton2.addActionListener(this);
```

```
saveButton1=new JButton("Save");  
otherButtonFonts(saveButton1);  
saveButton1.addActionListener(this);
```

```
saveButton2=new JButton("Save");  
otherButtonFonts(saveButton2);  
saveButton2.addActionListener(this);
```

```
saveToFileButton1=new JButton("Save to File");  
otherButtonFonts(saveToFileButton1);  
saveToFileButton1.addActionListener(this);
```

```
saveToFileButton2=new JButton("Save to File");  
otherButtonFonts(saveToFileButton2);  
saveToFileButton2.addActionListener(this);
```

```
readFromFileButton1=new JButton("Read from File");  
otherButtonFonts(readFromFileButton1);  
readFromFileButton1.addActionListener(this);
```

```
readFromFileButton2=new JButton("Read from File");  
otherButtonFonts(readFromFileButton2);  
readFromFileButton2.addActionListener(this);
```

```
addARegularMemberButtonBottom=new JButton("Add a Regular Member");  
otherButtonFonts(addARegularMemberButtonBottom);  
addARegularMemberButtonBottom.addActionListener(this);
```

```
addAPremiumMemberButtonBottom=new JButton("Add a Premium Member");  
otherButtonFonts(addAPremiumMemberButtonBottom);  
addAPremiumMemberButtonBottom.addActionListener(this);
```

```
activateButton1=new JButton("Activate Membership");  
otherButtonFonts(activateButton1);  
activateButton1.addActionListener(this);
```

```
activateButton2=new JButton("Activate Membership");  
otherButtonFonts(activateButton2);  
activateButton2.addActionListener(this);
```

```
deactivateButton1=new JButton("Deactivate Membership");  
otherButtonFonts(deactivateButton1);  
deactivateButton1.addActionListener(this);
```

```
deactivateButton2=new JButton("Deactivate Membership");  
otherButtonFonts(deactivateButton2);  
deactivateButton2.addActionListener(this);
```

```
upgradePlanButton=new JButton("Upgrade Plan");  
otherButtonFonts(upgradePlanButton);  
upgradePlanButton.addActionListener(this);
```

```
attendanceButton1=new JButton("Mark Attendance");  
otherButtonFonts(attendanceButton1);  
attendanceButton1.addActionListener(this);
```

```
attendanceButton2=new JButton("Mark Attendance");  
otherButtonFonts(attendanceButton2);  
attendanceButton2.addActionListener(this);
```

```
revertButton1=new JButton("Revert");  
otherButtonFonts(revertButton1);  
revertButton1.addActionListener(this);
```

```
revertButton2=new JButton("Revert");  
otherButtonFonts(revertButton2);  
revertButton2.addActionListener(this);
```

```
displayButton1=new JButton("Display all Members");  
otherButtonFonts(displayButton1);
```



```
displayButton1.addActionListener(this);

displayButton2=new JButton("Display all Members");
otherButtonFonts(displayButton2);
displayButton2.addActionListener(this);

discountButton=new JButton("Discount");
otherButtonFonts(discountButton);
discountButton.addActionListener(this);

paymentButton=new JButton("Payment");
otherButtonFonts(paymentButton);
paymentButton.addActionListener(this);

backbutton1=new JButton("Back");
otherButtonFonts(backbutton1);
backbutton1.addActionListener(this);

backbutton2=new JButton("Back");
otherButtonFonts(backbutton2);
backbutton2.addActionListener(this);

readBackButton1=new JButton("Back");
otherButtonFonts(readBackButton1);
readBackButton1.addActionListener(this);

readBackButton2=new JButton("Back");
otherButtonFonts(readBackButton2);
readBackButton2.addActionListener(this);

//radio button
```

```
maleRadioButton1=new JRadioButton("Male");
maleRadioButton1.setOpaque(false);           //blends the radio buttons
background with the panel's background
maleRadioButton1.setFont(new Font("Century Gothic", Font.PLAIN,17));

maleRadioButton2=new JRadioButton("Male");
maleRadioButton2.setOpaque(false);           //blends the radio buttons
background with the panel's background
maleRadioButton2.setFont(new Font("Century Gothic", Font.PLAIN,17));

femaleRadioButton1=new JRadioButton("Female");
femaleRadioButton1.setOpaque(false);         //blends the radio buttons
background with the panel's background
femaleRadioButton1.setFont(new Font("Century Gothic", Font.PLAIN,17));

femaleRadioButton2=new JRadioButton("Female");
femaleRadioButton2.setOpaque(false);         //blends the radio buttons
background with the panel's background
femaleRadioButton2.setFont(new Font("Century Gothic", Font.PLAIN,17));

ButtonGroup group=new ButtonGroup();
//creating a button group, since without adding radio buttons to it, both the buttons
would be selectable, when there must be restriction to choosing only 1
group.add(maleRadioButton1);
group.add(femaleRadioButton1);
group.add(maleRadioButton2);
group.add(femaleRadioButton2);

//combo boxes
//date of birth
```

```
String []
dobYear={"1995","1996","1997","1998","1999","2000","2001","2002","2003","2004","
2005","2006","2007","2008","2009"};
DOBYear1=new JComboBox(dobYear);
comboBoxColour(DOBYear1);

DOBYear2=new JComboBox(dobYear);
comboBoxColour(DOBYear2);

String [] dobMonth={"1","2","3","4","5","6","7","8","9","10","11","12"};
DOBMonth1=new JComboBox(dobMonth);
comboBoxColour(DOBMonth1);

DOBMonth2=new JComboBox(dobMonth);
comboBoxColour(DOBMonth2);

String []
dobDay={"01","02","03","04","05","06","07","08","09","10","11","12","13","14","15","16
","17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"};
DOBDay1=new JComboBox(dobDay);
comboBoxColour(DOBDay1);

DOBDay2=new JComboBox(dobDay);
comboBoxColour(DOBDay2);

//start date
String []
startYear={"2015","2016","2017","2018","2019","2020","2021","2022","2023","2024","
2025"};
startDateYear1=new JComboBox(startYear);
comboBoxColour(startDateYear1);
```

```
startDateYear2=new JComboBox(startYear);
comboBoxColour(startDateYear2);
```

```
String [] startMonth={"1","2","3","4","5","6","7","8","9","10","11","12"};
startDateMonth1=new JComboBox(startMonth);
comboBoxColour(startDateMonth1);
```

```
startDateMonth2=new JComboBox(startMonth);
comboBoxColour(startDateMonth2);
```

```
String []
startDateDay={"01","02","03","04","05","06","07","08","09","10","11","12","13","14","15","16","17","18","19","20","21","22","23","24","25","26","27","28","29","30","31"};
startDateDay1=new JComboBox(startDateDay);
comboBoxColour(startDateDay1);
```

```
startDateDay2=new JComboBox(startDateDay);
comboBoxColour(startDateDay2);
```

```
//plan
String [] gymPlan={"Basic","Standard","Deluxe"};
planComboBox=new JComboBox(gymPlan);
comboBoxColour(planComboBox);
```

```
//textfield
nameText1=new JTextField();
nameText2=new JTextField();
locationText1=new JTextField();
locationText2=new JTextField();
idText1=new JTextField();
```

```
idText2=new JTextField();
phoneNoText1=new JTextField();
phoneNoText2=new JTextField();
emailText1=new JTextField();
emailText2=new JTextField();
personalTrainerText=new JTextField();
referralSourceText= new JTextField();
removalReasonText=new JTextField();

//add a member left panel
addAMemberLeftPanel.setBounds(1,1,250,800); //x,y,width,height

//quick access
quickAccess1.setBounds(50,120,150,25);
addAMemberLeftPanel.add(quickAccess1);

//buttons
//add a member
addAMemberButton1.setBounds(1,190,245,70);
addAMemberButton1.setBackground(Color.decode("#b08968"));
addAMemberButton1.setBorder(new MatteBorder(2,0,2,0,Color.BLACK));
//top, left, bottom, right, color
addAMemberLeftPanel.add(addAMemberButton1);

//membership details
membershipDetailsButton1.setBounds(1,260,245,70);
membershipDetailsButton1.setBackground(Color.decode("#F2E3BC"));
membershipDetailsButton1.setBorder(new MatteBorder(0,0,2,0,Color.BLACK));
//top, left, bottom, right, color
addAMemberLeftPanel.add(membershipDetailsButton1);
```

```
//all members
allMembersButton1.setBounds(1,330,245,70);
allMembersButton1.setBackground(Color.decode("#F2E3BC"));
allMembersButton1.setBorder(new MatteBorder(0,0,2,0,Color.BLACK));
//top, left, bottom, right, color
addAMemberLeftPanel.add(allMembersButton1);

//add a member panel
//regular member
addARegularMemberPanel.setBounds(250,1,950,800);
//heading
gymName1.setBounds(380,20,180,30);
addARegularMemberPanel.add(gymName1);
//member buttons
addARegularMemberButtonPanel1.setBounds(1,80,470,45);
addARegularMemberButtonPanel1.setBackground(Color.decode("#b08968"));
addARegularMemberButtonPanel1.setBorder(new
MatteBorder(2,0,2,1,Color.BLACK)); //top, left, bottom, right, color
addARegularMemberPanel.add(addARegularMemberButtonPanel1);

addAPremiumMemberButtonPanel1.setBounds(470,80,470,45);
addAPremiumMemberButtonPanel1.setBackground(Color.WHITE);
addAPremiumMemberButtonPanel1.setBorder(new
MatteBorder(2,1,2,0,Color.BLACK)); //top, left, bottom, right, color
addARegularMemberPanel.add(addAPremiumMemberButtonPanel1);

//name
nameLabel1.setBounds(15,250,70,20);
addARegularMemberPanel.add(nameLabel1);

nameText1.setBounds(110,250,170,25);
```

```
addARegularMemberPanel.add(nameText1);

//location
locationLabel1.setBounds(315,250,90,20);
addARegularMemberPanel.add(locationLabel1);

locationText1.setBounds(440,250,170,25);
addARegularMemberPanel.add(locationText1);

//id
idLabel1.setBounds(650,250,90,20);
addARegularMemberPanel.add(idLabel1);

idText1.setBounds(740,250,170,25);
addARegularMemberPanel.add(idText1);

//phone number
phoneNoLabel1.setBounds(15,350,95,20);
addARegularMemberPanel.add(phoneNoLabel1);

phoneNoText1.setBounds(110,350,170,25);
addARegularMemberPanel.add(phoneNoText1);

//email address
emailLabel1.setBounds(315,350,90,20);
addARegularMemberPanel.add(emailLabel1);

emailText1.setBounds(440,350,170,25);
addARegularMemberPanel.add(emailText1);

//gender
```

```
genderLabel1.setBounds(650,350,90,20);
addARegularMemberPanel.add(genderLabel1);

//radio buttons
maleRadioButton1.setBounds(740,350,70,20);
maleRadioButton1.setFocusPainted(false);
addARegularMemberPanel.add(maleRadioButton1);

femaleRadioButton1.setBounds(820,350,90,20);
femaleRadioButton1.setFocusPainted(false);
addARegularMemberPanel.add(femaleRadioButton1);

//date of birth
dateOfBirthLabel1.setBounds(15,450,70,20);
addARegularMemberPanel.add(dateOfBirthLabel1);

//combo boxes
DOBYear1.setBounds(115,450,60,25);
addARegularMemberPanel.add(DOBYear1);

DOBMonth1.setBounds(180,450,45,25);
addARegularMemberPanel.add(DOBMonth1);

DOBDay1.setBounds(230,450,50,25);
addARegularMemberPanel.add(DOBDay1);

//start date
startDateLabel1.setBounds(315,450,90,20);
addARegularMemberPanel.add(startDateLabel1);

//combo boxes
```



```
startDateYear1.setBounds(440,450,60,25);  
addARegularMemberPanel.add(startDateYear1);
```

```
startDateMonth1.setBounds(505,450,45,25);  
addARegularMemberPanel.add(startDateMonth1);
```

```
startDateDay1.setBounds(555,450,50,25);  
addARegularMemberPanel.add(startDateDay1);
```

```
referralSourceLabel1.setBounds(650,450,140,20);  
addARegularMemberPanel.add(referralSourceLabel1);
```

```
referralSourceText.setBounds(795,450,115,25);  
addARegularMemberPanel.add(referralSourceText);
```

```
//add button
```

```
addARegularMemberButtonBottom.setBounds(530,550,240,35);  
addARegularMemberButtonBottom.setBackground(Color.decode("#b08968"));  
addARegularMemberButtonBottom.setBorder(new  
MatteBorder(1,1,1,1,Color.BLACK));  
addARegularMemberPanel.add(addARegularMemberButtonBottom);
```

```
//clear button
```

```
addClearButton1.setBounds(780,550,120,35);  
addClearButton1.setBackground(Color.decode("#b08968"));  
addClearButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));  
addARegularMemberPanel.add(addClearButton1);
```

```
//add a premium member panel
```

```
addAPremiumMemberPanel.setBounds(250,1,950,800);
```

```
//heading
```

```
gymName2.setBounds(380,20,180,30);
addAPremiumMemberPanel.add(gymName2);
//member buttons
addARegularMemberButtonPanel2.setBounds(1,80,470,45);
addARegularMemberButtonPanel2.setBackground(Color.WHITE);
addARegularMemberButtonPanel2.setBorder(new
MatteBorder(2,0,2,1,Color.BLACK)); //top, left, bottom, right, color
addAPremiumMemberPanel.add(addARegularMemberButtonPanel2);

addAPremiumMemberButtonPanel2.setBounds(470,80,470,45);
addAPremiumMemberButtonPanel2.setBackground(Color.decode("#b08968"));
addAPremiumMemberButtonPanel2.setBorder(new
MatteBorder(2,1,2,0,Color.BLACK)); //top, left, bottom, right, color
addAPremiumMemberPanel.add(addAPremiumMemberButtonPanel2);

//name
nameLabel2.setBounds(15,250,70,20);
addAPremiumMemberPanel.add(nameLabel2);

nameText2.setBounds(110,250,170,25);
addAPremiumMemberPanel.add(nameText2);

//location
locationLabel2.setBounds(315,250,90,20);
addAPremiumMemberPanel.add(locationLabel2);

locationText2.setBounds(440,250,170,25);
addAPremiumMemberPanel.add(locationText2);

//id
idLabel2.setBounds(650,250,90,20);
```

```
addAPremiumMemberPanel.add(idLabel2);

idText2.setBounds(740,250,170,25);
addAPremiumMemberPanel.add(idText2);

//phone number
phoneNoLabel2.setBounds(15,350,95,20);
addAPremiumMemberPanel.add(phoneNoLabel2);

phoneNoText2.setBounds(110,350,170,25);
addAPremiumMemberPanel.add(phoneNoText2);

//email address
emailLabel2.setBounds(315,350,90,20);
addAPremiumMemberPanel.add(emailLabel2);

emailText2.setBounds(440,350,170,25);
addAPremiumMemberPanel.add(emailText2);

//gender
genderLabel2.setBounds(650,350,90,20);
addAPremiumMemberPanel.add(genderLabel2);

//radio buttons
maleRadioButton2.setBounds(740,350,70,20);
maleRadioButton2.setFocusPainted(false);
addAPremiumMemberPanel.add(maleRadioButton2);

femaleRadioButton2.setBounds(820,350,90,20);
femaleRadioButton2.setFocusPainted(false);
addAPremiumMemberPanel.add(femaleRadioButton2);
```

```
//date of birth
dateOfBirthLabel2.setBounds(15,450,70,20);
addAPremiumMemberPanel.add(dateOfBirthLabel2);

//combo boxes
DOBYear2.setBounds(115,450,60,25);
addAPremiumMemberPanel.add(DOBYear2);

DOBMonth2.setBounds(180,450,45,25);
addAPremiumMemberPanel.add(DOBMonth2);

DOBDay2.setBounds(230,450,50,25);
addAPremiumMemberPanel.add(DOBDay2);

//start date
startDateLabel2.setBounds(315,450,90,20);
addAPremiumMemberPanel.add(startDateLabel2);

//combo boxes
startDateYear2.setBounds(440,450,60,25);
addAPremiumMemberPanel.add(startDateYear2);

startDateMonth2.setBounds(505,450,45,25);
addAPremiumMemberPanel.add(startDateMonth2);

startDateDay2.setBounds(555,450,50,25);
addAPremiumMemberPanel.add(startDateDay2);

//trainer
personalTrainerLabel.setBounds(650,450,90,20);
```

```
addAPremiumMemberPanel.add(personalTrainerLabel);

personalTrainerText.setBounds(740,450,170,25);
addAPremiumMemberPanel.add(personalTrainerText);

//add button
addAPremiumMemberButtonBottom.setBounds(530,550,240,35);
addAPremiumMemberButtonBottom.setBackground(Color.decode("#b08968"));
addAPremiumMemberButtonBottom.setBorder(new
MatteBorder(1,1,1,1,Color.BLACK));
addAPremiumMemberPanel.add(addAPremiumMemberButtonBottom);

//clear button
addClearButton2.setBounds(780,550,120,35);
addClearButton2.setBackground(Color.decode("#b08968"));
addClearButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));
addAPremiumMemberPanel.add(addClearButton2);

//MEMBER MANAGEMENT
//add a member left panel
memberManageLeftPanel.setBounds(1,1,250,800);

//quick access
quickAccess2.setBounds(50,120,150,25);
memberManageLeftPanel.add(quickAccess2);

//buttons
//membership details
addAMemberButton2.setBounds(1,190,245,70);
addAMemberButton2.setBackground(Color.decode("#F2E3BC"));
```

```
        addAMemberButton2.setBorder(new MatteBorder(2,0,2,0,Color.BLACK));
//top, left, bottom, right, color
        memberManageLeftPanel.add(addAMemberButton2);

//membership details
        membershipDetailsButton2.setBounds(1,260,245,70);
        membershipDetailsButton2.setBackground(Color.decode("#b08968"));
        membershipDetailsButton2.setBorder(new MatteBorder(0,0,2,0,Color.BLACK));
//top, left, bottom, right, color
        memberManageLeftPanel.add(membershipDetailsButton2);

//all members
        allMembersButton2.setBounds(1,330,245,70);
        allMembersButton2.setBackground(Color.decode("#F2E3BC"));
        allMembersButton2.setBorder(new MatteBorder(0,0,2,0,Color.BLACK)); //top,
left, bottom, right, color
        memberManageLeftPanel.add(allMembersButton2);

//REGULAR MEMBER
//panel
        regularMemberPanel.setBounds(250,1,950,800);
        regularMemberPanel.add(regularInfoPanel);
//heading
        gymName3.setBounds(380,20,180,30);
        regularMemberPanel.add(gymName3);
//member buttons
        regularMemberInfoButton1.setBounds(1,80,470,45);
        regularMemberInfoButton1.setBackground(Color.decode("#b08968"));
        regularMemberInfoButton1.setBorder(new MatteBorder(2,0,2,1,Color.BLACK));
//top, left, bottom, right, color
        regularMemberPanel.add(regularMemberInfoButton1);
```

```
premiumMemberInfoButton1.setBounds(470,80,470,45);
premiumMemberInfoButton1.setBackground(Color.WHITE);
premiumMemberInfoButton1.setBorder(new
MatteBorder(2,1,2,0,Color.BLACK)); //top, left, bottom, right, color
regularMemberPanel.add(premiumMemberInfoButton1);

//name
nameLabel3.setBounds(50,170,95,20);
regularMemberPanel.add(nameLabel3);

nameOutputLabel1.setBounds(150,170,310,25);
regularMemberPanel.add(nameOutputLabel1);

//id
idLabel3.setBounds(470,170,95,20);
regularMemberPanel.add(idLabel3);

idOutputLabel1.setBounds(580 ,170,95,25);
regularMemberPanel.add(idOutputLabel1);

//location
locationLabel3.setBounds(50,220,95,20);
regularMemberPanel.add(locationLabel3);

locationOutputLabel1.setBounds(150,220,310,25);
regularMemberPanel.add(locationOutputLabel1);

//phone no
phoneNoLabel3.setBounds(470,220,95,20);
regularMemberPanel.add(phoneNoLabel3);
```

```
phoneOutputLabel1.setBounds(580,220,150,25);  
regularMemberPanel.add(phoneOutputLabel1);
```

```
//email  
emailLabel3.setBounds(50,270,95,20);  
regularMemberPanel.add(emailLabel3);
```

```
emailOutputLabel1.setBounds(150,270,460,25);  
regularMemberPanel.add(emailOutputLabel1);
```

```
//gender  
genderLabel3.setBounds(470,270,95,20);  
regularMemberPanel.add(genderLabel3);
```

```
genderOutputLabel1.setBounds(580,270,105,25);  
regularMemberPanel.add(genderOutputLabel1);
```

```
//date of birth  
dateOfBirthLabel3.setBounds(50,320,95,20);  
regularMemberPanel.add(dateOfBirthLabel3);
```

```
dobOutputLabel1.setBounds(150,320,300,25);  
regularMemberPanel.add(dobOutputLabel1);
```

```
//start date  
startDateLabel3.setBounds(470,320,95,20);  
regularMemberPanel.add(startDateLabel3);
```

```
startDateOutputLabel1.setBounds(580,320,300,25);  
regularMemberPanel.add(startDateOutputLabel1);
```



```
//buttons
activateButton1.setBounds(50,400,200,40);
activateButton1.setBackground(Color.decode("#b08968"));
activateButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));    //top,
left, bottom, right, color
regularMemberPanel.add(activateButton1);

deactivateButton1.setBounds(255,400,220,40);
deactivateButton1.setBackground(Color.decode("#b08968"));
deactivateButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));    //top,
left, bottom, right, color
regularMemberPanel.add(deactivateButton1);

upgradePlanButton.setBounds(50,450,200,40);
upgradePlanButton.setBackground(Color.decode("#b08968"));
upgradePlanButton.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));
//top, left, bottom, right, color
regularMemberPanel.add(upgradePlanButton);

attendanceButton1.setBounds(255,450,220,40);
attendanceButton1.setBackground(Color.decode("#b08968"));
attendanceButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));
//top, left, bottom, right, color
regularMemberPanel.add(attendanceButton1);

revertButton1.setBounds(50,500,200,40);
revertButton1.setBackground(Color.decode("#b08968"));
revertButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));    //top,
left, bottom, right, color
regularMemberPanel.add(revertButton1);
```

```
regularInfoPanel.setBounds(495,400,420,140);
regularInfoPanel.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));    //top,
left, bottom, right, color
regularInfoPanel.setLayout(null);

planLabel1.setBounds(10,20,160,20);
regularInfoPanel.add(planLabel1);

planOutputLabel.setBounds(180,20,160,20);
regularInfoPanel.add(planOutputLabel);

priceLabel2.setBounds(10,60,160,20);
regularInfoPanel.add(priceLabel2);

priceOutputLabel.setBounds(180,60,180,25);
regularInfoPanel.add(priceOutputLabel);

removalReasonLabel2.setBounds(10,100,160,20);
regularInfoPanel.add(removalReasonLabel2);

removalReasonOutputLabel.setBounds(180,100,180,25);
regularInfoPanel.add(removalReasonOutputLabel);

attendanceLabel1.setBounds(50,580,140,25);
regularMemberPanel.add(attendanceLabel1);

attendanceOutputLabel1.setBounds(200,580,90,25);
regularMemberPanel.add(attendanceOutputLabel1);

loyaltyPointsLabel1.setBounds(370,580,140,25);
```

```
regularMemberPanel.add(loyaltyPointsLabel1);
```

```
loyaltyPointsOutputLabel1.setBounds(520,580,90,25);  
regularMemberPanel.add(loyaltyPointsOutputLabel1);
```

```
planLabel2.setBounds(600,580,60,25);  
regularMemberPanel.add(planLabel2);
```

```
planComboBox.setBounds(670,580,90,25);  
regularMemberPanel.add(planComboBox);
```

```
removalReasonLabel.setBounds(50,635,160,25);  
regularMemberPanel.add(removalReasonLabel);  
removalReasonText.setBounds(220,635,130,35);  
regularMemberPanel.add(removalReasonText);
```

```
referralSourceLabel2.setBounds(370,635,140,25);  
regularMemberPanel.add(referralSourceLabel2);  
referralSourceOutputLabel.setBounds(520,635,130,25);  
regularMemberPanel.add(referralSourceOutputLabel);
```

```
memberClearButton1.setBounds(150,680,100,30);  
memberClearButton1.setBackground(Color.decode("#b08968"));  
memberClearButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));  
//top, left, bottom, right, color  
regularMemberPanel.add(memberClearButton1);
```

```
saveButton1.setBounds(270,680,100,30);  
saveButton1.setBackground(Color.decode("#b08968"));  
saveButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top,  
left, bottom, right, color
```

```
regularMemberPanel.add(saveButton1);

saveToFileButton1.setBounds(390, 680,150,30);
saveToFileButton1.setBackground(Color.decode("#b08968"));
saveToFileButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));
//top, left, bottom, right, color
regularMemberPanel.add(saveToFileButton1);

readFromFileButton1.setBounds(560, 680,150,30);
readFromFileButton1.setBackground(Color.decode("#b08968"));
readFromFileButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));
//top, left, bottom, right, color
regularMemberPanel.add(readFromFileButton1);

displayButton1.setBounds(730,680,190,30);
displayButton1.setBackground(Color.decode("#b08968"));
displayButton1.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top,
left, bottom, right, color
regularMemberPanel.add(displayButton1);

//PREMIUM MEMBER
//panel
premiumMemberPanel.setBounds(250,1,950,800);
premiumMemberPanel.add(premiumInfoPanel);
//heading
gymName4.setBounds(380,20,180,30);
premiumMemberPanel.add(gymName4);

//member buttons
regularMemberInfoButton2.setBounds(1,80,470,45);
regularMemberInfoButton2.setBackground(Color.WHITE);
```

```
        regularMemberInfoButton2.setBorder(new MatteBorder(2,0,2,1,Color.BLACK));
//top, left, bottom, right, color
        premiumMemberPanel.add(regularMemberInfoButton2);

        premiumMemberInfoButton2.setBounds(470,80,470,45);
        premiumMemberInfoButton2.setBackground(Color.decode("#b08968"));
        premiumMemberInfoButton2.setBorder(new
MatteBorder(2,1,2,0,Color.BLACK));    //top, left, bottom, right, color
        premiumMemberPanel.add(premiumMemberInfoButton2);

//name
        nameLabel4.setBounds(50,170,95,20);
        premiumMemberPanel.add(nameLabel4);

        nameOutputLabel2.setBounds(150,170,310,25);
        premiumMemberPanel.add(nameOutputLabel2);

//id
        idLabel4.setBounds(470,170,95,20);
        premiumMemberPanel.add(idLabel4);

        idOutputLabel2.setBounds(580,170,95,25);
        premiumMemberPanel.add(idOutputLabel2);

//location
        locationLabel4.setBounds(50,220,95,20);
        premiumMemberPanel.add(locationLabel4);

        locationOutputLabel2.setBounds(150,220,310,25);
        premiumMemberPanel.add(locationOutputLabel2);
```

```
//phone no
phoneNoLabel4.setBounds(470,220,95,20);
premiumMemberPanel.add(phoneNoLabel4);

phoneOutputLabel2.setBounds(580,220,150,25);
premiumMemberPanel.add(phoneOutputLabel2);

//email
emailLabel4.setBounds(50,270,95,20);
premiumMemberPanel.add(emailLabel4);

emailOutputLabel2.setBounds(150,270,460,25);
premiumMemberPanel.add(emailOutputLabel2);

//gender
genderLabel4.setBounds(470,270,95,20);
premiumMemberPanel.add(genderLabel4);

genderOutputLabel2.setBounds(580,270,95,25);
premiumMemberPanel.add(genderOutputLabel2);

//date of birth
dateOfBirthLabel4.setBounds(50,320,95,20);
premiumMemberPanel.add(dateOfBirthLabel4);

dobOutputLabel2.setBounds(150,320,300,25);
premiumMemberPanel.add(dobOutputLabel2);

//start date
startDateLabel4.setBounds(470,320,95,20);
premiumMemberPanel.add(startDateLabel4);
```

```
startDateOutputLabel2.setBounds(580,320,300,25);
premiumMemberPanel.add(startDateOutputLabel2);

//buttons
activateButton2.setBounds(50,400,200,40);
activateButton2.setBackground(Color.decode("#b08968"));
activateButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top, left,
bottom, right, color
premiumMemberPanel.add(activateButton2);

deactivateButton2.setBounds(255,400,220,40);
deactivateButton2.setBackground(Color.decode("#b08968"));
deactivateButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top,
left, bottom, right, color
premiumMemberPanel.add(deactivateButton2);

discountButton.setBounds(50,450,200,40);
discountButton.setBackground(Color.decode("#b08968"));
discountButton.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top, left,
bottom, right, color
premiumMemberPanel.add(discountButton);

attendanceButton2.setBounds(255,450,220,40);
attendanceButton2.setBackground(Color.decode("#b08968"));
attendanceButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top,
left, bottom, right, color
premiumMemberPanel.add(attendanceButton2);

paymentButton.setBounds(50,500,200,40);
paymentButton.setBackground(Color.decode("#b08968"));
```

```
        paymentButton.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));    //top, left,
bottom, right, color
        premiumMemberPanel.add(paymentButton);

        revertButton2.setBounds(255,500,220,40);
        revertButton2.setBackground(Color.decode("#b08968"));
        revertButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));    //top, left,
bottom, right, color
        premiumMemberPanel.add(revertButton2);

        premiumInfoPanel.setBounds(495,400,420,140);
        premiumInfoPanel.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));    //top,
left, bottom, right, color
        premiumInfoPanel.setLayout(null);

        personalTrainerLabel2.setBounds(10,20,130,25);
        premiumInfoPanel.add(personalTrainerLabel2);

        personalTrainerOutputLabel2.setBounds(150,20,130,25);
        premiumInfoPanel.add(personalTrainerOutputLabel2);

        paidAmountLabel.setBounds(10,60,130,25);
        premiumInfoPanel.add(paidAmountLabel);

        paidAmountOutputLabel.setBounds(150,60,160,25);
        premiumInfoPanel.add(paidAmountOutputLabel);

        fullPaymentLabel.setBounds(10,100,130,25);
        premiumInfoPanel.add(fullPaymentLabel);

        fullPaymentOutputLabel.setBounds(150,100,160,25);
```



```
premiumInfoPanel.add(fullPaymentOutputLabel);
```

```
attendanceLabel2.setBounds(50,585,140,25);  
premiumMemberPanel.add(attendanceLabel2);
```

```
attendanceOutputLabel2.setBounds(200,585,90,25);  
premiumMemberPanel.add(attendanceOutputLabel2);
```

```
loyaltyPointsLabel2.setBounds(360,585,140,25);  
premiumMemberPanel.add(loyaltyPointsLabel2);
```

```
loyaltyPointsOutputLabel2.setBounds(510,585,90,25);  
premiumMemberPanel.add(loyaltyPointsOutputLabel2);
```

```
totalChargeLabel.setBounds(670,585,140,25);  
premiumMemberPanel.add(totalChargeLabel);
```

```
totalChargeOutputLabel.setBounds(820,585,90,25);  
premiumMemberPanel.add(totalChargeOutputLabel);
```

```
memberClearButton2.setBounds(150,650,100,30);  
memberClearButton2.setBackground(Color.decode("#b08968"));  
memberClearButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top,  
left, bottom, right, color  
premiumMemberPanel.add(memberClearButton2);
```

```
saveButton2.setBounds(270,650,100,30);  
saveButton2.setBackground(Color.decode("#b08968"));  
saveButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK)); //top, left,  
bottom, right, color  
premiumMemberPanel.add(saveButton2);
```

```

    saveToFileButton2.setBounds(390, 650,150,30);
    saveToFileButton2.setBackground(Color.decode("#b08968"));
    saveToFileButton2.setBorder(new           MatteBorder(1,1,1,1,Color.BLACK));
//top, left, bottom, right, color
    premiumMemberPanel.add(saveToFileButton2);

    readFromFileButton2.setBounds(560, 650,150,30);
    readFromFileButton2.setBackground(Color.decode("#b08968"));
    readFromFileButton2.setBorder(new           MatteBorder(1,1,1,1,Color.BLACK));
//top, left, bottom, right, color
    premiumMemberPanel.add(readFromFileButton2);

    displayButton2.setBounds(730,650,190,30);
    displayButton2.setBackground(Color.decode("#b08968"));
    displayButton2.setBorder(new MatteBorder(1,1,1,1,Color.BLACK));           //top,
left, bottom, right, color
    premiumMemberPanel.add(displayButton2);

//left panel
allMembersLeftPanel.setBounds(1,1,250,800);
allMembersLeftPanel.setBackground(Color.decode("#F2E3BC"));
quickAccess3.setBounds(50,120,150,25);
allMembersLeftPanel.add(quickAccess3);

//add a member
addAMemberLeftPanel.add(quickAccess1);
addAMemberButton3.setBounds(1,190,245,70);
addAMemberButton3.setBackground(Color.decode("#F2E3BC"));
addAMemberButton3.setBorder(new           MatteBorder(2,0,2,0,Color.BLACK));
//top, left, bottom, right, color

```

```
allMembersLeftPanel.add(addAMemberButton3);

//membership details
membershipDetailsButton3.setBounds(1,260,245,70);
membershipDetailsButton3.setBackground(Color.decode("#F2E3BC"));
membershipDetailsButton3.setBorder(new MatteBorder(0,0,2,0,Color.BLACK));
//top, left, bottom, right, color
allMembersLeftPanel.add(membershipDetailsButton3);

//all members
allMembersButton3.setBounds(1,330,245,70);
allMembersButton3.setBackground(Color.decode("#b08968"));
allMembersButton3.setBorder(new MatteBorder(0,0,2,0,Color.BLACK));
//top, left, bottom, right, color
allMembersLeftPanel.add(allMembersButton3);

//REGULAR MEMBER
allMembersRegularPanel.setBounds(250,1,950,800);
//heading
gymName5.setBounds(380,20,180,30);
allMembersRegularPanel.add(gymName5);
//member buttons
allMembersRegularButton1.setBounds(1,80,470,45);
allMembersRegularButton1.setBackground(Color.decode("#b08968"));
allMembersRegularButton1.setBorder(new MatteBorder(2,0,2,1,Color.BLACK));
//top, left, bottom, right, color
allMembersRegularPanel.add(allMembersRegularButton1);

allMembersPremiumButton1.setBounds(470,80,470,45);
allMembersPremiumButton1.setBackground(Color.WHITE);
```

```
allMembersPremiumButton1.setBorder(new
MatteBorder(2,1,2,0,Color.BLACK));    //top, left, bottom, right, color
allMembersRegularPanel.add(allMembersPremiumButton1);

//PREMIUM MEMBER
allMembersPremiumPanel.setBounds(250,1,950,800);
//heading
gymName6.setBounds(380,20,180,30);
allMembersPremiumPanel.add(gymName6);
//member buttons
allMembersRegularButton2.setBounds(1,80,470,45);
allMembersRegularButton2.setBackground(Color.WHITE);
allMembersRegularButton2.setBorder(new MatteBorder(2,0,2,1,Color.BLACK));
//top, left, bottom, right, color
allMembersPremiumPanel.add(allMembersRegularButton2);

allMembersPremiumButton2.setBounds(470,80,470,45);
allMembersPremiumButton2.setBackground(Color.decode("#b08968"));
allMembersPremiumButton2.setBorder(new
MatteBorder(2,1,2,0,Color.BLACK));    //top, left, bottom, right, color
allMembersPremiumPanel.add(allMembersPremiumButton2);

//DISPLAY PANEL- REGULAR
displayPanel1.setBounds(0,0,1200,800);

backbutton1.setBounds(10,10,80,45);
backbutton1.setBackground(Color.decode("#b08968"));
backbutton1.setBorder(new MatteBorder(2,1,2,0,Color.BLACK));    //top, left,
bottom, right, color
displayPanel1.add(backbutton1);
```

```
//adding the headings to the text area
displayAreaRegular.append(String.format("%-6s %-25s %-25s %-15s %-30s %-
10s %-15s %-30s %-15s %-15s %-15s %-15s %-25s %-25s\n",
    "ID", "Name", "Location", "Phone", "Email","Gender","DOB","Membership
Start Date",
    "Attendance", "Loyalty Points", "ActiveStatus","Plan", "Price","Eligible for
Upgrade",
    "Referral Source"));
```

```
//adding text area to the scroll pane
scrollPane1= new JScrollPane(displayAreaRegular);
//providing dimensions for scrollpane which also includes the text area
scrollPane1.setBounds(10,65,1170,700);
scrollPane1.setBorder(new MatteBorder(2,2,2,2,Color.BLACK));    //top, left,
bottom, right, color
//adding scroll pane to the panel
displayPanel1.add(scrollPane1);
```

```
//READ FROM FILE PANEL-REGULAR
readFromFilePanel1.setBounds(0,0,1200,800);

readBackButton1.setBounds(10,10,80,45);
readBackButton1.setBackground(Color.decode("#b08968"));
readBackButton1.setBorder(new MatteBorder(2,1,2,0,Color.BLACK));    //top,
left, bottom, right, color
readFromFilePanel1.add(readBackButton1);
```

```
//adding text area to the scroll pane
scrollRead1= new JScrollPane(readAreaRegular);
//providing dimensions for scrollpane which also includes the text area
scrollRead1.setBounds(10,65,1170,700);
```

```

        scrollRead1.setBorder(new MatteBorder(2,2,2,2,Color.BLACK));    //top, left,
bottom, right, color
        //adding scroll pane to the panel
        readFromFilePanel1.add(scrollRead1);

        //DISPLAY PANEL- PREMIUM
        displayPanel2.setBounds(0,0,1200,800);

        backbutton2.setBounds(10,10,80,45);
        backbutton2.setBackground(Color.decode("#b08968"));
        backbutton2.setBorder(new MatteBorder(2,1,2,0,Color.BLACK));    //top, left,
bottom, right, color
        displayPanel2.add(backbutton2);

        //adding the headings to the text area
        displayAreaPremium.append(String.format("%-6s  %-25s  %-15s  %-20s  %-30s
        %-10s  %-15s  %-30s  %-15s  %-15s  %-15s  %-15s  %-18s  %-25s\n",
                "ID", "Name", "Location", "Phone", "Email","Gender","DOB","Membership
        Start Date",
                "Attendance", "Loyalty  Points", "ActiveStatus","Full  Payment", "Paid
        Amount","Discount Amount",
                "Premium Charge"));
        //adding text area to the scroll pane
        scrollPane2= new JScrollPane(displayAreaPremium);
        //providing dimensions for scrollpane which also includes the text area
        scrollPane2.setBounds(10,65,1170,700);
        scrollPane2.setBorder(new MatteBorder(2,2,2,2,Color.BLACK));    //top, left,
bottom, right, color
        //adding scroll pane to the panel
        displayPanel2.add(scrollPane2);

```

```

//READ FROM FILE PANEL- PREMIUM
readFromFilePanel2.setBounds(0,0,1200,800);

readBackButton2.setBounds(10,10,80,45);
readBackButton2.setBackground(Color.decode("#b08968"));
readBackButton2.setBorder(new MatteBorder(2,1,2,0,Color.BLACK));    //top,
left, bottom, right, color
readFromFilePanel2.add( readBackButton2);

//adding text area to the scroll pane
scrollRead2= new JScrollPane(readAreaPremium);
//providing dimensions for scrollpane which also includes the text area
scrollRead2.setBounds(10,65,1170,700);
scrollRead2.setBorder(new MatteBorder(2,2,2,2,Color.BLACK));    //top, left,
bottom, right, color
//adding scroll pane to the panel
readFromFilePanel2.add(scrollRead2);

frame.add(addARegularMemberPanel);
frame.add(addAMemberLeftPanel);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true);    //setting the frame visibility as true
frame.setResizable(false);

}

public void actionPerformed(ActionEvent e)
{
    if(e.getSource()==addAMemberButton1)
    {
        //adding and removing panels according to need
    }
}

```

```
        frame.add(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(memberManageLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.add(addAMemberLeftPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();    //this is used to re-draw the component/ refresh the
window
    }
    else if(e.getSource()==addAMemberButton2)
    {
        frame.add(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(memberManageLeftPanel);
        frame.add(addAMemberLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.remove(allMembersRegularPanel);
        frame.remove(allMembersPremiumPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();    //this is used to re-draw the component
    }
    else if(e.getSource()==addAMemberButton3)
    {
        frame.add(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
```



```
        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(memberManageLeftPanel);
        frame.add(addAMemberLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.remove(allMembersRegularPanel);
        frame.remove(allMembersPremiumPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();    //this is used to re-draw the component
    }
    else if(e.getSource()==addARegularMemberButtonPanel1)
    {
        frame.add(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();    //this is used to re-draw the component
    }
    else if(e.getSource()==addARegularMemberButtonPanel2)
    {
        frame.add(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that the
layout updates properly
        frame.repaint();    //this is used to re-draw the component
    }
```

```
else if(e.getSource()==addAPremiumMemberButtonPanel1)
{
    frame.remove(addARegularMemberPanel);
    frame.add(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();      //this is used to re-draw the component
}
else if(e.getSource()==addAPremiumMemberButtonPanel2)
{
    frame.remove(addARegularMemberPanel);
    frame.add(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();      //this is used to re-draw the component
}
else if(e.getSource()==membershipDetailsButton1)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.add(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.add(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.remove(allMembersRegularPanel);
    frame.remove(allMembersPremiumPanel);
```

```

        frame.revalidate();           //this is used to re-calculate the layout/ ensures
that the layout updates properly
        frame.repaint();             //this is used to re-draw the component

        try{
            int
regularInputId=Integer.parseInt(JOptionPane.showInputDialog(regularMemberPanel,
"Enter your ID:"));

            for (GymMember member: members)           //for each member in arraylist
members
            {
                if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered and if it
belongs to regular member
                {
                    //setting the values in labels
                    nameOutputLabel1.setText(member.getName());
                    idOutputLabel1.setText(String.valueOf(member.getId()));
//converting to string
                    locationOutputLabel1.setText(member.getLocation());
                    phoneOutputLabel1.setText(member.getPhone());
                    emailOutputLabel1.setText(member.getEmail());
                    genderOutputLabel1.setText(member.getGender());
                    dobOutputLabel1.setText(member.getDOB());
                    startDateOutputLabel1.setText(member.getMembershipStartDate());
                    //casting from gym member to regular member
                    RegularMember regularMember=(RegularMember) member;

                    referralSourceOutputLabel.setText(regularMember.getReferralSource());
                    planOutputLabel.setText(regularMember.getPlan());
                    priceOutputLabel.setText(String.valueOf(regularMember.getPrice()));

```

```

        return;           //to immediately exit the method when information is
displayed else, the second dialog box will also pop up
    }
}
//else is not written since it shows this dialog box everytime an id doesnt
match even though it exists
//only if no member was found, i.e if the member doesn't exist at all
JOptionPane.showMessageDialog(regularMemberPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(regularMemberPanel, "ID must be a
number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
}
else if(e.getSource()==membershipDetailsButton2)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.add(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.add(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.remove(allMembersRegularPanel);
    frame.remove(allMembersPremiumPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();       //this is used to re-draw the component

```

```

        try
        {
            int
regularInputId=Integer.parseInt(JOptionPane.showInputDialog(regularMemberPanel,
"Enter your ID:"));
            for (GymMember member: members)          //for each member in arraylist
members
            {
                if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered
                {
                    //setting the values in labels
                    nameOutputLabel1.setText(member.getName());
                    idOutputLabel1.setText(String.valueOf(member.getId()));
//converting to string
                    locationOutputLabel1.setText(member.getLocation());
                    phoneOutputLabel1.setText(member.getPhone());
                    emailOutputLabel1.setText(member.getEmail());
                    genderOutputLabel1.setText(member.getGender());
                    dobOutputLabel1.setText(member.getDOB());
                    startDateOutputLabel1.setText(member.getMembershipStartDate());
                    //casting from gym member to regular member
                    RegularMember regularMember=(RegularMember) member;

                    referralSourceOutputLabel.setText(regularMember.getReferralSource());
                    planOutputLabel.setText(regularMember.getPlan());
                    priceOutputLabel.setText(String.valueOf(regularMember.getPrice()));
                    return;          //to immediately exit the method when information is
displayed else, the second dialog box will also pop up

```

```

        }
    }

    //else is not written since it shows this dialog box everytime an id doesnt
    match even though it exists

    //only if no member was found, i.e if the member doesn't exist at all
    JOptionPane.showMessageDialog(regularMemberPanel, "Member not
    found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(regularMemberPanel, "ID must be a
    number", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
}
else if(e.getSource()==membershipDetailsButton3)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.add(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.add(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.remove(allMembersRegularPanel);
    frame.remove(allMembersPremiumPanel);
    frame.revalidate(); //this is used to re-calculate the layout/ ensures that the
    layout updates properly
    frame.repaint(); //this is used to re-draw the component

    try
    {

```

```

        int
        regularInputId=Integer.parseInt(JOptionPane.showInputDialog(regularMemberPanel,
        "Enter your ID:"));
        for (GymMember member: members)      //for each member in arraylist
members
        {
            if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered and if it
belongs to regular member
            {
                //setting the values in labels
                nameOutputLabel1.setText(member.getName());
                idOutputLabel1.setText(String.valueOf(member.getId()));
//converting to string
                locationOutputLabel1.setText(member.getLocation());
                phoneOutputLabel1.setText(member.getPhone());
                emailOutputLabel1.setText(member.getEmail());
                genderOutputLabel1.setText(member.getGender());
                dobOutputLabel1.setText(member.getDOB());
                startDateOutputLabel1.setText(member.getMembershipStartDate());
                //casting from gym member to regular member
                RegularMember regularMember=(RegularMember) member;

                referralSourceOutputLabel.setText(regularMember.getReferralSource());
                planOutputLabel.setText(regularMember.getPlan());
                priceOutputLabel.setText(String.valueOf(regularMember.getPrice()));
                return;    //to immediately exit the method when information is
displayed else, the second dialog box will also pop up

            }
        }
    }

```

```

        //else is not written since it shows this dialog box everytime an id doesnt
match even though it exists

        //only if no member was found, i.e if the member doesn't exist at all
        JOptionPane.showMessageDialog(regularMemberPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
    catch(NumberFormatException ex)
    {
        JOptionPane.showMessageDialog(regularMemberPanel, "ID must be a
number", "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
}
else if(e.getSource()==regularMemberInfoButton1)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.add(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.add(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.revalidate();        //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();          //this is used to re-draw the component

    try{
        int
regularInputId=Integer.parseInt(JOptionPane.showInputDialog(regularMemberPanel,
"Enter your ID:"));
        for (GymMember member: members)        //for each member in arraylist
members

```



```

        {
            if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered
            {
                //setting the values in labels
                nameOutputLabel1.setText(member.getName());
                idOutputLabel1.setText(String.valueOf(member.getId()));
//converting to string
                locationOutputLabel1.setText(member.getLocation());
                phoneOutputLabel1.setText(member.getPhone());
                emailOutputLabel1.setText(member.getEmail());
                genderOutputLabel1.setText(member.getGender());
                dobOutputLabel1.setText(member.getDOB());
                startDateOutputLabel1.setText(member.getMembershipStartDate());
                //casting from gym member to regular member
                RegularMember regularMember=(RegularMember) member;

                referralSourceOutputLabel.setText(regularMember.getReferralSource());
                planOutputLabel.setText(regularMember.getPlan());
                priceOutputLabel.setText(String.valueOf(regularMember.getPrice()));
                return;    //to immediately exit the method when information is
displayed else, the second dialog box will also pop up
            }
        }

        //else is not written since it shows this dialog box everytime an id doesnt
match even though it exists

        //only if no member was found, i.e if the member doesn't exist at all
        JOptionPane.showMessageDialog(regularMemberPanel, "Member    not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
    catch(NumberFormatException ex)

```

```

        {
            JOptionPane.showMessageDialog(regularMemberPanel,"ID must be a
number","Invalid ID", JOptionPane.ERROR_MESSAGE);
        }
    }

    else if(e.getSource()==regularMemberInfoButton2)
    {
        frame.remove(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.add(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(addAMemberLeftPanel);
        frame.add(memberManageLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();    //this is used to re-draw the component

        try{
            int
regularInputId=Integer.parseInt(JOptionPane.showInputDialog(regularMemberPanel,
"Enter your ID:"));

            for (GymMember member: members)    //for each member in arraylist
members
            {
                if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered
                {
                    //instanceof keyword
checks whether an object belongs to a specific class or an interface. The return value
is either true or false.

```

```

        //setting the values in labels
        nameOutputLabel1.setText(member.getName()); //set the value of
name from setter method
        idOutputLabel1.setText(String.valueOf(member.getId()));
//converting to string
        locationOutputLabel1.setText(member.getLocation());
        phoneOutputLabel1.setText(member.getPhone());
        emailOutputLabel1.setText(member.getEmail());
        genderOutputLabel1.setText(member.getGender());
        dobOutputLabel1.setText(member.getDOB());
        startDateOutputLabel1.setText(member.getMembershipStartDate());

        //casting from gym member to regular member
        RegularMember regularMember=(RegularMember) member;

        referralSourceOutputLabel.setText(regularMember.getReferralSource());
        planOutputLabel.setText(regularMember.getPlan());
        priceOutputLabel.setText(String.valueOf(regularMember.getPrice()));

        return; //to immediately exit the method when information is displayed
else, the second dialog box will also pop up
    }
}
//else is not written since it shows this dialog box everytime an id doesnt
match even though it exists
//only if no member was found, i.e if the member doesn't exist at all
JOptionPane.showMessageDialog(regularMemberPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);

}
catch(NumberFormatException ex)

```

```

        {
            JOptionPane.showMessageDialog(regularMemberPanel, "ID must be a
number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
        }
    }

    else if(e.getSource()==premiumMemberInfoButton1)
    {
        frame.remove(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.add(premiumMemberPanel);
        frame.remove(addAMemberLeftPanel);
        frame.add(memberManageLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.revalidate();        //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();        //this is used to re-draw the component
        try{
            int
premiumInputId=Integer.parseInt(JOptionPane.showInputDialog(premiumMemberPa
nel, "Enter your ID:"));
            for (GymMember member: members)        //for each member in arraylist
members
            {
                if(member.getId()==premiumInputId    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id entered
                {
                    //instanceof keyword checks whether an
object belongs to a specific class or an interface. The return value is either true or
false.

                    //setting the values in labels

```

```

        nameOutputLabel2.setText(member.getName()); //set the value of
name from setter method
        idOutputLabel2.setText(String.valueOf(member.getId()));
//converting to string
        locationOutputLabel2.setText(member.getLocation());
        phoneOutputLabel2.setText(member.getPhone());
        emailOutputLabel2.setText(member.getEmail());
        genderOutputLabel2.setText(member.getGender());
        dobOutputLabel2.setText(member.getDOB());
        startDateOutputLabel2.setText(member.getMembershipStartDate());

        //casting to set trainer
        PremiumMember premiumMember = (PremiumMember) member;

personalTrainerOutputLabel2.setText(premiumMember.getPersonalTrainer());

        return; //to immediately exit the method when information is displayed
else, the second dialog box will also pop up
    }
}
//else is not written since it shows this dialog box everytime an id doesnt
match even though it exists
//only if no member was found, i.e if the member doesn't exist at all
JOptionPane.showMessageDialog(premiumMemberPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(premiumMemberPanel, "ID must be a
number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}

```

```

    }

    else if(e.getSource()==premiumMemberInfoButton2)
    {
        frame.remove(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.add(premiumMemberPanel);
        frame.remove(addAMemberLeftPanel);
        frame.add(memberManageLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();    //this is used to re-draw the component

        try{
            int
premiumInputId=Integer.parseInt(JOptionPane.showInputDialog(premiumMemberPa
nel, "Enter your ID:"));

            for (GymMember member: members)    //for each member in arraylist
members
            {
                if(member.getId()==premiumInputId    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id entered
                {
                    //setting the values in labels
                    nameOutputLabel2.setText(member.getName());
                    idOutputLabel2.setText(String.valueOf(member.getId()));
//converting to string
                    locationOutputLabel2.setText(member.getLocation());
                    phoneOutputLabel2.setText(member.getPhone());

```

```

        emailOutputLabel2.setText(member.getEmail());
        genderOutputLabel2.setText(member.getGender());
        dobOutputLabel2.setText(member.getDOB());
        startDateOutputLabel2.setText(member.getMembershipStartDate());

        //casting to set trainer
        PremiumMember premiumMember = (PremiumMember) member;

        personalTrainerOutputLabel2.setText(premiumMember.getPersonalTrainer());
        return; //to immediately exit the method when information is displayed
else, the second dialog box will also pop up
    }
}

//else is not written since it shows this dialog box everytime an id doesnt
match even though it exists
    //only if no member was found, i.e if the member doesn't exist at all
    JOptionPane.showMessageDialog(premiumMemberPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
catch(NumberFormatException ex)
{
    JOptionPane.showMessageDialog(premiumMemberPanel, "ID must be a
number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
}

else if(e.getSource()==addARegularMemberButtonBottom)
{
    //getting value from fields
    String regularName= nameText1.getText();
    String regularLocation=locationText1.getText();

```

```

String regularId=idText1.getText();
String regularPhone= phoneNoText1.getText();
String regularEmail= emailText1.getText();
String regularReferralSource=referralSourceText.getText();
String regularGender;
if(maleRadioButton1.isSelected())
{
    regularGender="Male";
}
else{
    regularGender="Female";
}
String regularDOBYear= (String) DOBYear1.getSelectedItem();
String regularDOBMonth=(String) DOBMonth1.getSelectedItem();
String regularDOBDay=(String) DOBDay1.getSelectedItem();
String regularStartYear=(String) startDateYear1.getSelectedItem();
String regularStartMonth=(String) startDateMonth1.getSelectedItem();
String regularStartDay=(String) startDateDay1.getSelectedItem();

//confirmation message to ensure all fields are filled and the member is added
//.equals is used to compare strings
if(regularName.equals("") || regularLocation.equals("") || regularId.equals("") ||
regularPhone.equals("") || regularEmail.equals("") ||regularReferralSource.equals("") )
//isEmpty() checks if the field is empty
{
    JOptionPane.showMessageDialog(addARegularMemberPanel, "Please fill
out all the fields.", "Empty field(s)",JOptionPane.ERROR_MESSAGE);
    return;          //immediately exits this method when empty fields are found.
otherwise, the other dialog boxes will also pop up
}
try

```



```

{
    int regularIdInteger = Integer.parseInt(regularId);
    //checking if the id entered by the user is within the range
    if (regularIdInteger<=0 || regularIdInteger>90000)
    {
        throw new InvalidIdException("ID must be within 1 to 90000! Please enter
a valid ID.");
    }
    if(regularPhone.length()<10 || regularPhone.length()>10)
    {
        throw new InvalidPhoneException("Phone number must be of 10 digits!");
    }
    //checking for duplicate ID
    for (GymMember member : members)
    {
        if (member.getId() == regularIdInteger && member instanceof
RegularMember)    //if member is an object of regular member
        {
            JOptionPane.showMessageDialog(addARegularMemberPanel,    "A
member with ID " + regularIdInteger + " already exists!\nPlease enter another
ID.", "Existing ID", JOptionPane.ERROR_MESSAGE);
            //place of display, message, title and type
            return;    //immediately exits this method when duplicate id is found.
            if not written so, it will show the error message and also add the member with same id
        }
    }

    // Create and add new member
    //calling the constructor of regular member which is a subclass of the class
gym member

```

```

        GymMember      rMember      =      new
RegularMember(regularIdInteger,regularName, regularLocation, regularPhone,
               regularEmail,      regularGender,      regularDOBYear+      "-"      +
regularDOBMonth+ "-" + regularDOBDay,
               regularStartYear+ "-" +regularStartMonth+ "-" + regularStartDay,
regularReferralSource);
        members.add(rMember);      //adds the new object created to the arraylist

//success message
        JOptionPane.showMessageDialog(addARegularMemberPanel, "You have
been added as a regular member!");

//setting all fields empty
        nameText1.setText("");
        locationText1.setText("");
        idText1.setText("");
        phoneNoText1.setText("");
        emailText1.setText("");
        DOBYear1.setSelectedIndex(0);      //selects the first option
        DOBMonth1.setSelectedIndex(0);
        DOBDay1.setSelectedIndex(0);
        startDateYear1.setSelectedIndex(0);
        startDateMonth1.setSelectedIndex(0);
        startDateDay1.setSelectedIndex(0);
        referralSourceText.setText("");
    }
    catch (NumberFormatException ex) {      //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed
        JOptionPane.showMessageDialog(addARegularMemberPanel, "ID must
be a number!","Invalid ID", JOptionPane.ERROR_MESSAGE);

```

```

        //checking if ID entered is a number
    }
    catch(InvalidIdException i)
    {
        JOptionPane.showMessageDialog(addARegularMemberPanel,
i.getMessage(),"Invalid ID",JOptionPane.ERROR_MESSAGE);
        //if the ID is a number, checking if it is within the range as required
    }
    catch(InvalidPhoneException p)
    {

JOptionPane.showMessageDialog(addARegularMemberPanel,p.getMessage(),"Inva
lid phone number", JOptionPane.ERROR_MESSAGE);
    }
}
else if(e.getSource()==addAPremiumMemberButtonBottom)
{
    //getting value from fields
    String premiumName= nameText2.getText();
    String premiumLocation=locationText2.getText();
    String premiumId=idText2.getText();
    String premiumPhone= phoneNoText2.getText();
    String premiumEmail= emailText2.getText();
    String premiumGender;
    if(maleRadioButton2.isSelected())
    {
        premiumGender="Male";
    }
    else
    {
        premiumGender="Female";
    }
}

```

```

    }
    String premiumDOBYear= (String) DOBYear2.getSelectedItemId();
    String premiumDOBMonth=(String) DOBMonth2.getSelectedItemId();
    String premiumDOBDay=(String) DOBDay2.getSelectedItemId();
    String premiumStartYear=(String) startDateYear2.getSelectedItemId();
    String premiumStartMonth=(String) startDateMonth2.getSelectedItemId();
    String premiumStartDay=(String) startDateDay2.getSelectedItemId();
    String trainer= personalTrainerText.getText();

    //confirmation message to ensure all fields are filled and the member is added
    //equals is used to compare strings
    if(premiumName.equals("") || premiumLocation.equals("") ||
premiumId.equals("") || premiumPhone.equals("") || premiumEmail.equals("") ||
trainer.equals(""))
    {
        JOptionPane.showMessageDialog(addAPremiumMemberPanel, "Please
fill out all the fields.", "Empty field(s)", JOptionPane.ERROR_MESSAGE);
        return; //immediately exits this method when empty fields are found.
otherwise, the other dialog boxes will also pop up
    }
    try
    {
        int premiumIdInteger = Integer.parseInt(premiumId);
        if (premiumIdInteger<=0 || premiumIdInteger>90000)
        {
            throw new InvalidIdException("ID must be within 1 to 90000! Please enter
a valid ID.");
        }
        if(premiumPhone.length()<10 || premiumPhone.length()>10)
        {
            throw new InvalidPhoneException("Phone number must be of 10 digits!");

```

```

    }
    //checking for duplicate ID
    for (GymMember member : members)
    {
        if (member.getId() == premiumIdInteger && member instanceof
PremiumMember)    //if member is an object of premium member class
        {
            JOptionPane.showMessageDialog(addAPremiumMemberPanel, "A
member with ID " + premiumIdInteger + " already exists!\nPlease enter another
ID. ","Existing ID",JOptionPane.ERROR_MESSAGE);
            return;    //immediately exits this method when duplicate id is found.
if not written so, it will show the error message and also add the member with same id
        }
    }
    // Create and add new member
    //calling the constructor of premium member which is a subclass of the class
gym member
    GymMember          pMember          =          new
PremiumMember(premiumIdInteger,premiumName,          premiumLocation,
premiumPhone,
                premiumEmail,  premiumGender,  premiumDOBYear+  "-"  +
premiumDOBMonth+ "-" + premiumDOBDay,
                premiumStartYear+ "-" +premiumStartMonth+ "-" + premiumStartDay,
trainer);
    members.add(pMember);    //adds the new object created to the arraylist

    //success message
    JOptionPane.showMessageDialog(addAPremiumMemberPanel, "You
have been added as a premium member!");

    //setting all fields empty

```

```

        nameText2.setText("");
        locationText2.setText("");
        idText2.setText("");
        phoneNoText2.setText("");
        emailText2.setText("");
        DOBYear2.setSelectedIndex(0);           //selects the first option
        DOBMonth2.setSelectedIndex(0);
        DOBDay2.setSelectedIndex(0);
        startDateYear2.setSelectedIndex(0);
        startDateMonth2.setSelectedIndex(0);
        startDateDay2.setSelectedIndex(0);
        personalTrainerText.setText("");
    }
    catch (NumberFormatException ex) {           //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed
        JOptionPane.showMessageDialog(addAPremiumMemberPanel, "ID must
be a number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
    catch (InvalidIdException i)
    {
        JOptionPane.showMessageDialog(addAPremiumMemberPanel,
i.getMessage(), "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
    catch (InvalidPhoneException p)
    {
        JOptionPane.showMessageDialog(addAPremiumMemberPanel, p.getMessage(), "Inv
alid phone number", JOptionPane.ERROR_MESSAGE);
    }
}

```

```

else if(e.getSource()==allMembersButton1)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.remove(memberManageLeftPanel);
    frame.add(allMembersRegularPanel);
    frame.add(allMembersLeftPanel);
    frame.remove(allMembersPremiumPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();    //this is used to re-draw the component

    try
    {
        int
regularInputId=Integer.parseInt(JOptionPane.showInputDialog(allMembersRegularP
anel, "Enter your ID:"));
        for (GymMember member: members)    //for each member in arraylist
members
        {
            if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered
            {
                //casting from gym member to regular member
                RegularMember regularMember=(RegularMember) member;
                regularMember.display();
                return;    //immediately exits the method if id is found
            }
        }
    }
}

```

```

    }
}

//else is not written since it shows this dialog box everytime an id doesnt
match even though it exists in premium

//only if no member was found, i.e if the member doesn't exist at all
JOptionPane.showMessageDialog(allMembersRegularPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}

catch (NumberFormatException ex) {           //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed

    JOptionPane.showMessageDialog(allMembersRegularPanel, "ID must be
a number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
}

else if(e.getSource()==allMembersButton2)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.remove(memberManageLeftPanel);
    frame.add(allMembersLeftPanel);
    frame.add(allMembersRegularPanel);
    frame.remove(allMembersPremiumPanel);
    frame.revalidate();           //this is used to re-calculate the layout/ ensures that
the layout updates properly

    frame.repaint();           //this is used to re-draw the component

```



```

        try{
            int
regularInputId=Integer.parseInt(JOptionPane.showInputDialog(allMembersRegularP
anel, "Enter your ID:"));

            for (GymMember member: members)          //for each member in arraylist
members
            {
                if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered
                {
                    //casting from gym member to regular member
                    RegularMember regularMember=(RegularMember) member;
                    regularMember.display();
                    return;    //immediately exits the method if id is found
                }
            }

            //else is not written since it shows this dialog box everytime an id doesnt
match even though it exists in premium

            //only if no member was found, i.e if the member doesn't exist at all
            JOptionPane.showMessageDialog(allMembersRegularPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
        }

        catch (NumberFormatException ex) {          //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed

            JOptionPane.showMessageDialog(allMembersRegularPanel, "ID must be
a number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
        }
    }

    else if(e.getSource()==allMembersButton3)

```

```

{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.remove(memberManageLeftPanel);
    frame.add(allMembersLeftPanel);
    frame.add(allMembersRegularPanel);
    frame.remove(allMembersPremiumPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();    //this is used to re-draw the component

    try{
        int
        regularInputId=Integer.parseInt(JOptionPane.showInputDialog(allMembersRegularP
        anel, "Enter your ID:"));

        for (GymMember member: members)    //for each member in arraylist
members
        {
            if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered
        {
            //casting from gym member to regular member
            RegularMember regularMember=(RegularMember) member;
            regularMember.display();
            return;    //immediately exits the method if id is found
        }
    }
}

```

//else is not written since it shows this dialog box everytime an id doesnt match even though it exists in premium

//only if no member was found, i.e if the member doesn't exist at all

JOptionPane.showMessageDialog(allMembersRegularPanel, "Member not found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);

}

catch (NumberFormatException ex) { //if the user enters such risky code, catch catches the exception, and allows the user to be informed about what is to be changed

JOptionPane.showMessageDialog(allMembersRegularPanel, "ID must be a number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);

}

}

else if(e.getSource()==allMembersRegularButton1)

{

frame.remove(addARegularMemberPanel);

frame.remove(addAPremiumMemberPanel);

frame.remove(regularMemberPanel);

frame.remove(premiumMemberPanel);

frame.remove(addAMemberLeftPanel);

frame.remove(memberManageLeftPanel);

frame.add(allMembersLeftPanel);

frame.add(allMembersRegularPanel);

frame.remove(allMembersPremiumPanel);

frame.revalidate(); //this is used to re-calculate the layout/ ensures that the layout updates properly

frame.repaint(); //this is used to re-draw the component

try{

```

        int
        regularInputId=Integer.parseInt(JOptionPane.showInputDialog(allMembersRegularP
        anel, "Enter your ID:"));

        for (GymMember member: members)      //for each member in arraylist
members
        {
            if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered
        {
            //casting from gym member to regular member
            RegularMember regularMember=(RegularMember) member;
            regularMember.display();
            return;    //immediately exits the method if id is found
        }
    }

    //else is not written since it shows this dialog box everytime an id doesnt
match even though it exists in premium

    //only if no member was found, i.e if the member doesn't exist at all
    JOptionPane.showMessageDialog(allMembersRegularPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}

catch (NumberFormatException ex) {      //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed

    JOptionPane.showMessageDialog(allMembersRegularPanel, "ID must be
a number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
}
}

else if(e.getSource()==allMembersRegularButton2)
{
    frame.remove(addARegularMemberPanel);

```

```

        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(addAMemberLeftPanel);
        frame.remove(memberManageLeftPanel);
        frame.add(allMembersLeftPanel);
        frame.add(allMembersRegularPanel);
        frame.remove(allMembersPremiumPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that the
layout updates properly
        frame.repaint();    //this is used to re-draw the component
        try
        {
            int
regularInputId=Integer.parseInt(JOptionPane.showInputDialog(allMembersRegularP
anel, "Enter your ID:"));
            for (GymMember member: members)    //for each member in arraylist
members
            {
                if(member.getId()==regularInputId    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id entered and if it
belongs to regular member
                {
                    //casting from gym member to regular member
                    RegularMember regularMember=(RegularMember) member;
                    regularMember.display();
                    return;    //immediately exits the method if id is found
                }
            }
            //else is not written since it shows this dialog box everytime an id doesnt
match even though it exists in premium

```

```

        //only if no member was found, i.e if the member doesn't exist at all
        JOptionPane.showMessageDialog(allMembersRegularPanel, "Member not
found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }

    catch (NumberFormatException ex) {                //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed
        JOptionPane.showMessageDialog(allMembersRegularPanel, "ID must be
a number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
}

else if(e.getSource()==allMembersPremiumButton1)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(addAMemberLeftPanel);
    frame.remove(memberManageLeftPanel);
    frame.add(allMembersLeftPanel);
    frame.remove(allMembersRegularPanel);
    frame.add(allMembersPremiumPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that the
layout updates properly
    frame.repaint();        //this is used to re-draw the component
    try{
        int
premiumInputId=Integer.parseInt(JOptionPane.showInputDialog(allMembersPremiu
mPanel, "Enter your ID:"));

```

```

        for (GymMember member: members)           //for each member in arraylist
members
    {
        if(member.getId()==premiumInputId    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id entered and if it
belongs to premium member
        {
            //casting from gym member to regular member
            PremiumMember premiumMember=(PremiumMember) member;
            premiumMember.display();
            return;    //immediately exits the method if id is found
        }
    }

    //else is not written since it shows this dialog box everytime an id doesnt
match even though it exists in regular

    //only if no member was found, i.e if the member doesn't exist at all
    JOptionPane.showMessageDialog(allMembersPremiumPanel,    "Member
not found!","Invalid ID", JOptionPane.ERROR_MESSAGE);
}

    catch (NumberFormatException ex) {            //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed

        JOptionPane.showMessageDialog(allMembersPremiumPanel, "ID must be
a number!","Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
}

else if(e.getSource()==allMembersPremiumButton2)
{
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
}

```

```

        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(addAMemberLeftPanel);
        frame.remove(memberManageLeftPanel);
        frame.add(allMembersLeftPanel);
        frame.remove(allMembersRegularPanel);
        frame.add(allMembersPremiumPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that the
layout updates properly
        frame.repaint();      //this is used to re-draw the component
        try{
            int
premiumInputId=Integer.parseInt(JOptionPane.showInputDialog(allMembersPremiumPanel, "Enter your ID:"));
            for (GymMember member: members)    //for each member in arraylist
members
            {
                if(member.getId()==premiumInputId    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id entered
                {
                    //casting from gym member to regular member
                    PremiumMember premiumMember=(PremiumMember) member;
                    premiumMember.display();
                    return;    //immediately exits the method if id is found
                }
            }
            //else is not written since it shows this dialog box everytime an id doesnt
match even though it exists in regular
            //only if no member was found, i.e if the member doesn't exist at all
            JOptionPane.showMessageDialog(allMembersPremiumPanel,    "Member
not found!", "Invalid ID", JOptionPane.ERROR_MESSAGE);

```



```

    }
    catch (NumberFormatException ex) {           //if the user enters such risky
code, catch catches the exception, and allows the user to be informed about what is
to be changed
        JOptionPane.showMessageDialog(allMembersPremiumPanel, "ID must be
a number!", "Invalid ID", JOptionPane.ERROR_MESSAGE);
    }
}
else if(e.getSource()==addClearButton1)
{
    //setting all fields empty
    nameText1.setText("");
    locationText1.setText("");
    idText1.setText("");
    phoneNoText1.setText("");
    emailText1.setText("");
    referralSourceText.setText("");
    DOBYear1.setSelectedIndex(0);           //selects the first option
    DOBMonth1.setSelectedIndex(0);
    DOBDay1.setSelectedIndex(0);
    startDateYear1.setSelectedIndex(0);
    startDateMonth1.setSelectedIndex(0);
    startDateDay1.setSelectedIndex(0);
    //confirmation message
    JOptionPane.showMessageDialog(addARegularMemberPanel,    "Welcome!
Please enter information about the new member.");
}
else if(e.getSource()==addClearButton2)
{
    //setting all fields empty
    nameText2.setText("");

```

```
locationText2.setText("");
idText2.setText("");
phoneNoText2.setText("");
emailText2.setText("");
personalTrainerText.setText("");
DOBYear2.setSelectedIndex(0);           //selects the first option
DOBMonth2.setSelectedIndex(0);
DOBDay2.setSelectedIndex(0);
startDateYear2.setSelectedIndex(0);
startDateMonth2.setSelectedIndex(0);
startDateDay2.setSelectedIndex(0);

//confirmation message
JOptionPane.showMessageDialog(addAPremiumMemberPanel, "Welcome!
Please enter information about the new member.");
}
else if(e.getSource()==memberClearButton1)
{
    //setting all fields empty
    nameOutputLabel1.setText("");
    locationOutputLabel1.setText("");
    idOutputLabel1.setText("");
    phoneOutputLabel1.setText("");
    emailOutputLabel1.setText("");
    planOutputLabel.setText("");
    dobOutputLabel1.setText("");
    startDateOutputLabel1.setText("");
    attendanceOutputLabel1.setText("");
    loyaltyPointsOutputLabel1.setText("");
    genderOutputLabel1.setText("");
    priceOutputLabel.setText("");
}
```

```

        removalReasonText.setText("");
        referralSourceOutputLabel.setText("");
        removalReasonOutputLabel.setText("");
    }
    else if(e.getSource()==memberClearButton2)
    {
        //setting all fields empty
        nameOutputLabel2.setText("");
        locationOutputLabel2.setText("");
        idOutputLabel2.setText("");
        phoneOutputLabel2.setText("");
        emailOutputLabel2.setText("");
        personalTrainerOutputLabel2.setText("");
        dobOutputLabel2.setText("");
        startDateOutputLabel2.setText("");
        attendanceOutputLabel2.setText("");
        loyaltyPointsOutputLabel2.setText("");
        genderOutputLabel2.setText("");
        fullPaymentOutputLabel.setText("");
        paidAmountOutputLabel.setText("");
    }
    else if(e.getSource()==attendanceButton1)
    {
        String regularId=idOutputLabel1.getText();           //getting the id from the
displayed label
        int regularIdInteger = Integer.parseInt(regularId);

        for (GymMember member: members)                     //for each member in arraylist
members
        {

```

```

        if(member.getId()==regularIdInteger    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to regular member
    {
        if(member.getActiveStatus()==true)
        {
            member.markAttendance();
            int updatedAttendance=member.getAttendance();

attendanceOutputLabel1.setText(String.valueOf(updatedAttendance));
            double updatedLoyaltyPoints=member.getLoyaltyPoints();

loyaltyPointsOutputLabel1.setText(String.valueOf(updatedLoyaltyPoints));
        }
        else
        {
            JOptionPane.showMessageDialog(regularMemberPanel,"Please
activate your membership first to mark attendance!","Membership Activation",
JOptionPane.WARNING_MESSAGE);
        }
    }
}
else if(e.getSource()==attendanceButton2)
{
    String premiumId=idOutputLabel2.getText();    //getting the id from the
displayed label
    int premiumIdInteger = Integer.parseInt(premiumId);

    for (GymMember member: members)    //for each member in arraylist
members

```

```

        {
            if(member.getId()==premiumIdInteger    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to premium member
            {
                if(member.getActiveStatus()==true)
                {
                    member.markAttendance();
                    int updatedAttendance=member.getAttendance();

attendanceOutputLabel2.setText(String.valueOf(updatedAttendance));
                    double updatedLoyaltyPoints=member.getLoyaltyPoints();

loyaltyPointsOutputLabel2.setText(String.valueOf(updatedLoyaltyPoints));
                }
                else
                {
                    JOptionPane.showMessageDialog(premiumMemberPanel,"Please
activate your membership first to mark attendance!","Membership Activation",
JOptionPane.WARNING_MESSAGE);
                }
            }
        }

        else if(e.getSource()==activateButton1)
        {
            String regularId=idOutputLabel1.getText();    //getting the id from the
displayed label
            int regularIdInteger = Integer.parseInt(regularId);

```

```

        for (GymMember member: members)           //for each member in arraylist
members
    {
        if(member.getId()==regularIdInteger    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to regular member
    {
        member.activateMembership();
        JOptionPane.showMessageDialog(regularMemberPanel,"Your account
has been activated!");
    }
    }
    else if(e.getSource()==activateButton2)
    {
        String premiumId=idOutputLabel2.getText();           //getting the id from the
displayed label
        int premiumIdInteger = Integer.parseInt(premiumId);

        for (GymMember member: members)           //for each member in arraylist
members
    {
        if(member.getId()==premiumIdInteger    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to premium member
    {
        member.activateMembership();
        JOptionPane.showMessageDialog(premiumMemberPanel,"Your
account has been activated!");
    }
    }
    }

```

```
    }  
    else if(e.getSource()==deactivateButton1)  
    {  
        String regularId=idOutputLabel1.getText();           //getting the id from the  
displayed label  
        int regularIdInteger = Integer.parseInt(regularId);  
  
        for (GymMember member: members)           //for each member in arraylist  
members  
        {  
            if(member.getId()==regularIdInteger    &&    member    instanceof  
RegularMember)    //if the id returned from getter is equal to the id from the panel  
and if it belongs to regular member  
            {  
                member.deactivateMembership();  
            }  
        }  
    }  
    else if(e.getSource()==deactivateButton2)  
    {  
        String premiumId=idOutputLabel2.getText();           //getting the id from the  
displayed label  
        int premiumIdInteger = Integer.parseInt(premiumId);  
  
        for (GymMember member: members)           //for each member in arraylist  
members  
        {  
            if(member.getId()==premiumIdInteger    &&    member    instanceof  
PremiumMember)    //if the id returned from getter is equal to the id from the panel  
and if it belongs to premium member  
            {
```

```

        member.deactivateMembership();
    }
}
else if(e.getSource()==saveButton1)
{
    String regularId=idOutputLabel1.getText();           //getting the id from the
displayed label
    int regularIdInteger = Integer.parseInt(regularId);

    for (GymMember member: members)           //for each member in arraylist
members
    {
        if(member.getId()==regularIdInteger    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to regular member
        {
            String regularRemovalReason= removalReasonText.getText();
            removalReasonOutputLabel.setText(regularRemovalReason);
        }
    }
}
else if(e.getSource()==upgradePlanButton)
{
    String regularId=idOutputLabel1.getText();           //getting the id from the
displayed label
    int regularIdInteger = Integer.parseInt(regularId);

    for (GymMember member: members)           //for each member in arraylist
members
    {

```



```

        if(member.getId()==regularIdInteger    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to regular member

```

```

    {
        String newPlan=(String) planComboBox.getSelectedItem();
        //casting
        RegularMember regularMember = (RegularMember) member;
        String upgradedPlan=regularMember.upgradePlan(newPlan);

```

```

JOptionPane.showMessageDialog(regularMemberPanel,upgradedPlan);    //calling
the method through dialog box so that the returned statements can be displayed
        planOutputLabel.setText(regularMember.getPlan());

```

```

priceOutputLabel.setText(String.valueOf(regularMember.getPlanPrice(regularMember.getPlan())));

```

```

        //sets the value obtained from getPlanPrice method which again takes the
plan as parameter

```

```

    }

```

```

}

```

```

}

```

```

else if(e.getSource()==revertButton1)

```

```

{

```

```

    String regularId=idOutputLabel1.getText();    //getting the id from the
displayed label

```

```

    int regularIdInteger = Integer.parseInt(regularId);

```

```

        for (GymMember member: members)    //for each member in arraylist
members

```

```

    {

```

```

        if(member.getId()==regularIdInteger    &&    member    instanceof
RegularMember)    //if the id returned from getter is equal to the id from the panel

```

```

    {
        String reason=removalReasonText.getText();
        RegularMember regularMember = (RegularMember) member;
        regularMember.revertRegularMember(reason);
        planOutputLabel.setText(regularMember.getPlan());
        planComboBox.setSelectedIndex(0);
        priceOutputLabel.setText(String.valueOf(regularMember.getPrice()));
        attendanceOutputLabel1.setText("");
        loyaltyPointsOutputLabel1.setText("");
        removalReasonText.setText("");
        removalReasonOutputLabel.setText("");
        JOptionPane.showMessageDialog(regularMemberPanel,"Your
membership has been reset.");
    }
}
}
else if(e.getSource()==revertButton2)
{
    String premiumId=idOutputLabel2.getText();           //getting the id from the
displayed label
    int premiumIdInteger = Integer.parseInt(premiumId);

    for (GymMember member: members)           //for each member in arraylist
members
    {
        if(member.getId()==premiumIdInteger    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to premium member
        {
            PremiumMember premiumMember = (PremiumMember) member;
            premiumMember.revertPremiumMember();

```

```

        attendanceOutputLabel1.setText("");
        loyaltyPointsOutputLabel1.setText("");
        discountOutputLabel.setText("");
        paidAmountOutputLabel.setText("");
        personalTrainerOutputLabel2.setText("");
        fullPaymentOutputLabel.setText("");
        attendanceOutputLabel2.setText("");
        loyaltyPointsOutputLabel2.setText("");
        JOptionPane.showMessageDialog(regularMemberPanel,"Your
membership has been reset.");
    }
}
}
else if(e.getSource()==discountButton)
{
    String premiumId=idOutputLabel2.getText();           //getting the id from the
displayed label
    int premiumIdInteger = Integer.parseInt(premiumId);

    for (GymMember member: members)           //for each member in arraylist
members
    {
        if(member.getId()==premiumIdInteger    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id from the panel
and if it belongs to premium member
        {
            PremiumMember premiumMember = (PremiumMember) member;
            premiumMember.calculateDiscount();
        }
    }
}
}

```

```

        else if(e.getSource()==paymentButton)
        {
            String premiumId=idOutputLabel2.getText();           //getting the id from the
displayed label
            int premiumIdInteger = Integer.parseInt(premiumId);
            String
paidAmount=JOptionPane.showInputDialog(premiumMemberPanel,"Enter      the
amount you want to pay.");
            int premiumPaidAmt=Integer.parseInt(paidAmount);      //converting amount
entered by user from String to int
            for (GymMember member: members)           //for each member in arraylist
members
            {
                if(member.getId()==premiumIdInteger    &&    member    instanceof
PremiumMember)    //if the id returned from getter is equal to the id from the panel
                {
                    PremiumMember premiumMember = (PremiumMember) member;
                    String payAmt=premiumMember.payDueAmount(premiumPaidAmt);

                    JOptionPane.showMessageDialog(premiumMemberPanel,    payAmt);
//calling the method through dialog box so that the returned statements can be
displayed

                    paidAmountOutputLabel.setText(String.valueOf(premiumMember.getPaidAmount()))
;

                    fullPaymentOutputLabel.setText(String.valueOf(premiumMember.getIsFullPayment()
));

                }
            }

```

```

    }
    else if(e.getSource()==saveToFileButton1)
    {
        file=new File("MemberDetailsRegular.txt"); //making file in same directory
        //to actually create a file
        try{
            if(file.createNewFile()) //doesnt exist--> true
            {
                System.out.println("File is created");
            }
            else{ //MemberDetails.txt exists
                System.out.println("File already exists");
            }
            writer=new FileWriter(file, true); //location is stored in file variable
            //getting the values in string format
            //using String format
            String headLnFormat=String.format("%-6s %-25s %-15s %-20s %-30s %-10s %-15s %-30s %-15s %-15s %-15s %-15s %-15s %-25s %-25s\n",
                "ID", "Name", "Location", "Phone",
                "Email","Gender","DOB","Membership Start Date",
                "Attendance", "Loyalty Points", "ActiveStatus","Plan", "Price","Eligible
for Upgrade",
                "Referral Source");
            writer.write(headLnFormat+"\n");
            for (GymMember member: members) //for each member in arraylist
members
            {
                if(member instanceof RegularMember) //if the id belongs to regular
member
                {
                    //casting

```

```

        RegularMember regularMember = (RegularMember) member;
        String infoInFormat=String.format("%-6s %-25s %-15s %-20s %-30s
%-10s %-15s %-30s %-15s %-15s %-15s %-15s %-15s %-25s %-25s\n",
            member.getId(), member.getName(), member.getLocation(),
member.getPhone(), member.getEmail(), member.getGender(), member.getDOB(),
member.getMembershipStartDate(),
            member.getAttendance(), member.getLoyaltyPoints(),
member.getActiveStatus(), regularMember.getPlan(), regularMember.getPrice(),
regularMember.getIsEligibleForUpgrade(),
            regularMember.getReferralSource());

        writer.write(infoInFormat);
    }
}
}
catch(IOException f)
{
    System.out.println("An error has occurred while writing. Please try again.");
}
finally{
    //an exception might occur here too, so using try and catch blocks
    try{
        writer.close();
    }
    catch(IOException f)
    {
        System.out.println("Sorry! File couldn't be closed properly. Please try
again");
    }
}
}
}

```

```

else if(e.getSource()==readFromFileButton1)
{
    frame.add(readFromFilePanel1);
    frame.remove(readFromFilePanel2);
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.remove(addAMemberLeftPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();
    try{
        //to read
        reader=new FileReader(file);
        int ch; //reads in ascii value which is stored in ch
        while((ch=reader.read())!=-1) //until it is not the end of file (until the ascii
value is not -1)
        {
            char c=(char) ch;    //converting the ascii value to char, to obtain data
represented by ascii value
            readAreaRegular.append(String.valueOf(c));    //printing the data in the
form of string
        }
    }
    catch(IOException f)
    {
        System.out.println("An error has occurred while reading. Please try again.");
    }
}

```

```

finally{
    //an exception might occur here too, so using try and catch blocks
    try{
        reader.close();
    }
    catch(IOException f)
    {
        System.out.println("Sorry! File couldn't be closed properly. Please try
again");
    }
}
}
else if(e.getSource()==saveToFileButton2)
{
    file2=new File("MemberDetailsPremium.txt"); //making file in same directory
    //to actually create a file
    try{
        if(file2.createNewFile()) //doesnt exist--> true
        {
            System.out.println("File is created");
        }
        else{ //MemberDetails.txt exists
            System.out.println("File already exists");
        }
        writer=new FileWriter(file2, true); //location is stored in file variable
        //getting the values in string format
        //using String format
        String headLnFormat=String.format("%-6s %-25s %-15s %-20s %-30s %-
10s %-15s %-30s %-15s %-15s %-15s %-15s %-15s %-18s %-25s\n",
            "ID",           "Name",           "Location",           "Phone",
            "Email","Gender","DOB","Membership Start Date",

```



```

        "Attendance", "Loyalty Points", "ActiveStatus","Full Payment", "Paid
Amount","Discount Amount",
        "Premium Charge");
    writer.write(headlnFormat+"\n");
    for (GymMember member: members)        //for each member in arraylist
members
    {
        if(member instanceof PremiumMember)    //if the id belongs to regular
member
        {
            //casting
            PremiumMember premiumMember = (PremiumMember) member;
            String infoLnFormat=String.format("%-6s %-25s %-15s %-20s %-30s
%-10s %-15s %-30s %-15s %-15s %-15s %-15s %-15s %-18s %-25s\n",
                member.getId(), member.getName(), member.getLocation(),
member.getPhone(), member.getEmail(), member.getGender(), member.getDOB(),
member.getMembershipStartDate(),
                member.getAttendance(), member.getLoyaltyPoints(),
member.getActiveStatus(), premiumMember.getIsFullPayment(),
premiumMember.getPaidAmount(), premiumMember.getDiscountAmount(),
                premiumMember.getPremiumCharge());

            writer.write(infoLnFormat);
        }
    }
}
catch(IOException f)
{
    System.out.println("An error has occurred while writing. Please try again.");
}
finally{

```

```

        //an exception might occur here too, so using try and catch blocks
        try{
            writer.close();
        }
        catch(IOException f)
        {
            System.out.println("Sorry! File couldn't be closed properly. Please try
again");
        }
    }
}

else if(e.getSource()==readFromFileButton2)
{
    frame.add(readFromFilePanel2);
    frame.remove(readFromFilePanel1);
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.remove(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.remove(addAMemberLeftPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();
    try{
        //to read
        reader=new FileReader(file2);
        int ch; //reads in ascii value which is stored in ch
    }
}

```

```

        while((ch=reader.read())!=-1) //until it is not the end of file (until the ascii
value is not -1)
        {
            char c=(char) ch;        //converting the ascii value to char, to obtain data
represented by ascii value
            readAreaPremium.append(String.valueOf(c)); //printing the data in the
form of string
        }
    }
    catch(IOException f)
    {
        System.out.println("An error has occurred while reading. Please try again.");
    }
    finally{
        //an exception might occur here too, so using try and catch blocks
        try{
            reader.close();
        }
        catch(IOException f)
        {
            System.out.println("Sorry! File couldn't be closed properly. Please try
again");
        }
    }
}
else if(e.getSource()==displayButton1)
{
    frame.add(displayPanel1);
    frame.remove(displayPanel2);
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
}

```

```

        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(memberManageLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.remove(addAMemberLeftPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();
        for (GymMember member: members)    //for each member in arraylist
members
        {
            if(member instanceof RegularMember)    //if the id belongs to regular
member
            {
                //casting
                RegularMember regularMember = (RegularMember) member;
                displayAreaRegular.append(String.format("%-6s %-25s %-25s %-15s %-
30s %-10s %-15s %-30s %-15s %-15s %-15s %-15s %-15s %-25s %-25s\n",
                    member.getId(),    member.getName(),    member.getLocation(),
member.getPhone(), member.getEmail(), member.getGender(), member.getDOB(),
member.getMembershipStartDate(),
                    member.getAttendance(),    member.getLoyaltyPoints(),
member.getActiveStatus(),    regularMember.getPlan(),    regularMember.getPrice(),
regularMember.getisEligibleForUpgrade(),
                    regularMember.getReferralSource()));
            }
        }
    }
    else if(e.getSource()==displayButton2)
    {
        frame.add(displayPanel2);
    }
}

```

```

        frame.remove(displayPanel1);
        frame.remove(addARegularMemberPanel);
        frame.remove(addAPremiumMemberPanel);
        frame.remove(regularMemberPanel);
        frame.remove(premiumMemberPanel);
        frame.remove(memberManageLeftPanel);
        frame.remove(allMembersLeftPanel);
        frame.remove(addAMemberLeftPanel);
        frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
        frame.repaint();

        for (GymMember member: members)    //for each member in arraylist
members
        {
            if(member instanceof PremiumMember)    //if the id belongs to regular
member
            {
                //casting
                PremiumMember premiumMember = (PremiumMember) member;
                displayAreaPremium.append(String.format("%-6s  %-25s  %-15s  %-20s
%-30s  %-10s  %-15s  %-30s  %-15s  %-15s  %-15s  %-15s  %-18s  %-25s\n",
                    member.getId(),    member.getName(),    member.getLocation(),
member.getPhone(), member.getEmail(), member.getGender(), member.getDOB(),
member.getMembershipStartDate(),
                    member.getAttendance(),    member.getLoyaltyPoints(),
member.getActiveStatus(),    premiumMember.getIsFullPayment(),
premiumMember.getPaidAmount(), premiumMember.getDiscountAmount(),
                    premiumMember.getPremiumCharge()));
            }
        }

```

```
    }  
  }  
  else if(e.getSource()==backbutton1)  
  {  
    frame.remove(displayPanel2);  
    frame.remove(displayPanel1);  
    frame.remove(addARegularMemberPanel);  
    frame.remove(addAPremiumMemberPanel);  
    frame.add(regularMemberPanel);  
    frame.remove(premiumMemberPanel);  
    frame.add(memberManageLeftPanel);  
    frame.remove(allMembersLeftPanel);  
    frame.remove(addAMemberLeftPanel);  
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that  
the layout updates properly  
    frame.repaint();  
  }  
  else if(e.getSource()==backbutton2)  
  {  
    frame.remove(displayPanel2);  
    frame.remove(displayPanel1);  
    frame.remove(addARegularMemberPanel);  
    frame.remove(addAPremiumMemberPanel);  
    frame.remove(regularMemberPanel);  
    frame.add(premiumMemberPanel);  
    frame.add(memberManageLeftPanel);  
    frame.remove(allMembersLeftPanel);  
    frame.remove(addAMemberLeftPanel);  
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that  
the layout updates properly  
    frame.repaint();
```

```
}
else if(e.getSource()==readBackButton1)
{
    frame.remove(readFromFilePanel1);
    frame.remove(readFromFilePanel2);
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(premiumMemberPanel);
    frame.add(regularMemberPanel);
    frame.add(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.remove(addAMemberLeftPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();
}
else if(e.getSource()==readBackButton2)
{
    frame.remove(readFromFilePanel1);
    frame.remove(readFromFilePanel2);
    frame.remove(addARegularMemberPanel);
    frame.remove(addAPremiumMemberPanel);
    frame.remove(regularMemberPanel);
    frame.add(premiumMemberPanel);
    frame.add(memberManageLeftPanel);
    frame.remove(allMembersLeftPanel);
    frame.remove(addAMemberLeftPanel);
    frame.revalidate();    //this is used to re-calculate the layout/ ensures that
the layout updates properly
    frame.repaint();
}
```

```
    }  
    //main method  
    public static void main (String[] args)  
    {  
        new GymGUI();  
    }  
}
```

10.5 InvalidIdException Class:

```
public class InvalidIdException extends Exception  
{  
    public InvalidIdException(String message)  
    {  
        super(message);    //giving message to parent  
    }  
}
```

10.6 InvalidPhoneException Class:

```
public class InvalidPhoneException extends Exception  
{  
    public InvalidPhoneException(String message)  
    {  
        super(message);    //giving message to parent  
    }  
}
```