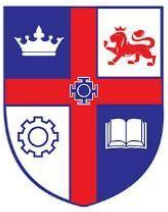
LONDON METROPOLITAN UNIVERSITY



Islington college
(इस्लिङ्टन कलेज)

CS4001NI Programming

60% Individual Coursework

2025 Spring

Student Name: Sichu Maharjan

London Met ID: 24046892

College ID: NP01CP4A240151

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.

24046892 Sichu Maharjan copy.docx

Islington College,Nepal

Document Details

Submission ID

trn:oid:::3618:96105175

Submission Date

May 16, 2025, 12:24 AM GMT+5:45

Download Date

May 16, 2025, 12:26 AM GMT+5:45

File Name

24046892 Sichu Maharjan copy.docx

File Size

75.5 KB

80 Pages

8,024 Words

48,202 Characters

5% Overall Similarity

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

Match Groups

- **28** Not Cited or Quoted 3%
  Matches with neither in-text citation nor quotation marks
- **2** Missing Quotations 0%
  Matches that are still very similar to source material
- **16** Missing Citation 2%
  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
1% Publications
4% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

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

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

Match Groups

- **28** Not Cited or Quoted 3%
  Matches with neither in-text citation nor quotation marks
- **2** Missing Quotations 0%
  Matches that are still very similar to source material
- **16** Missing Citation 2%
  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
1% Publications
4% Submitted works (Student Papers)

Top Sources

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

1	Submitted works	iacademy on 2025-05-14 <1%
2	Submitted works	University of Westminster on 2017-05-01 <1%
3	Internet	github.com <1%
4	Submitted works	University of London External System on 2011-04-27 <1%
5	Internet	www.geeksforgeeks.org <1%
6	Submitted works	Somerset College of Arts and Technology, Somerset on 2024-09-13 <1%
7	Submitted works	Southern New Hampshire University - Continuing Education on 2024-08-05 <1%
8	Submitted works	Asia Pacific University College of Technology and Innovation (UCTI) on 2023-10-30 <1%
9	Submitted works	Colorado Technical University Online on 2016-08-08 <1%
10	Submitted works	University of London External System on 2013-04-29 <1%

11	Publication	Usharani Bhimavarapu. "Java 22 for Healthcare and Medical Applications", CRC Pr... <1%
12	Submitted works	Informatics Education Limited on 2009-04-09 <1%
13	Submitted works	West Cheshire College on 2017-01-27 <1%
14	Submitted works	Tarumanagara University on 2022-12-20 <1%
15	Submitted works	Mansfield State High School on 2024-03-08 <1%
16	Submitted works	Swinburne University of Technology on 2016-10-28 <1%
17	Submitted works	University of Portsmouth on 2024-03-21 <1%
18	Submitted works	Sim University on 2022-04-23 <1%
19	Submitted works	University of Greenwich on 2025-04-27 <1%
20	Internet	archive.org <1%
21	Internet	javaindore.blogspot.in <1%
22	Submitted works	University of Strathclyde on 2024-03-28 <1%

Table of Contents

List of tables

Introduction

Introduction to the coursework:

As a part of first year coursework for programming module, this system is developed. This system is developed considering the scenario of a gym. This system allows the gym to maintain a record for their members which involves their name, contact information as well as their eligibility to use the facilities of the gym. This system not only works to store information but also to track the frequency of members attending the gym as well as counting their loyalty points. Not only that, but this system tracks also whether the member has paid the fee for using the gym or not and if it is paid in full or not.

This system has differentiated between regular members of the gym and premium members who have paid extra to use other facilities of the gym. This coursework required me to use all the Java components which has only been studied and practice but not used for a real-world scenario system. This system combines various components like class, encapsulation, inheritance, abstraction, etc. Different components are used in various places to ensure an easy and functional system.

In this the GymMember class is an abstract class which define the variable, which is used by the child class, RegularMember and PremiumMember. These classes extends the GymMember class and has functions of their own too. These functions are carried out by the help of a GUI named GymGUI class where an user interface is created to help the user navigate through different functions and methods. The buttons present in the GUI triggers different methods to be executed. In this way, this gym system takes life and becomes a fully functioning system.

The regular member frame contains buttons like add regular member, mark attendance, activate/deactivate/revert membership, upgrade plan, save, read from file and many more. As the title of the button suggests, it does these things. Similarly, the premium member frame contains similar button in addition to discount, and pay button.

Aim and Objectives:

This system was created with the primary goal of developing a working system for a gym. This website has various goals and objectives to be achieved for being a fully functioning system.

Aim:

The aim is to develop a functioning gym system which can do series of operations ranging from adding members, marking attendance, updating plans, calculating payment, calculating discount, and many more.

Objectives:

1. To create a user-friendly GUI interface.
2. To develop a system which manages different plans for the member and allowing them to upgrade to new plan or to deactivate their membership.
3. To track the information (id, name, location, contact information) of members
4. To implement the payment tracking to ensure members have paid full amount, neither less nor more than the requirement.
5. To make sure some variables values cannot be changed.
6. To give members accurate discounts.

Programming language used: Java

Java is an object-oriented programming language developed by James Gosling in the year 1995. Java is a widely used OOP language and software platform which runs on many devices worldwide(Anon., 2021). The syntax of Java is based on C and C++ languages. Java is used to develop various mobile application, desktop application, business system, scientific application, database connection and many more.

Features:

1. Java has open source and rich API (Application programming Interface) tools available for programmers to use.
2. Java code is portable as it works on different platforms known as platform independent.

3. Java is both complied and interpreted language.
4. Java supports running many small tasks simultaneously. (College, n.d.)

Concepts:

1. **Inhritance**: this allows reusability of code. In Java, variables and method used in one class can be used in another class with the concept of inheritance. It is achieved by using the keyword extends. (College, n.d.)
2. **Encapsulation**: the process of wrapping variable and methods into a single unit is called encapsulation. In this coursework encapsulation is achieved by using getter/setter methods. This ensures data security. (College, n.d.)
3. **Abstraction**: this is the process of hiding how the function works and only showing that the function works to the user. It makes sure that all the unnecessary information are hidden from the user. (College, n.d.)
4. **Polymorphism**: polymorphism is a concept which allows same name methods. These methods can do two different things but will have the same name. according to the situation, the required methods are triggered. It is of two types, overloading an overriding

Tools used:

1. BlueJ

BlueJ is a free windows-based platform for JDK (Java Development Kit) started in 1995. BlueJ was developed to help teach and learn OOP (Object-Oriented Programming). It can be freely download from its website. BlueJ is easy to use and has simple interface. It even makes debugging easier as it shows the error line. BlueJ runs on any platform whether it is windows, Mac or Linux (Anon., 2022). BlueJ has several features which ensures user has an easy time using it. It colour codes block of statements to make it easier to read and spot missing brackets. BlueJ allows the user to interact with objects within the program, pass them as parameters, call methods, etc. (Anon., n.d.) (Anon., n.d.)

2. Draw.io

It is a free website which allows the user to design variety of stuffs. It lets user create class diagram, flowchart, ER diagram, advanced drawing, anything. It contains variety of diagrammatic tools which assists users to create any visual diagrams of anything. It also allows user to save the created diagram in google drive, one drive or the device itself. There is no need to sign up on this website for basic use. It supports exporting to multiple formats such as PDF, PNG, JPEG, etc. (Anon., n.d.)

In this coursework, draw.io is used to create the class diagram. It already contained the template for the class diagram. It was used to create the class diagrams of the four classes made in this system.

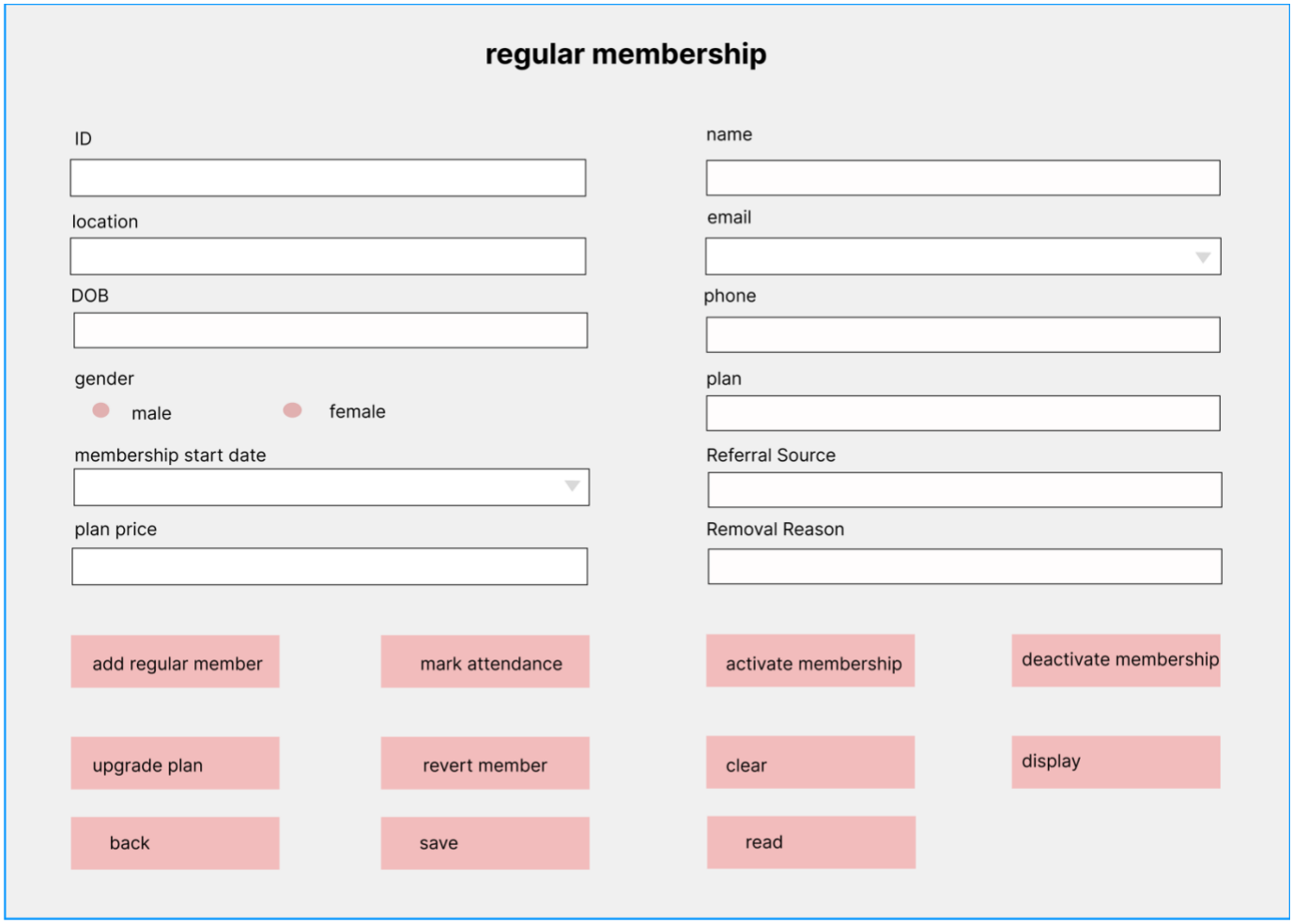
3. Figma

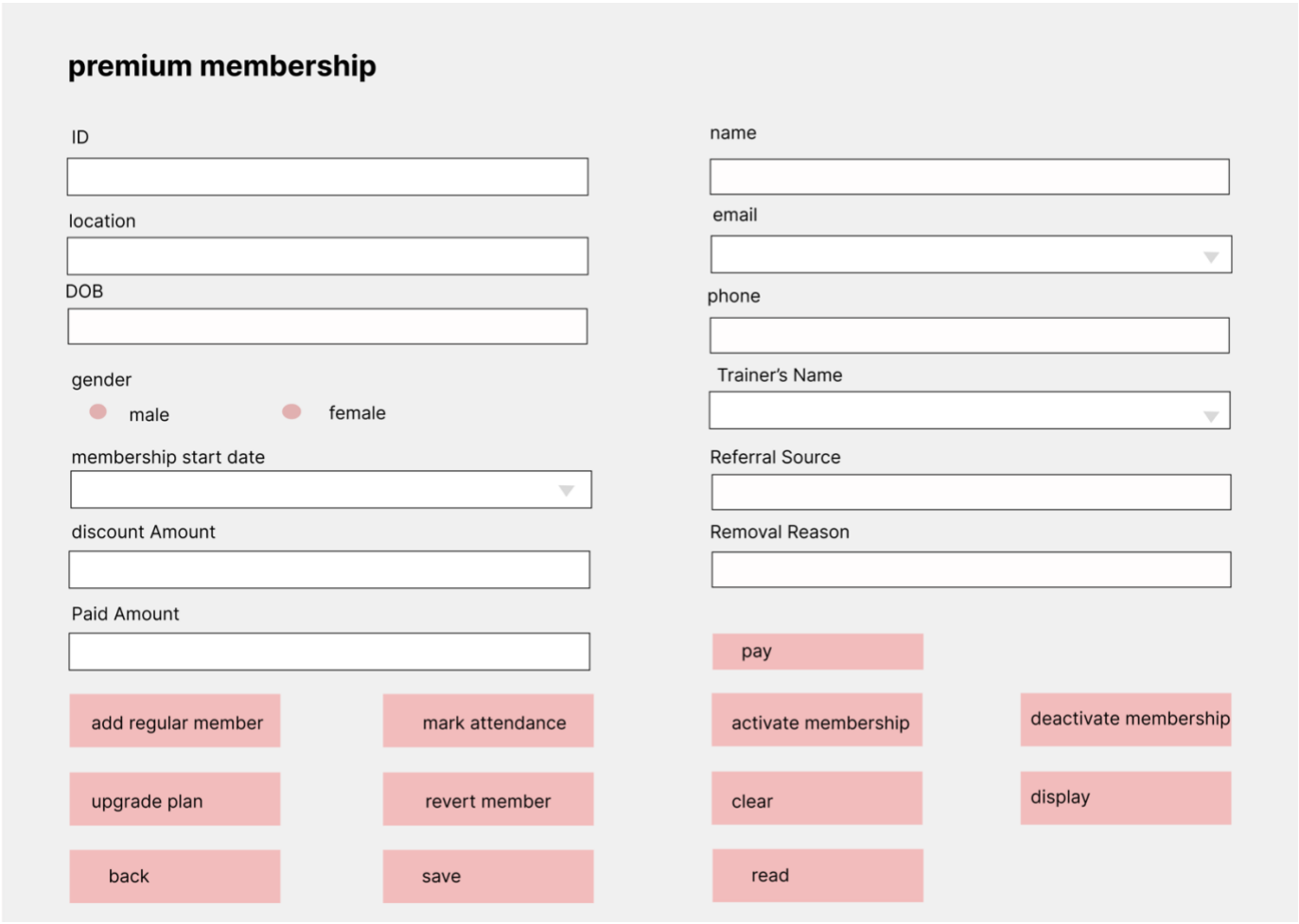
Figma is a design tool which is mainly used for UI/UX design, and wireframing websites or any other applications. It runs in your browser so there is no need for installing it. Developers, students and many people use figma for creating wireframes. It also allows user to collaborate with people to make different designs. (Anon., n.d.)

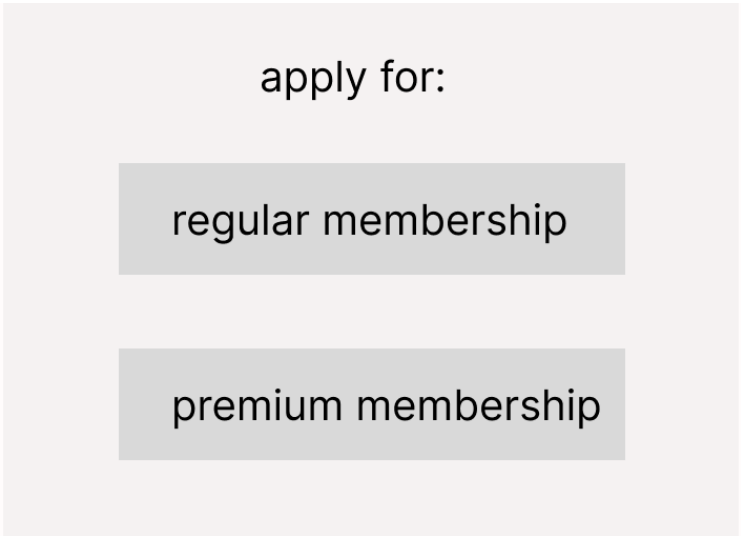
It was used to create the wireframe of the GUI. Different tools of figma was used inorder to create the wireframe of the GUI. Frame, shapes, text, color and many more tools were used to create the wireframe.

GUI wireframe

Wireframe is a simple visual layout that is made to give the developer some sort of idea on how to arrange stuffs. It might not contains any design, it may be of simple kind just pin pointing where the components should be set.In this coursework, the wireframe is made for the GUI part. This wireframe gives a general idea for arranging various components of the GUI such as label, button, combo box and many more.




premium membership
ID
location
DOB
gender
male
female
membership start date
discount Amount
Paid Amount
name
email
phone
Trainer's Name
Referral Source
Removal Reason
pay
add regular member
mark attendance
activate membership
deactivate membership
upgrade plan
revert member
clear
display
back
save
read


apply for:
regular membership
premium membership

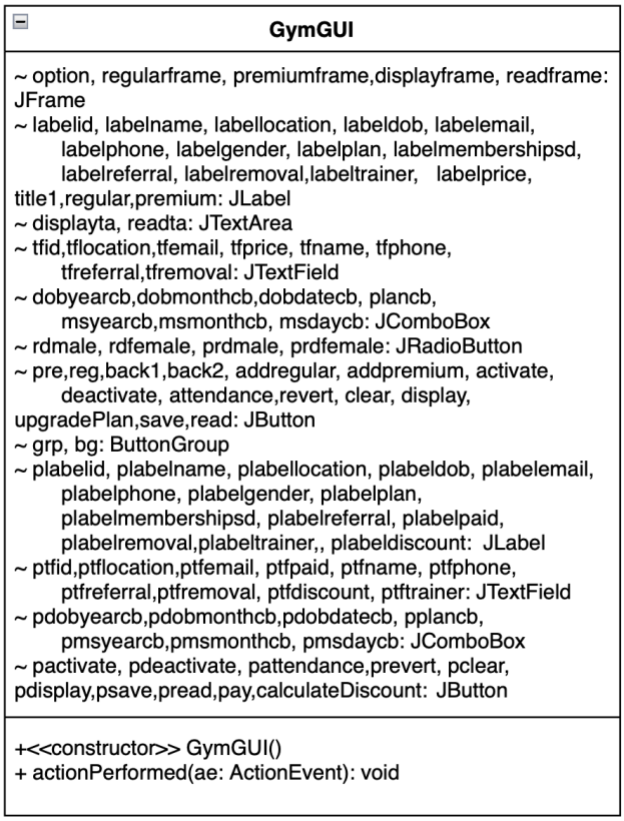
Class Diagram

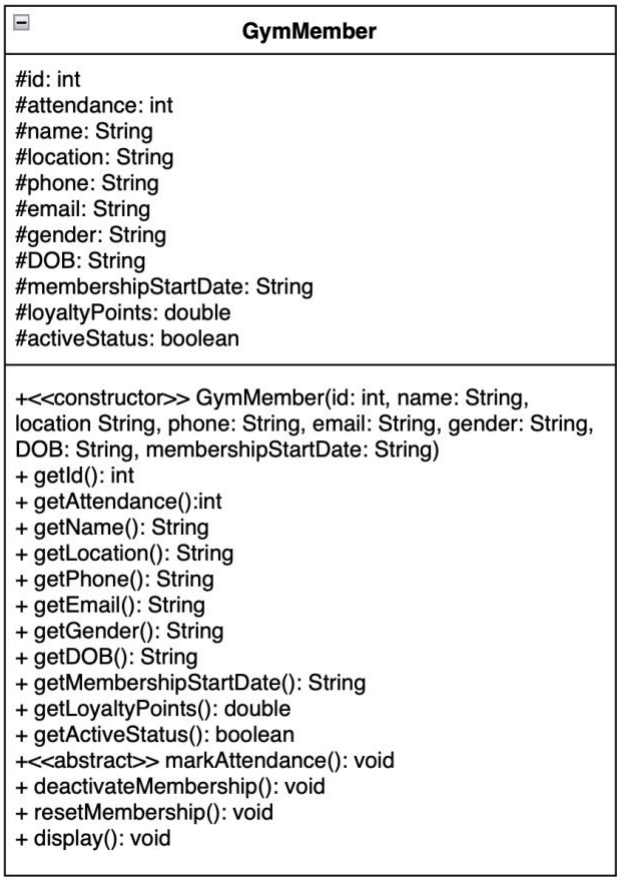
Class diagram is a structured diagram that describes the class structure which includes its attributes and methods. Class diagram is the visual representation of a class in object-oriented programming. It contains class name, attributes with datatype and access modifier and methods with return type and access modifiers.

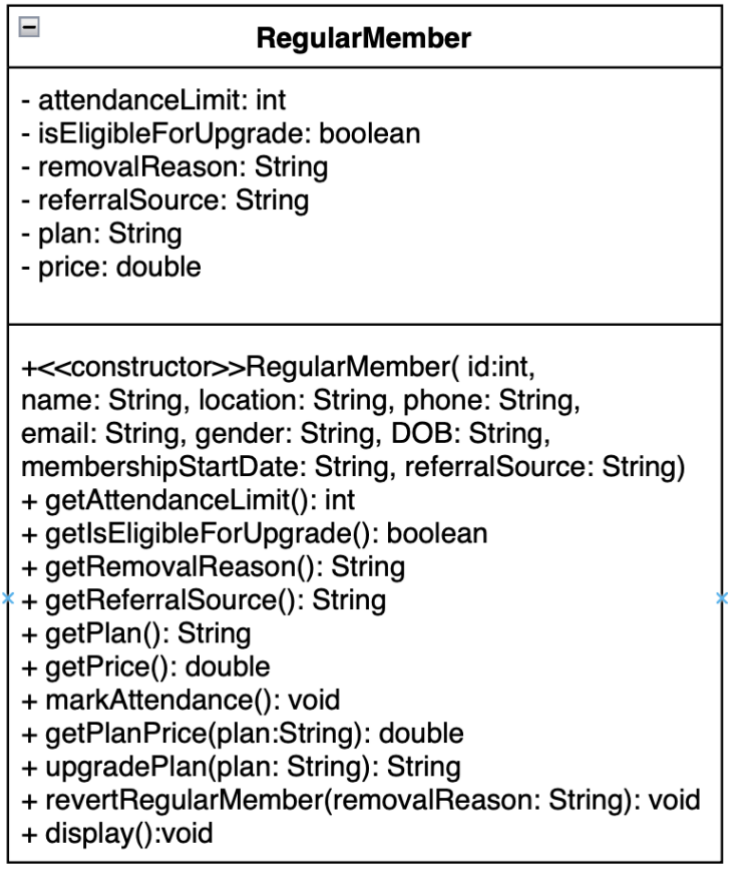
Attribute: access modifier variable_name : datatype

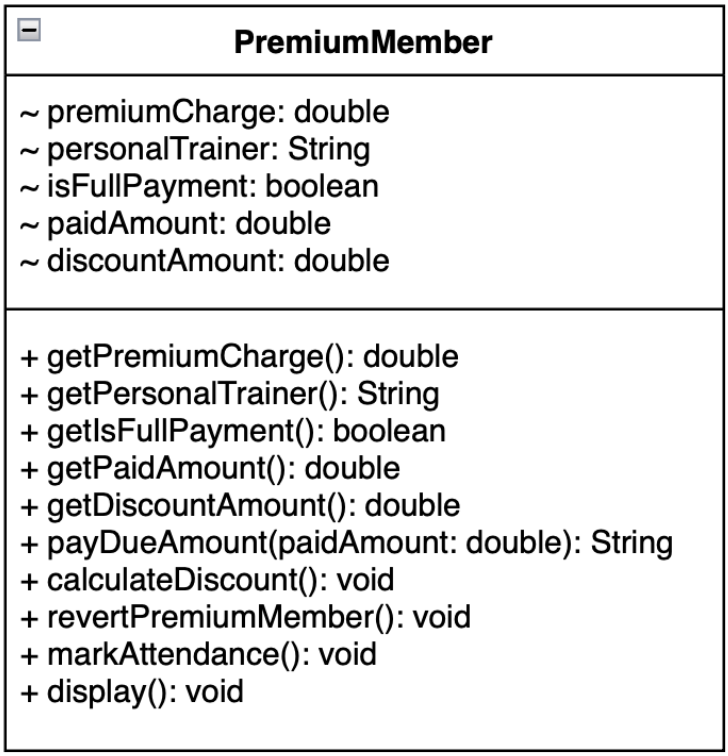
Methods: access modifier method_name : return_type

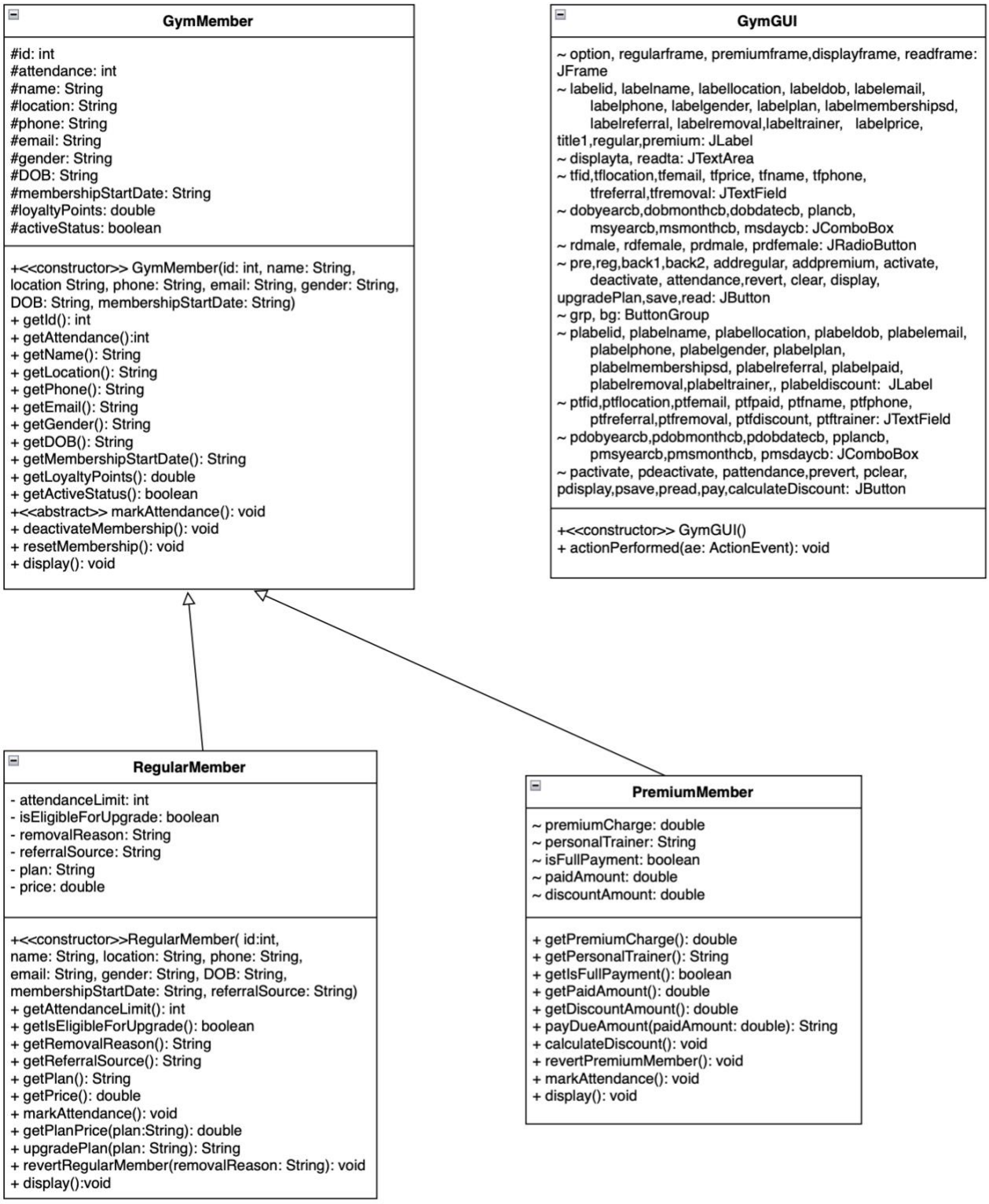
symbol	meaning
+	Public access modifier
-	Private access modifier
#	Protected access modifier
<<constructor>>	Constructor
<<abstract>>	Abstract


GymGUI
~ option, regularframe, premiumframe,displayframe, readframe: JFrame
~ labelid, labelname, labellocation, labeldob, labelemail, labelphone, labelgender, labelplan, labelmembershipsd, labelreferral, labelremoval,labeltrainer, labelprice, title1,regular,premium: JLabel
~ displayta, readta: JTextArea
~ tfid,tflocation,tfemail, tfprice, tfname, tfphone, tfreferral,tfremoval: JTextField
~ dobyearcb,dobmonthcb,dobdatecb, plancb, msyearcb,msmonthcb, msdaycb: JComboBox
~ rdmale, rdfemale, prdmale, prdfemale: JRadioButton
~ pre,reg,back1,back2, addregular, addpremium, activate, deactivate, attendance,revert, clear, display, upgradePlan,save,read: JButton
~ grp, bg: ButtonGroup
~ plabelid, plabelname, plabellocation, plabeldob, plabelemail, plabelphone, plabelgender, plabelplan, plabelmembershipsd, plabelreferral, plabelpaid, plabelremoval,plabeltrainer,, plabeldiscount: JLabel
~ ptfid,ptflocation,ptfemail, ptfpaid, ptfname, ptfphone, ptfreferral,ptfremoval, ptfdiscount, ptftrainer: JTextField
~ pdobyearcb,pdobmonthcb,pdobdatecb, pplancb, pmsyearcb,pmsmonthcb, pmsdaycb: JComboBox
~ pactivate, pdeactivate, pattendance,prevert, pclear, pdisplay,psave,pread,pay,calculateDiscount: JButton
+<<constructor>> GymGUI()
+ actionPerformed(ae: ActionEvent): void


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


RegularMember
- attendanceLimit: int
- isEligibleForUpgrade: boolean
- removalReason: String
- referralSource: String
- plan: String
- price: double
+<<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


PremiumMember
~ premiumCharge: double
~ personalTrainer: String
~ isFullPayment: boolean
~ paidAmount: double
~ discountAmount: double
+ getPremiumCharge(): double
+ getPersonalTrainer(): String
+ getIsFullPayment(): boolean
+ getPaidAmount(): double
+ getDiscountAmount(): double
+ payDueAmount(paidAmount: double): String
+ calculateDiscount(): void
+ revertPremiumMember(): void
+ markAttendance(): void
+ display(): void


GymMember
#id: int
#attendance: int
#name: String
#location: String
#phone: String
#email: String
#gender: String
#DOB: String
#membershipStartDate: String
#loyaltyPoints: double
#activeStatus: boolean
+<<constructor>> GymMember(id: int, name: String, location String, phone: String, email: String, gender: String, DOB: String, membershipStartDate: String)
+ getId(): int
+ getAttendance():int
+ getName(): String
+ getLocation(): String
+ getPhone(): String
+ getEmail(): String
+ getGender(): String
+ getDOB(): String
+ getMembershipStartDate(): String
+ getLoyaltyPoints(): double
+ getActiveStatus(): boolean
+<<abstract>> markAttendance(): void
+ deactivateMembership(): void
+ resetMembership(): void
+ display(): void
GymGUI
~ option, regularframe, premiumframe,displayframe, readframe: JFrame
~ labelid, labelname, labellocation, labeldob, labelemail, labelphone, labelgender, labelplan, labelmembershipsd, labelreferral, labelremoval,labeltrainer, labelprice, title1,regular,premium: JLabel
~ displayta, readta: JTextArea
~ tfid,tflocation,tfemail, tfprice, tfname, tfphone, tfreferral,tfremoval: JTextField
~ dobyearcb,dobmonthcb,dobdatecb, plancb, msyearcb,msmonthcb, msdaycb: JComboBox
~ rdmale, rdfemale, prdmale, prdfemale: JRadioButton
~ pre,reg,back1,back2, addregular, addpremium, activate, deactivate, attendance,revert, clear, display, upgradePlan,save,read: JButton
~ grp, bg: ButtonGroup
~ plabelid, plabelname, plabellocation, plabeldob, plabelemail, plabelphone, plabelgender, plabelplan, plabelmembershipsd, plabelreferral, plabelpaid, plabelremoval,plabeltrainer,, plabeldiscount: JLabel
~ ptfid,ptflocation,ptfemail, ptfpaid, ptfname, ptfphone, ptfreferral,ptfremoval, ptfdiscount, ptftrainer: JTextField
~ pdobyearcb,pdobmonthcb,pdobdatecb, pplancb, pmsyearcb,pmsmonthcb, pmsdaycb: JComboBox
~ pactivate, pdeactivate, pattendance,prevert, pclear, pdisplay,psave,pread,pay,calculateDiscount: JButton
+<<constructor>> GymGUI()
+ actionPerformed(ae: ActionEvent): void
RegularMember
- attendanceLimit: int
- isEligibleForUpgrade: boolean
- removalReason: String
- referralSource: String
- plan: String
- price: double
+<<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
PremiumMember
~ premiumCharge: double
~ personalTrainer: String
~ isFullPayment: boolean
~ paidAmount: double
~ discountAmount: double
+ getPremiumCharge(): double
+ getPersonalTrainer(): String
+ getIsFullPayment(): boolean
+ getPaidAmount(): double
+ getDiscountAmount(): double
+ payDueAmount(paidAmount: double): String
+ calculateDiscount(): void
+ revertPremiumMember(): void
+ markAttendance(): void
+ display(): void

Pseudocode

Pseudocode is a plain language translation of algorithm. It contains the programs core logic without any actual code. It is written before an actual code. It does. Not use technical terms and uses simple words like create, set, generate, assign and many more. It uses ident to show the hierarchy. It avoids actual syntax rules of programming. It shows how the things are done but not the whole process of it.

GymMember

CLASS GymMember

DECLARE INSTANCE VARIABLE id AS protected Integer

DECLARE INSTANCE VARIABLE attendance AS protected Integer

DECLARE INSTANCE VARIABLE name AS protected String

DECLARE INSTANCE VARIABLE location AS protected String

DECLARE INSTANCE VARIABLE phone AS protected String

DECLARE INSTANCE VARIABLE email AS protected String

DECLARE INSTANCE VARIABLE gender AS protected String

DECLARE INSTANCE VARIABLE DOB AS protected String

DECLARE INSTANCE VARIABLE membershipStartDate AS protected String

DECLARE INSTANCE VARIABLE loyaltyPoints AS protected Double

DECLARE INSTANCE VARIABLE activeStatus AS protected Boolean

CREATE CONSTRUCTOR GymMember (id AS Integer , name AS String, location AS String, phone AS String, email AS String, gender AS String, DOB AS String, membershipStartDate AS String)

SET INSTANCE VARIABLE id AS id;

SET INSTANCE VARIABLE name AS name;

```
    SET INSTANCE VARIABLE location AS location;
    SET INSTANCE VARIABLE phone AS phone;
    SET INSTANCE VARIABLE email AS email;
    SET INSTANCE VARIABLE gender AS gender;
    SET INSTANCE VARIABLE DOB AS DOB;
    SET INSTANCE VARIABLE membershipStartDate AS membershipStartDate;
    SET INSTANCE VARIABLE attendance AS 0;
    SET INSTANCE VARIABLE loyaltyPoints AS 0;
    SET INSTANCE VARIABLE activeStatus AS false;
END CONSTRUCTOR

DEFINE abstract METHOD markAttendance();
END METHOD

DEFINE METHOD activateMembership()
    SET activeStatus AS true;
 END METHOD

DEFINE METHOD deactivateMembership()
    IF(activeStatus)
       SET activeStatus AS false;
    ELSE
        OUTPUT("It seems like active status is already inactive");
END METHOD
```

```
    DEFINE METHOD resetMembership()
        SET activeStatus AS false;
        SET attendance AS 0;
        SET loyaltyPoints AS 0;
END METHOD


    DEFINE METHOD display()
        OUTPUT("Id: "+id);
        OUTPUT("Name: "+name);
        OUTPUT("Location: "+location);
        OUTPUT("Phone: "+phone);
        OUTPUT("Email: "+email);
        OUTPUT("Gender: "+gender);
        OUTPUT("DOB: "+DOB);
        OUTPUT("Membership Start Date: "+membershipStartDate);
        OUTPUT("Attendance: "+attendance);
        OUTPUT("Loyalty points: "+loyaltyPoints);
        OUTPUT("Active Status: "+(activeStatus?"active":"inactive"));
    END METHOD
END GymMember
```

Regular Member

CLASS RegularMember **EXTENDS** GymMember

{

DECLARE INSTANCE VARIABLE attendanceLimit AS private Integter

DECLARE INSTANCE VARIABLE isEligibleForUpgrade AS private boolean

DECLARE INSTANCE VARIABLE removalReason AS private String

DECLARE INSTANCE VARIABLE referralSource AS private String

DECLARE INSTANCE VARIABLE plan AS private String

DECLARE INSTANCE VARIABLE price AS private double

CREATE CONSTRUCTOR RegularMember (id AS Integer , name AS String, location AS String, phone AS String, email AS String, gender AS String, DOB AS String, membershipStartDate AS String, referralSource AS String)

CALL GymMember CONSTRUCTOR WITH id, name, location, phone, email, gender, DOB, membershipStartDate

SET INSTANCE VARIABLE isEligibleForUpgrade AS false

SET INSTANCE VARIABLE attendanceLimit AS 30

SET INSTANCE VARIABLE plan AS "basic"

SET INSTANCE VARIABLE price AS 6500

SET INSTANCE VARIABLE removalReason AS ""

SET INSTANCE VARIABLE referralSource AS referralSource

END CONSTRUCTOR

```
@Override
DEFINE METHOD markAttendance()
    SET attendance++
    SET loyaltyPoints +=  5
    IF (getAttendance() >= attendanceLimit)
      SET  isEligibleForUpgrade AS true
    END IF
END METHOD

DEFINE METHOD getPlanPrice(String plan) AS double
    DECLARE double x
    SWITCH(plan)
        case "basic":
            RETURN 6500
        case "standard":
            RETURN 12500
        case "deluxe":
            RETURN 18500
        default:
            RETURN -1
    END SWITCH
END METHOD

DEFINE METHOD upgradePlan(String plan) AS String
    IF (isEligibleForUpgrade != true)
```

```
        RETURN "the member is not eligible for upgrade"
    ELSE
        IF(INSTANCE VARIABLE plan == plan
            RETURN "the member has chosen the same plan"
        ELSE
            IF(getPlanPrice(plan) == -1)
                RETURN "the member has entered invalid plan"
            ELSE
                SET INSTANCE VARIABLE plan AS plan
                SET INSTANCE VARIABLE price AS getPlanPrice(plan)
                RETURN ("plan upgraded to "+plan+" price = "+price)
            END IF
        END IF
    END IF
END METHOD

DEFINE METHOD revertRegularMember(String removalReason)
    CALL PARENT CLASS METHOD resetMembership()
    SET INSTANCE VARIABLE isEligibleForUpgrade AS false
    SET INSTANCE VARIABLE plan AS "basic"
    SET INSTANCE VARIABLE price AS 6500
    SET INSTANCE VARIABLE removalReason AS removalReason
END METHOD

@Override
```

```
    DEFINE METHOD void display()
        CALL PARENT CLASS METHOD display()
        OUTPUT ("plan: "+plan)
        OUTPUT ("price: "+price)
        IF (removalReason != "")
            OUTPUT ("removal reason: "+removalReason)
        END IF
    END METHOD
END CLASS
```

Premium member

CLASS PremiumMember EXTENDS GymMember

{

DECLARE INSTANCE VARIABLE premiumCharge AS 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 CONSTRUCTOR PremiumMember (id AS Integer , name AS String, location AS String, phone AS String, email AS String, gender AS String, DOB AS String, membershipStartDate AS String, personalTrainer AS String)

CALL PARENT CLASS CONSTRUCTOR (id, name, location, phone, email, gender, DOB, membershipStartDate)

SET INSTANCE VARIABLE premiumCharge=50000

SET INSTANCE VARIABLE paidAmount=0

SET INSTANCE VARIABLE isFullPayment=false

SET INSTANCE VARIABLE discountAmount=0

SET INSTANCE VARIABLE personalTrainer=personalTrainer

END CONSTRUCTOR

DECLARE METHOD payDueAmount (double paidAmount) AS String

IF (INSTANCE VARIABLE paidAmount==premiumCharge)

INSTANCE VARIABLE isFullPayment=true

END IF

IF (INSTANCE VARIABLE isFullPayment)

OUTPUT ("full payment has been done")

```
        RETURN "the member has already paid full amount"
    ELSE
        SET INSTANCE VARIABLE paidAmount += paidAmount
        IF (INSTANCE VARIABLE paidAmount > premiumCharge)
    SET INSTANCE VARIABLE paidAmount -= paidAmount
            OUTPUT ("the member has paid more than premiumCharge.")
            RETURN "please pay the correct amount"
        ELSE
        OUTPUT ("payment was successful")
        RETURN "remaining Amount="+(premiumCharge-INSTANCE VARIABLE paidAmount)
        END IF
    END IF
END METHOD

DECLARE METHOD calculateDiscount()
    IF (INSTANCE VARIABLE isFullPayment)
       SET INSTANCE VARIABLE discountAmount = (10/100)*premiumCharge
        OUTPUT ("discount is calculated. discount amount= "
        +INSTANCE VARIABLE discountAmount)
    ELSE
        OUTPUT ("discount is not available")
    END IF
END METHOD

DECLARE METHOD revertPremiumMember()
```

```
        CALL PARENT CLASS METHOD resetMembership()
        SET INSTANCE VARIABLE personalTrainer AS EMPTY
        SET INSTANCE VARIABLE isFullPayment AS false
        SET INSTANCE VARIABLE paidAmount AS 0
        SET INSTANCE VARIABLE discountAmount AS 0
    END METHOD

    @Override
    DECLARE METHOD markAttendance()
        SET attendance++
        SET loyaltyPoints+=10
    END METHOD
    @Override
    DECLARE METHOD  display()
        CALL PARENT CLASS METHOD display()
        OUTPUT ("personal trainer= "+personalTrainer)
        OUTPUT ("paid amount= "+paidAmount)
        OUTPUT ("is the amount fully paid= "+(isFullPayment?"yes":"no"))
        IF (INSTANCE VARIABLE isFullPayment)
            OUTPUT ("discount amount= "+discountAmount)
        ELSE IF (INSTANCE VARIABLE paidAmount < premiumCharge)
            OUTPUT ("remaining Amount="+(premiumCharge-INSTANCE VARIABLE
paidAmount))
        END IF
    END METHOD
END CLASS
```

GymGUI

IMPORT ArrayList FROM java.util

IMPORT JFrame FROM javax.swing

IMPORT JLabel FROM javax.swing

IMPORT JTextField FROM javax. swing

IMPORT JTextArea FROM javax. swing

IMPORT JComboBox FROM javax. swing

IMPORT JRadioButton FROM javax. swing

IMPORT JButton FROM javax. swing

IMPORT ButtonGroup FROM javax. swing

IMPORT ActionListener FROM java.awt.event

IMPORT ActionEvent FROM java.awt.event

IMPORT JOptionPane FROM javax. swing

IMPORT Font FROM java.awt

IMPORT Color FROM java.awt

IMPORT File FROM java.io

IMPORT FileWriter FROM java.io

IMPORT IOException FROM java.io

IMPORT FileReader FROM java.io

IMPORT FileNotFoundException FROM java.io

CLASS GymGUI IMPLEMENTS ActionListener

INITIALIZE list to store GymMember objects

DEFINE METHOD main

    CALL CONSTRUC**TO**R GymGUI

END METHOD

DECLARE option, regularframe, premiumframe, readframe **AS** JFrame

DECLARE displayta, readta **AS** JTextArea

DECLARE labelid, labelname, labellocation, labeldob, labelemail, labelphone, labelgender, labelplan, labelmembershipsd, labelreferral, labelremoval,labeltrainer, labelprice, title1,regular,premium **AS** JLabel

DECLARE plabelid, plabelname, plabellocation, plabeldob, plabelemail, plabelphone, plabelgender, plabelplan, plabelmembershipsd, plabelreferral, plabelpaid, plabelremoval,plabeltrainer, plabeldiscount **AS** JLabel

DECLARE tfid,tflocation,tfemail, tfprice, tfname, tfphone, tfreferral,tfremoval **AS** JTextField

DECLARE dobyearcb,dobmonthcb,dobdatecb, plancb, msyearcb,msmonthcb, msdaycb, pdobyearcb,pdobmonthcb,pdobdatecb, pplancb, pmsyearcb,pmsmonthcb, pmsdaycb **AS** JComboBox

DECLARE rdmale, rdfemale, prdmale, prdfemale **AS** JRadioButton

DECLARE grp, bg **AS** ButtonGroup

DECLARE **ADD**regular, activate, deactivate, attendance,revert, clear, display, upgradePlan, save, read, pre, reg, back1, back2, **ADD**premium, pactivate, pdeactivate, pattendance, prevert, pclear, pdisplay,calculateDisco **AS** JButton

DECLARE ptfid, ptflocation, ptfemail, ptfpaid, ptfname, ptfphone, ptfreferral, ptfremoval, ptfdiscount, ptftrainer **AS** JTextField

CONSTRUC**TO**R GymGUI

CREATE INSTANCE VARIABLE option **AS** JFrame

SET INSTANCE VARIABLE title1 **AS** “apply for:”

SET INSTANCE VARIABLE reg **AS** “regular membership”

SET INSTANCE VARIABLE pre **AS** “premium membership”

SET BOUNDS title1 **TO** (110,5,120,30)

SET BOUNDS reg **TO** (50,50,200,60)

SET BOUNDS pre TP (50,120,200,60)

ADD title1, pre, reg **TO** option frame

SET layout null

SET visibility true

SET size (300,250)

CREATE INSTANCE VARIABLE regularframe **AS** JFrame

SET INSTANCE VARIABLE regular **AS** “regular membership application”

SET INSTANCE VARIABLE labelid, labelname, labellocation, labeldob, labelemail, labelphone, labelgender, labelplan, labelmembershipsd, labelreferral, labelremoval, labelprice **AS** “ID”, “name”, “location”, “DOB”, “email”, “phone”, “gender”, “plan”, “membership start date”, “referral source”, “removal reason”, “plan price” respectively

INITIALIZE all **DECLARE**d JTextField and ComboBox

//**SET**ting radio button

SET INSTANCE VARIABLE rdmale **AS** “male”

SET INSTANCE VARIABLE rdfemale **AS** “female”

//**SET**ting buttons

SET INSTANCE VARIABLE addregular **AS** ("**ADD** Regular Member")

SET INSTANCE VARIABLE addpremium **AS** ("**ADD** Premium Member")

SET INSTANCE VARIABLE activate **AS** ("Activate Membership")

SET INSTANCE VARIABLE deactivate **AS** ("Deactivate Membership")

SET INSTANCE VARIABLE attendance **AS** ("Mark Attendance")

SET INSTANCE VARIABLE revert **AS** ("Revert Membership")

SET INSTANCE VARIABLE clear **AS** ("Clear")

SET INSTANCE VARIABLE display **AS** ("Display")

SET INSTANCE VARIABLE back1 **AS** ("Back")

```
        SET INSTANCE VARIABLE upgradePlan AS ("Upgrade Plan")
        SET INSTANCE VARIABLE save AS ("Save")
        SET INSTANCE VARIABLE read AS ("Read from file")

    SET BOUNDS regular TO (465,15,400,50)
    SET BOUNDS labelid TO (50,110,100,25)
    SET BOUNDS labellocation TO (50,200,100,25)
    SET BOUNDS labeldob TO (50,290,100,25)
    SET BOUNDS labelgender TO (50,380,100,25)
    SET BOUNDS labelmembershipsd TO (50,470,150,25)
    SET BOUNDS labelprice TO (50,560,100,25)
    SET BOUNDS labelname TO (640,110,100,25)
    SET BOUNDS labelemail TO (640,200,100,25)
    SET BOUNDS labelphone TO (640,290,100,25)
    SET BOUNDS labelplan TO (640,380,100,25)
    SET BOUNDS labelreferral TO (640,470,100,25)
    SET BOUNDS labelremoval TO (640,560,150,25)
SET BOUNDS addregular TO (50,680,220,55)
    SET BOUNDS attendance TO (350,680,220,55)
    SET BOUNDS activate TO (670,680,220,55)
    SET BOUNDS deactivate TO (970,680,220,55)
    SET BOUNDS revert TO (350,760,220,55)
    SET BOUNDS clear TO (670,760,220,55)
    SET BOUNDS display TO (970,760,220,55)
    SET BOUNDS upgradePlan TO (50,760,220,55)
```

SET BOUNDS back1 **TO** (50,840,220,55)

SET BOUNDS save **TO** (350,840,220,55)

SET BOUNDS read **TO** (670,840,220,55)

SET BOUNDS tfid **TO** (48,140,550,40)

SET BOUNDS tflocation **TO** (48,230,550,40)

SET BOUNDS tfemail **TO** (638,230,550,40)

SET BOUNDS tfprice **TO** (48,590,550,40)

SET BOUNDS tfname **TO** (638,140,550,40)

SET BOUNDS tfphone **TO** (638,320,550,40)

SET BOUNDS tfreferral **TO** (638,500,550,40)

SET BOUNDS tfremoval **TO** (638,590,550,40)

SET BOUNDS rdmale **TO** (48,410,100,30)

SET BOUNDS rdfemale **TO** (150,410,100,30)

SET BOUNDS plancb **TO** (638,400,550,70)

SET BOUNDS dobyearcb **TO** (48,320,100,50)

SET BOUNDS dobmonthcb **TO** (148,320,100,50)

SET BOUNDS dobdatecb **TO** (248,320,100,50)

SET BOUNDS msyearcb **TO** (48,500,100,50)

SET BOUNDS msmonthcb **TO** (148,500,100,50)

SET BOUNDS msdaycb **TO** (248,500,100,50)

ADD SET initialized JLabel, JTextField, JButton, JRadioButton, JComboBox **TO** regularframe

SET layout null

SET visibility false

SET size (1250,950)

SET INSTANCE VARIABLE premiumframe **AS** JFrame("")

SET INSTANCE VARIABLE premium **AS** "Premium Membership Application"

SET INSTANCE VARIABLE plabelid **AS** ("ID")

SET INSTANCE VARIABLE plabelname **AS** ("Name")

SET INSTANCE VARIABLE plabellocation **AS** ("Location")

SET INSTANCE VARIABLE plabeldob **AS** ("DOB")

SET INSTANCE VARIABLE plabelemail **AS** ("Email")

SET INSTANCE VARIABLE plabelphone **AS** ("Phone")

SET INSTANCE VARIABLE plabelgender **AS** ("Gender")

SET INSTANCE VARIABLE plabelplan **AS** ("Plan")

SET INSTANCE VARIABLE plabelmembershipsd **AS** ("Membership Start Date")

SET INSTANCE VARIABLE plabelreferral **AS** ("Referral Source")

SET INSTANCE VARIABLE plabelpaid **AS** ("Paid Amount")

SET INSTANCE VARIABLE plabelremoval **AS** ("Removal Re**AS**on")

SET INSTANCE VARIABLE plabeltrainer **AS** ("Trainer's Name")

SET INSTANCE VARIABLE plabeldiscount **AS** ("Discount Amount")

INITIALIZE all declared JTextField and ComboBox

//**SET**ting radio button

SET INSTANCE VARIABLE prdmale **AS** “male”

SET INSTANCE VARIABLE prdfemale **AS** “female”

SET INSTANCE VARIABLE addpremium **AS** ("add Premium Member")

SET INSTANCE VARIABLE pactivate **AS** ("Activate Membership")

SET INSTANCE VARIABLE pdeactivate **AS** ("Deactivate Membership")

SET INSTANCE VARIABLE pattendance **AS** ("Mark Attendance")

SET INSTANCE VARIABLE prevert **AS** ("Revert Membership")

SET INSTANCE VARIABLE pclear **AS** ("Clear")

SET INSTANCE VARIABLE pdisplay **AS** ("Display")

SET INSTANCE VARIABLE back2 **AS** ("Back")

SET INSTANCE VARIABLE psave **AS** ("Save")

SET INSTANCE VARIABLE pread **AS** ("Read from file")

SET INSTANCE VARIABLE calculateDiscount **AS** ("Discount")

SET INSTANCE VARIABLE pay **AS** ("Pay")

SET BOUNDS premium **TO** (465,15,500,50)

SET BOUNDS plabelid **TO** (50,110,100,25)

SET BOUNDS plabellocation **TO** (50,200,100,25)

SET BOUNDS plabeldob **TO** (50,290,100,25)

SET BOUNDS plabelgender **TO** (50,380,100,25)

SET BOUNDS plabelmembershipsd **TO** (50,470,150,25)

SET BOUNDS plabeldiscount **TO** (50,560,150,25)

SET BOUNDS plabelpaid **TO** (50,650,100,25)

SET BOUNDS plabelname **TO** (640,110,100,25)

SET BOUNDS plabelemail **TO** (640,200,100,25)

SET BOUNDS plabelphone **TO** (640,290,100,25)

SET BOUNDS labeltrainer **TO** (640,380,100,25)

SET BOUNDS plabelreferral **TO** (640,470,100,25)

SET BOUNDS plabelremoval **TO** (640,560,150,25)

SET BOUNDS pay **TO** (638,680,220,45)

SET BOUNDS addpremium **TO** (50,760,220,55)

SET BOUNDS pattendance **TO** (350,760,220,55)

SET BOUNDS pactivate **TO** (670,760,220,55)

SET BOUNDS pdeactivate **TO** (970,760,220,55)

SET BOUNDS prevert **TO** (350,835,220,55)

SET BOUNDS pclear **TO** (670,835,220,55)

SET BOUNDS pdisplay **TO** (970,835,220,55)

SET BOUNDS calculateDiscount **TO** (50,835,220,55)

SET BOUNDS back2 **TO** (50,910,220,55)

SET BOUNDS psave **TO** (350,910,220,55)

SET BOUNDS pread **TO** (670,910,220,55)

SET BOUNDS ptfid **TO** (48,140,550,40)

SET BOUNDS ptflocation **TO** (48,230,550,40)

SET BOUNDS ptfemail **TO** (638,230,550,40)

SET BOUNDS ptfdiscount **TO** (48,590,550,40)

SET BOUNDS ptfpaid **TO** (48,680,550,40)

SET BOUNDS ptfname **TO** (638,140,550,40)

SET BOUNDS ptfphone **TO** (638,320,550,40)

SET BOUNDS ptfreferral **TO** (638,500,550,40)

SET BOUNDS ptfremoval **TO** (638,590,550,40)

SET BOUNDS ptftrainer **TO** (638,410,550,40)

SET BOUNDS prdmale **TO** (48,410,100,30)

SET BOUNDS prdfemale **TO** (150,410,100,30)

SET BOUNDS pdobyearcb **TO** (48,320,100,50)

SET BOUNDS pdobmonthcb **TO** (148,320,100,50)

SET BOUNDS pdobdatecb **TO** (248,320,100,50)

SET BOUNDS pmsyearcb **TO** (48,500,100,50)

SET BOUNDS pmsmonthcb **TO** (148,500,100,50)

SET BOUNDS pmsdaycb **TO** (248,500,100,50)

ADD SET initialized JLabel, JTextField, JButton, JRadioButton, JComboBox **TO** premiumframe

SET layout null

SET visibility false

SET size (1250,1000)

SET INSTANCE VARIABLE displayframe **AS** JFrame

SET INSTANCE VARIABLE displayta **AS** JTextArea

SET BOUNDS displayta **AS** (5,5,890,890)

ADD displayta **TO** displayframe

SET layout null

SET visibility false

SET size (900,900)

SET INSTANCE VARIABLE readframe **AS** JFrame

SET INSTANCE VARIABLE readta **AS** JTextArea

SET BOUNDS readta **AS** (5,5,890,890)

ADD readta **TO** readframe

SET layout null

SET visibility false

SET size (1350,300)

DEFINE METHOD actionPerformed (ae **AS** ActionEvent)

IF(event source is reg)

SET VISIBLE of option **AS** (false)

```
    SET VISIBLE of regularframe AS (true)
    SET VISIBLE of premiumframe AS (false)
  END IF

ELSE IF(event source is plancb)
    String s= (String)plancb.getSelectedItem()
    IF(s=="Basic")
       tfprice.setText("6500")
    END IF
    ELSE IF(s=="Standard")
       tfprice.setText("12500")
    END ELSE IF
    ELSE
       tfprice.setText("18500")
    END IF

 END ELSE IF
  ELSE IF event source is pre
    SET VISIBLE of option AS   (false)
    SET VISIBLE of premiumframe AS   (true)
    SET VISIBLE of regularframe AS   (false)
  END ELSE IF
  ELSE IF(ae.getSource()==back1)
    SET VISIBLE of option AS   (true)
    SET VISIBLE of regularframe AS   (false)
```

END ELSE IF

ELSE IF(event source is back2)

SET VISIBLE of option **AS** (true)

SET VISIBLE of premiumframe **AS** (false)

END ELSE IF

ELSE IF (event source is addregular)

IF (tfid.getText().isEmpty()|| tflocation.getText().isEmpty()|| tfemail.getText().isEmpty()|| tfname.getText().isEmpty()||tfphone.getText().isEmpty()||tfreferral.getText().isEmpty())

MESSAGE (regularframe,"please fill in the application");

ELSE

TRY{

SET int regId = Integer.parseInt(tfid.getText());

SET String regLocation = tflocation.getText();

SET String regEmail = tfemail.getText();

SET String regName = tfname.getText();

SET String regPhone = tfphone.getText();

SET String regReferral = tfreferral.getText();

SET String regGender="";

SET String regPlan= (String) plancb.getSelectedItem();

IF (rdmale.isSelected())

SET regGender="male";

ELSE IF (rdfemale.isSelected())

SET regGender="female";

SET String year= (String) dobyearcb.getSelectedItem();

SET String month= (String) dobmonthcb.getSelectedItem(); //combo box creates object and (String) converts it into string

SET String date= (String) dobdatecb.getSelectedItem();

SET String regDob= date+"-"+month+"-"+year;

SET String myear= (String) msyearcb.getSelectedItem();

SET String mmonth= (String) msmonthcb.getSelectedItem(); //combo box creates object and (String) converts it into string

SET String mdate= (String) msdaycb.getSelectedItem();

SET String regMembershipsd= mdate+"-"+mmonth+"-"+myear;

IF (al1.isEmpty())

CREATE RegularMember regobj = new RegularMember(regId,regName,regLocation,regPhone,regEmail,

regGender,regDob, regMembershipsd, regReferral);

al1.add(regobj);

MESSAGE (regularframe,"Regular Member added succesfully");

ELSE

SET boolean idexistence=false;

FOR (GymMember mem : al1)

IF (mem.getId()==regId) //**FOR** each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa

SET idexistence=true;

break;//loop bata baira niskinxa

END IF

END FOR

IF (idexistence)

MESSAGE (regularframe,"Member with this id already exists. please choose another ID.","Duplication",JOptionPane.ERROR_MESSAGE);

```
            ELSE
                RegularMember regobj = new
RegularMember(regId,regName,regLocation,regPhone,regEmail,
                regGender,regDob, regMembershipsd, regReferral);
                al1.add(regobj);
                MESSAGE (regularframe,"Regular Member added succesfully");
            END  IF
          END  IF
        END  IF
        CATCH(NumberFormatException ex)
          MESSAGE (regularframe,"ID can only have number
values!","Error",JOptionPane.WARNING_MESSAGE);
      END  IF
      END  ELSE  IF
ELSE  IF (event source is attendance)
        IF (tfid.getText().isEmpty())
          SHOW MESSAGE (please fill in the id")
        ELSE
          TRY{
          SET int regId AS Integer.parseInt(tfid.getText())
          SET boolean idexistence AS false
          SET GymMember memberexistence AS null
            FOR (GymMember member : al1)
              IF (member.getId()==regId)
                SET idexistence AS true
                SET memberexistence  AS member
```

```
            BREAK
          END IF
        END FOR
      IF (idexistence)
        IF (memberexistence.getActiveStatus()){
          CALL memberexistence.markAttendance()
          SHOW MESSAGE "Attendance marked"
        ELSE
          SHOW MESSAGE “member is not activate”
      ELSE
        SHOW MESSAGE "Member with this ID does not exist
  CATCH(NumberFormatException ex)
    SHOW MESSAGE "ID can only have number values!"
END IF
END IF

ELSE  IF (event source is activate)
  IF (tfid.getText().isEmpty())
    SHOW MESSAGE "please fill in the id")
  ELSE
    TRY{
    SET int regId=Integer.parseInt(tfid.getText())
    SET boolean idexistence=false
    SET GymMember memberexistence = null
      FOR (GymMember member : al1)
```

```
            IF (member.getId()==regId)
               SET idexistence=true
               SET memberexistence  = member
               BREAK
            END IF
         END FOR
      IF (idexistence)
         IF (memberexistence.getActiveStatus()){
            SHOW MESSAGE "Membership is already activated"
         ELSE
            CALL memberexistence.activateMembership()
            SHOW MESSAGE "Membership activated"
      ELSE
         SHOW MESSAGE  "Member with this ID does not exist.",
      END IF
   CATCH(NumberFormatException ex)
      SHOW MESSAGE "ID can only have number values!"
END IF
END IF

ELSE  IF (event source is deactivate)
   IF (tfid.getText().isEmpty())
      SHOW MESSAGE "please fill in the id")
   ELSE
      TRY{
```

```
        SET int regId AS Integer.parseInt(tfid.getText())
        SET boolean idexistence AS false
        SET GymMember memberexistence AS null
          FOR (GymMember member : al1)
            IF (member.getId()==regId) //FOR  each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa
              SET idexistence AS true
              SET memberexistence  AS member
              BREAK
            END IF
          END FOR
        IF (idexistence)
        {
          IF (memberexistence.getActiveStatus()==false){
            SHOW MESSAGE "Membership is already deactivated"
          ELSE
            memberexistence.deactivateMembership()
            SHOW MESSAGE "Membership deactivated"
        ELSE
          SHOW MESSAGE  "Member with this ID does not exist.",
      CATCH(NumberFormatException ex)
        SHOW MESSAGE "ID can only have number values!"
    END IF
    END IF

    ELSE  IF (event source isrevert)
```

```
IF (tfid.getText().isEmpty())
   SHOW MESSAGE "please fill in the id")
ELSE  IF (tfremoval.getText().isEmpty())
   SHOW MESSAGE "please fill in the removal reason")
ELSE
   TRY
   SET int regId AS Integer.parseInt(tfid.getText())
   SET String regRemoval AS tfremoval.getText()
   SET boolean idexistence AS false
   SET RegularMember memberexistence AS null
   FOR (GymMember member : al1)
      IF  (member instanceof RegularMember)
         RegularMember regMember = (RegularMember) member
      IF (regMember.getId()==regId)
         SET idexistence AS true
         memberexistence AS regMember
         BREAK
      END IF
      END IF
   END FOR
   IF (idexistence)
         CALL memberexistence.revertRegularMember(regRemoval)
         SHOW MESSAGE "Membership Reset","Reset"
   ELSE
      SHOW MESSAGE  "Member with this ID does not exist."
```

END IF

CATCH(NumberFormatException ex)

SHOW MESSAGE "ID can only have number values!"

END IF

END IF

ELSE IF (the event source is display)
THEN

MAKE displayframe visible

CLEAR the text area displayta

IF the member list al1 is empty

APPEND "NO MEMBER REGISTERED!!!" to displayta

ELSE

APPEND header "REGULAR MEMBERS" to displayta

INITIALIZE memberExistence as **false**

FOR EACH member IN al1

IF member IS INSTANCE OF RegularMember

SET memberExistence to **true**

BREAK loop

IF memberExistence is **true**

FOR EACH member IN al1

IF member IS INSTANCE OF RegularMember

CAST member to RegularMember

APPEND member details to displayta:

ID, Name, Location, Email, Phone

DOB, Gender, Plan, Price

Membership Start Date, Attendance

Loyalty Points, Active Status

IF removal reason is not empty

APPEND removal reason to displayta

APPEND “=”

END FOR

END IF

END IF

ELSEIF event source is clear OR pclear

SET tfid, tflocation, tfemail, tfprice, tfname, tfphone, tfreferral, tfremoval to empty

SET ptfid, ptflocation, ptfemail, ptfpaid, ptfname, ptfphone, ptfreferral, ptfremoval, ptfdiscount, ptftrainer to empty

SET dobmonthcb to "January"

SET dobyearcb to "2006"

SET dobdatecb to "1"

SET msdaycb to "1"

SET msmonthcb to "January"

SET msyearcb to "2006"

SET plancb to "basic"

END IF

IF event source is addpremium THEN

TRY

IF any of ptfid, ptflocation, ptfemail, ptfname, ptfphone, ptfreferral is empty THEN

Show error message: "please fill in the application"

ELSE

Read ptfid as integer AS preId

Read ptflocation AS preLocation

Read ptfemail AS preEmail

Read ptfname AS preName

Read ptfphone AS prePhone

Read ptftrainer AS preTrainer

SET preGender to ""

IF male radio button is selected THEN

    preGender AS "male"

ELSEIF female radio button is selected THEN

    preGender AS "female"

Read DOB from combo boxes: date, month, year

preDob as "date-month-year"

Read Membership Start Date from combo boxes: mdate, mmonth, myear

Combine into preMembershipsd as "mdate-mmonth-myear"

IF member list (al1) is empty THEN

    CREATE new PremiumMember object with all collected values

    Add object to al1

    Show success message

ELSE

    SET idexistence AS false

    FOR EACH member in al1

        IF member ID == preId THEN

            SET idexistence AS true

            BREAK loop

    IF idexistence is true THEN

        Show error: "ID already exists"

    ELSE

```
            CREATE new PremiumMember object with collected values
            Add object to al1
            Show success message
        CATCH NumberFormatException
            Show message "ID can only have number values!"
    END IF
    ELSE IF (event source ispattendance)
        IF(ptfid.getText().isEmpty())
            show message "please fill in the id"
        ELSE
            TRY
            SET int preId AS Integer.parseInt(ptfid.getText())
            SET boolean idexistence AS false
            SET GymMember memberexistence AS null
                FOR(GymMember member : al1)
                    IF(member.getId()==preId)
                        SET idexistence AS true
                        SET memberexistence  AS member
                        BREAK
                    END IF
                END FOR
            IF(idexistence)
                IF(memberexistence.getActiveStatus())
                    CALL memberexistence.markAttendance()
                    show message "Attendance marked
```

```
            ELSE

                show message "member is not activate","Attendance",JOptionPane.ERROR_MESSAGE)
            END IF
        END IF
        ELSE
            show message  "Member with this ID does not exist."
    END IF
    CATCH(NumberFormatException ex)
show message "ID can only have number values!"
END IF
END IF
ELSE IF(event source isprevert)
    IF(ptfid.getText().isEmpty())
        show message "please fill in the id")
    ELSE IF(ptfremoval.getText().isEmpty())
    show message "please fill in the removal reason")
    ELSE
        TRY
        SET int preId AS Integer.parseInt(ptfid.getText())
        SET String preRemoval AS ptfremoval.getText()
        SET boolean idexistence AS false
       SET PremiumMember memberexistence AS null
        FOR(GymMember member : al1)
            IF(member instanceof PremiumMember)
```

```
            PremiumMember preMember = (PremiumMember) member
            IF(preMember.getId()==preId)
               SET idexistence AS true
               SET memberexistence  AS preMember
               BREAK
            END IF
         END IF
      END FOR
      IF(idexistence)
            memberexistence.revertPremiumMember()
            show message "Membership Reset"
      ELSE
         show message  "Member with this ID does not exist."
   CATCH(NumberFormatException ex)
      show message "ID can only have number values!"
END IF
END IF
ELSE IF(event source ispactivate)
   IF(ptfid.getText().isEmpty())
      show message "please fill in the id")
   ELSE

      TRY
      SET int preId AS Integer.parseInt(ptfid.getText())
      SET boolean idexistence AS false
```

SET GymMember memberexistence **AS** null

FOR(GymMember member : al1)

IF(member.getId()==preId) //**FOR** each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa

SET idexistence **AS** true

SET memberexistence **AS** member

BREAK//loop bata baira niskinxa

}

}

IF(idexistence)

IF(memberexistence.getActiveStatus())

show message "Membership is already activated"

}

ELSE

CALL memberexistence.activateMembership()

show message "Membership activated"

END IF

ELSE

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

END IF

CATCH(NumberFormatException ex)

show message "ID can only have number values!"

END IF

END IF

```
ELSE IF (event source is pdisplay)
    SET visible displayframe AS true
    SET displayta AS “”

    IF member list (al1) is empty
        Append "NO MEMBER REGISTERED!!!" to displayframe
    ELSE
        Append "PREMIUM MEMBERS:" to displayframe

        SET memberExistence AS false

        FOR EACH member in al1
            IF member is instance of PremiumMember
                SET memberExistence AS true
                BREAK loop

        IF memberExistence is true
            FOR EACH member in al1
                IF member is instance of PremiumMember
                    Cast member to PremiumMember AS premium

                    Append premium.getId to displayframe
                    Append premium.getName to displayframe
                    Append premium.getLocation to displayframe
                    Append premium.getEmail to displayframe
```

```
            Append premium.getPhone to displayframe
            Append premium.getDOB to displayframe
            Append premium.getGender to displayframe
            Append premium.getMembershipStartDate to displayframe
            Append premium.getAttendance to displayframe
            Append premium.getLoyaltyPoints to displayframe
            Append premium.getPersonalTrainer to displayframe
            Append message: "Payment of Rs.<paidAmount> has been made"
            Append premium.getActiveStatus to displayframe

            IF premium.getIsFullPayment is true
               Append message: "Full payment has been done"
               Append premium.getDiscountAmount to displayframe
            ELSE
               Append message: "Full payment has not been done"
               Calculate remainingAmount = premium.getPremiumCharge -
premium.getPaidAmount
               Append message: "Payment of Rs.<remainingAmount> is left to be paid"

            Append separator line to displayframe
      ELSE
         Append "NO PREMIUM MEMBER REGISTERED!!!" to displayframe
END IF
ELSE IF(event source is pdeactivate)
         IF(ptfid.getText().isEmpty())
            show message,"please fill in the id")
```

```
        ELSE
            TRY
            SET int preId AS Integer.parseInt(ptfid.getText())
            SET boolean idexistence AS false
            SET GymMember memberexistence AS null
                for(GymMember member : al1)
                    IF(member.getId()==preId)
                        SET idexistence AS true
                        SET memberexistence  AS member
                        BREAK
            IF(idexistence)
                IF(memberexistence.getActiveStatus()==false){
                    show message,"Membership is already deactivated"
                ELSE
                    CALL memberexistence.deactivateMembership
                    show message,"Membership deactivated"
                END IF
            END IF
            ELSE
                show message, "Member with this ID does not exist."
            END IF
        CATCH(NumberFormatException ex){
            show message "ID can only have number values”
    END IF
IF (event source is calculateDiscount)
```

```
IF ID field is empty
    Show message: "please fill in the id"
ELSE
    TRY
        Convert ID to integer
        SET idExistence AS false
        SET memberExistence AS null

        FOR EACH member in member list (al1)
            IF member is PremiumMember AND member's ID matches input ID
                SET idExistence AS true
                SET memberExistence AS this member
                BREAK

        IF idExistence is true
            IF member's status is active
                IF full payment is done
                    Call calculateDiscount method
                    Show discount in text field
                    Show message with 10% discount confirmation
                ELSE
                    Show message "Member is not eligible for discount"
            ELSE
                Show message "Membership not activated"
        ELSE
```

```
            Show message "Member with this ID does not exist"
        CATCH NumberFormatException
            Show message "ID can only have number values"
IF event source is pay
    IF ID field is empty
        Show message "please fill in the id"
    ELSE IF paid amount field is empty
        Show message "please fill in the paid amount"
    ELSE
        TRY
            Convert ID to integer
            Convert paid amount to double
            Set idExistence AS false
            Set memberExistence AS null

            FOR EACH member in member list (al1)
                IF member is PremiumMember AND ID matches
                    Set idExistence AS true
                    Set memberExistence AS this member
                    BREAK

            IF idExistence is true
                IF membership is active
                    Call payDueAmount with paidAmount
                    Show returned message
```

```
            ELSE
                Show message: "Membership not activated"
            ELSE
                Show message "Member with this ID does not exist"
        CATCH NumberFormatException
            Show message "ID can only have number values"

IF event source is upgradePlan
    IF ID field is empty
        Show message: "please fill in the id"
    ELSE
        TRY
            Convert ID to integer
            Get selected plan from combo box
            Set idExistence AS false
            Set memberExistence AS null

            FOR EACH member in member list (al1)
                IF member is RegularMember AND ID matches
                    Set idExistence AS true
                    Set memberExistence AS this member
                    BREAK

            IF idExistence is true
                IF membership is active
```

```
                IF eligible for upgrade
                    Call upgradePlan with selected plan
                    Show returned message
                ELSE
                    Show message "Member is not eligible for upgrade. Attendance is low."
            ELSE
                Show message "Membership not activated"
        ELSE
            Show message "Member with this ID does not exist"
    CATCH NumberFormatException
        Show message "ID can only have number values"
IF event source is save OR psave
    TRY
        IF member list (al1) is empty
            Show message "NO MEMBER REGISTERED!"
        ELSE
            Open/create file named "MemberDetails.txt"
            Write header and table headers to file

            FOR EACH member in al1
                IF member is RegularMember
                    Format member details with "NONE" in premium fields
                    Write to file
                ELSE IF member is PremiumMember
                    Format all premium details (ID, name, charges, discount, payment, etc.)
```

```
            Write to file


        Close file
        Show message "member details added"
    CATCH IOException
        Show error message


IF event source is read OR pread
    Make readframe visible
    Clear read text area

    TRY
        Open file "MemberDetails.txt" for reading
        WHILE characters are available
            Read character
            Append character to text area
    CATCH FileNotFoundException
        Show message "FILE NOT FOUND"
    CATCH IOException
        Show error with exception message
```

Method Description

Methods in java is a block of code which performs certain tasks. A class can contain several methods, these methods are the behaviors. It is a organized form of code which can be reused as per the need. Methods can be either instance which belongs to an object or static which belongs to a class. Different access modifiers, return types and keywords can be used to declare a method as needed.

Methods in GymMember class

markAttendance()

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

It is a abstract methods which is overwritten by the child classes, RegularMember and PremiumMember.

activateMembership()

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

This method sets the active status variable which is declared as boolean to true when it is called upon.

deactivateMembership()

```
public void deactivateMembership()
{
    if(activeStatus)//checking if its active or not
    {
        activeStatus=false;
    }
    else
    {
        System.out.println("It seems like active status is already inactive");
    }
}
```

This method checks if the active status is false or not according to it, it decides to set the active status or display message. If the active status is on, it turns it off. If the active status is off, it displays a message stating that the active status is inactive.

resetMembership()

```
public void resetMembership()
{
    activeStatus=false;
    attendance=0;
    loyaltyPoints=0;
}
```

When this method is called upon, it resets the active status to false, attendance to 0 and loyalty points to 0.

display()

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

This method displays id, name, location, phone, email, gender, DOB, membership start date, attendance, loyalty points and active status of members.

Methods in RegularMember class

markAttendance()

```
@Override
public void markAttendance()
{
    attendance++;
    loyaltyPoints+=5;
    if(getAttendance()>=attendanceLimit)
    {
        isEligibleForUpgrade=true;
    }
}
```

This method is overridden as RegularMember inherits the GymMember. In this method, attendance is increased by 1 and loyalty point is increased by 5 every time the method is called. It checks if the attendance is more than the attendance limit or not, if it is more, then isEligibleForUpgrade is set to true.

getPlanPrice(String plan)

```
public double getPlanPrice(String plan)
{
    double x;
    switch(plan)
    {
        case "basic":
            return 6500;
        case "standard":
            return 12500;
        case "deluxe":
            return 18500;
        default:
            return -1;
    } /*no break statement because it directly returns
    in java once return is executed, it skips the whole block*/
}
```

This method takes a parameter, String variable named plan, and has double as the return type. In this method, a switch statement is placed to check the selected plan and return the plan price. If basic is selected, it returns 6500. If standard is selected, it returns 12500. If deluxe is selected, it returns 18500. For default, it return -1.

upgradePlan(String plan)

```
public String upgradePlan(String plan)
{
    if (isEligibleForUpgrade!=true)//check eligibility
    {
        return "the member is not eligible for upgrade";
    }
    else//if member is eligible
    {
        if(this.plan==plan)//checks for same plan
        {
            return "the member has chosen the same plan";
        }
        else //if not same
        {
            if(getPlanPrice(plan)==-1)//check invalid plan
            {
                return "the member has entered invalid plan";
            }
            else //all requirement matches
            {
                this.plan=plan;
                this.price=getPlanPrice(plan);
                return ("plan upgraded to "+plan+" price= "+price);
            }
        }
    }
}
```

In this method, there is an if else statement. It takes a string variable named plan as parameter. It checks if the member is eligible for update. if not, then it displays a message saying member is not eligible for upgrade. If they are eligible, then it checks if the selected plan is same as current plan and displays a message saying the same. If not, then it again checks if available plan is entered or not and displays a message if not. If everything checks out, then it upgrades the plan.

revertRegularMember(String removalReason)

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

This method triggers the resetMembership() method of the parent class, GymMember. It takes a string variable named removalReason as parameter. It proceeds to set every variable value to the

reset value which is false for isEligibleForUpgrade, basic for plan, 6500 for plan price and entered removal reason is also set.

display()

```
@Override
public void display()
{
     super.display();
     System.out.println("plan: "+plan);
     System.out.println("price: "+price);
     if(removalReason!="")
     {
         System.out.println("removal reason: "+removalReason);
     }
}
```

This method overrides the display() method in the GymMember class which is a parent class. It takes no parameter. It calls the display method of the parent class. It prints out plan, price and removal reason if any is entered.

Methods in PremiumMember class

payDueAmount(double paidAmount)

```
public String payDueAmount(double paidAmount)
{
    if(this.paidAmount==premiumCharge)
    {
        this.isFullPayment=true;
    }
    if(this.isFullPayment)
    {
        System.out.println("full payment has been done");
        return "the member has already paid full amount";
    }
    else
    {
        this.paidAmount+=paidAmount;
        if(this.paidAmount>premiumCharge)
        {
            this.paidAmount-=paidAmount;//get back the paid amount before it was exc
            System.out.println("the member has paid more than premiumCharge.");
            return "please pay the correct amount";
        }
        else
        {
        System.out.println("payment was successful");
        return "remaining Amount="+(premiumCharge-this.paidAmount);
        }
    }
}
```

This method in the premium member class has string as return type and takes double variable named paidAmount as parameter. It checks if the paidAmount is equal to the premium charge or not and sets true to isFullPayment which is Boolean. After this, another if statement checks if there has been a full payment with the variable isFullPayment. If yes, then it displays a message saying that. If not, it adds the paidAmount to the total paid amount and proceeds to check if the paid amount is more than premium charge or not. If it is more, then it says so, doesn't accept that payment and requests the correct amount. If not, it prints out the remaining amount.

calculateDiscount()

```
public void calculateDiscount()
{
    if(this.isFullPayment)
    {
        this.discountAmount=(10/100)*premiumCharge;
        System.out.println("discount is calculated. discount amount= "
        +this.discountAmount);
    }
    else
    {
        System.out.println("discount is not available");
    }
}
```

In this method, it has no return type nor any parameter. It checks if full payment is made or not. If it has, then it is eligible for discount and the discount is calculated and displayed. If not, then it prints out saying it is not eligible for discount.

revertPremiumMember()

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

This method calls the parent class resetMembership() method and sets default values to the variables in the PremiumMember class. It sets trainer's name to null, isFullPayment to false, paidAmount and discountAmount to zero.

markAttendance()

```
@Override
public void markAttendance()
{
    attendance++;
    loyaltyPoints+=10;
}
```

This method is overridden from the parent class method which is abstract. It inceases attendance by one and loyalty points by 10 when this method is called upon.

display()

```
@Override
public void display()
{
    super.display();
    System.out.println("personal trainer= "+personalTrainer);
    System.out.println("paid amount= "+paidAmount);
    System.out.println("is the amount fully paid= "+(isFullPayment?"yes":"no"));
    if(this.isFullPayment)
    {
        System.out.println("discount amount= "+discountAmount);
    }
    else if(this.paidAmount<premiumCharge)
    {
        System.out.println("remaining Amount="+(premiumCharge-this.paidAmount));
    }
}
```

This method is overridden from the parent class. It calls for the parent class display method. It displays personal trainer, paid amount and if the payment has been made full or not. It checks for discount and prints it too.

Method in GymGUI class

actionPerformed(ActionEvent ae)

This method in the GymGUI class does the work for all the button in it. Here it has separate working mechanism for all buttons. If any button is pressed, then it methods acts up to full fill what that button should do. The actions performed by the button are written in the form of if else if else statement where the condition is the button clicked.

The buttons work as follows:

Reg= opens regular member frame

Pre= opens premium member frame

Back1, back2= return back to the main option frame

Addregular = adds regular member

Addpremium = adds premium member

Markattendance = marks the attendance of the members

Activate = activates membership for the members

Deactivate = deactivates membership for the members

Clear = clears all text fields

Display = displays the information about the members

Revert membership = resets the membership back to the initial phase

Upgrade plan = upgrade the plan as per the user chooses only for regular member

Discount = calculates and returns the discount received by premium member

Save = saves the information about the users in a text file

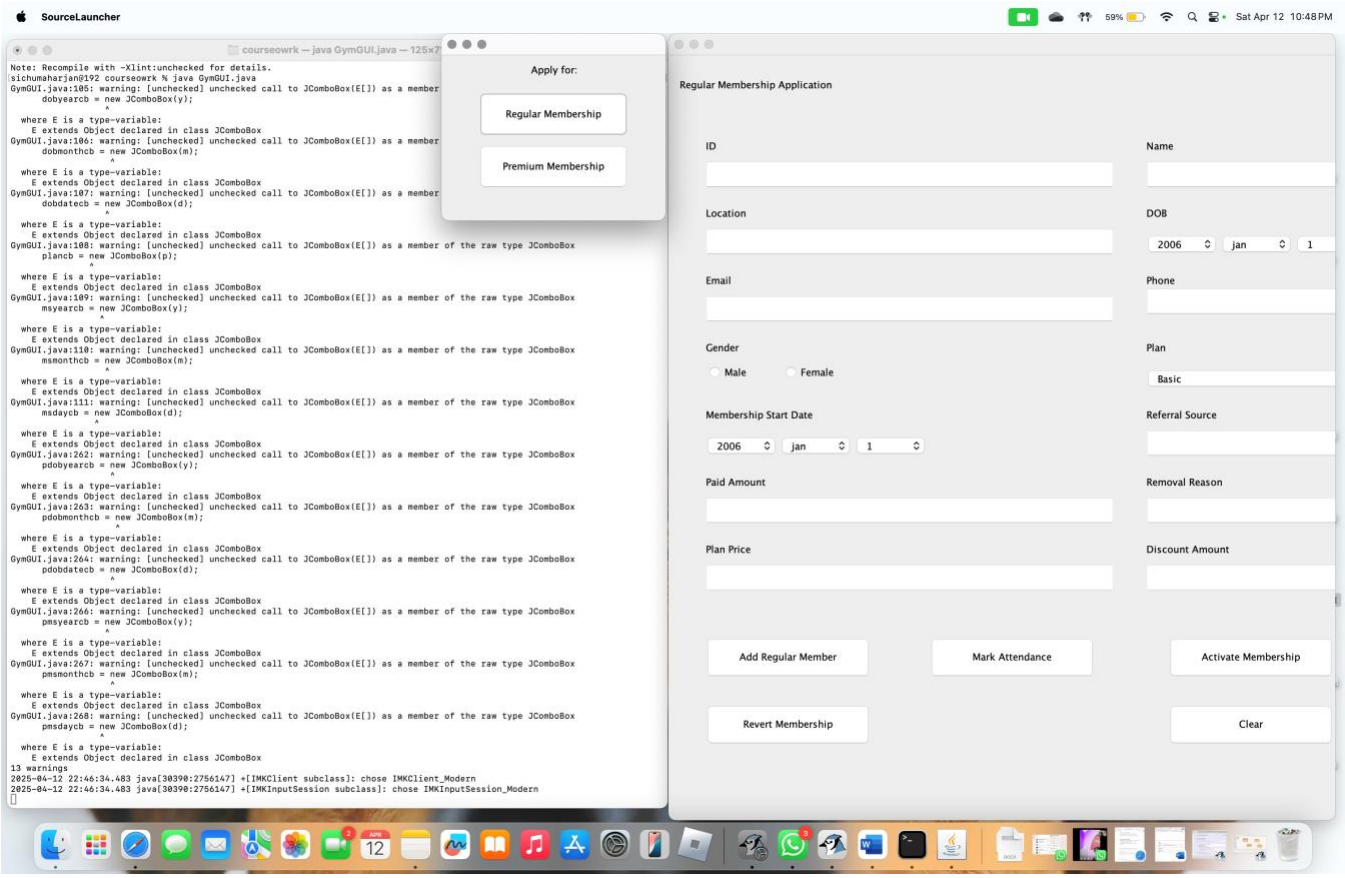
Read from file = reads the information in the text file and displays it.

Testing

Test-1

Table 1: test-1: compile and run the program

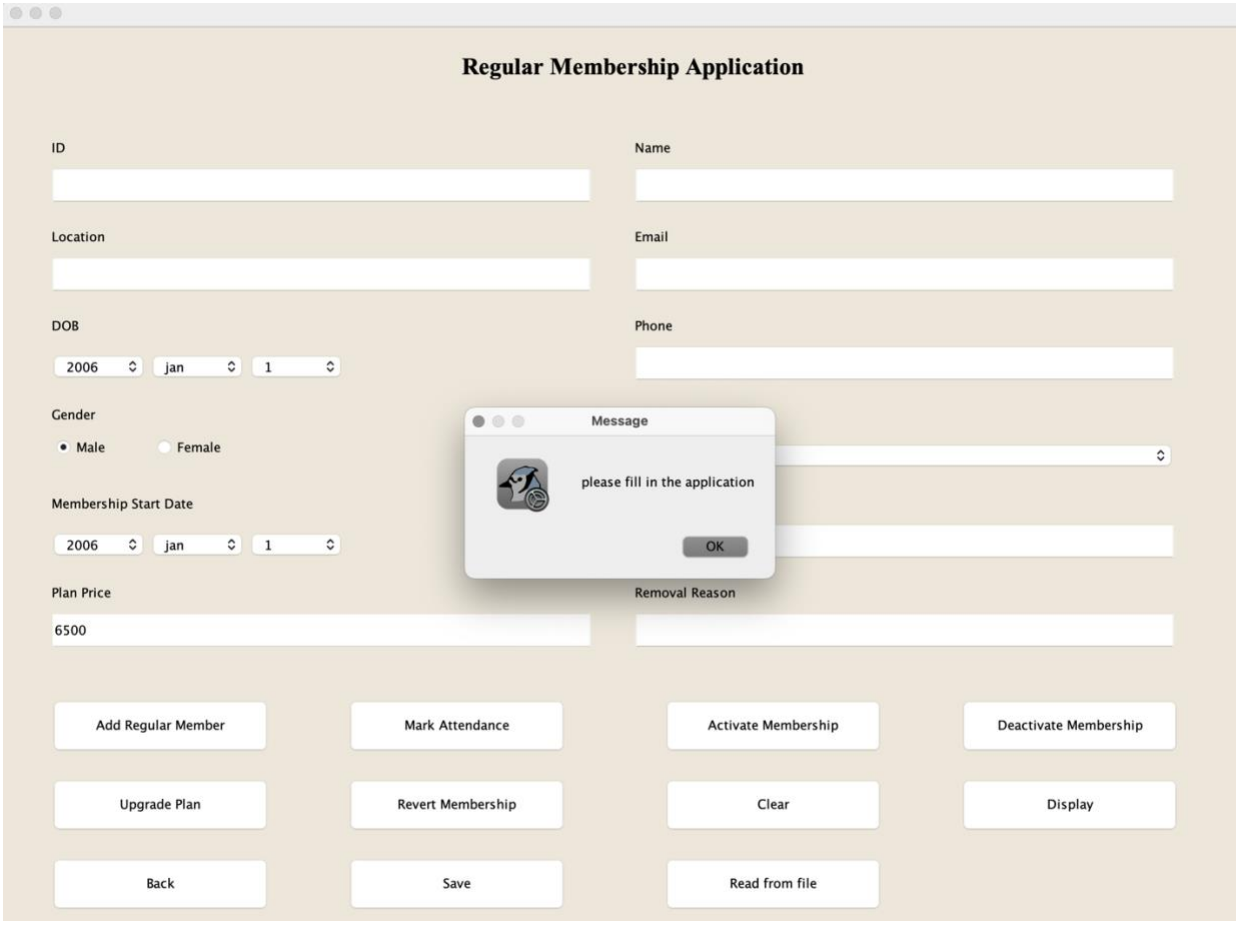
Objective	To compile and run the program in terminal.
Action	Write “javac GymGUI.java” & “java GymGUI.java” in terminal.
Expected Output	The option to choose for either regular membership or premium membership appears and when clicked on their respective GUI opens.
Actual Output	Option to choose appeared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

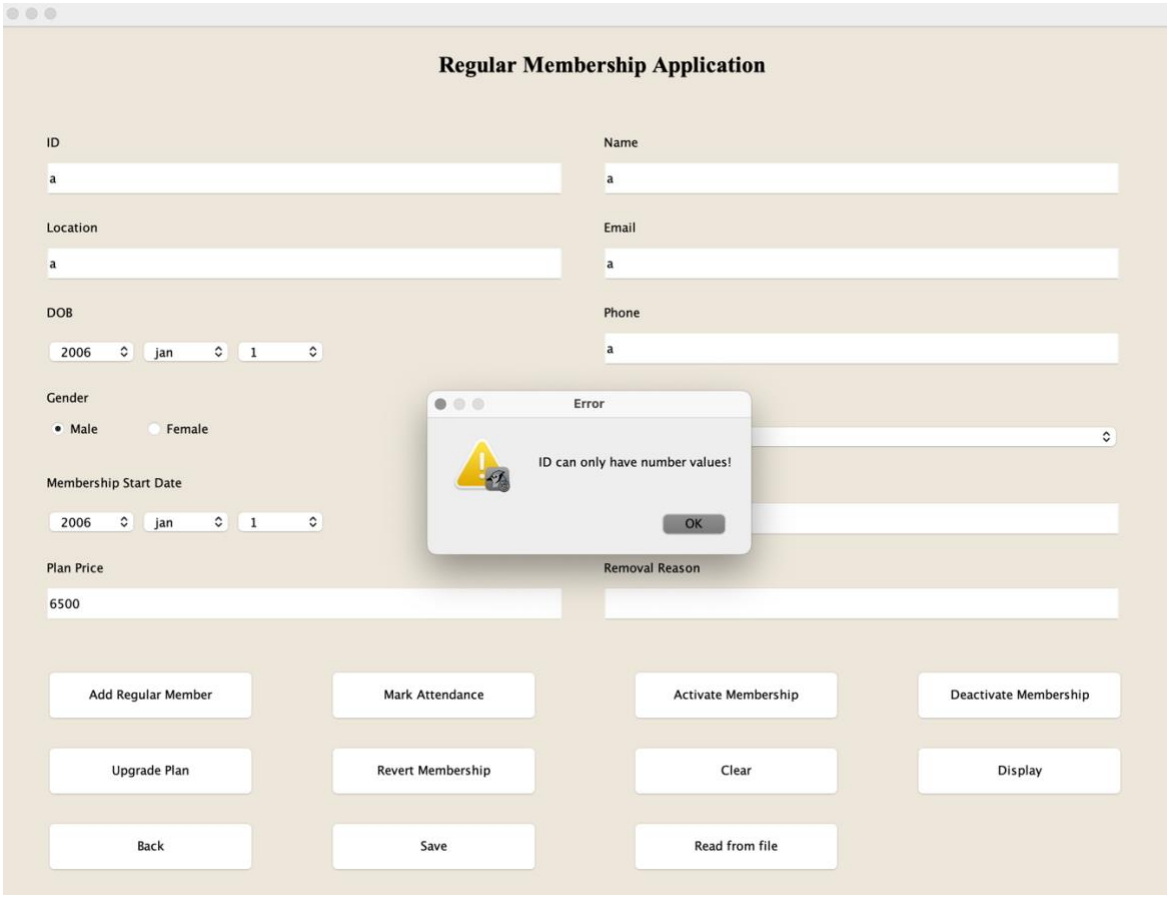


Test-2

Table 2: test-2.1: add regular member

Objective	To add regular member.
Action	1. click ‘add regular member’ button. 2. enter alphabetical id. 3. fill in the application in correct manner.
Expected Output	The regular member details should be added and a dialog box showcasing the message with appear.
Actual Output	The details were added, and the dialog box appeared with a success message.
Result	The test was successful.




Regular Membership Application
ID
a
Name
a
Location
a
Email
a
DOB
2006
jan
1
Phone
a
Gender
Male
Female
Error
ID can only have number values!
OK
Membership Start Date
2006
jan
1
Plan Price
6500
Removal Reason
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file

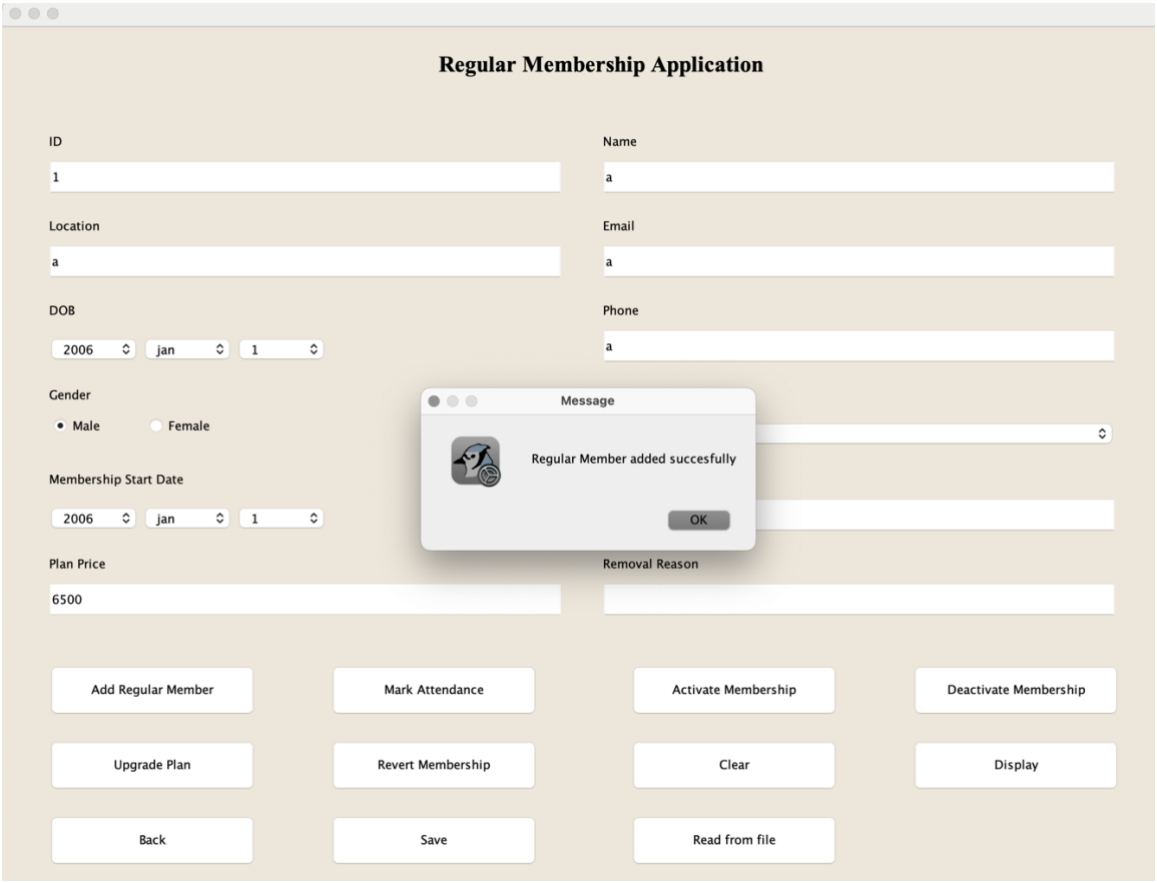
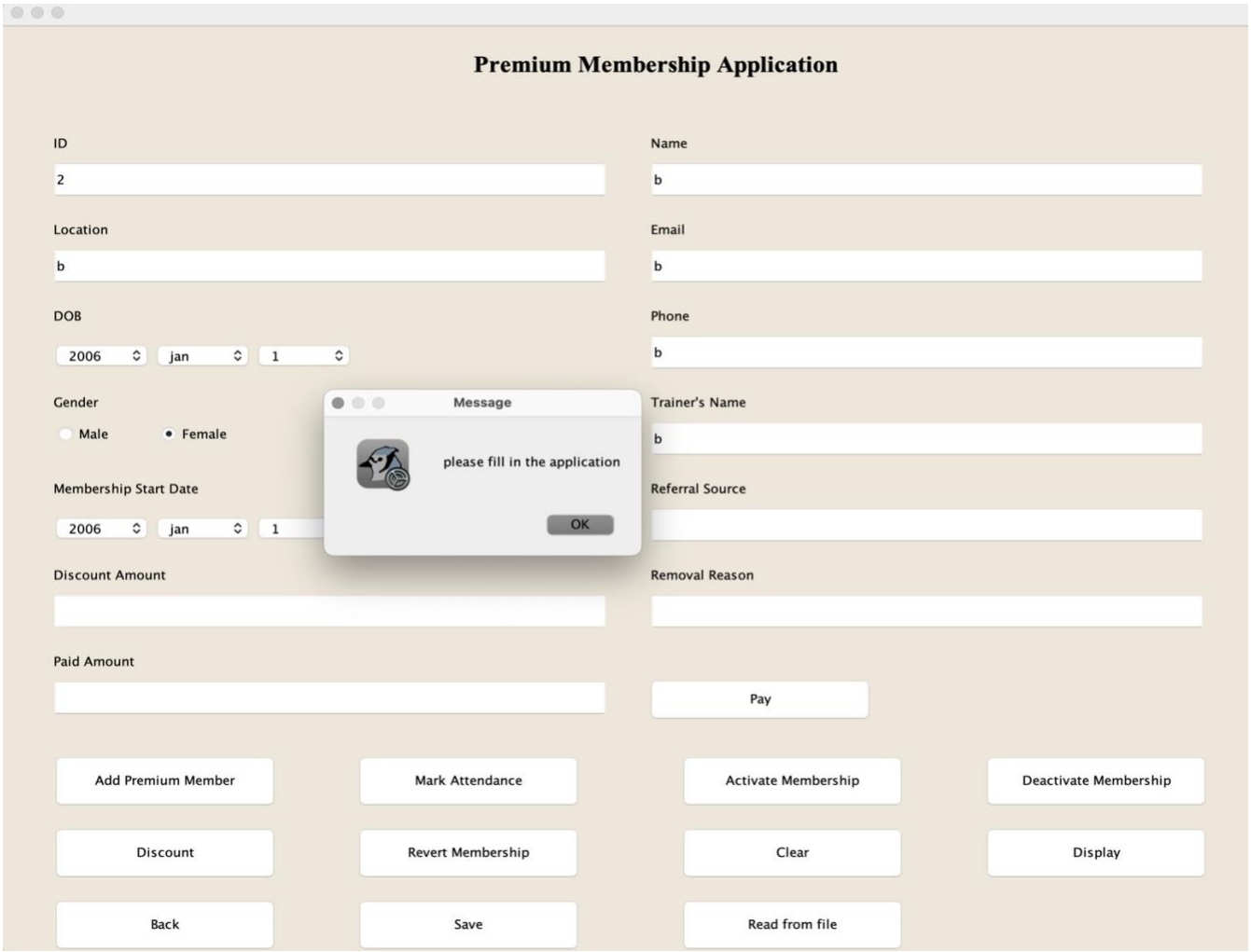
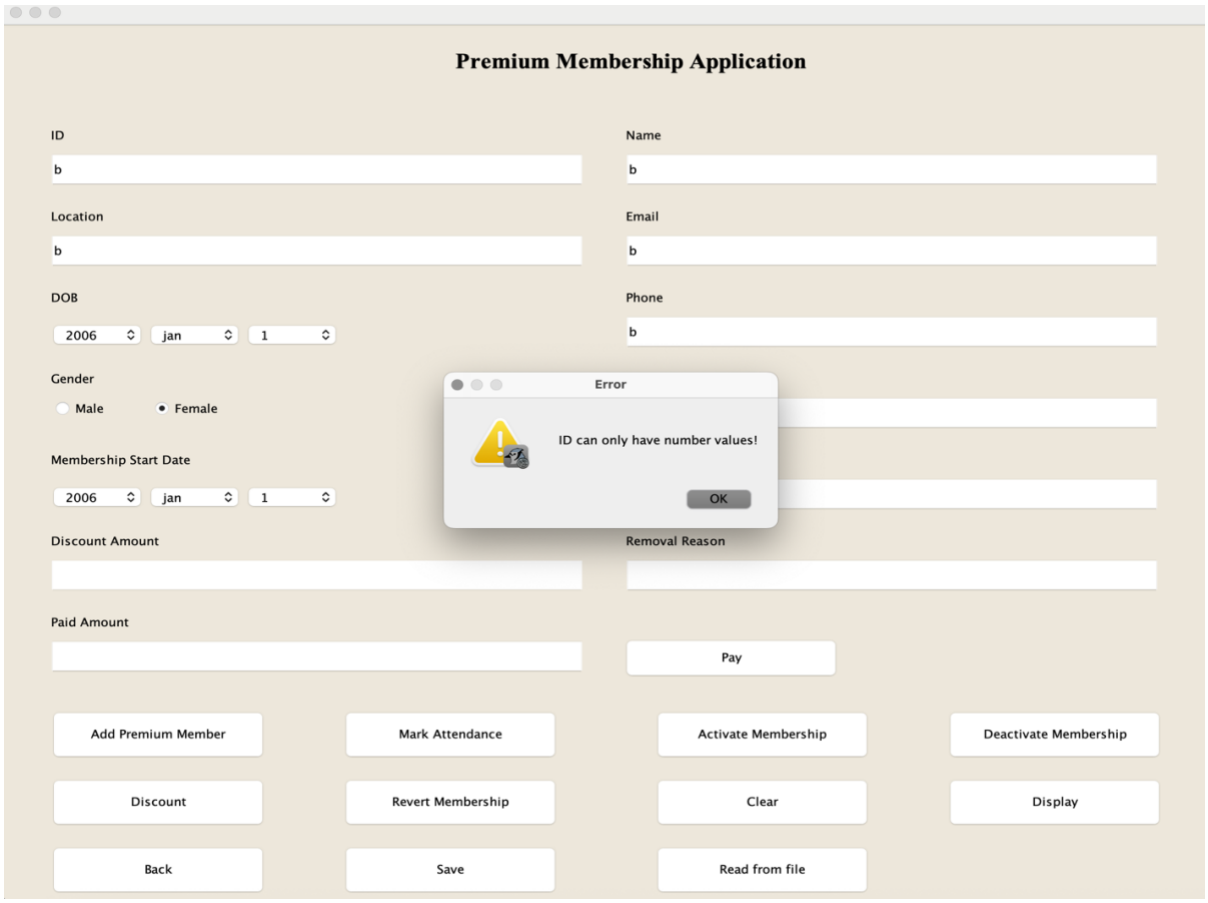
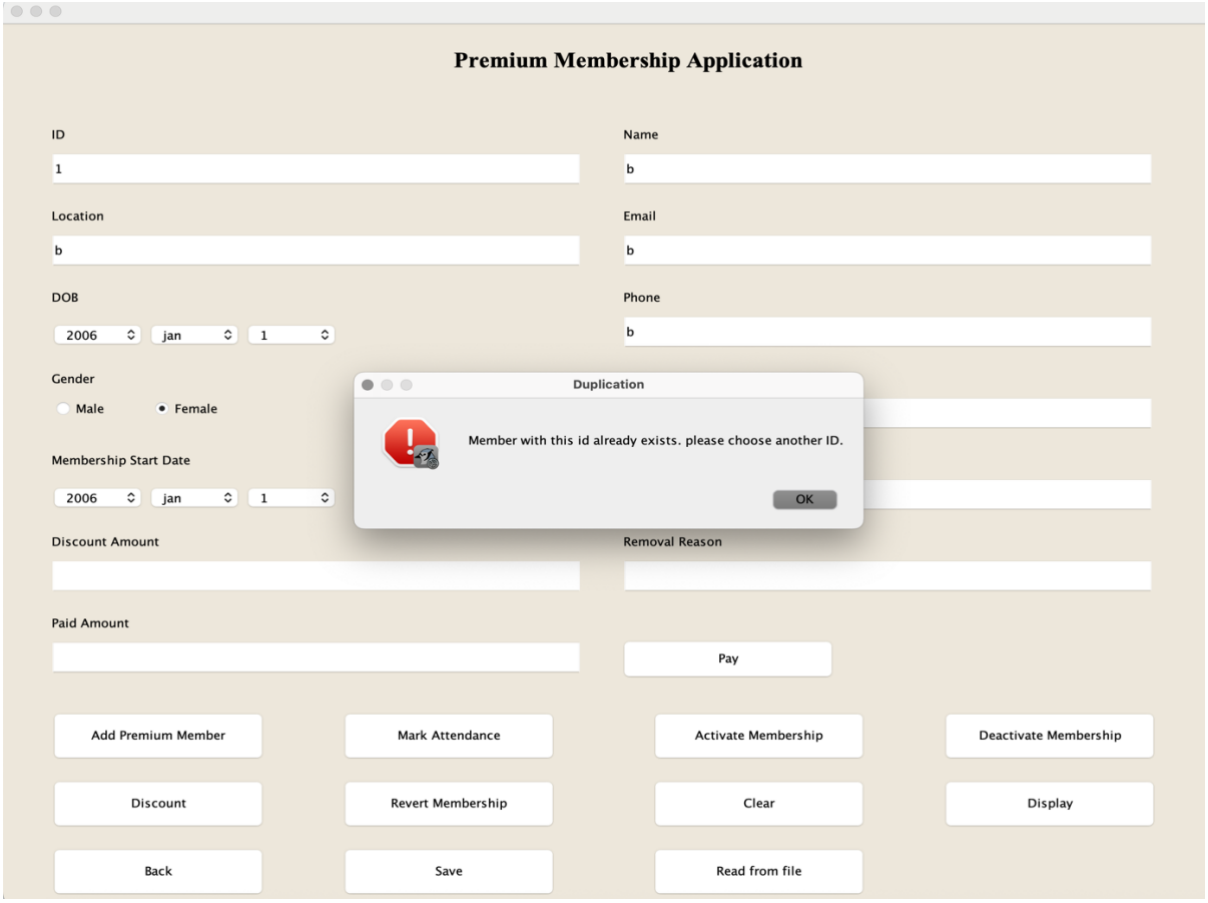

Regular Membership Application
ID
1
Name
a
Location
a
Email
a
DOB
2006
jan
1
Phone
a
Gender
Male
Female
Message
Regular Member added succesfully
OK
Membership Start Date
2006
jan
1
Plan Price
6500
Removal Reason
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file

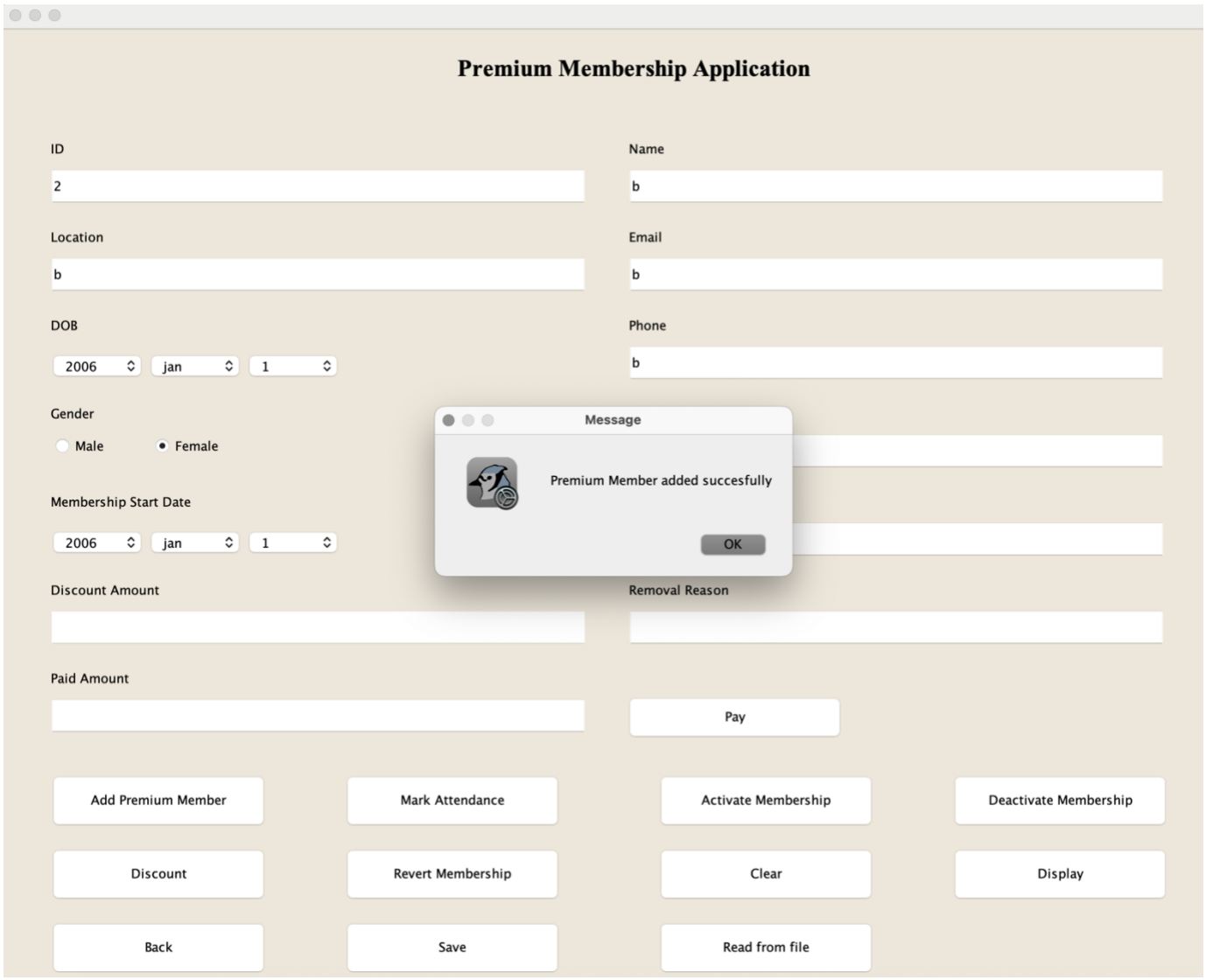
Table 3: test-2.2: add premium member

Objective	To add premium member.
Action	1. click ‘add premium member’ button. 2. enter alphabetical id. 3. already existing id was entered. 4. fill in the application in correct manner.
Expected Output	The premium member details will be added and a dialog box showcasing the message with appear.
Actual Output	The details were added, and the dialog box appeared with a success message.
Result	The test was successful.




Premium Membership Application
ID
b
Name
b
Location
b
Email
b
DOB
2006
jan
1
Phone
b
Gender
Male
Female
Membership Start Date
2006
jan
1
Error
ID can only have number values!
OK
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file

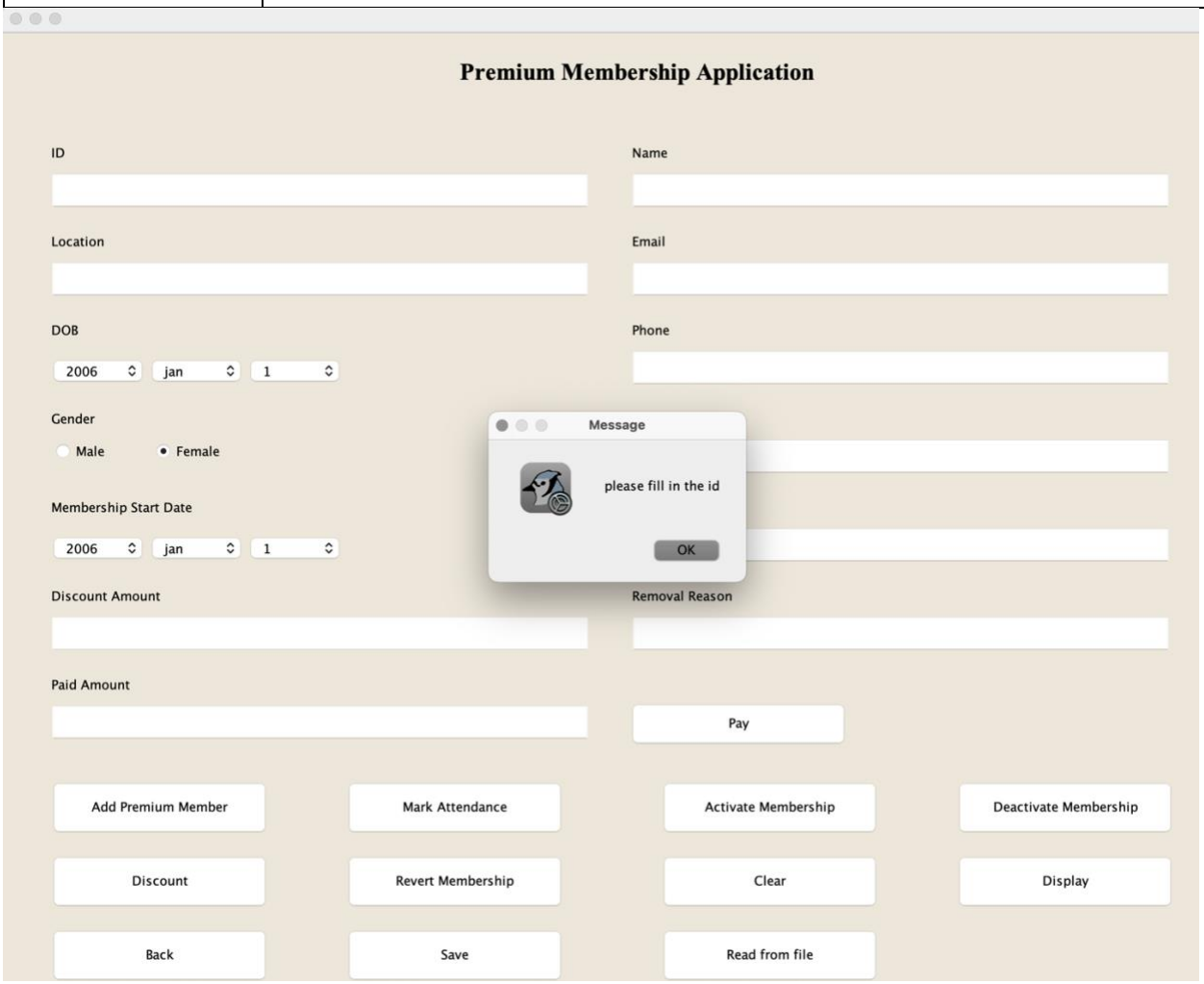

Premium Membership Application
ID
1
Name
b
Location
b
Email
b
DOB
2006
jan
1
Phone
b
Gender
Male
Female
Membership Start Date
2006
jan
1
Duplication
Member with this id already exists. please choose another ID.
OK
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file

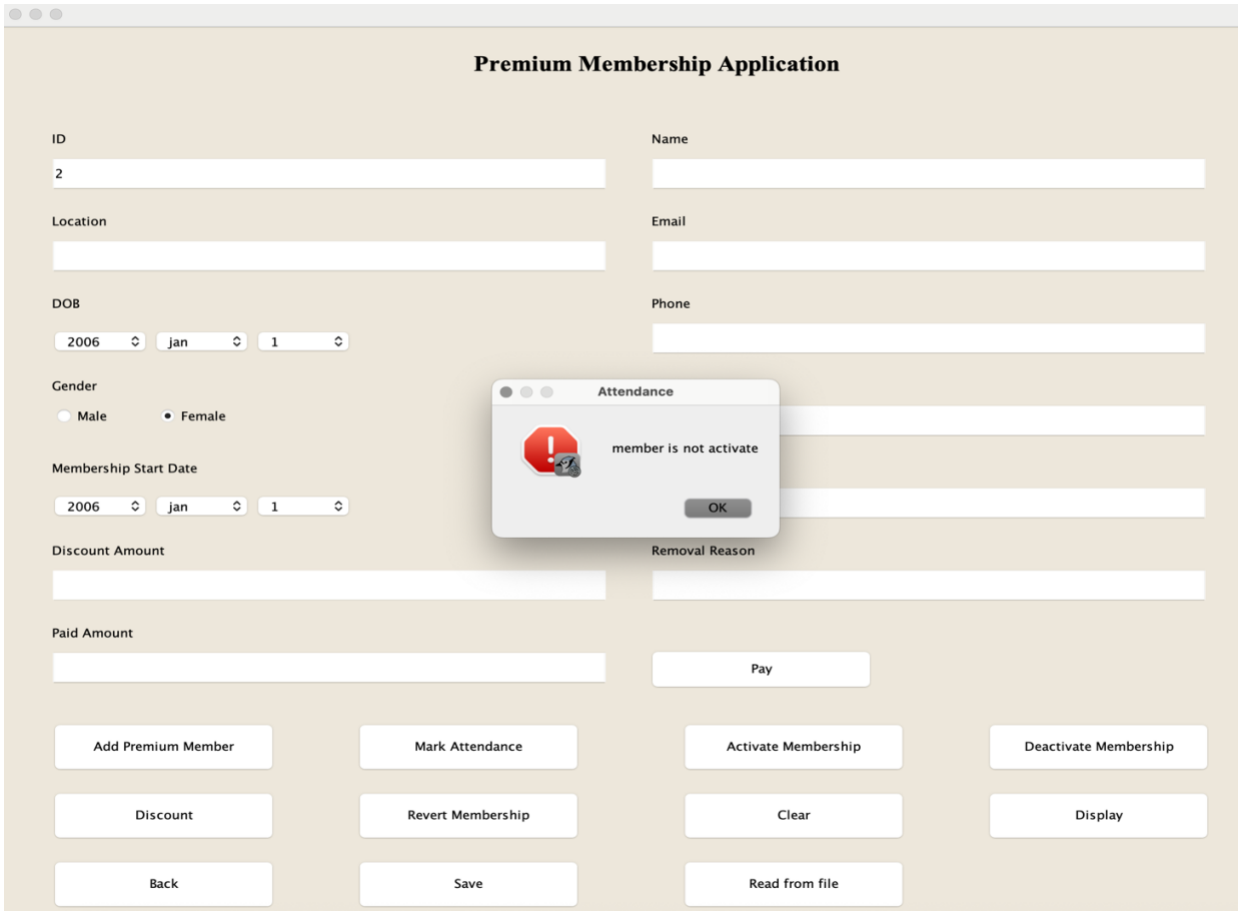

Premium Membership Application
ID
2
Name
b
Location
b
Email
b
DOB
2006
jan
1
Phone
b
Gender
Male
Female
Message
Premium Member added succesfully
OK
Membership Start Date
2006
jan
1
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file

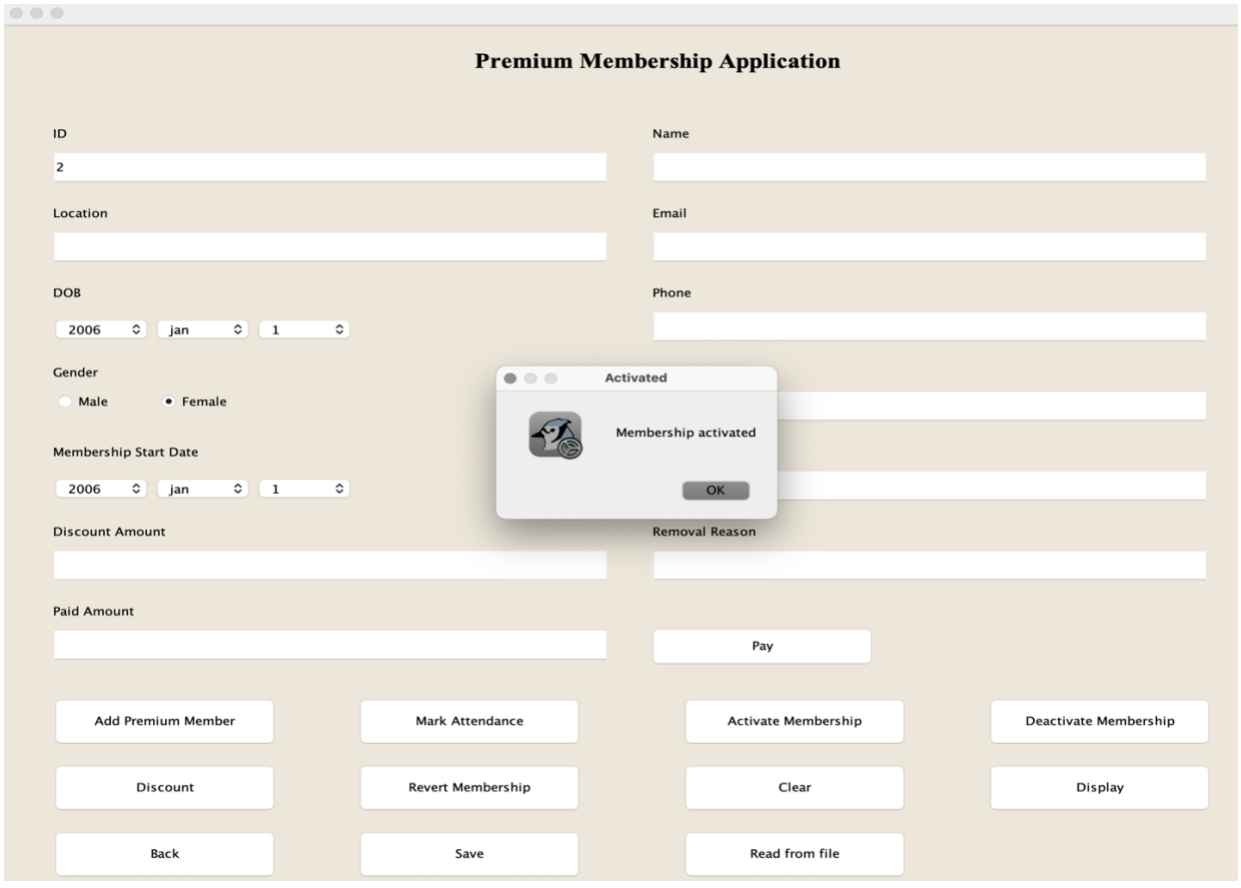
Test-3

Table 4: test-3.1: increase mark attendance

Objective	To increase mark attendance.
Action	1. click ‘mark attendance button. 2. enter id without activating membership. 3. activate membership. 4. click ‘mark attendance button again.
Expected Output	The premium member details will be added and a dialog box showcasing the message with appear.
Actual Output	The details were added, and the dialog box appeared with a success message.
Result	The test was successful.




Premium Membership Application
ID
2
Name
Location
Email
DOB
2006
jan
1
Phone
Gender
Male
Female
Attendance
member is not activate
OK
Membership Start Date
2006
jan
1
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file


Premium Membership Application
ID
2
Name
Location
Email
DOB
2006
jan
1
Phone
Gender
Male
Female
Activated
Membership activated
OK
Membership Start Date
2006
jan
1
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file

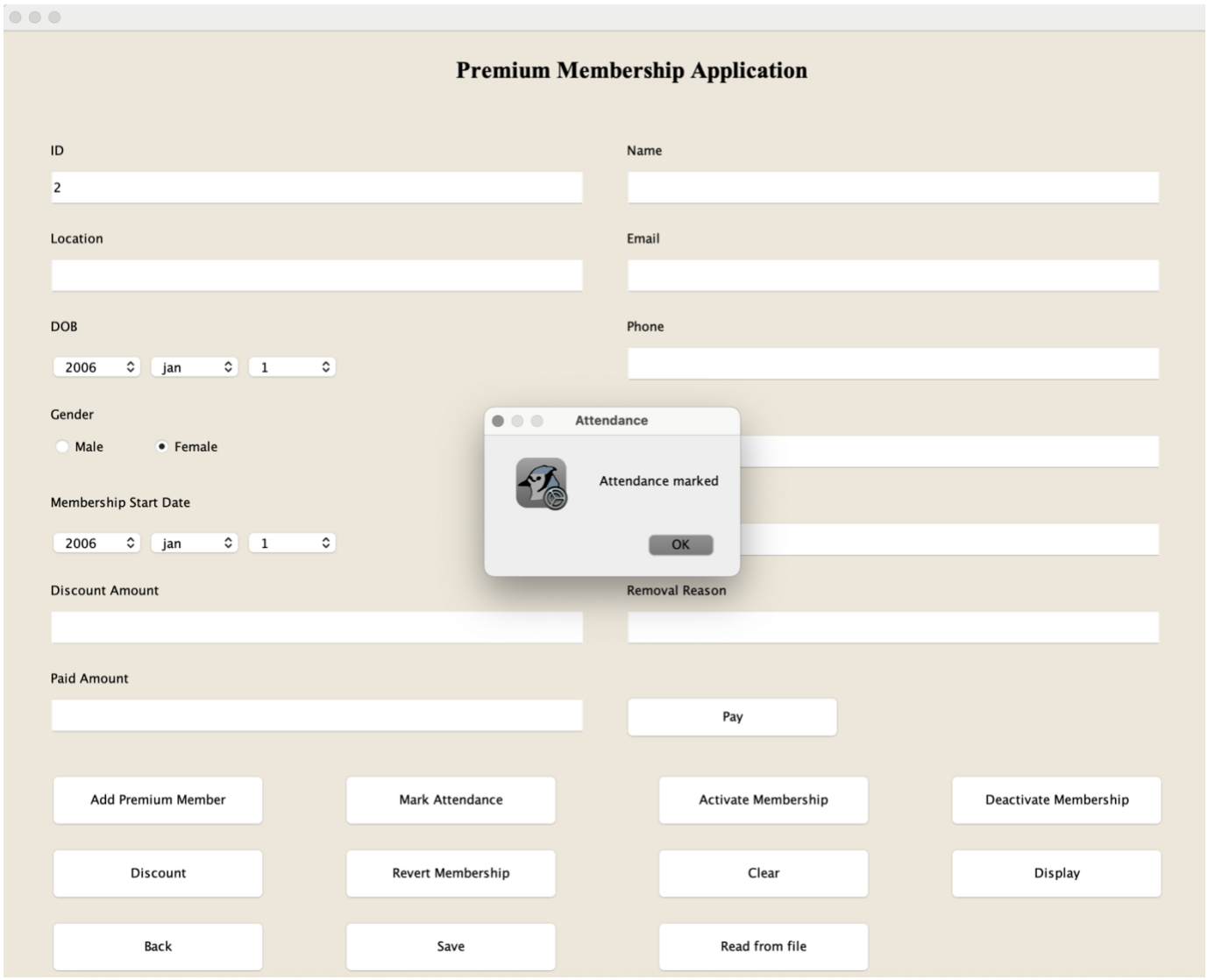
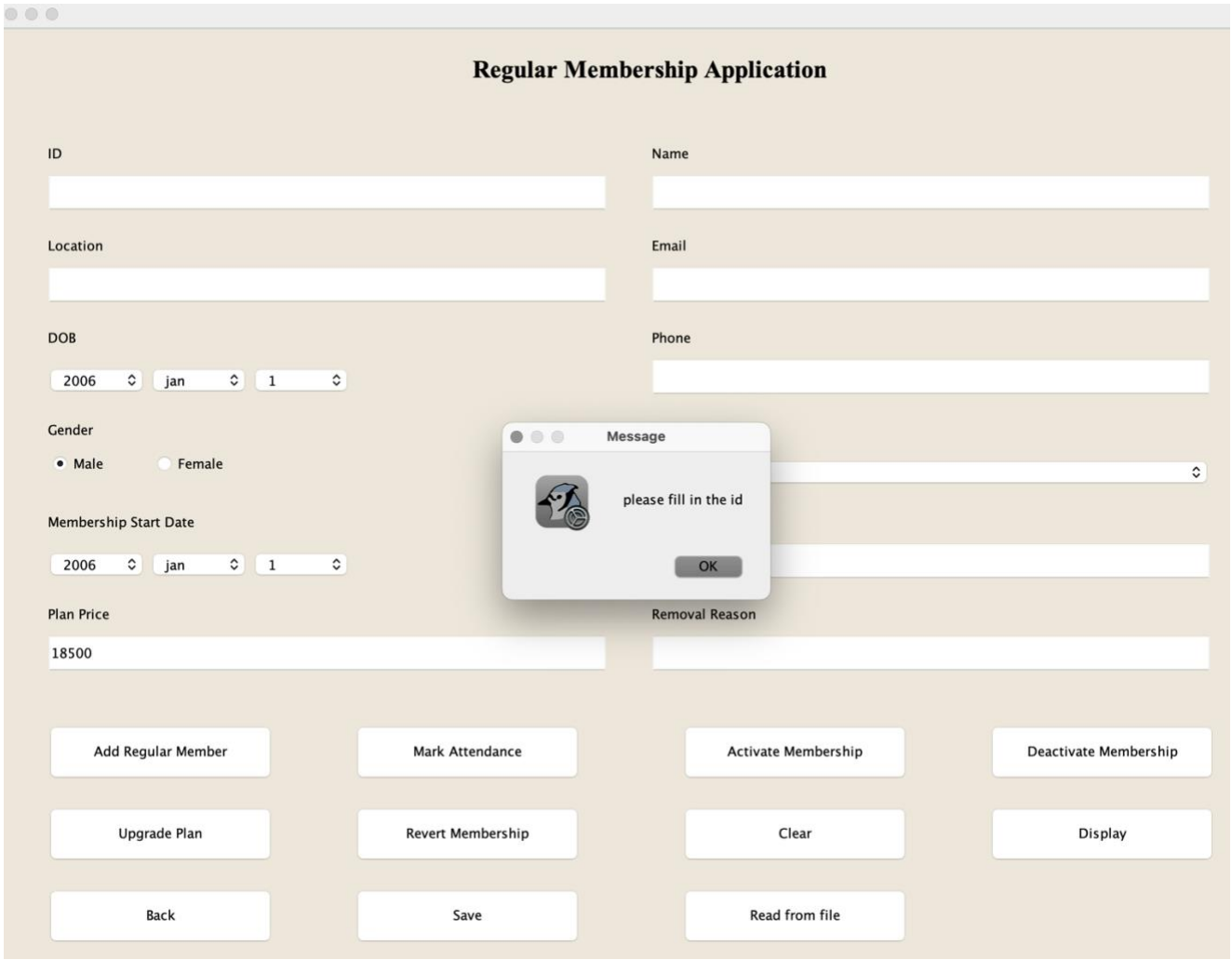
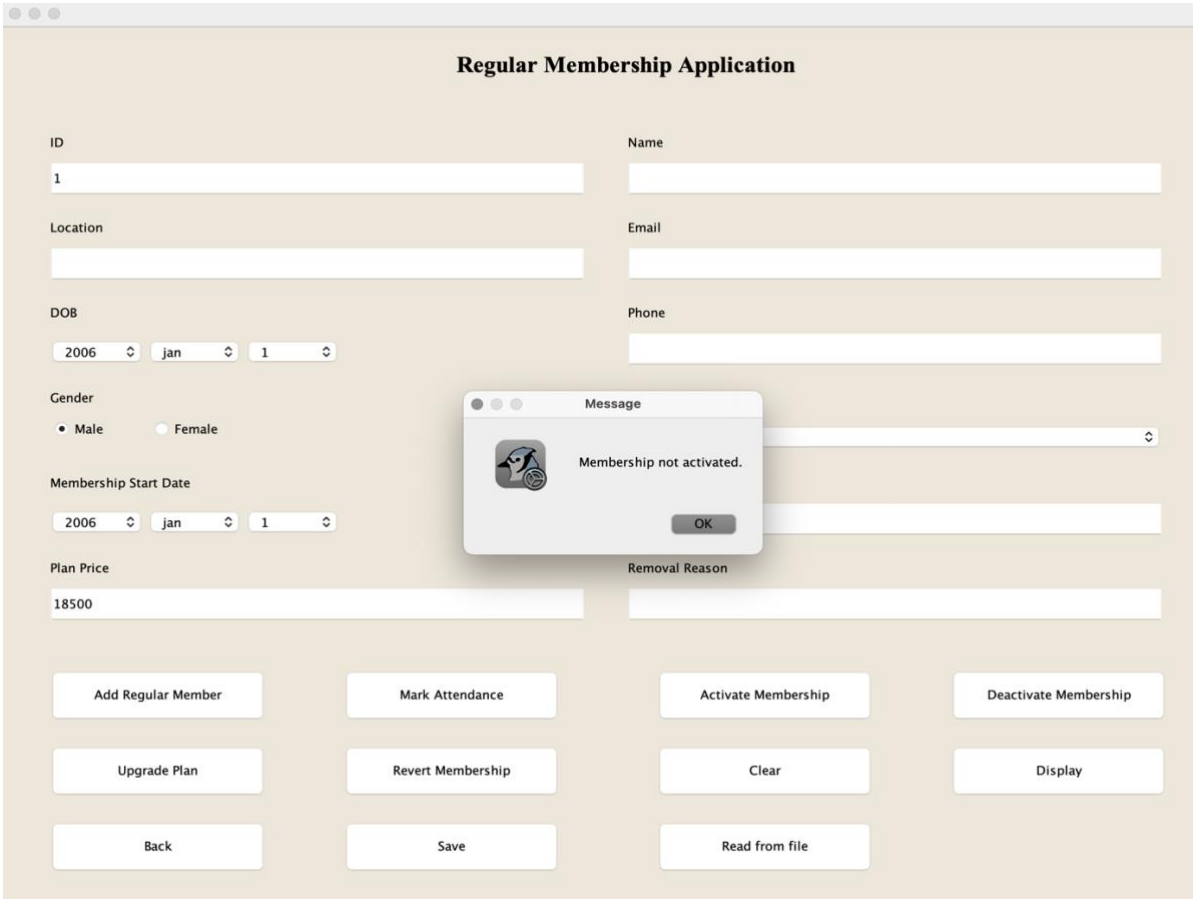
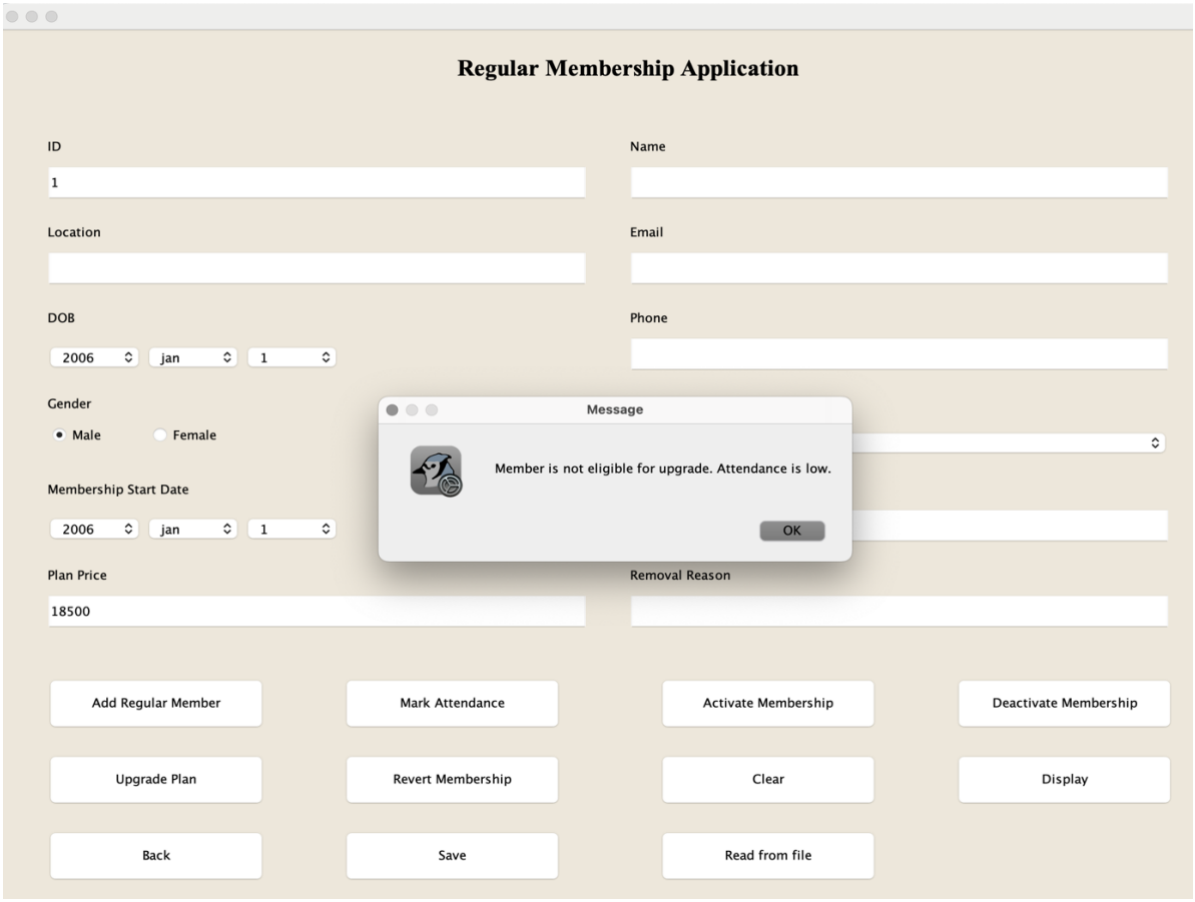

Premium Membership Application
ID
2
Name
Location
Email
DOB
2006
jan
1
Phone
Gender
Male
Female
Attendance
Attendance marked
OK
Membership Start Date
2006
jan
1
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file

Table 5: test-3.2: update plan

Objective	To upgrade the plan.
Action	1. click “upgrade plan” button without filling id. 2. fill existing id without activating the membership. 3. activate the membership and click “upgrade plan” button. 4. mark attendance 30 times and select the same plan. 5. select different plan.
Expected Output	A message dialog box stating that the plan is upgraded should be displayed.
Actual Output	A message dialog box stating that the plan is upgraded is displayed.
Result	The test was successful.

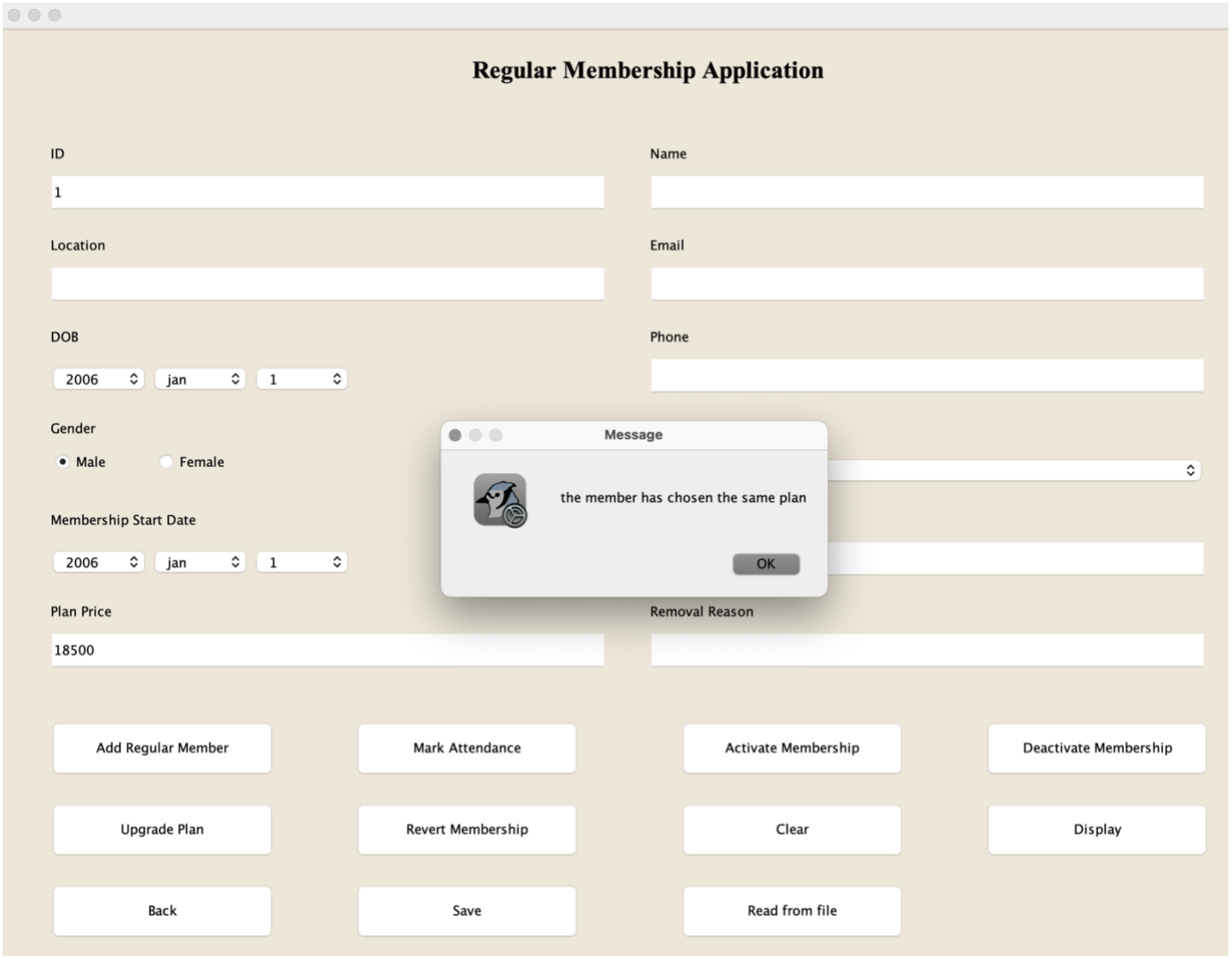


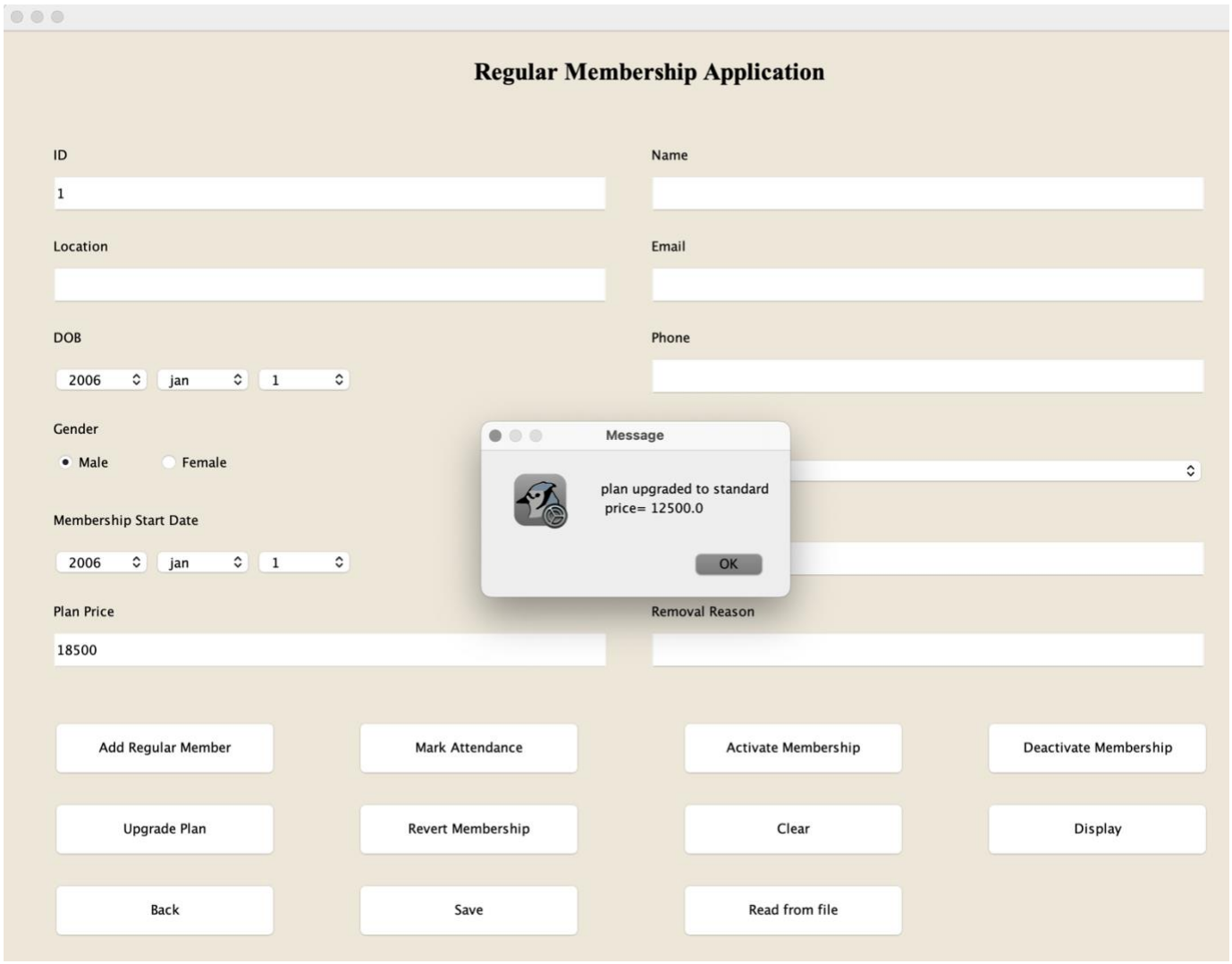

Regular Membership Application
ID
1
Name
Location
Email
DOB
2006
jan
1
Phone
Gender
Male
Female
Message
Membership not activated.
OK
Membership Start Date
2006
jan
1
Plan Price
18500
Removal Reason
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file


Regular Membership Application
ID
1
Name
Location
Email
DOB
2006
jan
1
Phone
Gender
Male
Female
Message
Member is not eligible for upgrade. Attendance is low.
OK
Membership Start Date
2006
jan
1
Plan Price
18500
Removal Reason
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file

REGULAR MEMBERS:

Member ID: 1
Name: a
Location: a
Email: a
Phone: a
Date of Birth: 1-jan-2006
Gender: male
Plan: basic
Price: 6500.0
Membership Start Date: 1-jan-2006
Attendance: 13
Loyalty Points: 65.0
Active Status: true
==========

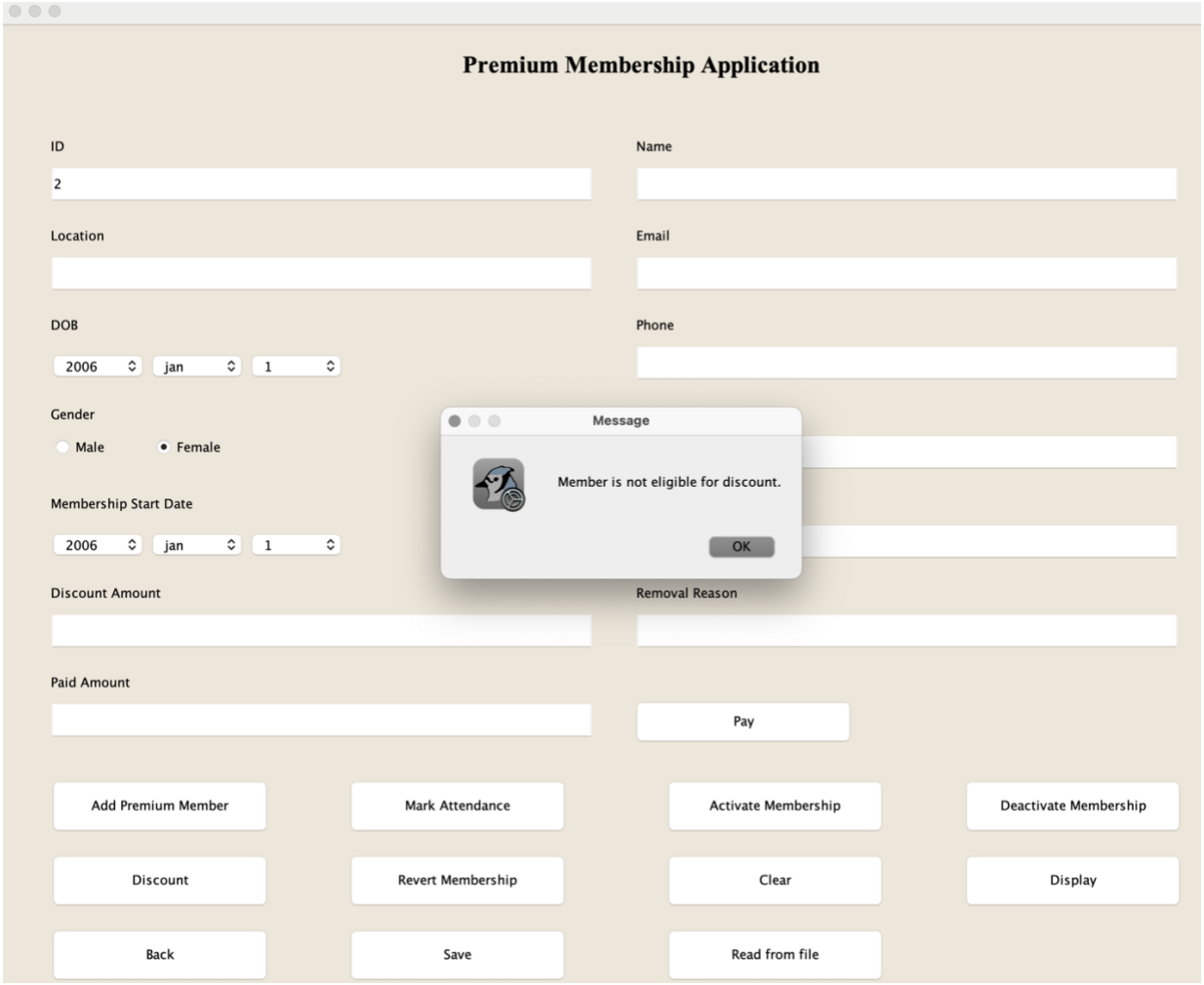



Regular Membership Application
ID
1
Name
Location
Email
DOB
2006
jan
1
Phone
Gender
Male
Female
Message
plan upgraded to standard
price= 12500.0
OK
Membership Start Date
2006
jan
1
Plan Price
18500
Removal Reason
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file

Test-4

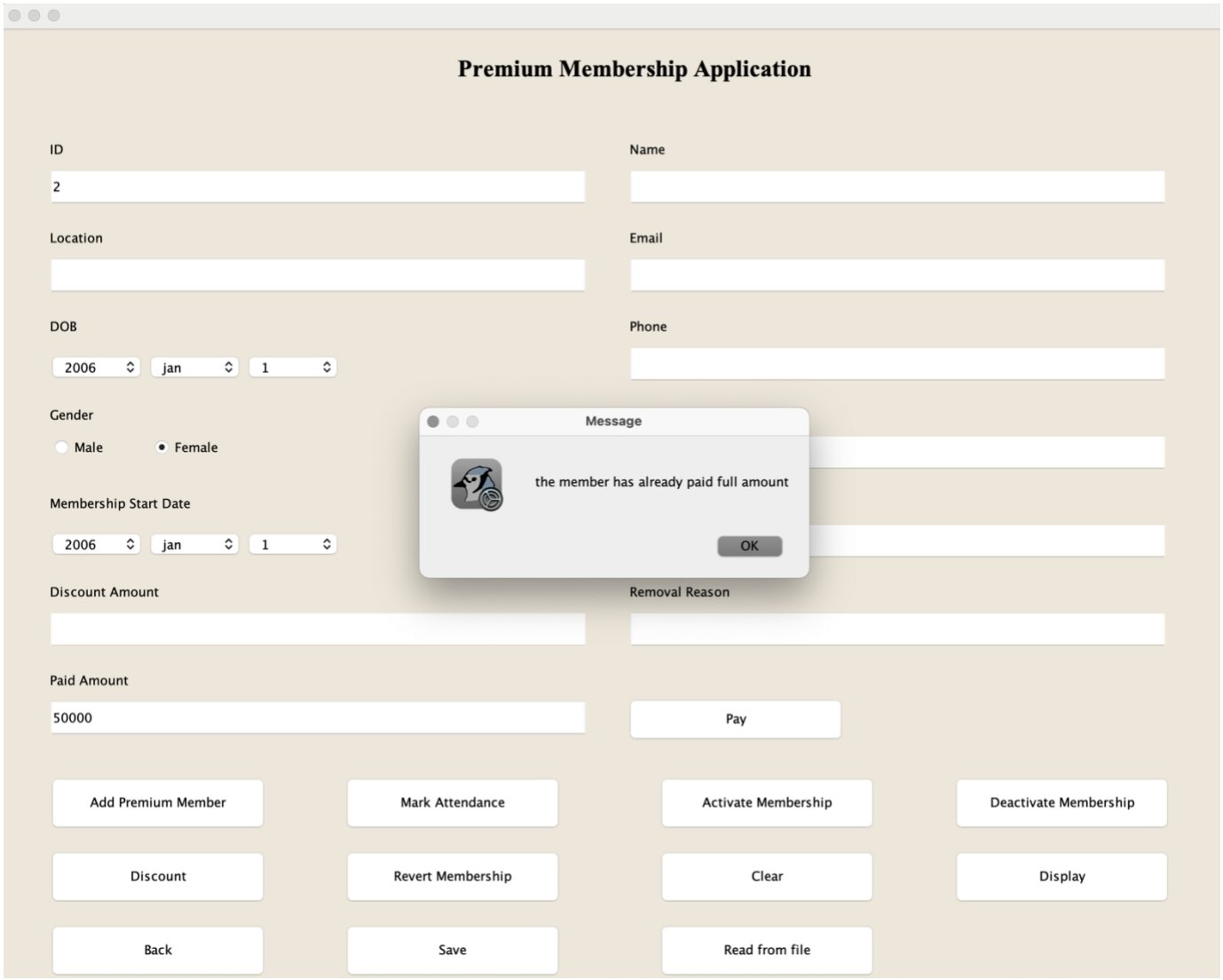
Table 6: test-4.1: calculate discount

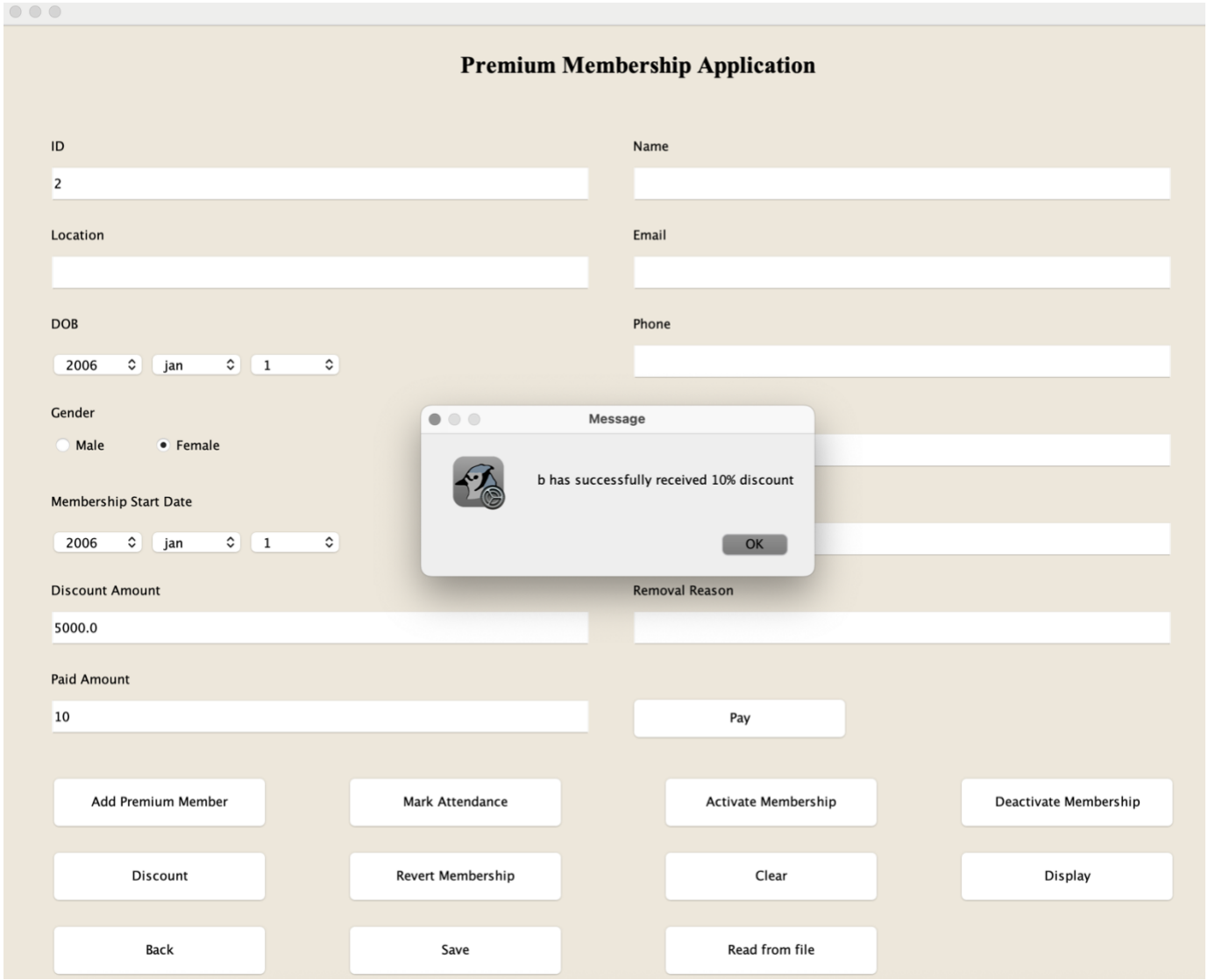
Objective	To calculate the discount.
Action	1. fill in the id but pay full amount. 2. pay full amount. 3. click “calculate discount” button.
Expected Output	A message dialog box stating that member has received 10% discount should be visible.
Actual Output	A message dialog box stating that member has received 10% discount is visible.
Result	The test was successful.



PREMIUM MEMBERS:

Member ID: 2
Name: b
Location: b
Email: b
Phone: b
Date of Birth: 1-jan-2006
Gender: female
Membership Start Date: 1-jan-2006
Attendance: 1
Loyalty Points: 10.0
Personal Trainer: b
Payment of Rs.0.0 has been made
Active Status: true
Full payment has not been done
Payment of Rs.50000.0 is left to be paid.
==========





PREMIUM MEMBERS:

Member ID: 2
Name: b
Location: b
Email: b
Phone: b
Date of Birth: 1-jan-2006
Gender: female
Membership Start Date: 1-jan-2006
Attendance: 1
Loyalty Points: 10.0
Personal Trainer: b
Payment of Rs.50000.0 has been made
Active Status: true
Full payment has been done
Discount: Rs.5000.0
==========

Table 7: test-4.2: pay due

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

```
PREMIUM MEMBERS:


Member ID: 2
Name: b
Location: b
Email: b
Phone: b
Date of Birth: 1-jan-2006
Gender: female
Membership Start Date: 1-jan-2006
Attendance: 1
Loyalty Points: 10.0
Personal Trainer: b
Payment of Rs.0.0 has been made
Active Status: true
Full payment has not been done
Payment of Rs.50000.0 is left to be paid.
==================================
```

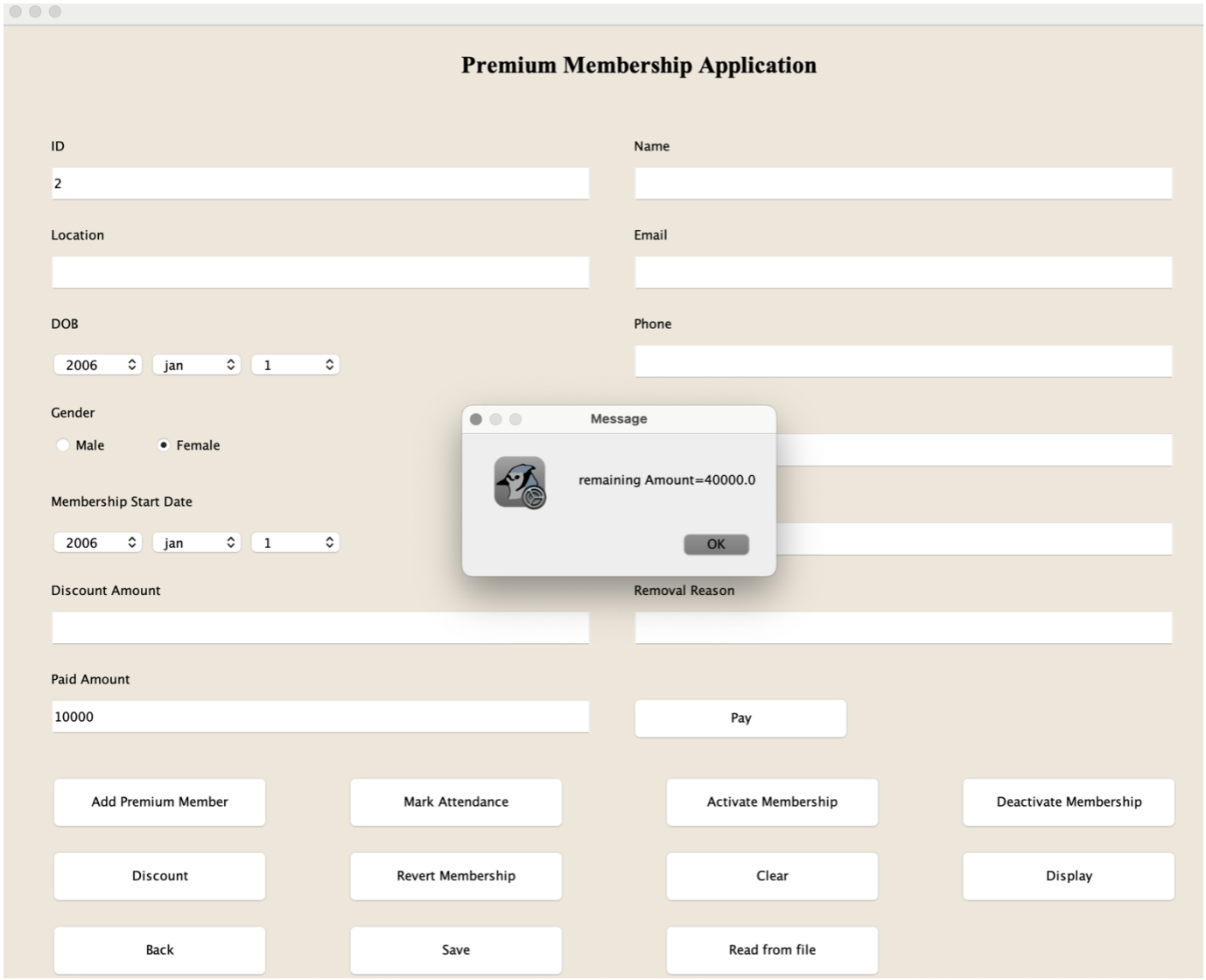
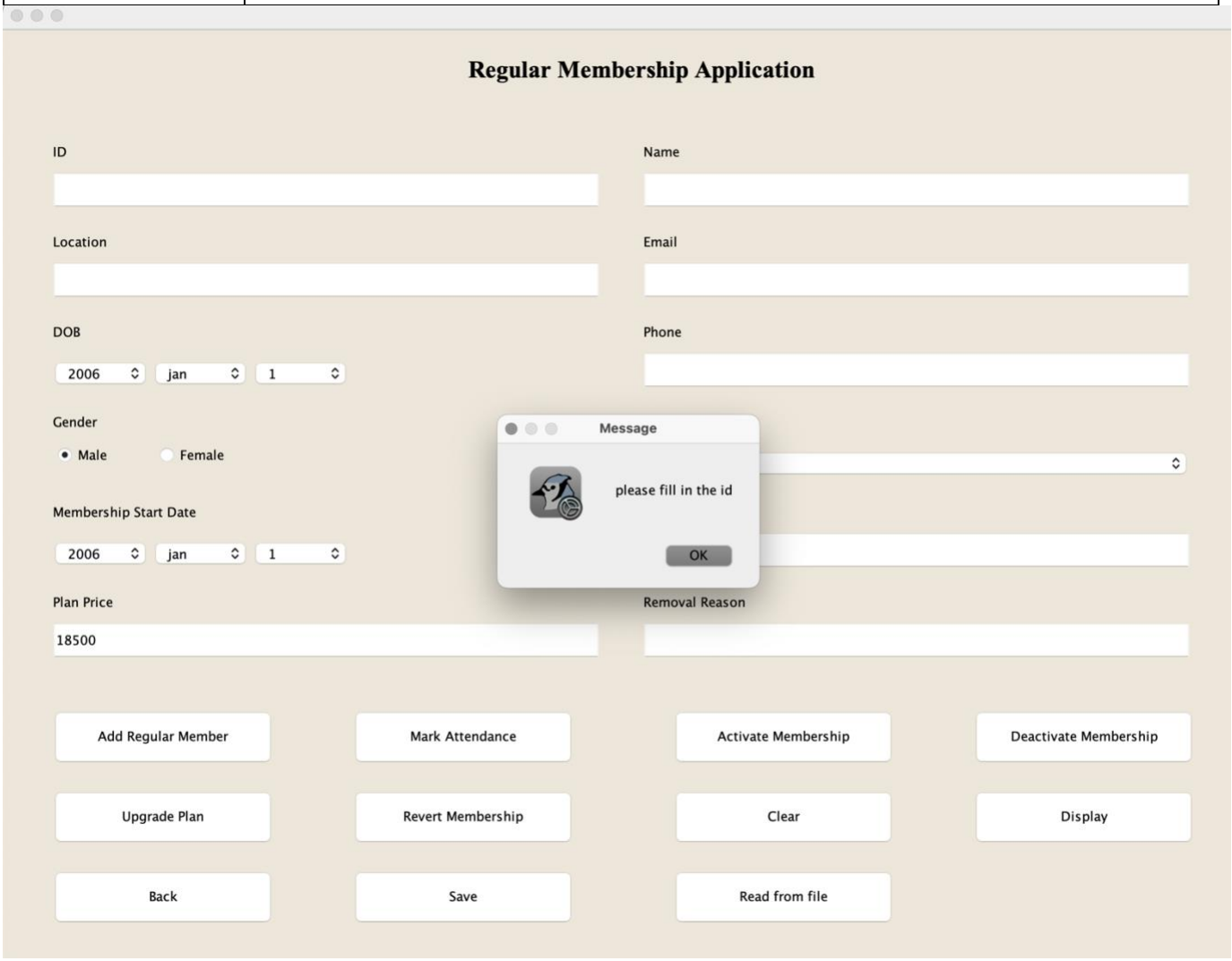

Premium Membership Application
ID
2
Name
Location
Email
DOB
2006
jan
1
Phone
Gender
Male
Female
Message
remaining Amount=40000.0
OK
Membership Start Date
2006
jan
1
Discount Amount
Removal Reason
Paid Amount
10000
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file

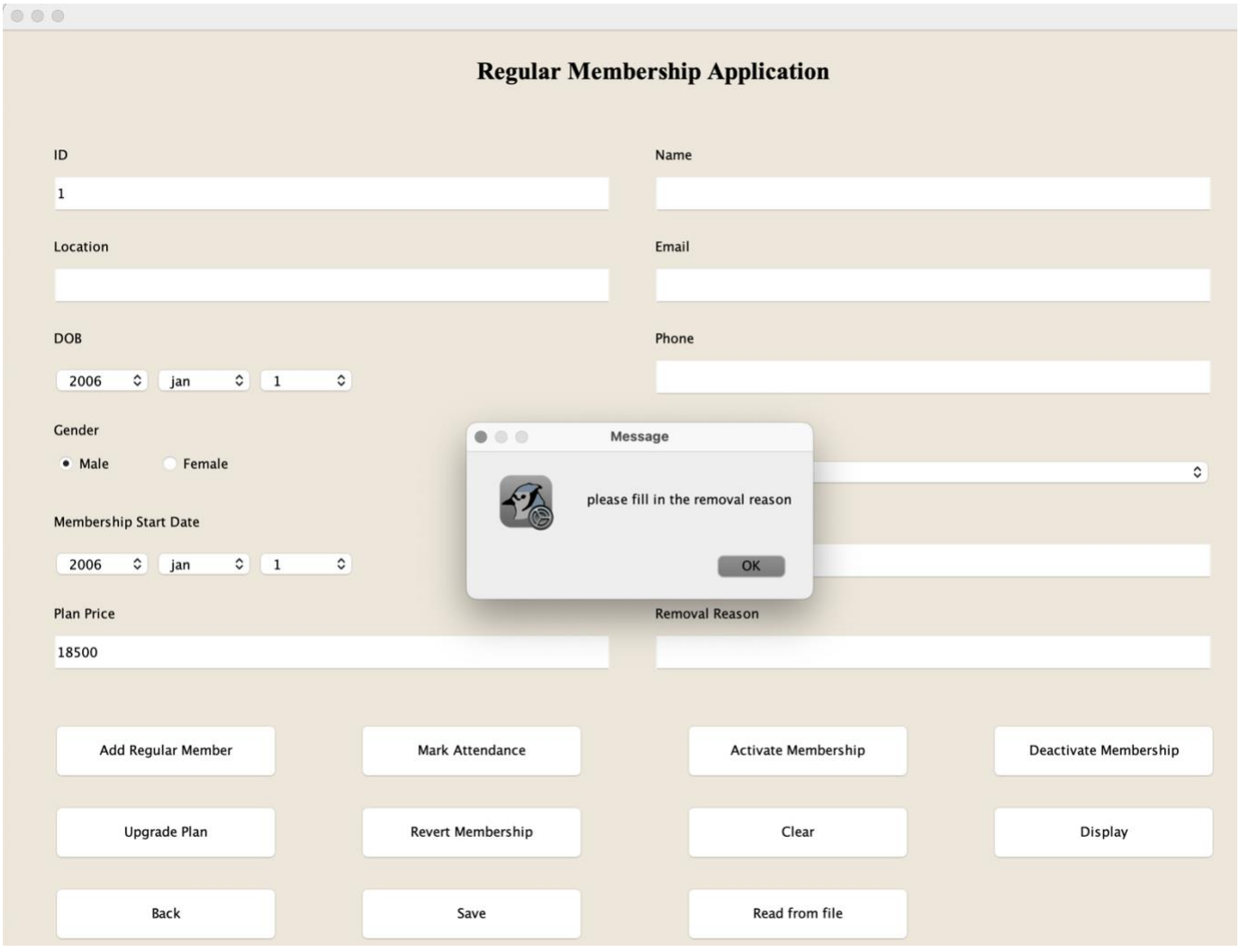
Table 8: test-4.3: revert member

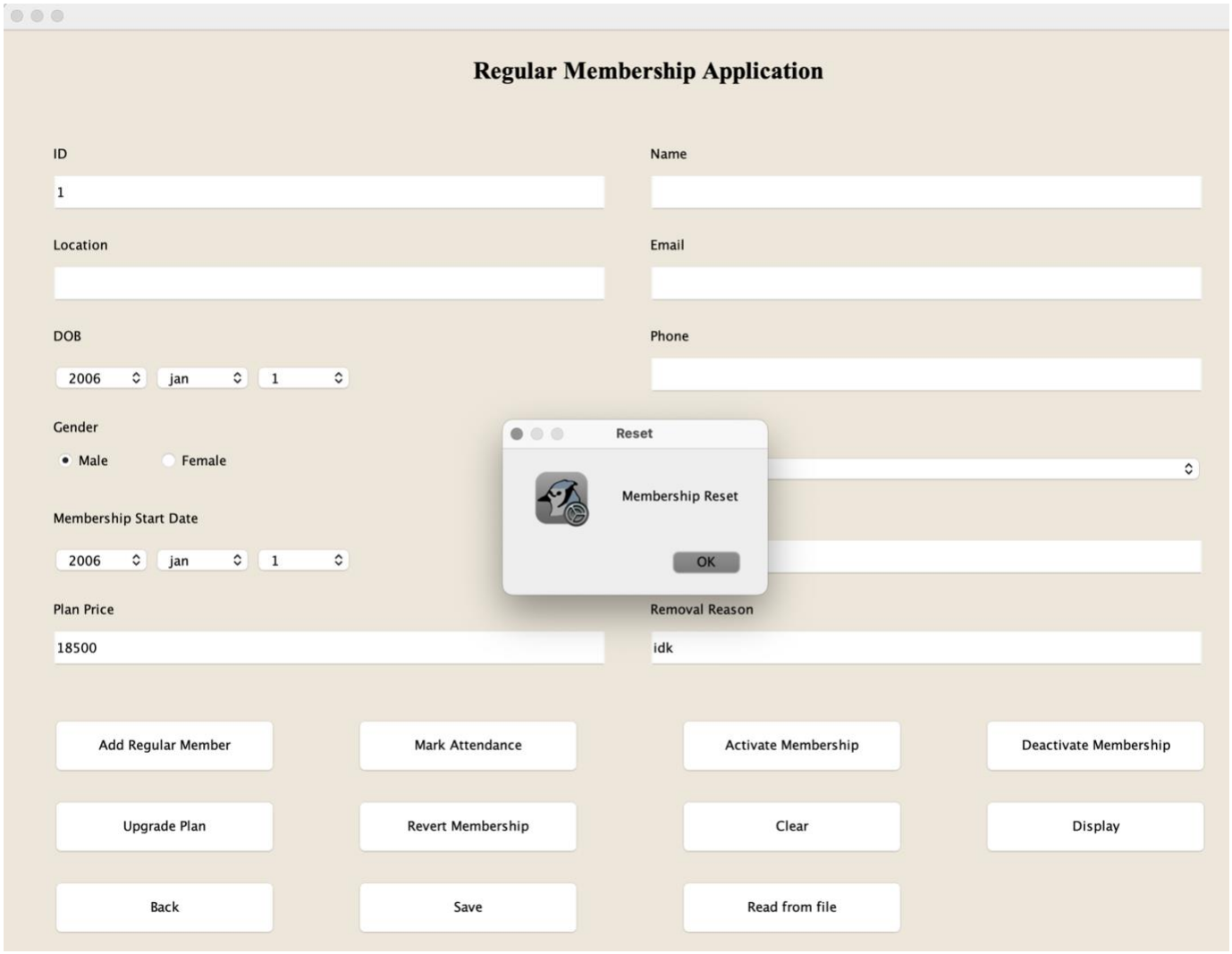
Objective	To revert the membership.
Action	Fill in the id, removal reason and click “revert membership” button.
Expected Output	A message dialog box stating that the membership is reverted should be displayed and the active status should be inactive, plan should be basic, attendance and loyalty points should be set to zero.
Actual Output	A message dialog box stating that the membership is reverted is displayed and the active status is inactive, plan is basic, attendance and loyalty points is set to zero.
Result	The test was successful.



REGULAR MEMBERS:

Member ID: 1
Name: a
Location: a
Email: a
Phone: a
Date of Birth: 1-jan-2006
Gender: male
Plan: deluxe
Price: 18500.0
Membership Start Date: 1-jan-2006
Attendance: 39
Loyalty Points: 195.0
Active Status: true
==========





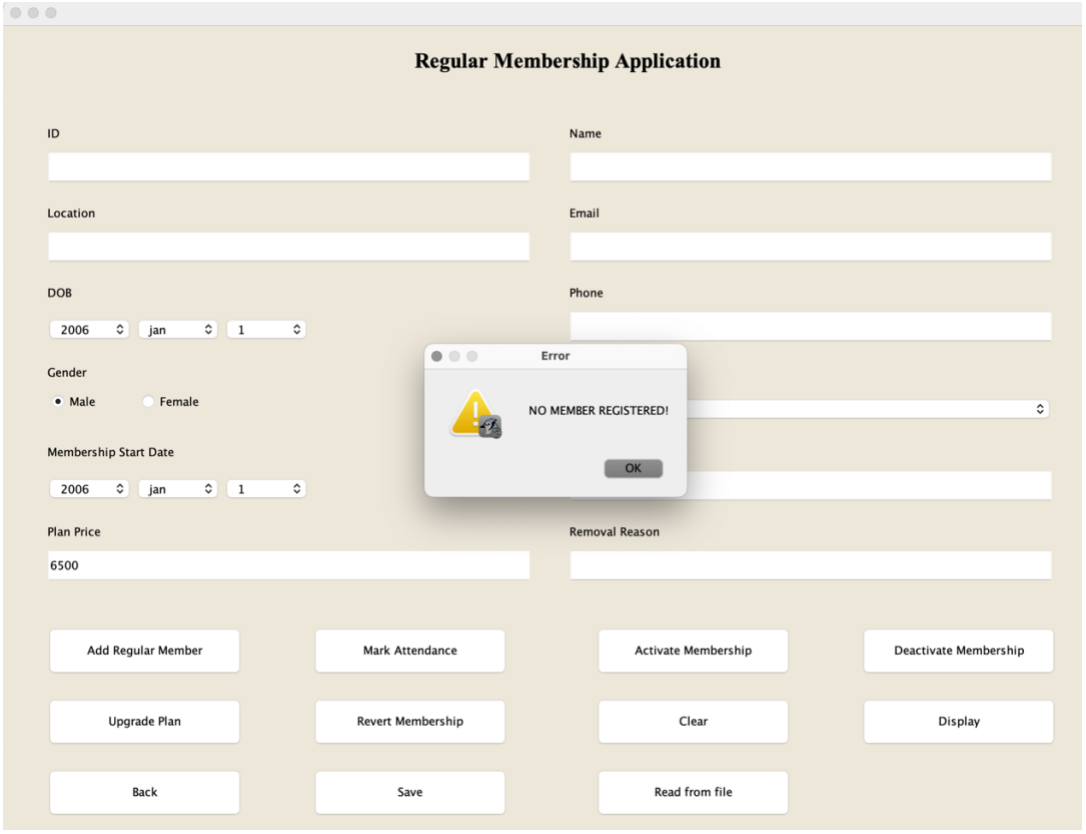
REGULAR MEMBERS:

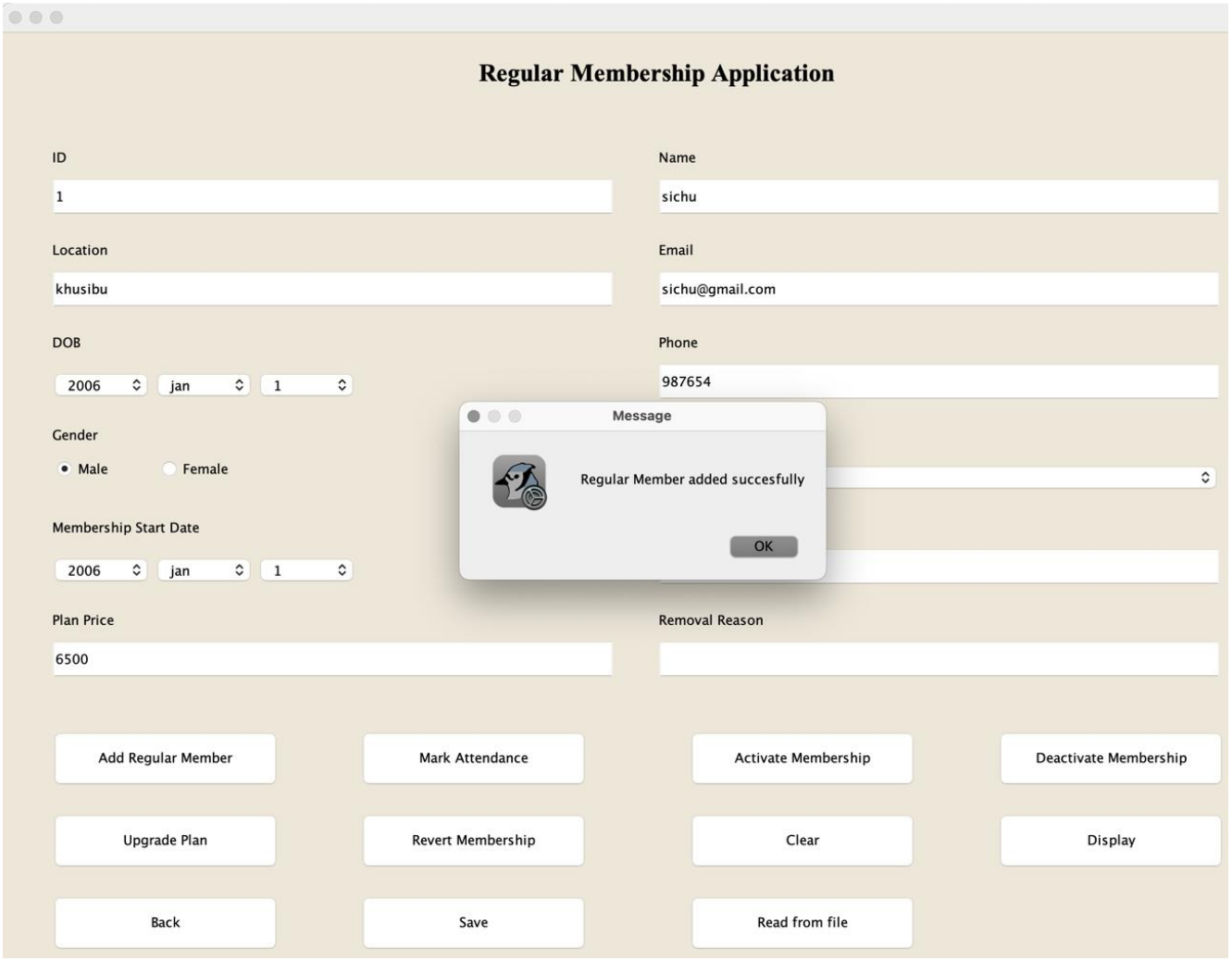
Member ID: 1
Name: a
Location: a
Email: a
Phone: a
Date of Birth: 1-jan-2006
Gender: male
Plan: basic
Price: 6500.0
Membership Start Date: 1-jan-2006
Attendance: 0
Loyalty Points: 0.0
Active Status: false
Removal Reason: idk
==========

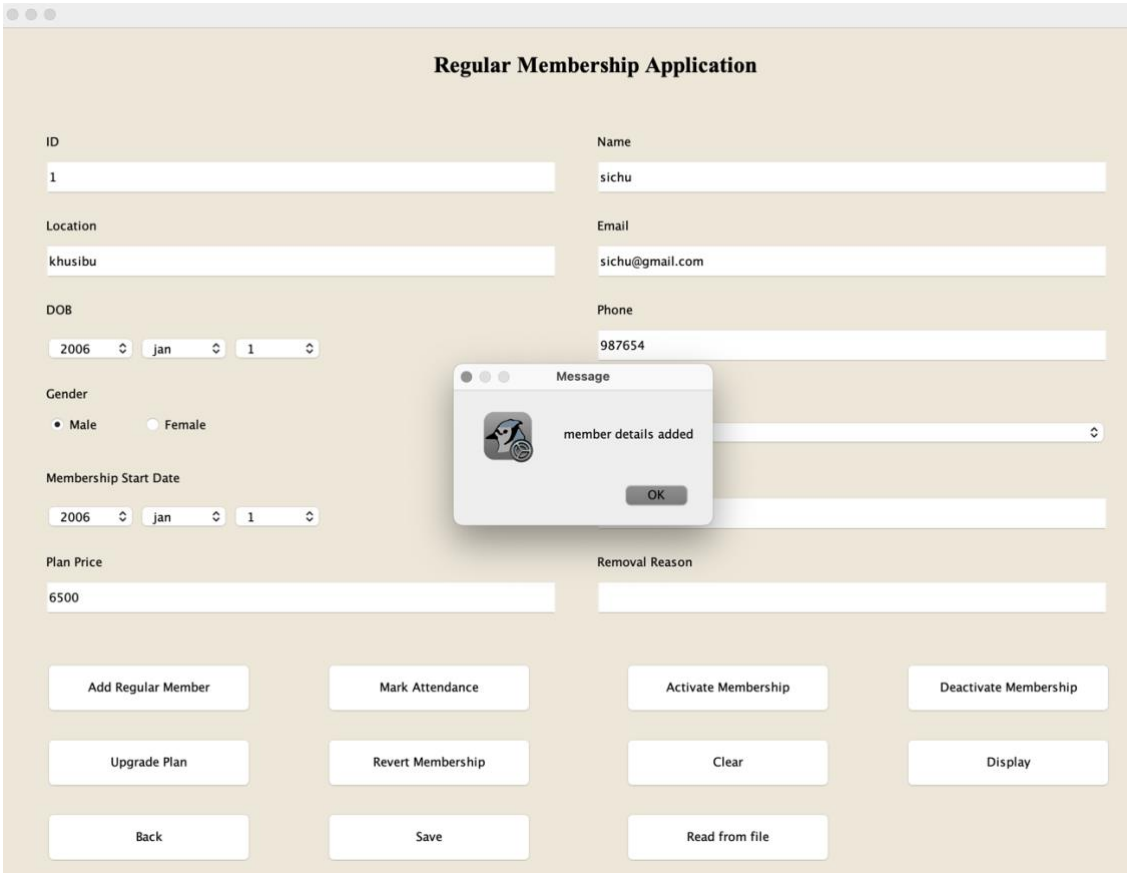
Test-5

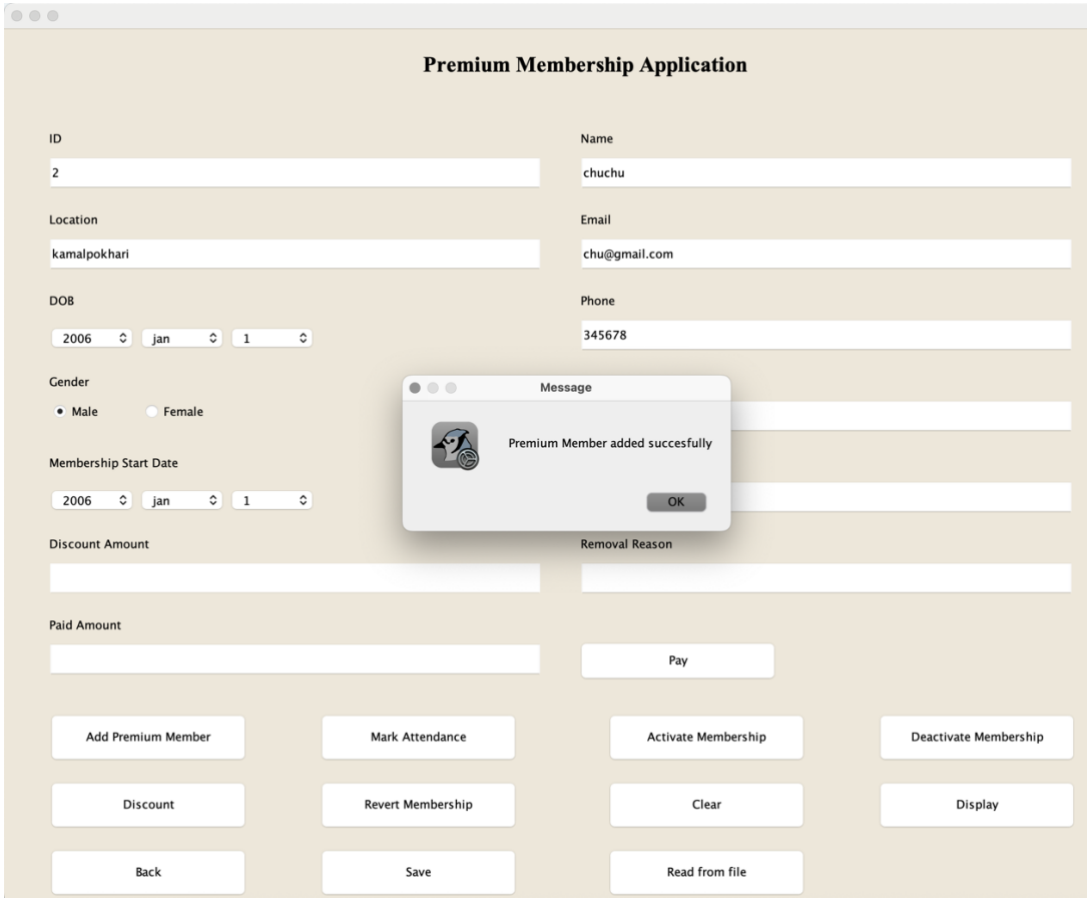
Table 9: test-5.1: save to file

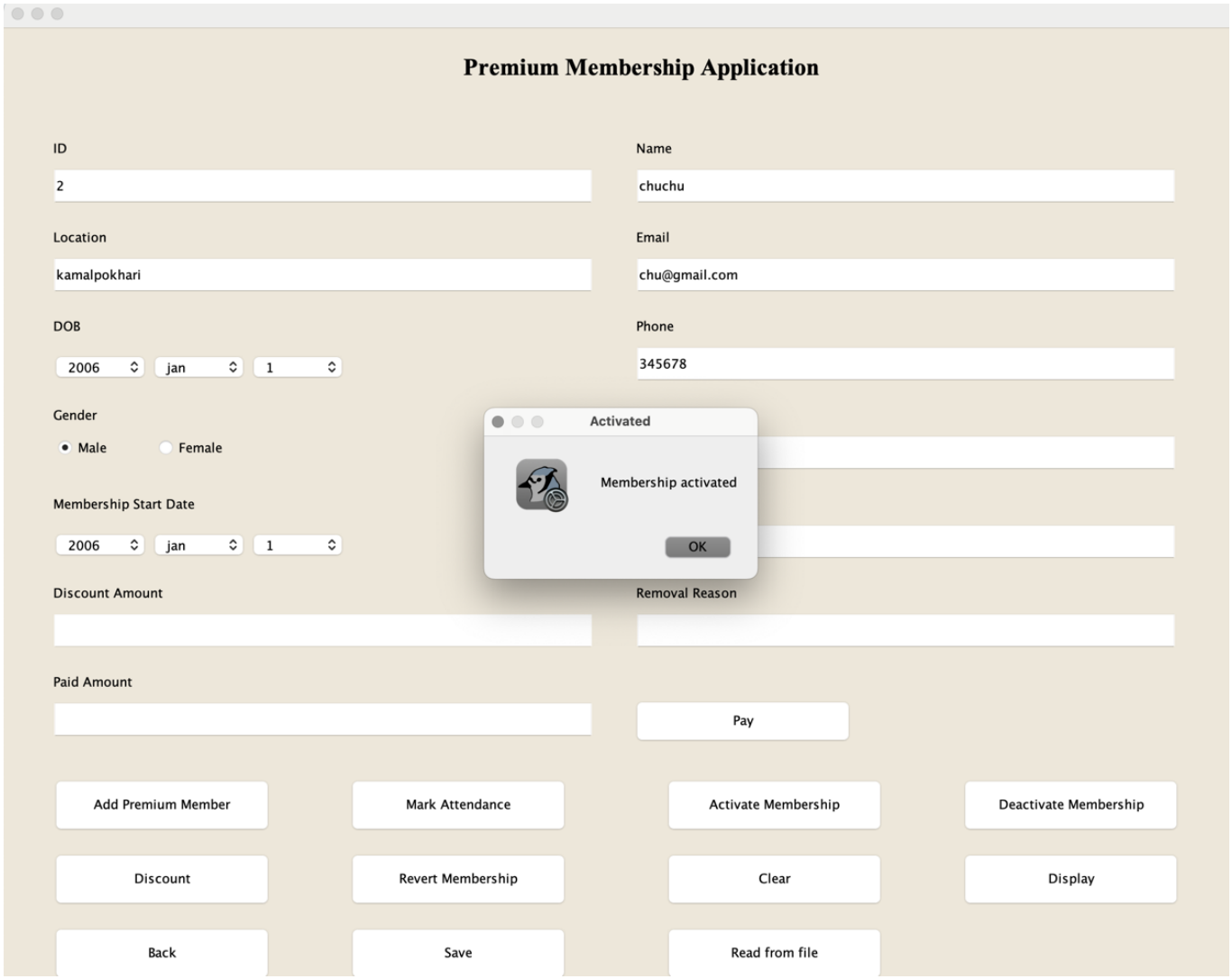
Objective	To save the file
Action	1. click “save” button without adding any member 2. fill in the application. click “add premium member” button. 3. do the same in regular member frame as well as mark attendance and activate membership. 4. click “save” button.
Expected Output	A message dialog stating the file is saved should be displayed along with a text file with the name “MemberDetails.txt” containing those information should be visible.
Actual Output	A message dialog stating the file is saved is displayed along with a text file with the name “MemberDetails.txt” containing those information is visible.
Result	The test was successful.

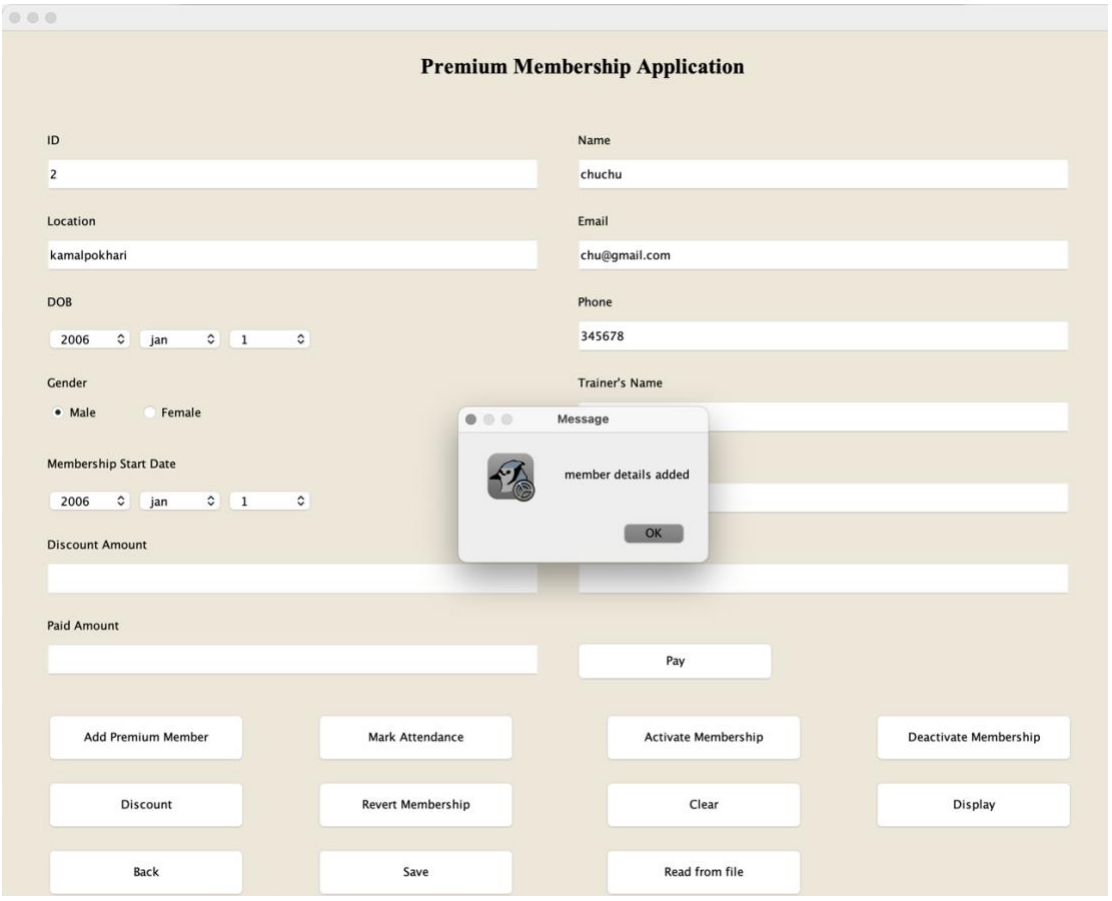


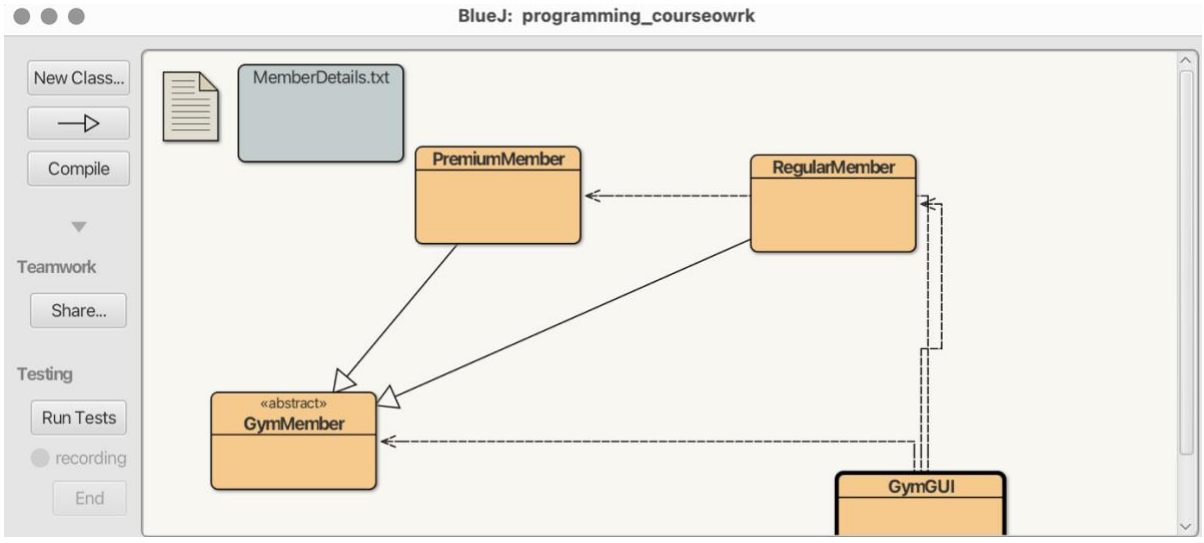

Regular Membership Application
ID
1
Name
sichu
Location
khusibu
Email
sichu@gmail.com
DOB
2006
jan
1
Phone
987654
Gender
Male
Female
Membership Start Date
2006
jan
1
Plan Price
6500
Removal Reason
Message
Regular Member added succesfully
OK
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file


Regular Membership Application
ID
1
Name
sichu
Location
khusibu
Email
sichu@gmail.com
DOB
2006
jan
1
Phone
987654
Message
member details added
OK
Gender
Male
Female
Membership Start Date
2006
jan
1
Plan Price
6500
Removal Reason
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file


Premium Membership Application
ID
2
Name
chuchu
Location
kamalpokhari
Email
chu@gmail.com
DOB
2006
jan
1
Phone
345678
Message
Premium Member added succesfully
OK
Gender
Male
Female
Membership Start Date
2006
jan
1
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file


Premium Membership Application
ID
2
Name
chuchu
Location
kamalpokhari
Email
chu@gmail.com
DOB
2006
jan
1
Phone
345678
Gender
Male
Female
Activated
Membership activated
OK
Membership Start Date
2006
jan
1
Discount Amount
Removal Reason
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file


Premium Membership Application
ID
2
Name
chuchu
Location
kamalpokhari
Email
chu@gmail.com
DOB
2006
jan
1
Phone
345678
Gender
Male
Female
Trainer's Name
Message
member details added
OK
Membership Start Date
2006
jan
1
Discount Amount
Paid Amount
Pay
Add Premium Member
Mark Attendance
Activate Membership
Deactivate Membership
Discount
Revert Membership
Clear
Display
Back
Save
Read from file


BlueJ: programming_courseowrk
New Class...
Compile
Teamwork
Share...
Testing
Run Tests
recording
End
MemberDetails.txt
PremiumMember
RegularMember
«abstract»
GymMember
GymGUI

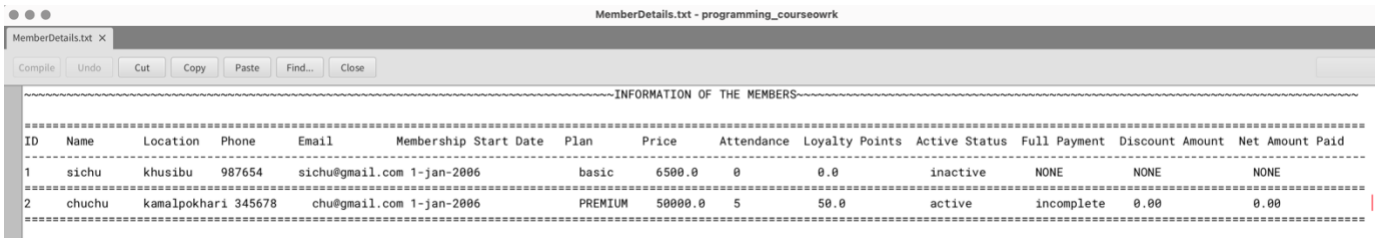
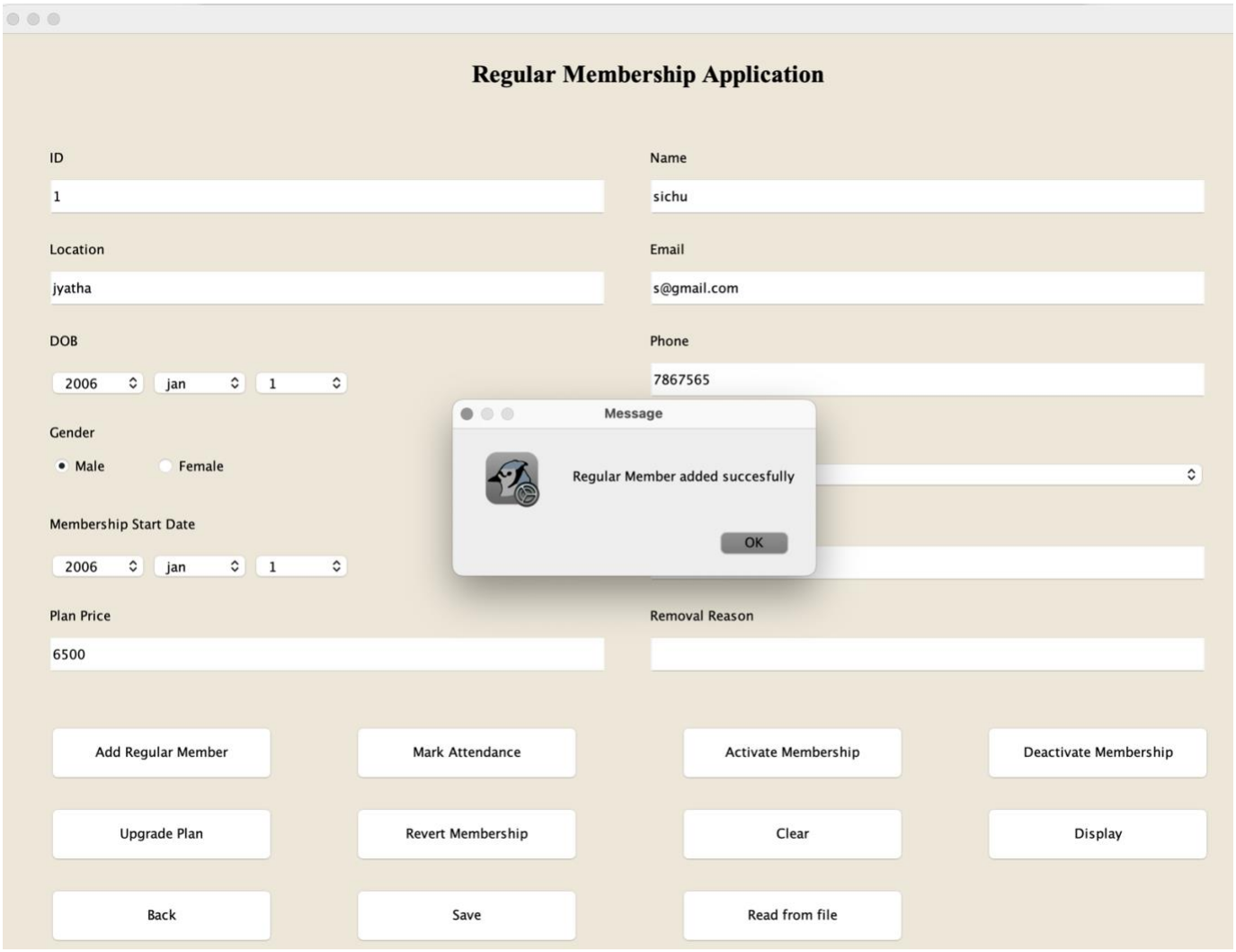
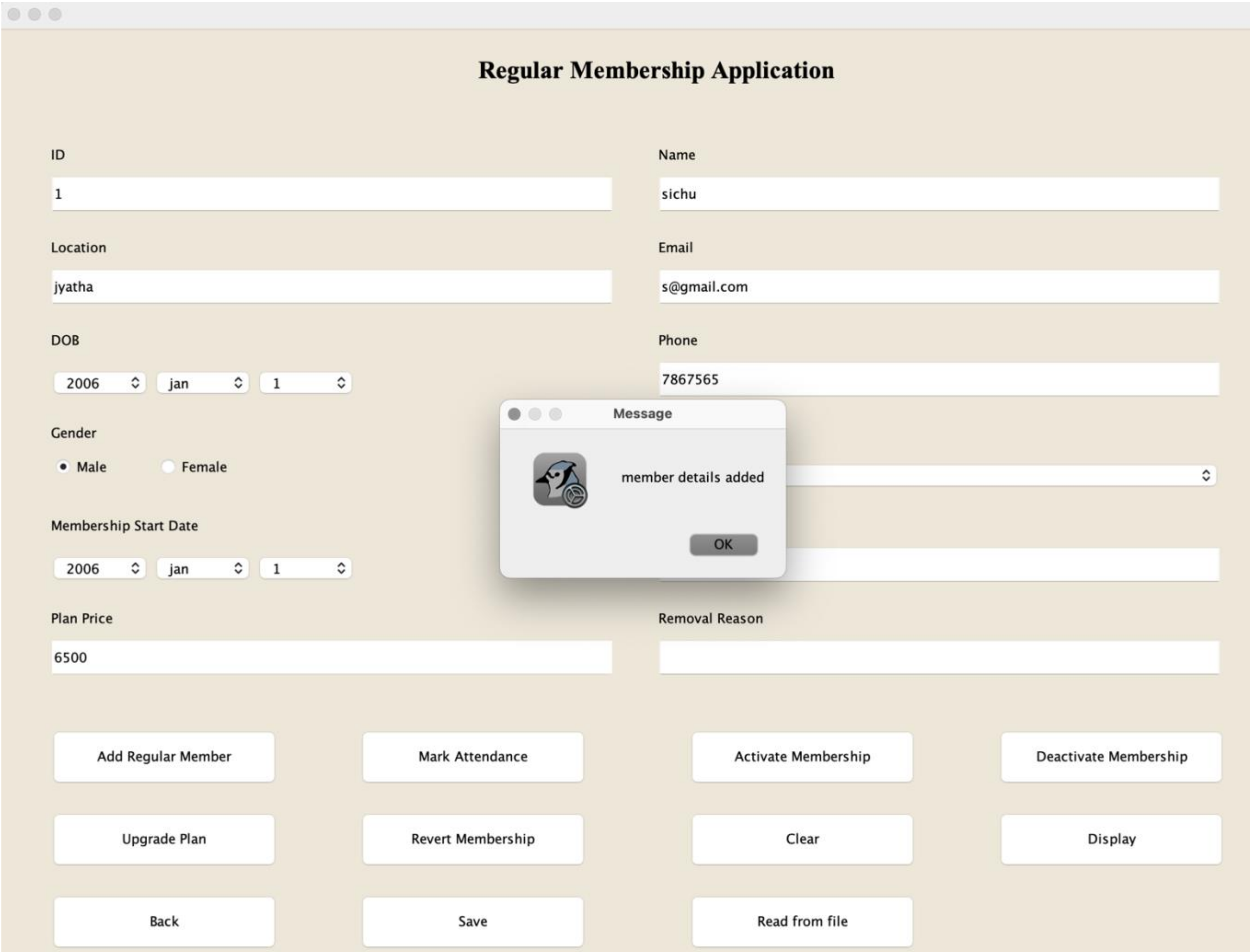
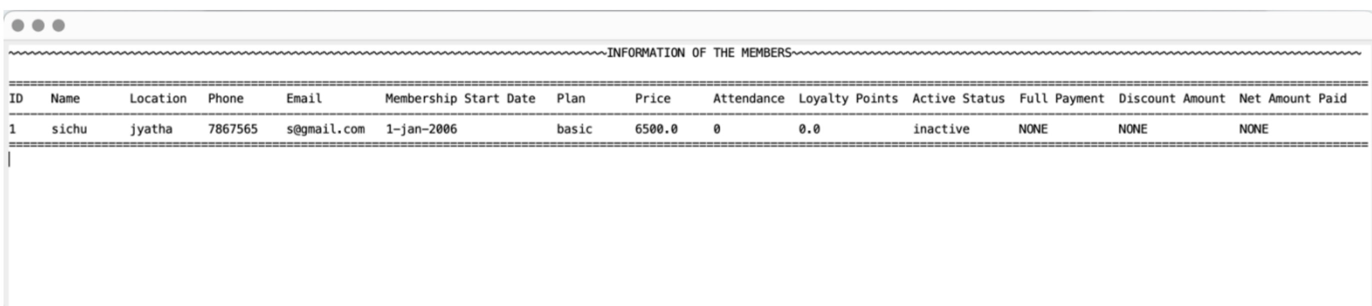

MemberDetails.txt - programming_courseowrk
MemberDetails.txt
Compile
Undo
Cut
Copy
Paste
Find...
Close
INFORMATION OF THE MEMBERS
ID Name Location Phone Email Membership Start Date Plan Price Attendance Loyalty Points Active Status Full Payment Discount Amount Net Amount Paid
1 sichu khusibu 987654 sichu@gmail.com 1-jan-2006 basic 6500.0 0 0.0 inactive NONE NONE NONE
2 chuchu kamalpokhari 345678 chu@gmail.com 1-jan-2006 PREMIUM 50000.0 5 50.0 active incomplete 0.00 0.00

Table 10: test-5.2: read from file

Objective	To read from file
Action	1. fill in the application. 2. save the application. 3. click “read from file” button.
Expected Output	A frame should be opened where the information are stored.
Actual Output	A frame is opened where the information are stored.
Result	The test was successful.




Regular Membership Application
ID
1
Name
sichu
Location
jyatha
Email
s@gmail.com
DOB
2006 jan 1
Phone
7867565
Gender
Male
Female
Membership Start Date
2006 jan 1
Plan Price
6500
Removal Reason
Message
member details added
OK
Add Regular Member
Mark Attendance
Activate Membership
Deactivate Membership
Upgrade Plan
Revert Membership
Clear
Display
Back
Save
Read from file


INFORMATION OF THE MEMBERS
ID Name Location Phone Email Membership Start Date Plan Price Attendance Loyalty Points Active Status Full Payment Discount Amount Net Amount Paid
1 sichu jyatha 7867565 s@gmail.com 1-jan-2006 basic 6500.0 0 0.0 inactive NONE NONE NONE

Error Detection and Correction

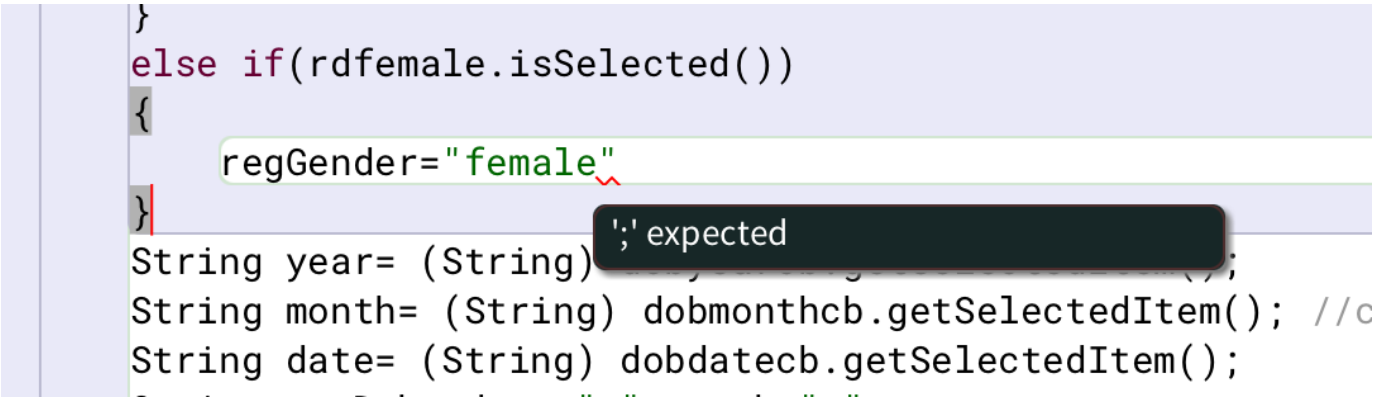
Errors in programming are generally bugs which interfere with the execution of the program. These can be of any type. Errors prevents the program from running as expected. Crucial errors can harm the program to great extent. Error detection is a feature built into many programming platforms. Some errors are detected during the execution, and some are detected while writing the codes itself. Error detection and correction are important steps of programming to ensure that the codes are running smoothly without any disruption from any errors that may occur. There are many types of error but the are mostly divided into 3 types, syntax, runtime and logical error.

Syntax Error

Programs have grammar the same way the language we spoke does. These grammar rules are known as syntax. It makes a errors in the language itself. It is detected during writing the program.

For example, missing a semicolon at the end or writing incorrect spelling to keywords can cause syntax error.

The main syntax error that occurred in this program was forgetting semicolon and forgetting to close a curly bracket. These errors were detected at compile time and corrected.



As you can see in this figure, a missing semicolon generates syntax error. These errors are detected as it BlueJ underlines them in red to make it easier to spot.

Runtime error

The errors which occur during the execution of the program is known as runtime errors. These errors are not detected during compile time. These errors executes the program however the output is not there. For example, dividing by zero, accessing index which is not used, etc. are some runtime errors. These errors often causes the program to crash. These errors also include number format exception, file not found exception which were handled in the program.

```
catch(FileNotFoundException e)
{
    JOptionPane.showMessageDialog(regularframe,"FILE NOT FOUND.","Error",JOptionPane.ERROR_MESSAGE);
}
catch(IOException e)
{
    JOptionPane.showMessageDialog(regularframe,"FILE NOT READING"+e.getMessage(),"Error",JOptionPane.ERROR_MESSAGE);
}
```

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

In this figure, the runtime errors are being handled to make sure the program doesn't end abruptly. The file not found exception catches when the file isn't found. The IO exception occurs when the program can't read the file. The number format exception occurs due to entering alphabetical value for integer id.

Logical error

The error which runs the program smoothly but will provide unexpected output is known as logical errors. These errors are mainly due to human mistakes. For example, using * instead of + or running an infinite loop. These kinds of errors are hardest to find.

```
public void calculateDiscount()
{
    if(this.isFullPayment)
    {
        this.discountAmount=(10/100)*premiumCharge;
        System.out.println("discount is calculated. discount amount= "+this.discountAmount);
    }
    else
    {
        System.out.println("Discount is not available");
    }
}
```

As you can see in this code, nothing seems to be the problem. There is no error during compile nor during runtime. However, this will give the discount amount 0.0 which is not our expected output. As the variable discountAmount is in double data type, atleast one part of that equation should be double.

```
public void calculateDiscount()
{
    if(this.isFullPayment)
    {
        this.discountAmount=(10.0/100)*premiumCharge;
        System.out.println("discount is calculated. discount amount= "+this.discountAmount);
    }
    else
    {
        System.out.println("Discount is not available");
    }
}
```

In this figure, we can see how the logical error was handled. By adding .0 in the equation, we get the desired output.

Errors are bugs which makes our program stop abruptly or doesn't let it execute at all. These errors should be handled inorder to run the program smoothly.

Conclusion

This coursework utilizes the knowledge and lessons taught on java to develop a gym system which lets the user add regular member and premium member as per the user's want. When adding the member, it requires few information like id, name, location, phone, email and many more. This system even saves the information to a text file and reads from that file too. This system does various works. We can activate and deactivate the membership as per our want. We can mark attendance. We can be eligible for upgrading to any other plan if your attendance exceeds 30.

While doing this coursework, I learned any new thing about java program. I learned topics which weren't discussed in class. I got to know that there are many things which I don't know about. This coursework presented an opportunity to leaned and use the java codes and know how to operate through them. It made me know just how much I am yet to learn. I learned the uses of object-oriented language's concept, polymorphism, abstraction, inheritance and encapsulation.

Many difficulties were faced during the completion of this program. The difficulty ranged from not understanding the logic to few things to handling mistakes generated through various programming errors. Sometimes syntax was typed wrong and sometimes object name was different. During the start, understanding the function of few methods was troublesome. Trying to understand how and where to place few codes were tiring. As progressing through the coursework, the issues were raised from forgetting to add addActionListener to the button to the button not working even if all steps were done correctly due to frame's visibility being set to false.

These difficulties were overcomed by asking teachers or discussing with friends. These difficulties were solved by simply adding actionlistener or putting .0 behind a number during calculation. Any difficulties faced during the completion of this program was handled. The program also handles several exceptions which were thrown. It may not handle every known exception, but it handles most of them which might occur in this system.

Bibliography

Anon., n.d. *about bluej.* [Online]
Available at: https://www.bluej.org/about.html

Anon., n.d. *draw.io.* [Online]
Available at: https://www.drawio.com/?utm_source=chatgpt.com

Anon., n.d. *figma.* [Online]
Available at: https://www.figma.com/about/

Anon., n.d. *introduction to BlueJ.* [Online]
Available at: https://www.geeksforgeeks.org/introduction-of-bluej/

College, I., n.d. s.l.: s.n.

Appendix (Code Listing)

```
public abstract class GymMember
{
    protected int id;
    protected int attendance;
    protected String name;
    protected String location;
    protected String phone;
    protected String email;
    protected String gender;
    protected String DOB;
    protected String membershipStartDate;
    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)
    {
        //assigning
        this.id=id;
        this.name=name;
        this.location=location;
        this.phone=phone;
        this.email=email;
        this.gender=gender;
        this.DOB=DOB;
        this.membershipStartDate=membershipStartDate;

        //initializing
        this.attendance=0;
        this.loyaltyPoints=0;
        this.activeStatus=false;
```

```
}

//accessor methods
public int getId()
{
   return this.id;
}
public int getAttendance()
{
   return this.attendance;
}
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 double getLoyaltyPoints()
{
    return this.loyaltyPoints;
}
public boolean getActiveStatus()
{
    return this.activeStatus;
}

public abstract void markAttendance();

public void activateMembership()
{
    activeStatus=true;
}

public void deactivateMembership()
{
    if(activeStatus)//checking if its active or not
    {
        activeStatus=false;
    }
    else
    {
        System.out.println("It seems like active status is already inactive");
    }
}
```

```
    public void resetMembership()
    {
        activeStatus=false;
        attendance=0;
        loyaltyPoints=0;
    }
    public void display()
    {
        System.out.println("Id: "+id);
        System.out.println("Name: "+name);
        System.out.println("Location: "+location);
        System.out.println("Phone: "+phone);
        System.out.println("Email: "+email);
        System.out.println("Gender: "+gender);
        System.out.println("DOB: "+DOB);
        System.out.println("Membership Start Date: "+membershipStartDate);
        System.out.println("Attendance: "+attendance);
        System.out.println("Loyalty points: "+loyaltyPoints);
        System.out.println("Active Status: "+(activeStatus?"active":"inactive"));
    }
}

public class RegularMember extends GymMember
{
    private final int attendanceLimit; //final keyword is used so that the value cannot be changed
    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)
{
   //calling parent class constructor
   super(id, name, location, phone, email, gender, DOB, membershipStartDate);

   //initializing
   this.isEligibleForUpgrade=false;
   this.attendanceLimit=30;
   this.plan="basic";
   this.price=6500;
   this.removalReason="";
   this.referralSource=referralSource;
}

//getter 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()
{
    attendance++;
    loyaltyPoints+=5;
    if(getAttendance()>=attendanceLimit)
    {
        isEligibleForUpgrade=true;
    }
}

public double getPlanPrice(String plan)
{
    double x;
    switch(plan)
    {
        case "basic":
            return 6500;
        case "standard":
            return 12500;
        case "deluxe":
            return 18500;
        default:
            return -1;
    } /*no break statement because it directly returns
    in java once return is executed, it skips the whole block*/
```

```
}

public String upgradePlan(String plan)
{
   if (isEligibleForUpgrade!=true)//check eligibility
   {
      return "the member is not eligible for upgrade";
   }
   else//if member is eligible
   {
      if(this.plan==plan)//checks for same plan
      {
         return "the member has chosen the same plan";
      }
      else //if not same
      {
         if(getPlanPrice(plan)==-1)//check invalid plan
         {
            return "the member has entered invalid plan";
         }
         else //all requirement matches
         {
            this.plan=plan;
            this.price=getPlanPrice(plan);
            return ("plan upgraded to "+plan+"\n price= "+price);
         }
      }
   }
}

public void revertRegularMember(String removalReason)
{
   super.resetMembership();
```

```
        this.isEligibleForUpgrade=false;
        this.plan="basic";
        this.price=6500;
        this.removalReason=removalReason;
    }

    @Override
    public void display()
    {
        super.display();
        System.out.println("plan: "+plan);
        System.out.println("price: "+price);
        if(removalReason!="")
        {
            System.out.println("removal reason: "+removalReason);
        }
    }
}

public class PremiumMember extends GymMember
{
    final double premiumCharge;
    String personalTrainer;
    boolean isFullPayment;
    double paidAmount;
    double discountAmount;
    public PremiumMember(int id, String name, String location, String phone, String
    email, String gender, String DOB, String membershipStartDate, String personalTrainer)
    {
        //calling parent class constructor
        super(id,name,location,phone,email,gender,DOB,membershipStartDate);

        //initializing
```

```
    this.premiumCharge=50000;
    this.paidAmount=0;
    this.isFullPayment=false;
    this.discountAmount=0;
    this.personalTrainer=personalTrainer;
}

//accessor/getter 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;
}

public String payDueAmount(double paidAmount)
{
    if(this.paidAmount==premiumCharge)
    {
```

```
        this.isFullPayment=true;
    }
    if(this.isFullPayment)
    {
        System.out.println("full payment has been done");
        return "the member has already paid full amount";
    }
    else
    {
        this.paidAmount+=paidAmount;
        if(this.paidAmount>premiumCharge)
        {
            this.paidAmount-=paidAmount;//get back the paid amount before it was excceded
            System.out.println("the member has paid more than premiumCharge.");
            return "please pay the correct amount";
        }
        else
        {
        System.out.println("payment was successful");
        return "remaining Amount="+(premiumCharge-this.paidAmount);
        }
    }
}
public void calculateDiscount()
{
    if(this.isFullPayment)
    {
        this.discountAmount=(10.0/100)*premiumCharge;
        System.out.println("discount is calculated. discount amount= "+this.discountAmount);
    }
    else
    {
        System.out.println("Discount is not available");
```

```
    }
}

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

@Override
public void markAttendance()
{
    attendance++;
    loyaltyPoints+=10;
}

@Override
public void display()
{
    super.display();
    System.out.println("personal trainer= "+personalTrainer);
    System.out.println("paid amount= "+paidAmount);
    System.out.println("is the amount fully paid= "+(isFullPayment?"yes":"no"));
    if(this.isFullPayment)
    {
        System.out.println("discount amount= "+discountAmount);
    }
    else if(this.paidAmount<premiumCharge)
    {
        System.out.println("remaining Amount="+(premiumCharge-this.paidAmount));
```

```
        }
    }
}

import java.util.ArrayList;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JTextArea;
import javax.swing.JComboBox;
import javax.swing.JRadioButton;
import javax.swing.JButton;
import javax.swing.ButtonGroup;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JOptionPane;
import java.awt.Font;
import java.awt.Color;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.FileReader;
import java.io.FileNotFoundException;

//import javax.swing.JSeparator;
public class GymGUI implements ActionListener
{
    ArrayList<GymMember> al1 = new ArrayList<GymMember>();
    public static void main(String[]args)
    {
        new GymGUI();
    }
    JFrame option, regularframe, premiumframe,displayframe,readframe;
```

```
JTextArea displayta,readta;
JLabel labelid, labelname, labellocation, labeldob, labelemail,
labelphone, labelgender, labelplan, labelmembershipsd,
labelreferral, labelremoval,labeltrainer, labelprice,
title1,regular,premium;
JTextField tfid,tflocation,tfemail, tfprice, tfname, tfphone, tfreferral,tfremoval;
JComboBox dobyearcb,dobmonthcb,dobdatecb, plancb, msyearcb,msmonthcb, msdaycb;
JRadioButton rdmale, rdfemale;
ButtonGroup grp;
JButton pre,reg,back1,back2;
JButton addregular, activate, deactivate, attendance,revert, clear, display, upgradePlan, save, read;

JLabel plabelid, plabelname, plabellocation, plabeldob, plabelemail,
plabelphone, plabelgender, plabelplan, plabelmembershipsd,
plabelreferral, plabelpaid, plabelremoval,plabeltrainer, plabeldiscount;
JTextField ptfid,ptflocation,ptfemail, ptfpaid, ptfname, ptfphone, ptfreferral,ptfremoval, ptfdiscount,
ptftrainer;
JComboBox pdobyearcb,pdobmonthcb,pdobdatecb, pplancb, pmsyearcb,pmsmonthcb, pmsdaycb;
JRadioButton prdmale, prdfemale;
ButtonGroup bg;

JButton addpremium, pactivate, pdeactivate, pattendance,prevert, pclear,
pdisplay,calculateDiscount,pay,psave,pread;
public GymGUI()
{
  //option gui
 option = new JFrame("");
 title1 = new JLabel("Apply for:");
 title1.setForeground(Color.WHITE);
 title1.setFont(new Font("Times New Roman",Font.BOLD,20));
 reg = new JButton("Regular Membership");
 pre = new JButton("Premium Membership");
 option.getContentPane().setBackground(Color.decode("#1c382c"));
```

```
title1.setBounds(110,5,120,30);
reg.setBounds(50,50,200,60);
pre.setBounds(50,120,200,60);



reg.setBackground(Color.decode("#f0e6d3"));
pre.setBackground(Color.decode("#f0e6d3"));
reg.setOpaque(true);
reg.setBorderPainted(false);
pre.setOpaque(true);//false=no bg color
pre.setBorderPainted(false);//true=no custom color

option.add(title1);
option.add(reg);
option.add(pre);



option.setLayout(null);
option.setVisible(true);
option.setSize(300,250);

//application member gui
regularframe = new JFrame("");
regularframe.getContentPane().setBackground(Color.decode("#ede7da"));

regular = new JLabel("Regular Membership Application");
regular.setFont(new Font("Times New Roman",Font.BOLD,24));
labelid = new JLabel("ID");
labelname = new JLabel("Name");
labellocation = new JLabel("Location");
labeldob = new JLabel("DOB");
labelemail = new JLabel("Email");
```

```
labelphone = new JLabel("Phone");
labelgender = new JLabel("Gender");
labelplan = new JLabel("Plan");
labelmembershipsd = new JLabel("Membership Start Date");
labelreferral = new JLabel("Referral Source");

labelremoval = new JLabel("Removal Reason");
labeltrainer = new JLabel("Trainer's Name");
labelprice = new JLabel("Plan Price");



tfid = new JTextField();
tflocation = new JTextField();
tfemail = new JTextField();

tfprice = new JTextField("6500");
tfname = new JTextField();
tfphone = new JTextField();
tfreferral = new JTextField();
tfremoval = new JTextField();



String y[]={"2006","2007","2008","2009","2010","2011","2012","2013","2014","2015","2016"};
String m[]={"jan","feb","mar","apr","may","jun","jul","aug",
"sept","oct","nov","dec"};
String d[]={"1","2","3","4","5","6","7","8","9","10","11","12",
"13","14","15","16","17","18","19","20","21","22","23","24",
"25","26","27","28","29","30","31"};
String p[]= {"basic","standard","deluxe"};
dobyearcb = new JComboBox(y);
dobmonthcb = new JComboBox(m);
```

```
dobdatecb = new JComboBox(d);
plancb = new JComboBox(p);
msyearcb = new JComboBox(y);
msmonthcb = new JComboBox(m);
msdaycb = new JComboBox(d);

rdmale = new JRadioButton("Male",true);
rdfemale = new JRadioButton("Female");
grp = new ButtonGroup();

addregular = new JButton("Add Regular Member");
addpremium = new JButton("Add Premium Member");
activate  = new JButton("Activate Membership");
deactivate = new JButton("Deactivate Membership");
attendance = new JButton("Mark Attendance");
revert = new JButton("Revert Membership");
clear = new JButton("Clear");
display = new JButton("Display");
back1 = new JButton("Back");
upgradePlan = new JButton("Upgrade Plan");
save = new JButton("Save");
read = new JButton("Read from file");

tfprice.setEditable(false);



regular.setBounds(465,15,400,50);



labelid.setBounds(50,110,100,25);
labellocation.setBounds(50,200,100,25);
labeldob.setBounds(50,290,100,25);
labelgender.setBounds(50,380,100,25);
```

```
labelmembershipsd.setBounds(50,470,150,25);
labelprice.setBounds(50,560,100,25);



labelname.setBounds(640,110,100,25);
labelemail.setBounds(640,200,100,25);
labelphone.setBounds(640,290,100,25);
labelplan.setBounds(640,380,100,25);
labelreferral.setBounds(640,470,100,25);
labelremoval.setBounds(640,560,150,25);



addregular.setBounds(50,680,220,55);
attendance.setBounds(350,680,220,55);
activate.setBounds(670,680,220,55);
deactivate.setBounds(970,680,220,55);
revert.setBounds(350,760,220,55);
clear.setBounds(670,760,220,55);
display.setBounds(970,760,220,55);
upgradePlan.setBounds(50,760,220,55);
back1.setBounds(50,840,220,55);
save.setBounds(350,840,220,55);
read.setBounds(670,840,220,55);

tfid.setBounds(48,140,550,40);
tflocation.setBounds(48,230,550,40);
tfemail.setBounds(638,230,550,40);
tfprice.setBounds(48,590,550,40);



tfname.setBounds(638,140,550,40);
tfphone.setBounds(638,320,550,40);
```

```
tfreferral.setBounds(638,500,550,40);
tfremoval.setBounds(638,590,550,40);


rdmale.setBounds(48,410,100,30);
rdfemale.setBounds(150,410,100,30);

plancb.setBounds(638,400,550,70);
dobyearcb.setBounds(48,320,100,50);
dobmonthcb.setBounds(148,320,100,50);
dobdatecb.setBounds(248,320,100,50);
msyearcb.setBounds(48,500,100,50);
msmonthcb.setBounds(148,500,100,50);
msdaycb.setBounds(248,500,100,50);



regularframe.add(regular);
regularframe.add(labelid);
regularframe.add(labelname);
regularframe.add(labellocation);
regularframe.add(labelemail);
regularframe.add(labelphone);
regularframe.add(labelgender);
regularframe.add(labelplan);
regularframe.add(labeldob);
regularframe.add(labelmembershipsd);
regularframe.add(labelreferral);

regularframe.add(labelremoval);

regularframe.add(labelprice);
```

```
regularframe.add(tfid);
regularframe.add(tflocation);
regularframe.add(tfemail);

regularframe.add(tfprice);
regularframe.add(tfname);
regularframe.add(tfphone);
regularframe.add(tfreferral);
regularframe.add(tfremoval);



regularframe.add(rdmale);
regularframe.add(rdfemale);
regularframe.add(dobyearcb);
regularframe.add(dobmonthcb);
regularframe.add(dobdatecb);
regularframe.add(plancb);
regularframe.add(msyearcb);
regularframe.add(msmonthcb);
regularframe.add(msdaycb);
regularframe.add(addregular);
regularframe.add(attendance);
regularframe.add(activate);
regularframe.add(deactivate);
regularframe.add(clear);
regularframe.add(display);
regularframe.add(revert);
regularframe.add(back1);
regularframe.add(upgradePlan);
regularframe.add(save);
regularframe.add(read);
grp.add(rdmale);
grp.add(rdfemale);
```

```
regularframe.setLayout(null);
regularframe.setVisible(false);
regularframe.setSize(1250,950);

//premium frame

premiumframe = new JFrame("");
premiumframe.getContentPane().setBackground(Color.decode("#ede7da"));

premium = new JLabel("Premium Membership Application");
premium.setFont(new Font("Times New Roman",Font.BOLD,24));
plabelid = new JLabel("ID");
plabelname = new JLabel("Name");
plabellocation = new JLabel("Location");
plabeldob = new JLabel("DOB");
plabelemail = new JLabel("Email");
plabelphone = new JLabel("Phone");
plabelgender = new JLabel("Gender");
plabelplan = new JLabel("Plan");
plabelmembershipsd = new JLabel("Membership Start Date");
plabelreferral = new JLabel("Referral Source");
plabelpaid = new JLabel("Paid Amount");
plabelremoval = new JLabel("Removal Reason");
plabeltrainer = new JLabel("Trainer's Name");

plabeldiscount = new JLabel("Discount Amount");

ptfid = new JTextField();
ptflocation = new JTextField();
ptfemail = new JTextField();
ptfpaid = new JTextField();
```

```
ptfname = new JTextField();
ptfphone = new JTextField();
ptfreferral = new JTextField();
ptfremoval = new JTextField();
ptfdiscount = new JTextField();
ptftrainer = new JTextField();

pdobyearcb = new JComboBox(y);
pdobmonthcb = new JComboBox(m);
pdobdatecb = new JComboBox(d);

pmsyearcb = new JComboBox(y);
pmsmonthcb = new JComboBox(m);
pmsdaycb = new JComboBox(d);

prdmale = new JRadioButton("Male",true);
prdfemale = new JRadioButton("Female");
bg = new ButtonGroup();


addpremium = new JButton("Add Premium Member");
pactivate  = new JButton("Activate Membership");
pdeactivate = new JButton("Deactivate Membership");
pattendance = new JButton("Mark Attendance");
prevert = new JButton("Revert Membership");
pclear = new JButton("Clear");
pdisplay = new JButton("Display");
back2 = new JButton("Back");
psave = new JButton("Save");
pread = new JButton("Read from file");
calculateDiscount = new JButton("Discount");
pay = new JButton("Pay");
```

```
ptfdiscount.setEditable(false);

premium.setBounds(465,15,500,50);
plabelid.setBounds(50,110,100,25);
plabellocation.setBounds(50,200,100,25);
plabeldob.setBounds(50,290,100,25);
plabelgender.setBounds(50,380,100,25);
plabelmembershipsd.setBounds(50,470,150,25);
plabeldiscount.setBounds(50,560,150,25);

plabelpaid.setBounds(50,650,100,25);

plabelname.setBounds(640,110,100,25);
plabelemail.setBounds(640,200,100,25);
plabelphone.setBounds(640,290,100,25);
labeltrainer.setBounds(640,380,100,25);
plabelreferral.setBounds(640,470,100,25);
plabelremoval.setBounds(640,560,150,25);

pay.setBounds(638,680,220,45);
addpremium.setBounds(50,760,220,55);
pattendance.setBounds(350,760,220,55);
pactivate.setBounds(670,760,220,55);
pdeactivate.setBounds(970,760,220,55);
prevert.setBounds(350,835,220,55);
pclear.setBounds(670,835,220,55);
pdisplay.setBounds(970,835,220,55);
calculateDiscount.setBounds(50,835,220,55);
back2.setBounds(50,910,220,55);
psave.setBounds(350,910,220,55);
```

```
pread.setBounds(670,910,220,55);

ptfid.setBounds(48,140,550,40);
ptflocation.setBounds(48,230,550,40);
ptfemail.setBounds(638,230,550,40);
ptfdiscount.setBounds(48,590,550,40);
ptfpaid.setBounds(48,680,550,40);
ptfname.setBounds(638,140,550,40);
ptfphone.setBounds(638,320,550,40);
ptfreferral.setBounds(638,500,550,40);
ptfremoval.setBounds(638,590,550,40);
ptftrainer.setBounds(638,410,550,40);

prdmale.setBounds(48,410,100,30);
prdfemale.setBounds(150,410,100,30);


pdobyearcb.setBounds(48,320,100,50);
pdobmonthcb.setBounds(148,320,100,50);
pdobdatecb.setBounds(248,320,100,50);
pmsyearcb.setBounds(48,500,100,50);
pmsmonthcb.setBounds(148,500,100,50);
pmsdaycb.setBounds(248,500,100,50);


premiumframe.add(premium);
premiumframe.add(plabelid);
premiumframe.add(plabelname);
premiumframe.add(plabellocation);
premiumframe.add(plabelemail);
premiumframe.add(plabelgender);
premiumframe.add(plabeldob);
premiumframe.add(plabelphone);
```

```
premiumframe.add(plabelmembershipsd);
premiumframe.add(plabelreferral);
premiumframe.add(plabelpaid);
premiumframe.add(plabelremoval);
premiumframe.add(labeltrainer);

premiumframe.add(plabeldiscount);
premiumframe.add(ptfid);
premiumframe.add(ptflocation);
premiumframe.add(ptfemail);
premiumframe.add(ptfpaid);

premiumframe.add(ptfname);
premiumframe.add(ptfphone);
premiumframe.add(ptfreferral);
premiumframe.add(ptfremoval);
premiumframe.add(ptfdiscount);
premiumframe.add(ptftrainer);
premiumframe.add(prdmale);
premiumframe.add(prdfemale);
premiumframe.add(pdobyearcb);
premiumframe.add(pdobmonthcb);
premiumframe.add(pdobdatecb);

premiumframe.add(pmsyearcb);
premiumframe.add(pmsmonthcb);
premiumframe.add(pmsdaycb);
premiumframe.add(addpremium);
premiumframe.add(pattendance);
premiumframe.add(pactivate);
premiumframe.add(pdeactivate);
premiumframe.add(pclear);
```

```
premiumframe.add(pdisplay);
premiumframe.add(prevert);
premiumframe.add(back2);
premiumframe.add(calculateDiscount);
premiumframe.add(pay);
premiumframe.add(psave);
premiumframe.add(pread);
bg.add(prdmale);
bg.add(prdfemale);



reg.addActionListener(this);
pre.addActionListener(this);
addregular.addActionListener(this);
addpremium.addActionListener(this);
activate.addActionListener(this);
deactivate.addActionListener(this);
attendance.addActionListener(this);
revert.addActionListener(this);
clear.addActionListener(this);
display.addActionListener(this);
save.addActionListener(this);
read.addActionListener(this);
upgradePlan.addActionListener(this);
back1.addActionListener(this);
back2.addActionListener(this);

pactivate.addActionListener(this);
pdeactivate.addActionListener(this);
pattendance.addActionListener(this);
prevert.addActionListener(this);
```

```
  pclear.addActionListener(this);
  pdisplay.addActionListener(this);
  calculateDiscount.addActionListener(this);
  pay.addActionListener(this);
  psave.addActionListener(this);
  pread.addActionListener(this);

  plancb.addActionListener(this);

  premiumframe.setLayout(null);
  premiumframe.setVisible(false);
  premiumframe.setSize(1250,1000);
  //display frame
  displayframe = new JFrame();
  displayta = new JTextArea();
  displayta.setBounds(5,5,890,890);
  displayframe.add(displayta);
  displayframe.setLayout(null);
  displayframe.setVisible(false);
  displayframe.setSize(900,900);

  //read frame
  readframe = new JFrame();
  readta = new JTextArea();
  readta.setBounds(5,5,1340,290);
  readframe.add(readta);
  readta.setFont(new Font("Monospaced", Font.PLAIN, 12));
  //when the file is read, monospaced makes sure all letter, symbol take same amount of space
  //for format purpose
  readframe.setLayout(null);
  readframe.setVisible(false);
  readframe.setSize(1350,300);
}
```

```
@Override
public void actionPerformed(ActionEvent ae)
{

   if(ae.getSource()==reg)
   {
      option.setVisible(false);
      regularframe.setVisible(true);
      premiumframe.setVisible(false);
   }

 else if(ae.getSource()==plancb)
  {
     String s= (String)plancb.getSelectedItem();
     if(s=="Basic")
     {
        tfprice.setText("6500");
     }
     else if(s=="Standard")
     {
        tfprice.setText("12500");
     }
     else
     {
        tfprice.setText("18500");
     }

  }
   else if(ae.getSource()==pre)
   {
      option.setVisible(false);
      premiumframe.setVisible(true);
```

```
    regularframe.setVisible(false);
}
else if(ae.getSource()==back1)
{
    option.setVisible(true);
    regularframe.setVisible(false);
}
else if(ae.getSource()==back2)
{
    option.setVisible(true);
    premiumframe.setVisible(false);
}
else if(ae.getSource()==addregular)
{
    if(tfid.getText().isEmpty()|| tflocation.getText().isEmpty()|| tfemail.getText().isEmpty()||
    tfname.getText().isEmpty()||tfphone.getText().isEmpty()||tfreferral.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(regularframe,"please fill in the application");
    }
    else
    {
        try{
        int regId = Integer.parseInt(tfid.getText());
        String regLocation = tflocation.getText();
        String regEmail = tfemail.getText();
        String regName = tfname.getText();
        String regPhone = tfphone.getText();
        String regReferral = tfreferral.getText();
        String regGender="";
        String regPlan= (String) plancb.getSelectedItem();
        if(rdmale.isSelected())
        {
            regGender="male";
```

```
        }
        else if(rdfemale.isSelected())
        {
           regGender="female";
        }
        String year= (String) dobyearcb.getSelectedItem();
        String month= (String) dobmonthcb.getSelectedItem(); //combo box creates object and (String) converts it into string
        String date= (String) dobdatecb.getSelectedItem();
        String regDob= date+"-"+month+"-"+year;
        String myear= (String) msyearcb.getSelectedItem();
        String mmonth= (String) msmonthcb.getSelectedItem(); //combo box creates object and (String) converts it into string
        String mdate= (String) msdaycb.getSelectedItem();
        String regMembershipsd= mdate+"-"+mmonth+"-"+myear;

        if(al1.isEmpty())
        {
           //creating object
           RegularMember regobj = new RegularMember(regId,regName,regLocation,regPhone,regEmail,
           regGender,regDob, regMembershipsd, regReferral);
           al1.add(regobj);
           JOptionPane.showMessageDialog(regularframe,"Regular Member added succesfully");
        }
        else
        {
           boolean idexistence=false;
           for(GymMember mem : al1)
           {
              //not displaying message in it because otherwise it is add same id again
              if(mem.getId()==regId) //for each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa
```

```
                {
                    idexistence=true;
                    break;//loop bata baira niskinxa
                }
            }
            if(idexistence)
            {
                JOptionPane.showMessageDialog(regularframe,"Member with this id already exists. please choose another ID.","Duplication",JOptionPane.ERROR_MESSAGE);
            }
            else
            {
                RegularMember regobj = new RegularMember(regId,regName,regLocation,regPhone,regEmail,
                regGender,regDob, regMembershipsd, regReferral);
                al1.add(regobj);
                JOptionPane.showMessageDialog(regularframe,"Regular Member added succesfully");
            }
          }
        }
        catch(NumberFormatException ex)
        {
          JOptionPane.showMessageDialog(regularframe,"ID can only have number values!","Error",JOptionPane.WARNING_MESSAGE);
        }
      }
      }

      else if(ae.getSource()==attendance)
      {
        if(tfid.getText().isEmpty())
        {
          JOptionPane.showMessageDialog(regularframe,"please fill in the id");
```

```
        }
        else
        {
           try{
           int regId=Integer.parseInt(tfid.getText());
           boolean idexistence=false;
           GymMember memberexistence = null;
              for(GymMember member : al1)
              {

                 if(member.getId()==regId)
                 {
                    idexistence=true;
                    memberexistence  = member;
                    break;
                 }
              }
           if(idexistence)
           {
              if(memberexistence.getActiveStatus()){
                 memberexistence.markAttendance();
                 JOptionPane.showMessageDialog(regularframe,"Attendance
marked","Attendance",JOptionPane.INFORMATION_MESSAGE);
              }
              else
              {
                 JOptionPane.showMessageDialog(regularframe,"member is not
activate","Attendance",JOptionPane.ERROR_MESSAGE);
              }
           }
           else
           {
```

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

      else if(ae.getSource()==activate)
      {
        if(tfid.getText().isEmpty())
        {
          JOptionPane.showMessageDialog(regularframe,"please fill in the id");
        }
        else
        {
          try{
          int regId=Integer.parseInt(tfid.getText());
          boolean idexistence=false;
          GymMember memberexistence = null;
            for(GymMember member : al1)
            {
              if(member.getId()==regId)
              {
                idexistence=true;
                memberexistence  = member;
                break;
              }
            }
```

```
            if(idexistence)
            {
               if(memberexistence.getActiveStatus()){
                  JOptionPane.showMessageDialog(regularframe,"Membership is already
activated","Activated",JOptionPane.INFORMATION_MESSAGE);
               }
               else
               {
                  memberexistence.activateMembership();//calling method to activate membership
                  JOptionPane.showMessageDialog(regularframe,"Membership
activated","Activated",JOptionPane.INFORMATION_MESSAGE);

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

      }

      else if(ae.getSource()==deactivate)
      {
         if(tfid.getText().isEmpty())
         {
```

```
            JOptionPane.showMessageDialog(regularframe,"please fill in the id");
        }
        else
        {
            try{
            int regId=Integer.parseInt(tfid.getText());
            boolean idexistence=false;
            GymMember memberexistence = null;
               for(GymMember member : al1)
               {
                  //not displaying message in it because otherwise it is add same id again
                  if(member.getId()==regId) //for each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa
                  {
                     idexistence=true;
                     memberexistence  = member;
                     break;//loop bata baira niskinxa
                  }
               }
            if(idexistence)
            {
               if(memberexistence.getActiveStatus()==false){
                  JOptionPane.showMessageDialog(regularframe,"Membership is already deactivated","Deactivated",JOptionPane.INFORMATION_MESSAGE);
               }
               else
               {
                  memberexistence.deactivateMembership();//calling method to activate membership
                  JOptionPane.showMessageDialog(regularframe,"Membership deactivated","Deactivated",JOptionPane.INFORMATION_MESSAGE);
               }
            }
            else
```

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

    else if(ae.getSource()==revert)
    {
        if(tfid.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(regularframe,"please fill in the id");
        }
        else if(tfremoval.getText().isEmpty())
        {
        JOptionPane.showMessageDialog(regularframe,"please fill in the removal reason");
        }
        else
        {
            try{
            int regId=Integer.parseInt(tfid.getText());
            String regRemoval=tfremoval.getText();
            boolean idexistence=false;
            RegularMember memberexistence = null;
            for(GymMember member : all)
            {
                if (member instanceof RegularMember)
```

```
            {
               RegularMember regMember = (RegularMember) member;
              if(regMember.getId()==regId)
              {
                 idexistence=true;
                 memberexistence  = regMember;
                 break;
              }
              }
           }
           if(idexistence)
           {
                 memberexistence.revertRegularMember(regRemoval);
                 JOptionPane.showMessageDialog(regularframe,"Membership
Reset","Reset",JOptionPane.INFORMATION_MESSAGE);
           }
           else
           {
              JOptionPane.showMessageDialog(regularframe,  "Member  with  this  ID  does  not
exist.","Error", JOptionPane.ERROR_MESSAGE);
           }
         }
         catch(NumberFormatException ex)
         {
           JOptionPane.showMessageDialog(regularframe,"ID  can  only  have  number
values!","Error",JOptionPane.WARNING_MESSAGE);
         }
      }
      }

      else if(ae.getSource()==display)
      {
         displayframe.setVisible(true);
```

```
displayta.setText("");
if(al1.isEmpty())
{

   displayta.append("NO MEMBER REGISTERED!!!");
}
else
{
   displayta.append("\nREGULAR MEMBERS: \n\n ");
   boolean memberExistence=false;
   //checking if there are regular members
   for(GymMember member : al1)
   {
      if(member instanceof RegularMember)
      {
         memberExistence= true;
         break;
      }
   }
   if(memberExistence)
   {
      for(GymMember member:al1)
      {
         if(member instanceof RegularMember)
         {
            RegularMember regular = (RegularMember) member;
            displayta.append("\nMember ID: "+regular.getId()+"\n");
            displayta.append("Name: "+regular.getName()+"\n");
            displayta.append("Location: "+regular.getLocation()+"\n");
            displayta.append("Email: "+regular.getEmail()+"\n");
            displayta.append("Phone: "+regular.getPhone()+"\n");
            displayta.append("Date of Birth: "+regular.getDOB()+"\n");
            displayta.append("Gender: "+regular.getGender()+"\n");
```

```
                displayta.append("Plan: "+regular.getPlan()+"\n");
                displayta.append("Price: "+regular.getPrice()+"\n");
                displayta.append("Membership Start Date: "+regular.getMembershipStartDate()+"\n");
                displayta.append("Attendance: "+regular.getAttendance()+"\n");
                displayta.append("Loyalty Points: "+regular.getLoyaltyPoints()+"\n");
                displayta.append("Active Status: "+regular.getActiveStatus()+"\n");
                if(regular.getRemovalReason()!="")
                {
                   displayta.append("Removal Reason: "+regular.getRemovalReason()+"\n");
                }

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

    }

    else if(ae.getSource()==clear || ae.getSource()==pclear)
    {
       tfid.setText("");
       tflocation.setText("");
       tfemail.setText("");

       tfprice.setText("");
       tfname.setText("");
       tfphone.setText("");
```

```
    tfreferral.setText("");
    tfremoval.setText("");


    ptfid.setText("");
    ptflocation.setText("");
    ptfemail.setText("");
    ptfpaid.setText("");

    ptfname.setText("");
    ptfphone.setText("");
    ptfreferral.setText("");
    ptfremoval.setText("");
    ptfdiscount.setText("");
    ptftrainer.setText("");


    dobmonthcb.setSelectedItem("January");
    dobyearcb.setSelectedItem("2006");
    dobdatecb.setSelectedItem("1");
    msdaycb.setSelectedItem("1");
    msmonthcb.setSelectedItem("January");
    msyearcb.setSelectedItem("2006");

    plancb.setSelectedItem("basic");
}

else if(ae.getSource()==addpremium)
{
    try{
    if(ptfid.getText().isEmpty()|| ptflocation.getText().isEmpty()|| ptfemail.getText().isEmpty()||
    ptfname.getText().isEmpty()||ptfphone.getText().isEmpty()||ptfreferral.getText().isEmpty())
    {
```

```
            JOptionPane.showMessageDialog(premiumframe,"please fill in the application");
        }
        else
        {
            int preId = Integer.parseInt(ptfid.getText());
            String preLocation = ptflocation.getText();
            String preEmail = ptfemail.getText();
            String preName = ptfname.getText();
            String prePhone = ptfphone.getText();
            String preTrainer = ptftrainer.getText();
            String preGender="";
            if(prdmale.isSelected())
            {
               preGender="male";
            }
            else if(prdfemale.isSelected())
            {
               preGender="female";
            }
            String year= (String) pdobyearcb.getSelectedItem();
            String month= (String) pdobmonthcb.getSelectedItem(); //combo box creates object and (String)
converts it into string
            String date= (String) pdobdatecb.getSelectedItem();
            String preDob= date+"-"+month+"-"+year;
            String myear= (String) pmsyearcb.getSelectedItem();
            String mmonth= (String) pmsmonthcb.getSelectedItem(); //combo box creates object and (String)
converts it into string
            String mdate= (String) pmsdaycb.getSelectedItem();
            String preMembershipsd= mdate+"-"+mmonth+"-"+myear;

            if(al1.isEmpty())
            {
               //creating object
```

```
            PremiumMember preobj = new PremiumMember(preId,preName,preLocation,prePhone,preEmail,
            preGender,preDob, preMembershipsd, preTrainer);
            al1.add(preobj);
            JOptionPane.showMessageDialog(premiumframe,"Premium Member added succesfully");
        }
        else
        {
            boolean idexistence=false;
            for(GymMember mem : al1)
            {
                //not displaying message in it because otherwise it is add same id again
                if(mem.getId()==preId) //for each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa
                {
                    idexistence=true;
                    break;//loop bata baira niskinxa
                }
            }
            if(idexistence)
            {
                JOptionPane.showMessageDialog(premiumframe,"Member with this id already exists. please choose another ID.","Duplication",JOptionPane.ERROR_MESSAGE);
            }
            else
            {
                PremiumMember preobj = new PremiumMember(preId,preName,preLocation,prePhone,preEmail,
                preGender,preDob, preMembershipsd, preTrainer);
                al1.add(preobj);
                JOptionPane.showMessageDialog(premiumframe,"Premium Member added succesfully");
            }
        }
```

```
        }
      }
      catch(NumberFormatException ex){
      JOptionPane.showMessageDialog(premiumframe,"ID can only have number values!","Error",JOptionPane.WARNING_MESSAGE);
    }
    }
      else if (ae.getSource()==pattendance)
      {
        if(ptfid.getText().isEmpty())
        {
          JOptionPane.showMessageDialog(premiumframe,"please fill in the id");
        }
        else
        {
          try{
          int preId=Integer.parseInt(ptfid.getText());
          boolean idexistence=false;
          GymMember memberexistence = null;
            for(GymMember member : al1)
            {
              if(member.getId()==preId)
              {
                idexistence=true;
                memberexistence  = member;
                break;
              }
            }
          if(idexistence)
          {
            if(memberexistence.getActiveStatus()){
              memberexistence.markAttendance(); //calling to mark attendance
```

```
                JOptionPane.showMessageDialog(premiumframe,"Attendance
marked","Attendance",JOptionPane.INFORMATION_MESSAGE);
                }
                else
                {
                JOptionPane.showMessageDialog(premiumframe,"member is not
activate","Attendance",JOptionPane.ERROR_MESSAGE);
                }
            }
            else
            {
              JOptionPane.showMessageDialog(premiumframe, "Member with this ID does not
exist.","Error", JOptionPane.ERROR_MESSAGE);
            }
          }
          catch(NumberFormatException ex){
      JOptionPane.showMessageDialog(premiumframe,"ID can only have number
values!","Error",JOptionPane.WARNING_MESSAGE);
   }
      }
      }
      else if(ae.getSource()==prevert)
      {
        if(ptfid.getText().isEmpty())
        {
          JOptionPane.showMessageDialog(premiumframe,"please fill in the id");
        }
        else if(ptfremoval.getText().isEmpty())
        {
        JOptionPane.showMessageDialog(premiumframe,"please fill in the removal reason");
        }
        else
        {
```

```
            try{
            int preId=Integer.parseInt(ptfid.getText());
            String preRemoval=ptfremoval.getText();
            boolean idexistence=false;
            PremiumMember memberexistence = null;
            for(GymMember member : all)
            {
               if(member instanceof PremiumMember)
               {
                  PremiumMember preMember = (PremiumMember) member;
                  if(preMember.getId()==preId)
                  {
                     idexistence=true;
                     memberexistence  = preMember;
                     break;
                  }
               }
            }
            if(idexistence)
            {
                  memberexistence.revertPremiumMember();
                  JOptionPane.showMessageDialog(premiumframe,"Membership
Reset","Reset",JOptionPane.INFORMATION_MESSAGE);
            }
            else
            {
               JOptionPane.showMessageDialog(premiumframe, "Member with this ID does not
exist.","Error", JOptionPane.ERROR_MESSAGE);
            }
         }
         catch(NumberFormatException ex)
         {
```

```
            JOptionPane.showMessageDialog(premiumframe,"ID can only have number values!","Error",JOptionPane.WARNING_MESSAGE);
        }
    }
    }
    else if(ae.getSource()==pactivate)
    {
        if(ptfid.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(premiumframe,"please fill in the id");
        }
        else
        {
            try{
            int preId=Integer.parseInt(ptfid.getText());
            boolean idexistence=false;
            GymMember memberexistence = null;
                for(GymMember member : al1)
                {
                    //not displaying message in it because otherwise it is add same id again
                    if(member.getId()==preId) //for each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa
                    {
                        idexistence=true;
                        memberexistence  = member;
                        break;//loop bata baira niskinxa
                    }
                }
            if(idexistence)
            {
                if(memberexistence.getActiveStatus()){
                    JOptionPane.showMessageDialog(premiumframe,"Membership is already activated","Activated",JOptionPane.INFORMATION_MESSAGE);
```

```
            }
            else
            {
               memberexistence.activateMembership();//calling method to activate membership
               JOptionPane.showMessageDialog(premiumframe,"Membership
activated","Activated",JOptionPane.INFORMATION_MESSAGE);
            }
         }
         else
         {
            JOptionPane.showMessageDialog(premiumframe, "Member with this ID does not
exist.","Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        catch(NumberFormatException ex)
        {
           JOptionPane.showMessageDialog(premiumframe,"ID can only have number
values!","Error",JOptionPane.WARNING_MESSAGE);
        }
     }
     }
     else if(ae.getSource()==pdeactivate)
     {
        if(ptfid.getText().isEmpty())
        {
           JOptionPane.showMessageDialog(premiumframe,"please fill in the id");
        }
        else
        {
           try{
           int preId=Integer.parseInt(ptfid.getText());
           boolean idexistence=false;
           GymMember memberexistence = null;
```

```
            for(GymMember member : al1)
            {
               //not displaying message in it because otherwise it is add same id again
               if(member.getId()==preId) //for each loop le array ma vako id haru tanera bhakhar input gareko id sanga same xa xaina check garxa
               {
                  idexistence=true;
                  memberexistence  = member;
                  break;//loop bata baira niskinxa
               }
            }
         if(idexistence)
         {
            if(memberexistence.getActiveStatus()==false){
               JOptionPane.showMessageDialog(premiumframe,"Membership is already deactivated","Deactivated",JOptionPane.INFORMATION_MESSAGE);
            }
            else
            {
               memberexistence.deactivateMembership();//calling method to activate membership
               JOptionPane.showMessageDialog(premiumframe,"Membership deactivated","Deactivated",JOptionPane.INFORMATION_MESSAGE);
            }
         }
         else
         {
            JOptionPane.showMessageDialog(premiumframe, "Member with this ID does not exist.","Error", JOptionPane.ERROR_MESSAGE);
         }
      }
      catch(NumberFormatException ex){
         JOptionPane.showMessageDialog(premiumframe,"ID can only have number values!","Error",JOptionPane.WARNING_MESSAGE);
```

```
    }
}
}
else if(ae.getSource()==pdisplay)
{
    displayframe.setVisible(true);
    displayta.setText("");
    if(al1.isEmpty())
    {
        displayta.append("NO MEMBER REGISTERED!!!");
    }
    else
    {
        displayta.append("\nPREMIUM MEMBERS: \n\n ");
        boolean memberExistence=false;
        //checking if there are premium members
        for(GymMember member : al1)
        {
            if(member instanceof PremiumMember)
            {
                memberExistence= true;
                break;
            }
        }
        if(memberExistence)
        {
            for(GymMember member:al1)
            {
                if(member instanceof PremiumMember)
                {
                    PremiumMember premium = (PremiumMember) member;
                    displayta.append("\nMember ID: "+premium.getId()+"\n");
                    displayta.append("Name: "+premium.getName()+"\n");
```

```
                displayta.append("Location: "+premium.getLocation()+"\n");
                displayta.append("Email: "+premium.getEmail()+"\n");
                displayta.append("Phone: "+premium.getPhone()+"\n");
                displayta.append("Date of Birth: "+premium.getDOB()+"\n");
                displayta.append("Gender: "+premium.getGender()+"\n");
                displayta.append("Membership Start Date: "+premium.getMembershipStartDate()+"\n");
                displayta.append("Attendance: "+premium.getAttendance()+"\n");
                displayta.append("Loyalty Points: "+premium.getLoyaltyPoints()+"\n");
                displayta.append("Personal Trainer: "+premium.getPersonalTrainer()+"\n");
                displayta.append("Payment of Rs."+premium.getPaidAmount()+" has been made\n");
                displayta.append("Active Status: "+premium.getActiveStatus()+"\n");
                displayta.append(premium.getIsFullPayment()?"Full payment has been done":"Full
payment has not been done"+"\n");
                if(premium.getIsFullPayment()==false)
                {
                   double remainingAmount=premium.getPremiumCharge()-premium.getPaidAmount();
                   displayta.append("Payment of Rs."+remainingAmount+" is left to be paid.\n");
                }
                else
                {
                   displayta.append("\nDiscount: Rs."+premium.getDiscountAmount()+"\n");
                }

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

```
}
else if(ae.getSource()==calculateDiscount)
{
   if(ptfid.getText().isEmpty())
   {
      JOptionPane.showMessageDialog(premiumframe,"please fill in the id");
   }
   try
   {
      int preId=Integer.parseInt(ptfid.getText());
      boolean idexistence=false;
      PremiumMember memberexistence = null;
      for(GymMember member: al1)
      {
         if(member instanceof PremiumMember)
         {
            PremiumMember preMember = (PremiumMember) member;
            if(member.getId()==preId)
            {
               idexistence=true;
               memberexistence = preMember;
               break;
            }
         }
      }
      if(idexistence)
      {
         if(memberexistence.getActiveStatus())
         {
            if(memberexistence.getIsFullPayment())
            {
               memberexistence.calculateDiscount();
```

```
                    ptfdiscount.setText(String.valueOf(memberexistence.getDiscountAmount()));
                    JOptionPane.showMessageDialog (premiumframe,""+memberexistence.getName()+" has successfully received 10% discount");
                }
                else
                {
                    JOptionPane.showMessageDialog (premiumframe,"Member is not eligible for discount.");
                }
            }
            else
            {
                JOptionPane.showMessageDialog (premiumframe,"Membership not activated.");
            }
        }
        else
        {
            JOptionPane.showMessageDialog(premiumframe,"Member with this ID does not exist.","Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        catch(NumberFormatException e)
        {
            JOptionPane.showMessageDialog(premiumframe,"ID can only have number values.","Error",JOptionPane.WARNING_MESSAGE);
        }
    }
    else if(ae.getSource()==pay)
    {
        if(ptfid.getText().isEmpty())
        {
            JOptionPane.showMessageDialog(premiumframe,"please fill in the id");
        }
```

```
else if(ptfpaid.getText().isEmpty())
{
   JOptionPane.showMessageDialog(premiumframe,"please fill in the paid amount.");
}
else
{
   try
   {
      int preId=Integer.parseInt(ptfid.getText());
      double paidAmount=Double.parseDouble(ptfpaid.getText());
      boolean idexistence=false;
      PremiumMember memberexistence = null;
      for(GymMember member: al1)
      {
         if(member instanceof PremiumMember)
         {
            PremiumMember preMember = (PremiumMember) member;
            if(member.getId()==preId)
            {
               idexistence=true;
               memberexistence = preMember;
               break;
            }
         }
      }
      if(idexistence)
      {
         if(memberexistence.getActiveStatus())
         {
            String message = memberexistence.payDueAmount(paidAmount);
            JOptionPane.showMessageDialog(premiumframe,message);
         }
         else
```

```
                {
                    JOptionPane.showMessageDialog (premiumframe,"Membership not activated.");
                }
            }
            else
            {
                JOptionPane.showMessageDialog(premiumframe,"Member with this ID does not exist.","Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        catch(NumberFormatException e)
        {
            JOptionPane.showMessageDialog(premiumframe,"ID can only have number values.","Error",JOptionPane.WARNING_MESSAGE);
        }
    }
}
else if(ae.getSource()==upgradePlan)
{
    if(tfid.getText().isEmpty())
    {
        JOptionPane.showMessageDialog(regularframe,"please fill in the id");
    }
    else
    {
        try
        {
            int regId=Integer.parseInt(tfid.getText());
            String plan = (String) plancb.getSelectedItem();
            boolean idexistence=false;
            RegularMember memberexistence = null;
            for(GymMember member: al1)
            {
```

```
            if(member instanceof RegularMember)
            {
                RegularMember regMember = (RegularMember) member;
                if(regMember.getId()==regId)
                {
                    idexistence=true;
                    memberexistence = regMember;
                    break;
                }
            }
        }
        if(idexistence)
        {
            if(memberexistence.getActiveStatus())
            {
                if(memberexistence.getIsEligibleForUpgrade())
                {
                    String message = memberexistence.upgradePlan(plan);
                    JOptionPane.showMessageDialog(regularframe,message);
                }
                else
                {
                    JOptionPane.showMessageDialog(regularframe,"Member is not eligible for upgrade. Attendance is low.");
                }
            }
            else
            {
                JOptionPane.showMessageDialog (regularframe,"Membership not activated.");
            }
        }
        else
        {
```

```
            JOptionPane.showMessageDialog(regularframe,"Member with this ID does not
exist.","Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        catch(NumberFormatException e)
        {
          JOptionPane.showMessageDialog(regularframe,"ID can only have number
values.","Error",JOptionPane.WARNING_MESSAGE);
        }
      }
    }
    else if(ae.getSource()==save || ae.getSource()==psave)
    {
      try{
        if(al1.isEmpty())
      {
        JOptionPane.showMessageDialog(regularframe,"NO MEMBER
REGISTERED!","Error",JOptionPane.WARNING_MESSAGE);
      }
      else
      {
        File file = new File("MemberDetails.txt");
        FileWriter writer = new FileWriter(file);


writer.write("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~INFORMATION OF THE
MEMBERS~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~\n\n");

writer.write("=========================================================
=============================================================
===============================================\n");
```

```
        String header = String.format("%-5s %-10s %-10s %-10s %-13s %-23s %-10s %-10s %-11s %-
15s %-14s %-13s %-16s %-16s \n",
        "ID", "Name", "Location", "Phone", "Email","Membership Start Date", "Plan",
        "Price", "Attendance", "Loyalty Points", "Active Status", "Full Payment",
        "Discount Amount", "Net Amount Paid");
        //s- string d-int f-float/double
        //% begining
        //- left align
        //.2 for 2 decimal place
        writer.write(header);
        writer.write("----------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------\n");
        for(GymMember member:al1)
        {
           if(member instanceof RegularMember)
           {
              RegularMember regMember = (RegularMember) member;
              String line = String.format("%-5d %-10s %-10s %-10s %-13s %-23s %-10s %-10s %-11s
%-15s %-14s %-13s %-16s %-16s \n",
              regMember.getId(),
              regMember.getName(),
              regMember.getLocation(),
              regMember.getPhone(),
              regMember.getEmail(),
              regMember.getMembershipStartDate(),
              regMember.getPlan(),
              regMember.getPrice(),
              regMember.getAttendance(),
              regMember.getLoyaltyPoints(),
              regMember.getActiveStatus()?"active":"inactive",
              "NONE","NONE","NONE");

              writer.write(line);
```

```
writer.write("==========================================================================
==========================================================================
==========================================\n");

            }
            else if(member instanceof PremiumMember)
            {
               PremiumMember preMember = (PremiumMember) member;
               String line = String.format("%-5d %-10s %-10s %-10s %-13s %-23s %-10s %-10s %-11s
%-15s %-14s %-13s %-16.2f %-16.2f \n",
               preMember.getId(),
               preMember.getName(),
               preMember.getLocation(),
               preMember.getPhone(),
               preMember.getEmail(),
               preMember.getMembershipStartDate(),
               "PREMIUM",
               preMember.getPremiumCharge(),
               preMember.getAttendance(),
               preMember.getLoyaltyPoints(),
               preMember.getActiveStatus()?"active":"inactive",
               preMember.getIsFullPayment()?"complete":"incomplete",
               preMember.getDiscountAmount(),
               preMember.getPaidAmount());

               writer.write(line);

writer.write("==========================================================================
==========================================================================
==========================================\n");
            }
```

```
            }
            writer.close();
            JOptionPane.showMessageDialog(regularframe,"member details added");
            //JOptionPane.showMessageDialog(premiumframe,"member details added");
        }
        }
        catch(IOException e)
        {
            JOptionPane.showMessageDialog(regularframe, "An error has occured."+e.getMessage()) ;
        }
    }
    else if(ae.getSource()==read || ae.getSource()==pread)
    {
        readframe.setVisible(true);
        readta.setText("");
        try
        {
            FileReader fileRead = new FileReader("MemberDetails.txt");
            int ch;
            while((ch = fileRead.read())!= -1)
            {
                char cha=(char) ch; //converting ascii values into character
                readta.append(String.valueOf(cha));
            }
        }
        catch(FileNotFoundException e)
        {
            JOptionPane.showMessageDialog(regularframe,"FILE                                    NOT
FOUND.","Error",JOptionPane.ERROR_MESSAGE);
        }
        catch(IOException e)
        {
```

```
            JOptionPane.showMessageDialog(regularframe,"FILE NOT
READING"+e.getMessage(),"Error",JOptionPane.ERROR_MESSAGE);
          }
        }

        }
    }
```