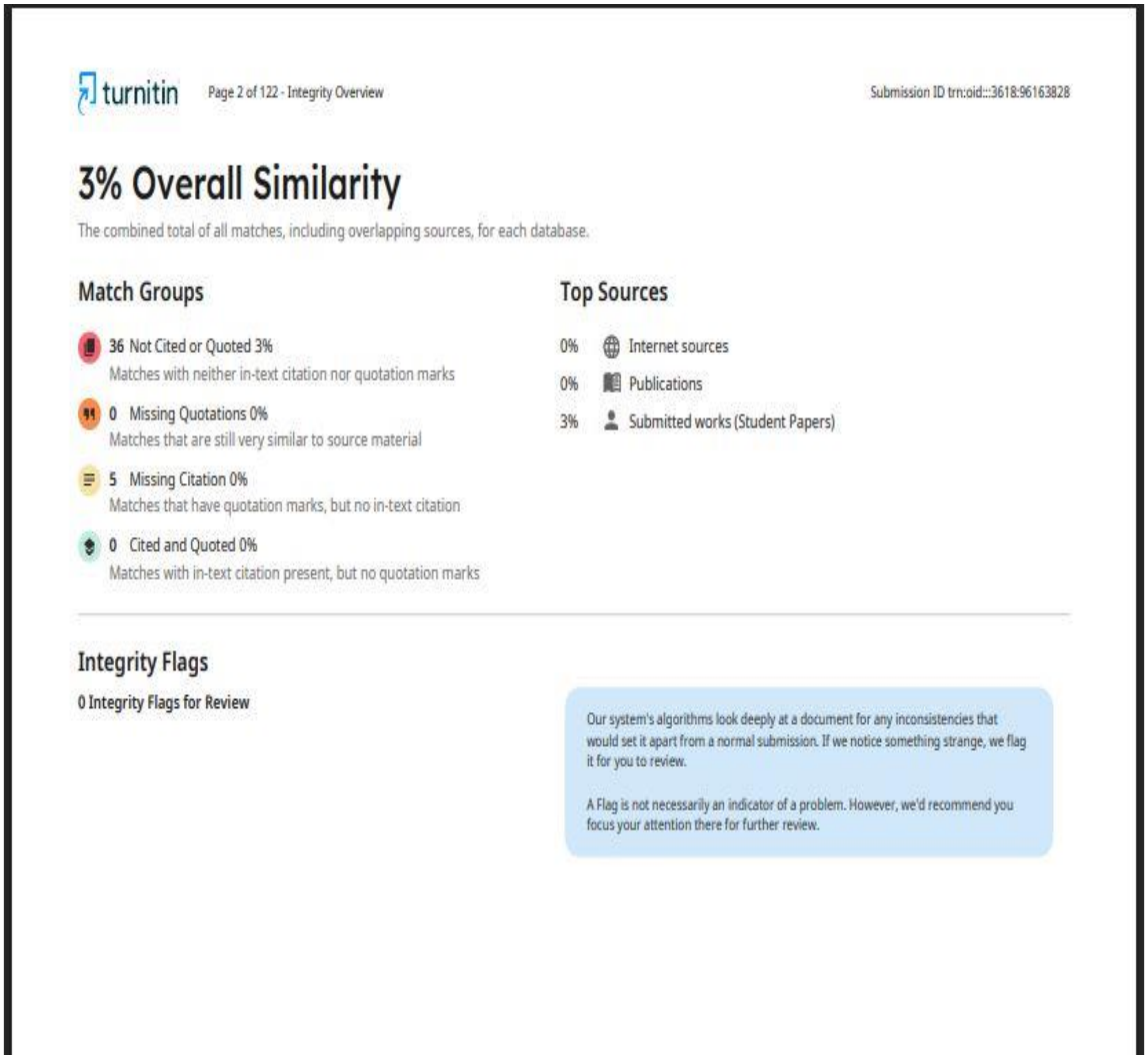
CS4001NI Programming

60% Individual Coursework

2025 Spring

Student Name: Abhishek Kumar Shah
London Met ID: 24046697
College ID: NP01CP4A240235
Group: L1C1
Assignment Due Date: Friday, May 16, 2025
Assignment Submission Date: Friday, May 16, 2025



turnitin Page 2 of 122 - Integrity Overview Submission ID trn:oid:::3618:96163828

3% Overall Similarity

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

Match Groups

- **36** Not Cited or Quoted 3%
  Matches with neither in-text citation nor quotation marks
- **0** Missing Quotations 0%
  Matches that are still very similar to source material
- **5** Missing Citation 0%
  Matches that have quotation marks, but no in-text citation
- **0** Cited and Quoted 0%
  Matches with in-text citation present, but no quotation marks

Top Sources

0%	Internet sources
0%	Publications
3%	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.

Table of Contents

List of Figures

List of Tables

1. Introduction

The primary objective of this coursework is to design and implement a comprehensive Gym Management System using the principles of Object-Oriented Programming (OOP) in Java. The project aims to simulate a real-world scenario where a gym manages different types of members—Regular and Premium—through an application that facilitates member registration, attendance tracking, membership activation/deactivation, upgrade eligibility, payment handling, and loyalty management. The system is divided into two key components: the core OOP implementation and a Graphical User Interface (GUI) built using Java's Abstract Window Toolkit (AWT) and Swing libraries.

At the core of the system lies an abstract class named GymMember, which encapsulates shared member details such as ID, name, contact information, gender, date of birth, membership start date, attendance, loyalty points, and membership status. This class enforces a common structure across all member types while allowing subclass-specific behavior through the use of abstract and inherited methods. Two concrete subclasses, RegularMember and PremiumMember, inherit from GymMember, each adding their own unique features and functionalities. Regular members, for example, have referral sources, attendance limits, and different gym plans (basic, standard, deluxe), while premium members have access to personal trainers and require full or partial payments for their membership with discounts applied upon full payment.

To allow user interaction with the system, a GUI is developed in a separate class named GymGUI. The interface includes labeled text fields for input, combo boxes for plan selection, radio buttons for gender choice, and various buttons to perform specific operations such as adding members, activating or deactivating memberships, marking attendance, upgrading plans, calculating discounts, and handling payments. The GUI not only provides a user-friendly means of accessing the system's features but also ensures that data is managed consistently and securely using a single ArrayList of GymMember objects, which stores both Regular and Premium members.

Throughout the development process, key OOP principles were applied:

- **Encapsulation** ensured that data fields were protected and accessible only through getters/setters.
- **Inheritance** enabled shared functionality and reusability between the parent and child classes.
- **Abstraction** was used through the abstract mark Attendance() method, enforcing method implementation in the subclasses.
- **Polymorphism** allowed for dynamic behavior, such as calling subclass-specific methods through superclass references during GUI operations.

Robust error handling using try-catch blocks was implemented to manage exceptions, especially during numeric input parsing and file operations. The system also includes features to save member data to a .txt file and retrieve it later, ensuring persistence between sessions.

This report documents every step of the development process in detail. It includes class diagrams, pseudocode for all essential methods, method descriptions, testing evidence (with screenshots), error-handling explanations, and a reflective conclusion. The development and testing were conducted in BlueJ and run via the command prompt, fulfilling all technical requirements as outlined in the coursework brief.

Ultimately, this project serves as a demonstration of how Java OOP and GUI concepts can be applied to solve a realistic, structured problem. It not only strengthens understanding of core programming concepts but also develops practical skills in user interface design, input validation, exception handling, file management, and modular software development.

2. GUI Wireframe

I used Balsamiq to make wireframes for [project, like an app or website]. Balsamiq was easy to use with its drag-and-drop tools, helping me create simple sketches of how the [app/website] will look and work. These wireframes show the basic layout and main features, making it clear for everyone to understand the plan before building the final design.

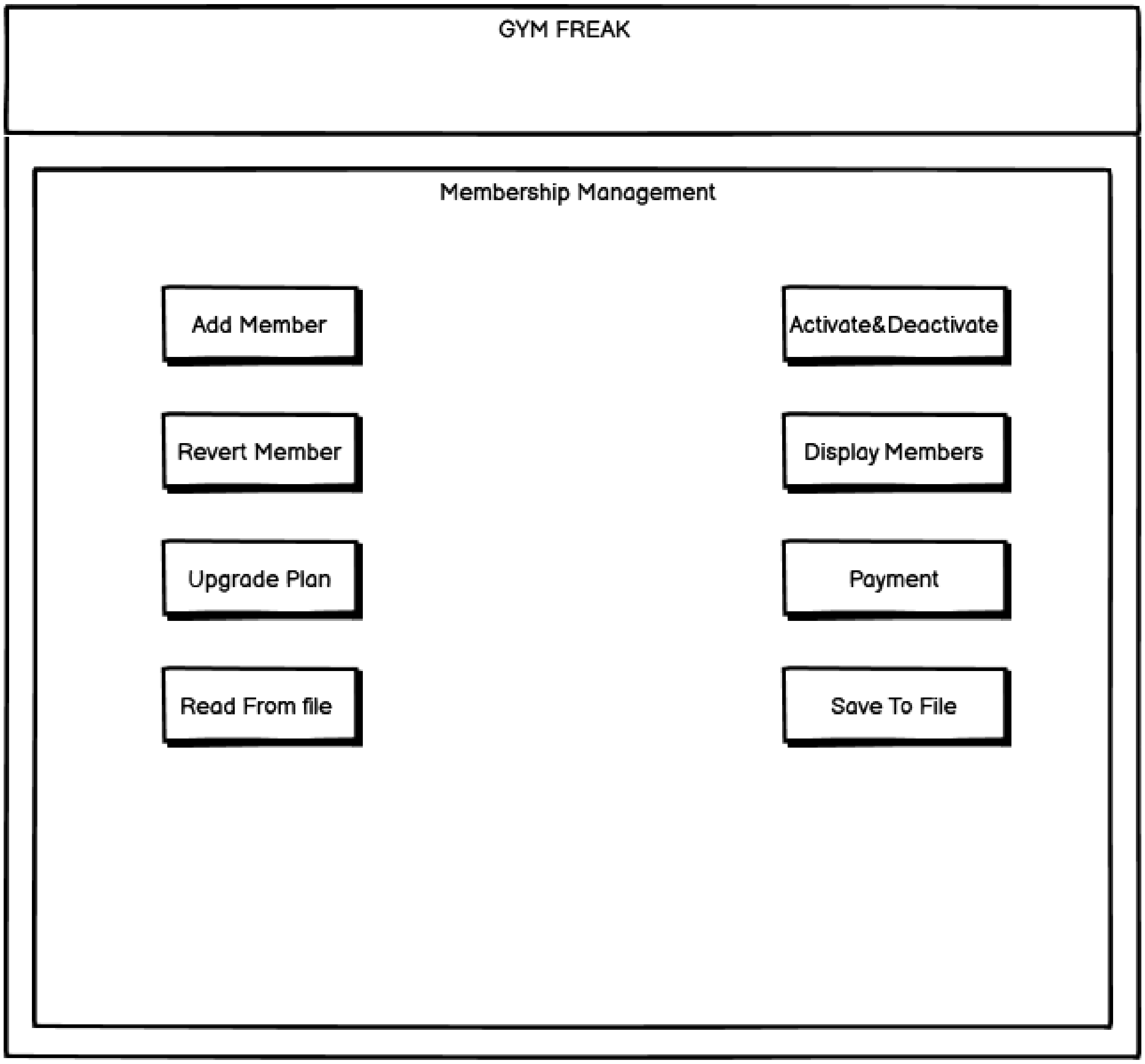


Figure 1: Main Frame

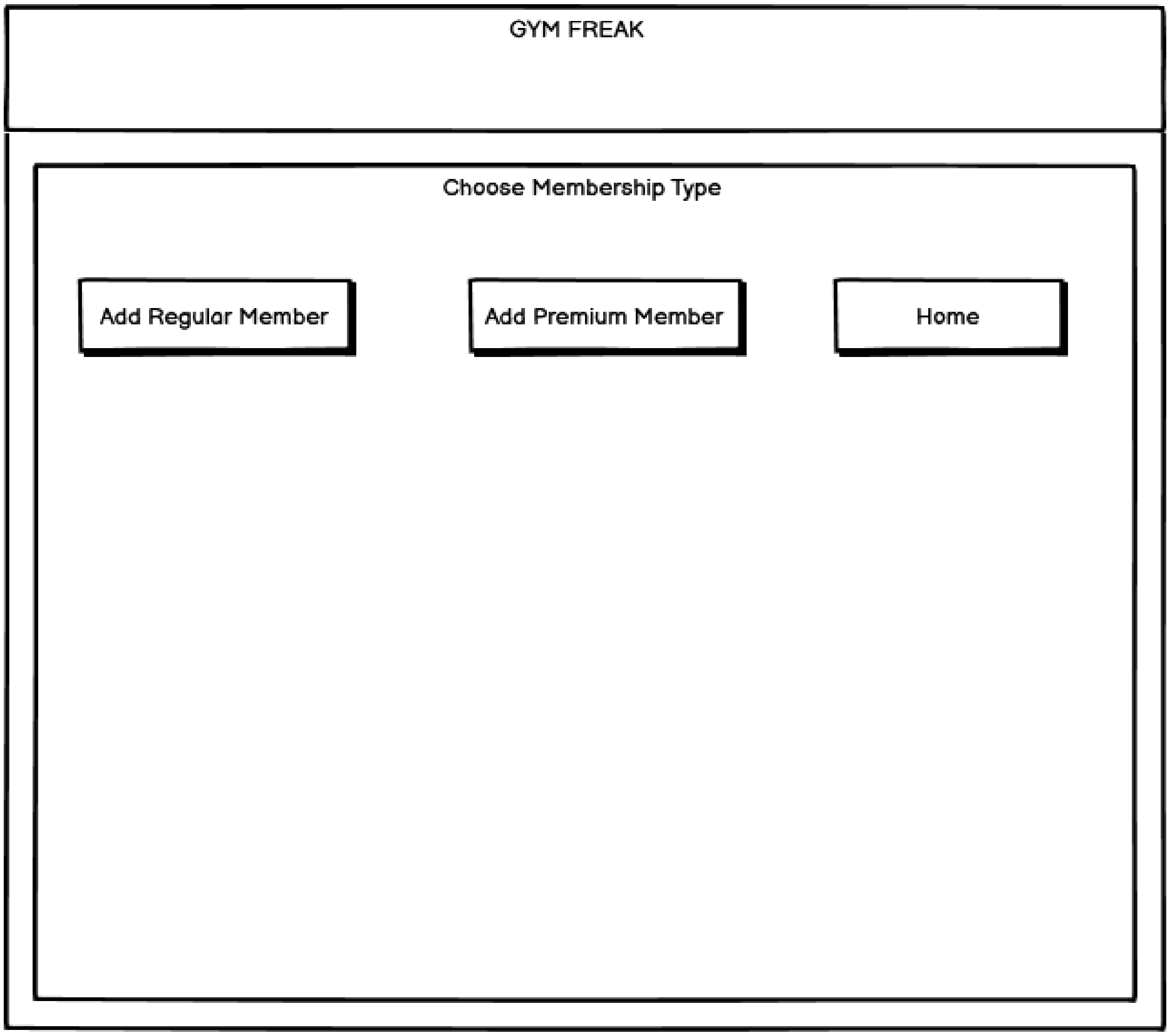


Figure 2: Choose Membership Type

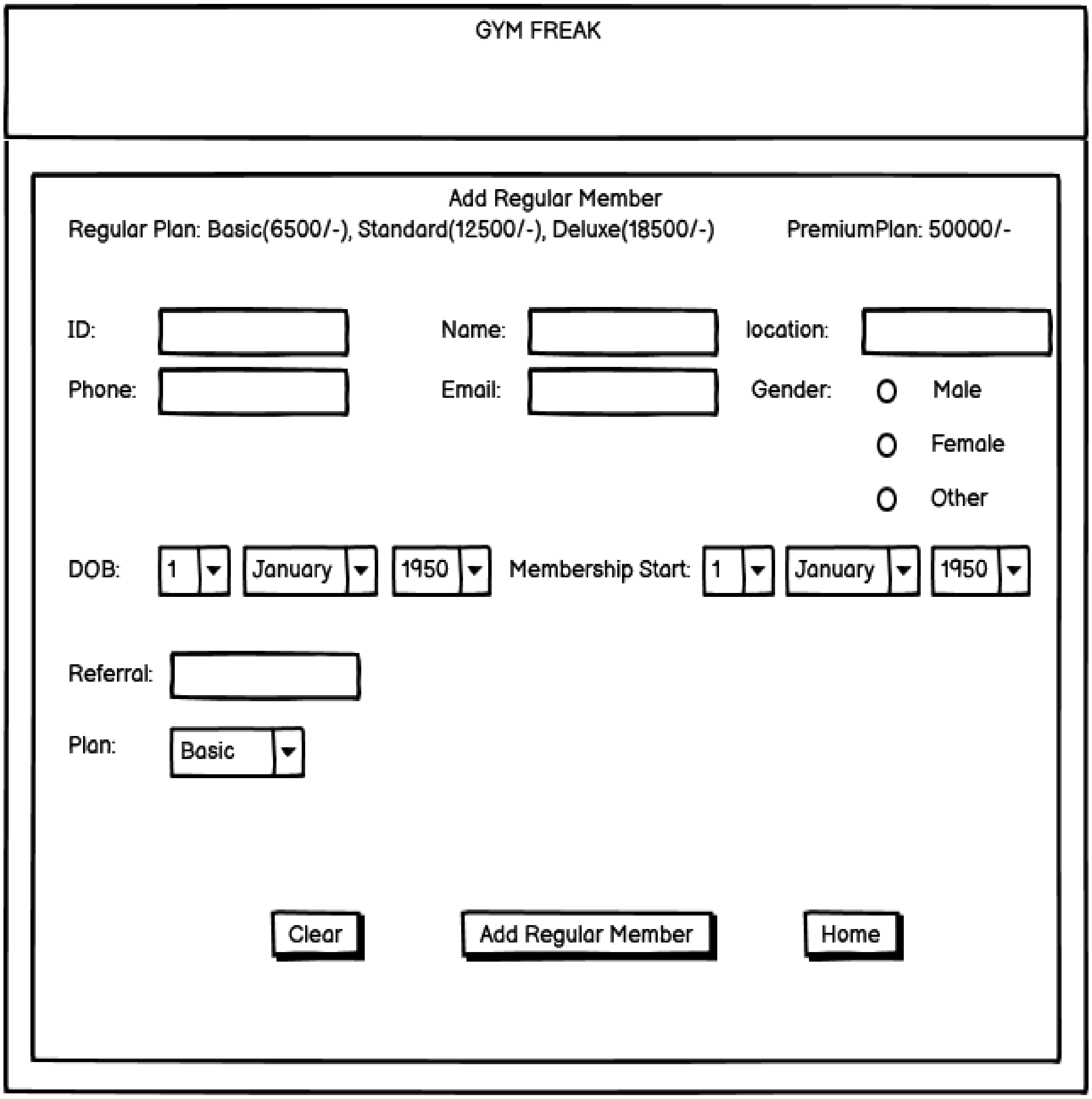


Figure 3: Add Regular Member

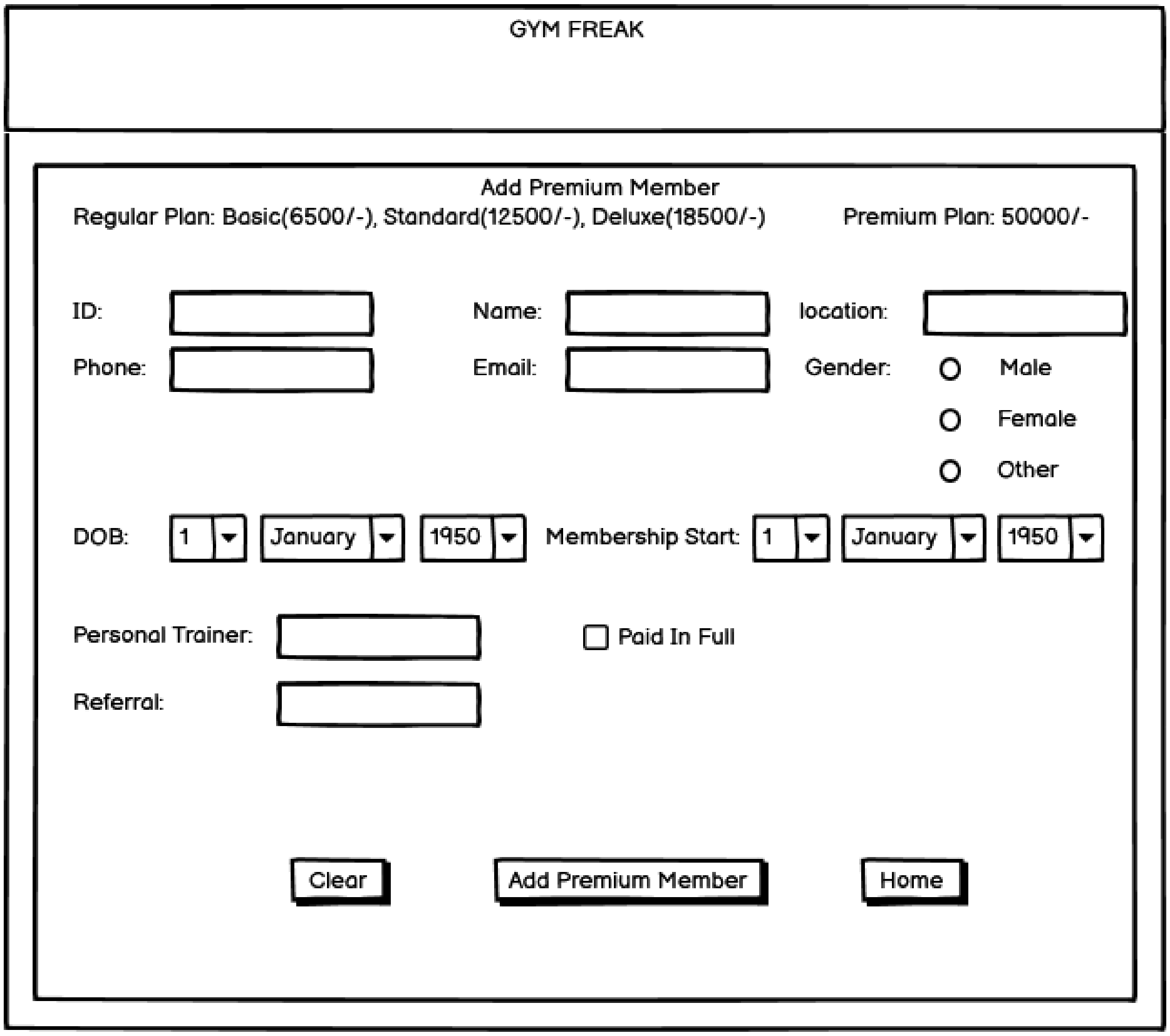


Figure 4: Add Premium Member

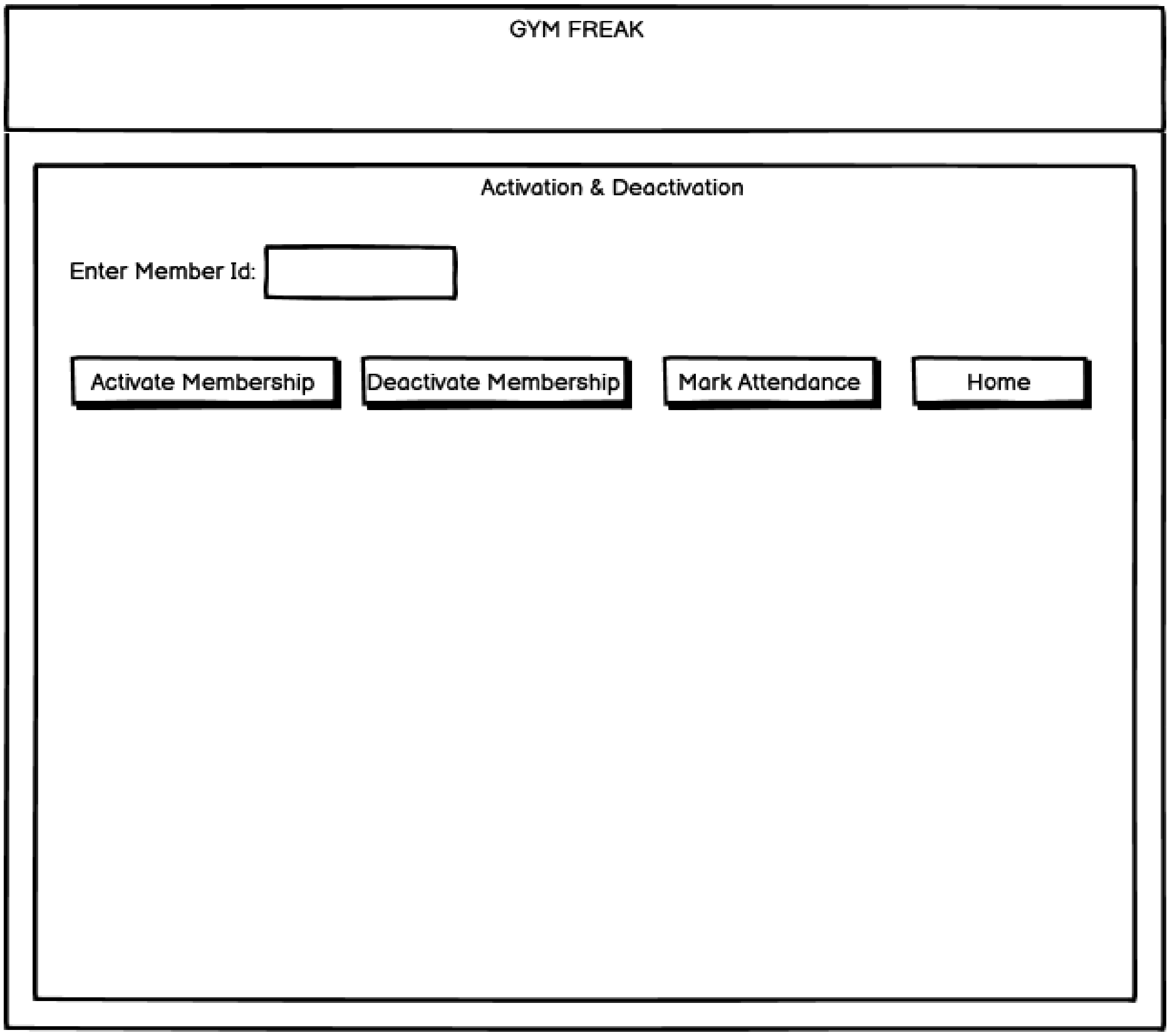


Figure 5: Activation and Deactivation

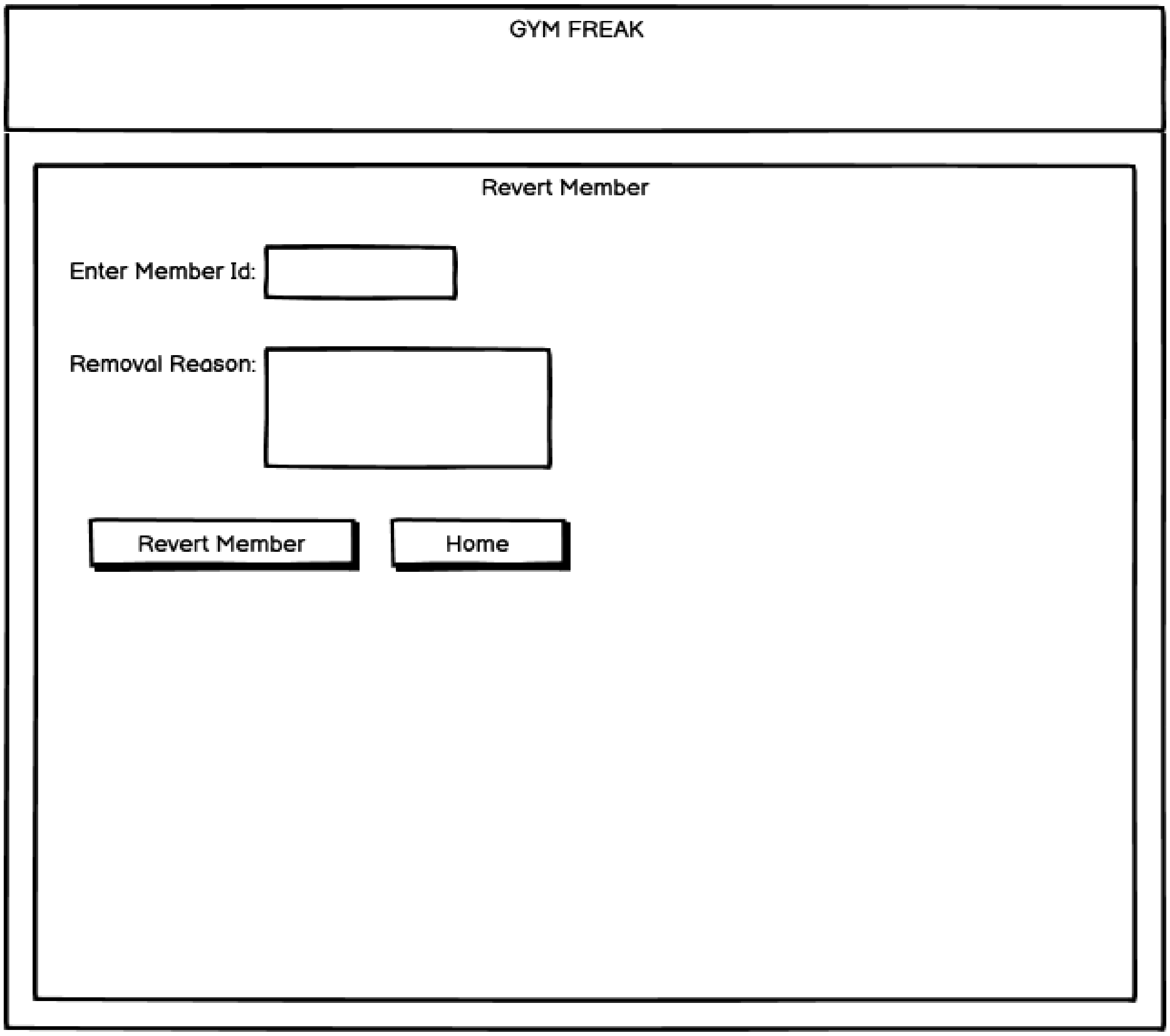


Figure 6: Revert Member

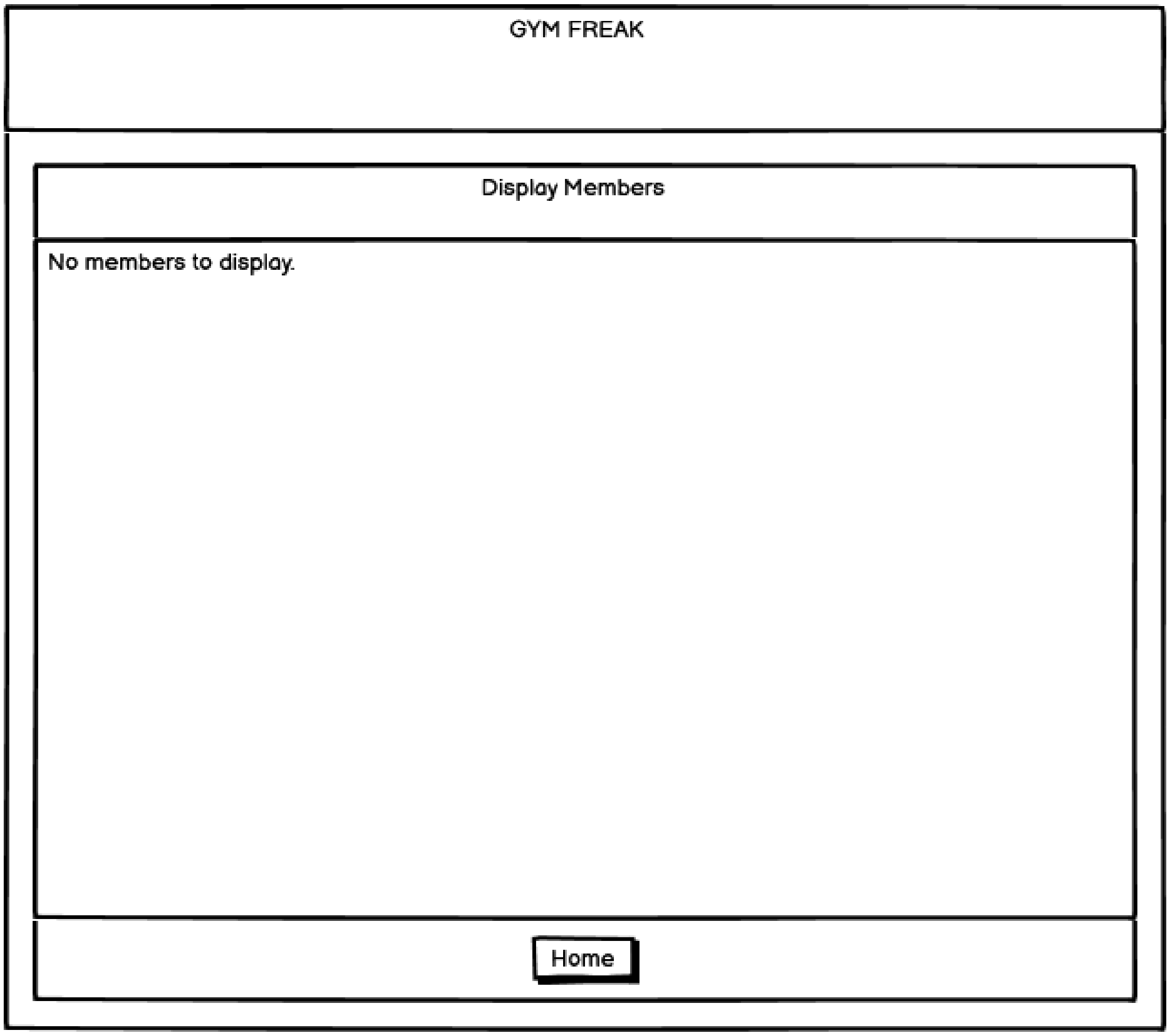


Figure 7: Display Members

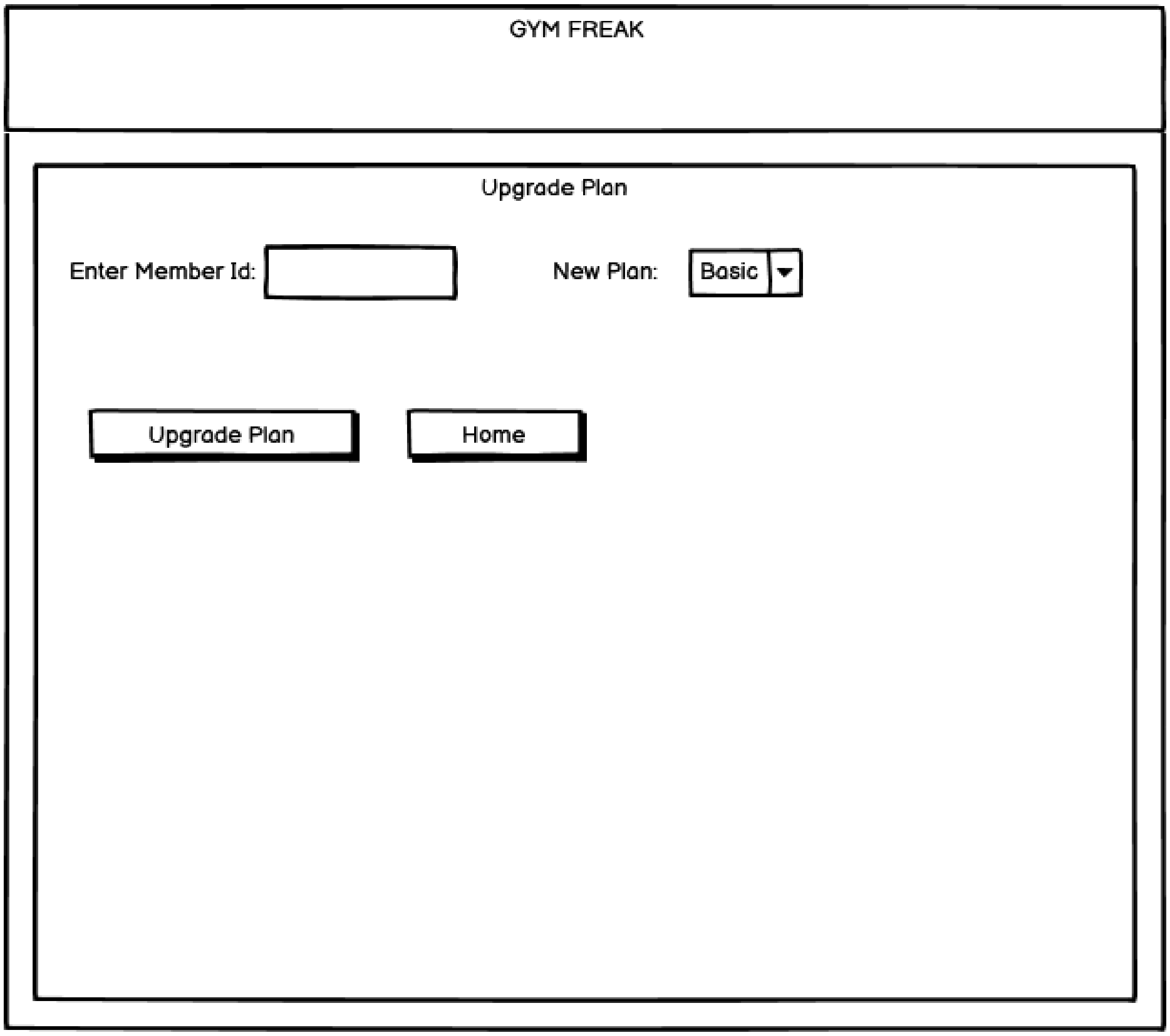


Figure 8: Upgrade Plan

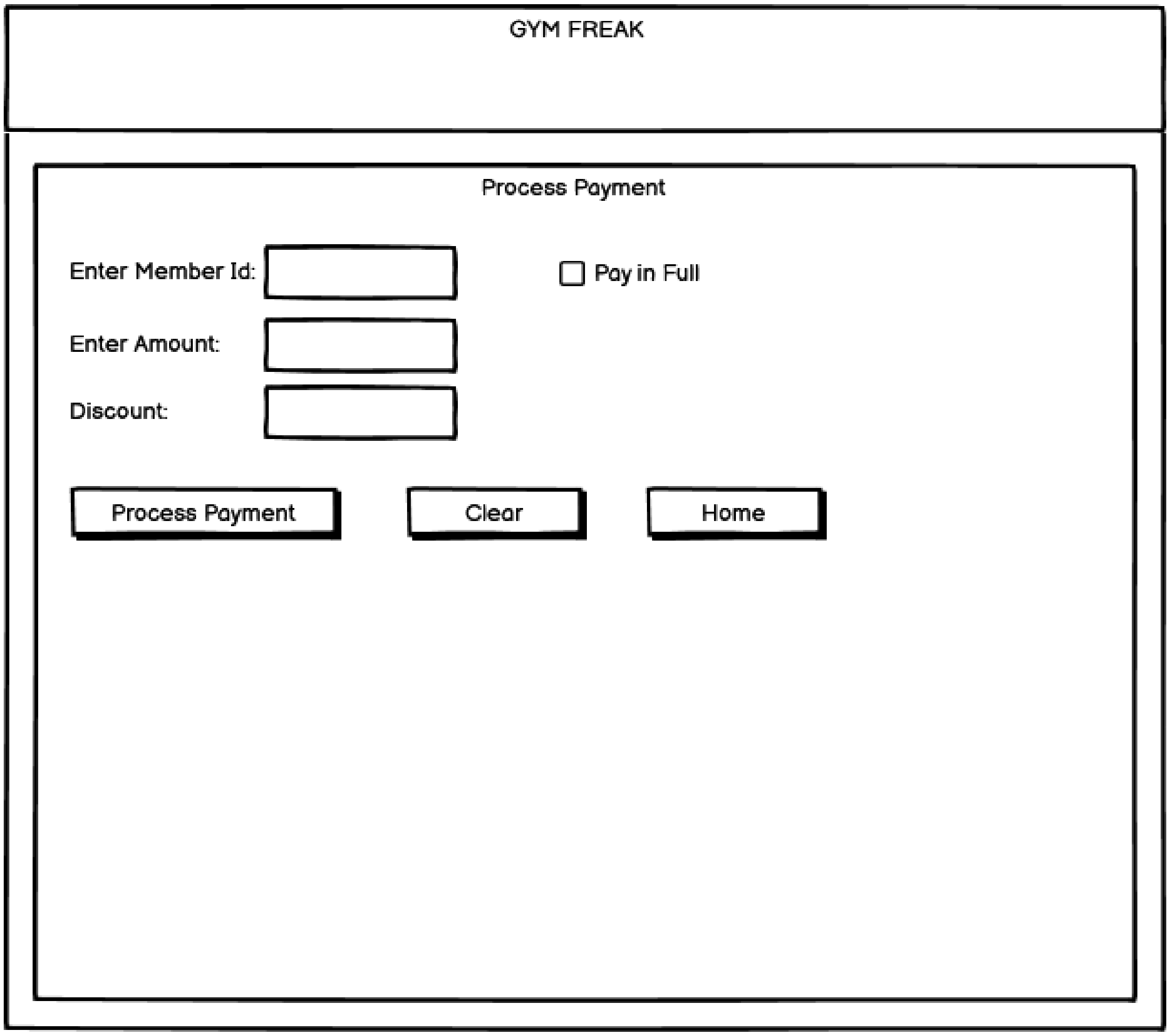


Figure 9: Process Payment

3. Class Diagram

i. Class diagram for GymMember Class

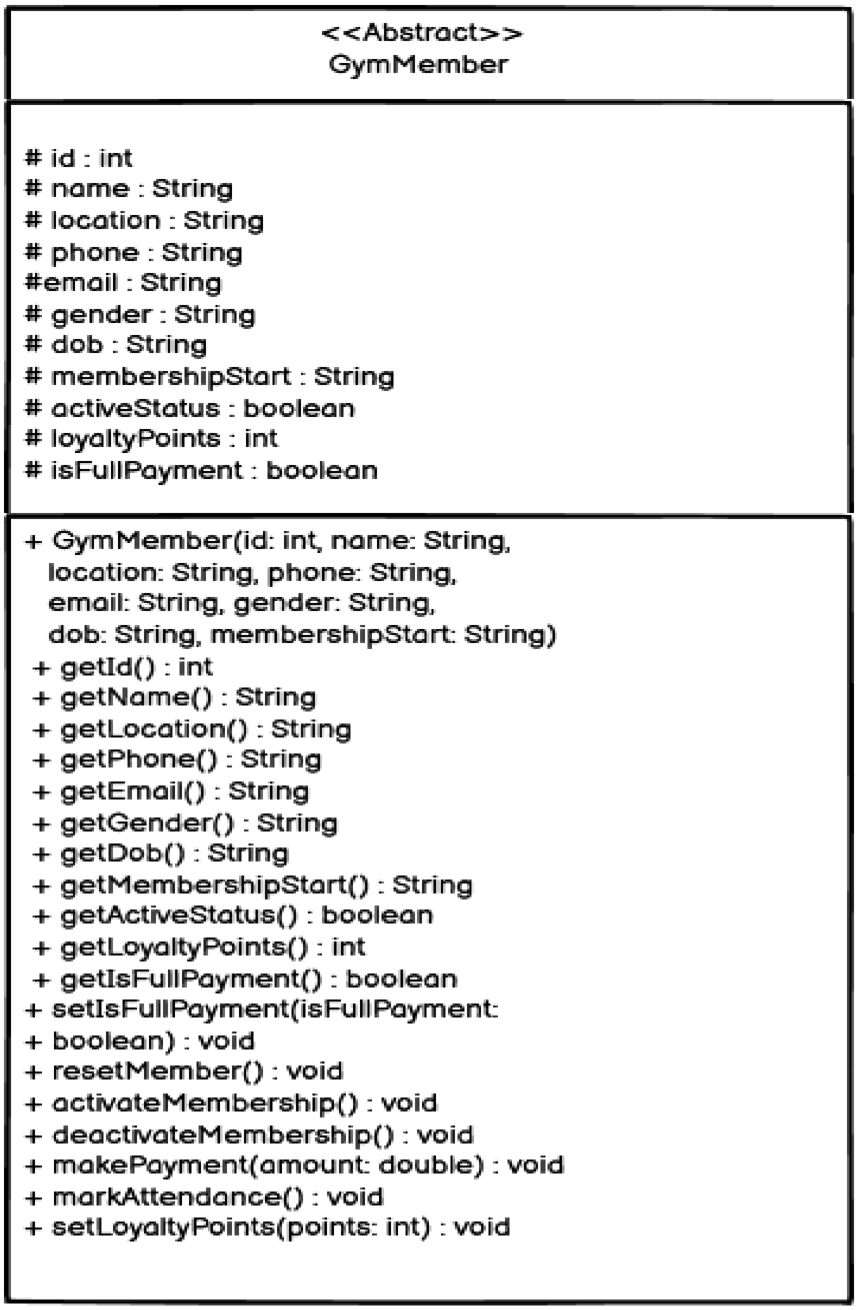
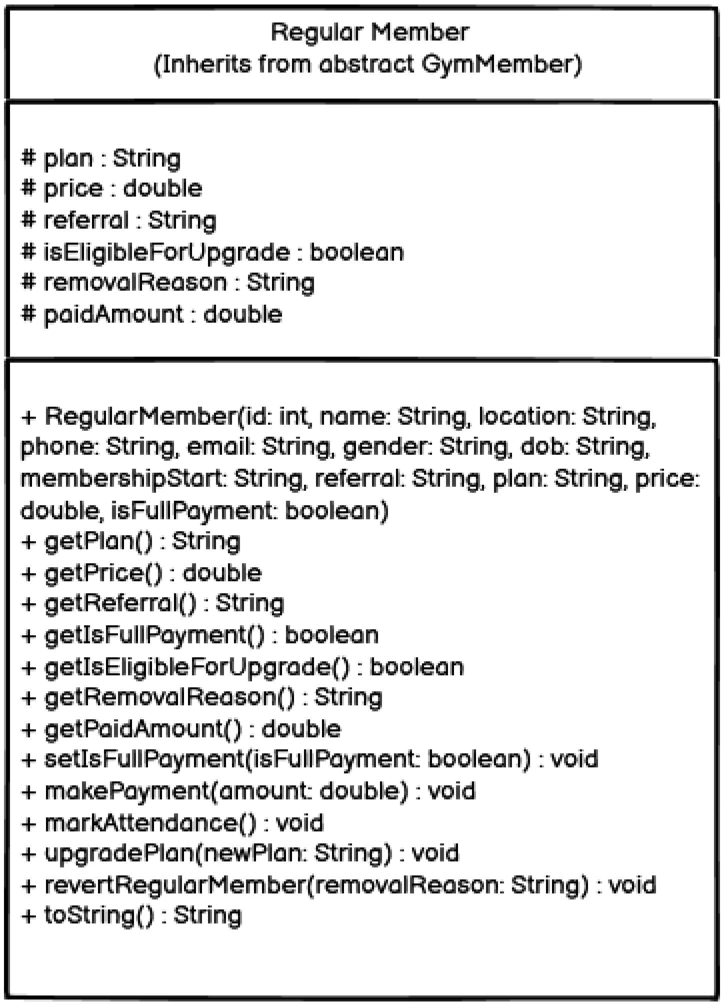

<<Abstract>>
GymMember
# id : int
# name : String
# location : String
# phone : String
#email : String
# gender : String
# dob : String
# membershipStart : String
# activeStatus : boolean
# loyaltyPoints : int
# isFullPayment : boolean
+ GymMember(id: int, name: String, location: String, phone: String, email: String, gender: String, dob: String, membershipStart: String)
+ getId() : int
+ getName() : String
+ getLocation() : String
+ getPhone() : String
+ getEmail() : String
+ getGender() : String
+ getDob() : String
+ getMembershipStart() : String
+ getActiveStatus() : boolean
+ getLoyaltyPoints() : int
+ getIsFullPayment() : boolean
+ setIsFullPayment(isFullPayment:
+ boolean) : void
+ resetMember() : void
+ activateMembership() : void
+ deactivateMembership() : void
+ makePayment(amount: double) : void
+ markAttendance() : void
+ setLoyaltyPoints(points: int) : void

Figure 10: Class diagram for GymMember class

ii. Class Diagram for RegularMember class



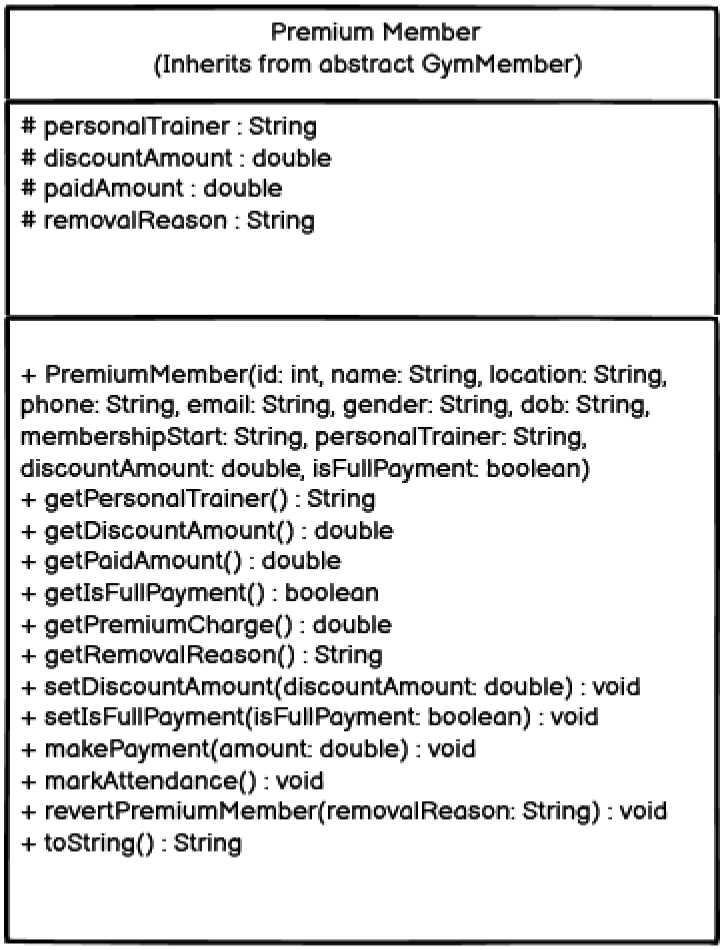
Regular Member
(Inherits from abstract GymMember)

plan : String
price : double
referral : String
isEligibleForUpgrade : boolean
removalReason : String
paidAmount : double

+ RegularMember(id: int, name: String, location: String, phone: String, email: String, gender: String, dob: String, membershipStart: String, referral: String, plan: String, price: double, isFullPayment: boolean)
+ getPlan() : String
+ getPrice() : double
+ getReferral() : String
+ getIsFullPayment() : boolean
+ getIsEligibleForUpgrade() : boolean
+ getRemovalReason() : String
+ getPaidAmount() : double
+ setIsFullPayment(isFullPayment: boolean) : void
+ makePayment(amount: double) : void
+ markAttendance() : void
+ upgradePlan(newPlan: String) : void
+ revertRegularMember(removalReason: String) : void
+ toString() : String

Figure 11: Class diagram for RegularMember class

iii. Class Diagram for PremiumMember class



Premium Member
(Inherits from abstract GymMember)

personalTrainer : String
discountAmount : double
paidAmount : double
removalReason : String

+ PremiumMember(id: int, name: String, location: String, phone: String, email: String, gender: String, dob: String, membershipStart: String, personalTrainer: String, discountAmount: double, isFullPayment: boolean)
+ getPersonalTrainer() : String
+ getDiscountAmount() : double
+ getPaidAmount() : double
+ getIsFullPayment() : boolean
+ getPremiumCharge() : double
+ getRemovalReason() : String
+ setDiscountAmount(discountAmount: double) : void
+ setIsFullPayment(isFullPayment: boolean) : void
+ makePayment(amount: double) : void
+ markAttendance() : void
+ revertPremiumMember(removalReason: String) : void
+ toString() : String

Figure 12: Class diagram for Premium Member class

iv. Class diagram for GymGUI class

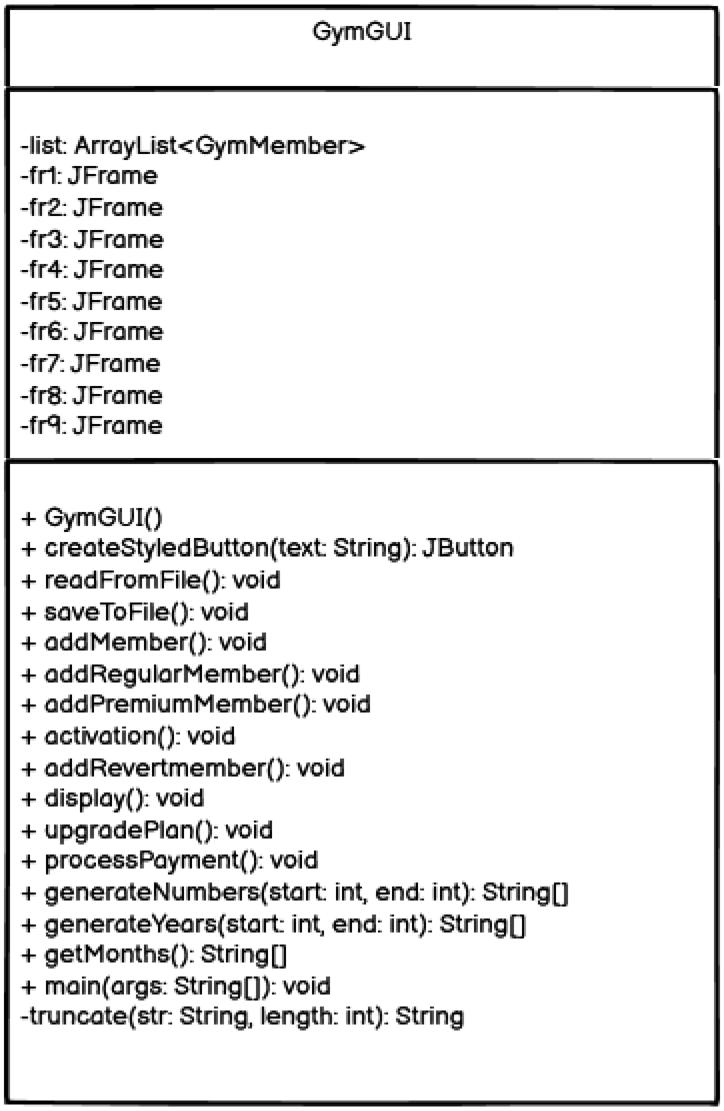

GymGUI
-list: ArrayList<GymMember>
-fr1: JFrame
-fr2: JFrame
-fr3: JFrame
-fr4: JFrame
-fr5: JFrame
-fr6: JFrame
-fr7: JFrame
-fr8: JFrame
-fr9: JFrame
+ GymGUI()
+ createStyledButton(text: String): JButton
+ readFromFile(): void
+ saveToFile(): void
+ addMember(): void
+ addRegularMember(): void
+ addPremiumMember(): void
+ activation(): void
+ addRevertmember(): void
+ display(): void
+ upgradePlan(): void
+ processPayment(): void
+ generateNumbers(start: int, end: int): String[]
+ generateYears(start: int, end: int): String[]
+ getMonths(): String[]
+ main(args: String[]): void
-truncate(str: String, length: int): String

Figure 13: Class diagram for GymGUI class

v. Combined class diagram of all class

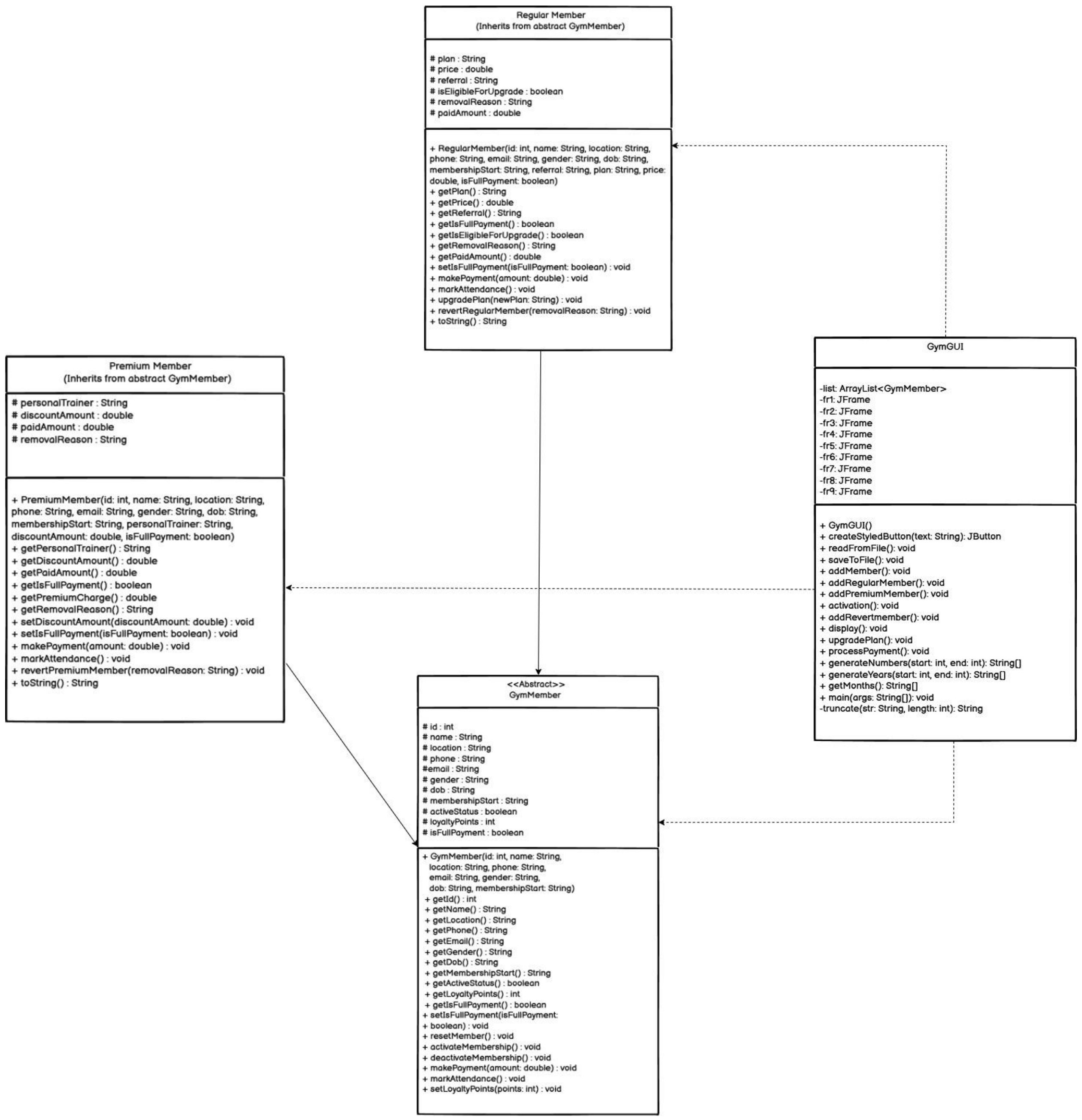

Regular Member
(Inherits from abstract GymMember)
# plan : String
# price : double
# referral : String
# isEligibleForUpgrade : boolean
# removalReason : String
# paidAmount : double
+ RegularMember(id: int, name: String, location: String, phone: String, email: String, gender: String, dob: String, membershipStart: String, referral: String, plan: String, price: double, isFullPayment: boolean)
+ getPlan() : String
+ getPrice() : double
+ getReferral() : String
+ getIsFullPayment() : boolean
+ getIsEligibleForUpgrade() : boolean
+ getRemovalReason() : String
+ getPaidAmount() : double
+ setIsFullPayment(isFullPayment: boolean) : void
+ makePayment(amount: double) : void
+ markAttendance() : void
+ upgradePlan(newPlan: String) : void
+ revertRegularMember(removalReason: String) : void
+ toString() : String
Premium Member
(Inherits from abstract GymMember)
# personalTrainer : String
# discountAmount : double
# paidAmount : double
# removalReason : String
+ PremiumMember(id: int, name: String, location: String, phone: String, email: String, gender: String, dob: String, membershipStart: String, personalTrainer: String, discountAmount: double, isFullPayment: boolean)
+ getPersonalTrainer() : String
+ getDiscountAmount() : double
+ getPaidAmount() : double
+ getIsFullPayment() : boolean
+ getPremiumCharge() : double
+ getRemovalReason() : String
+ setDiscountAmount(discountAmount: double) : void
+ setIsFullPayment(isFullPayment: boolean) : void
+ makePayment(amount: double) : void
+ markAttendance() : void
+ revertPremiumMember(removalReason: String) : void
+ toString() : String
GymGUI
-list: ArrayList<GymMember>
-fr1: JFrame
-fr2: JFrame
-fr3: JFrame
-fr4: JFrame
-fr5: JFrame
-fr6: JFrame
-fr7: JFrame
-fr8: JFrame
-fr9: JFrame
+ GymGUI()
+ createStyledButton(text: String): JButton
+ readFromFile(): void
+ saveToFile(): void
+ addMember(): void
+ addRegularMember(): void
+ addPremiumMember(): void
+ activation(): void
+ addRevertmember(): void
+ display(): void
+ upgradePlan(): void
+ processPayment(): void
+ generateNumbers(start: int, end: int): String[]
+ generateYears(start: int, end: int): String[]
+ getMonths(): String[]
+ main(args: String[]): void
-truncate(str: String, length: int): String
<<Abstract>>
GymMember
# id : int
# name : String
# location : String
# phone : String
#email : String
# gender : String
# dob : String
# membershipStart : String
# activeStatus : boolean
# loyaltyPoints : int
# isFullPayment : boolean
+ GymMember(id: int, name: String, location: String, phone: String, email: String, gender: String, dob: String, membershipStart: String)
+ getId() : int
+ getName() : String
+ getLocation() : String
+ getPhone() : String
+ getEmail() : String
+ getGender() : String
+ getDob() : String
+ getMembershipStart() : String
+ getActiveStatus() : boolean
+ getLoyaltyPoints() : int
+ getIsFullPayment() : boolean
+ setIsFullPayment(isFullPayment: boolean) : void
+ resetMember() : void
+ activateMembership() : void
+ deactivateMembership() : void
+ makePayment(amount: double) : void
+ markAttendance() : void
+ setLoyaltyPoints(points: int) : void

Figure 14: Combined class diagram (Relation)

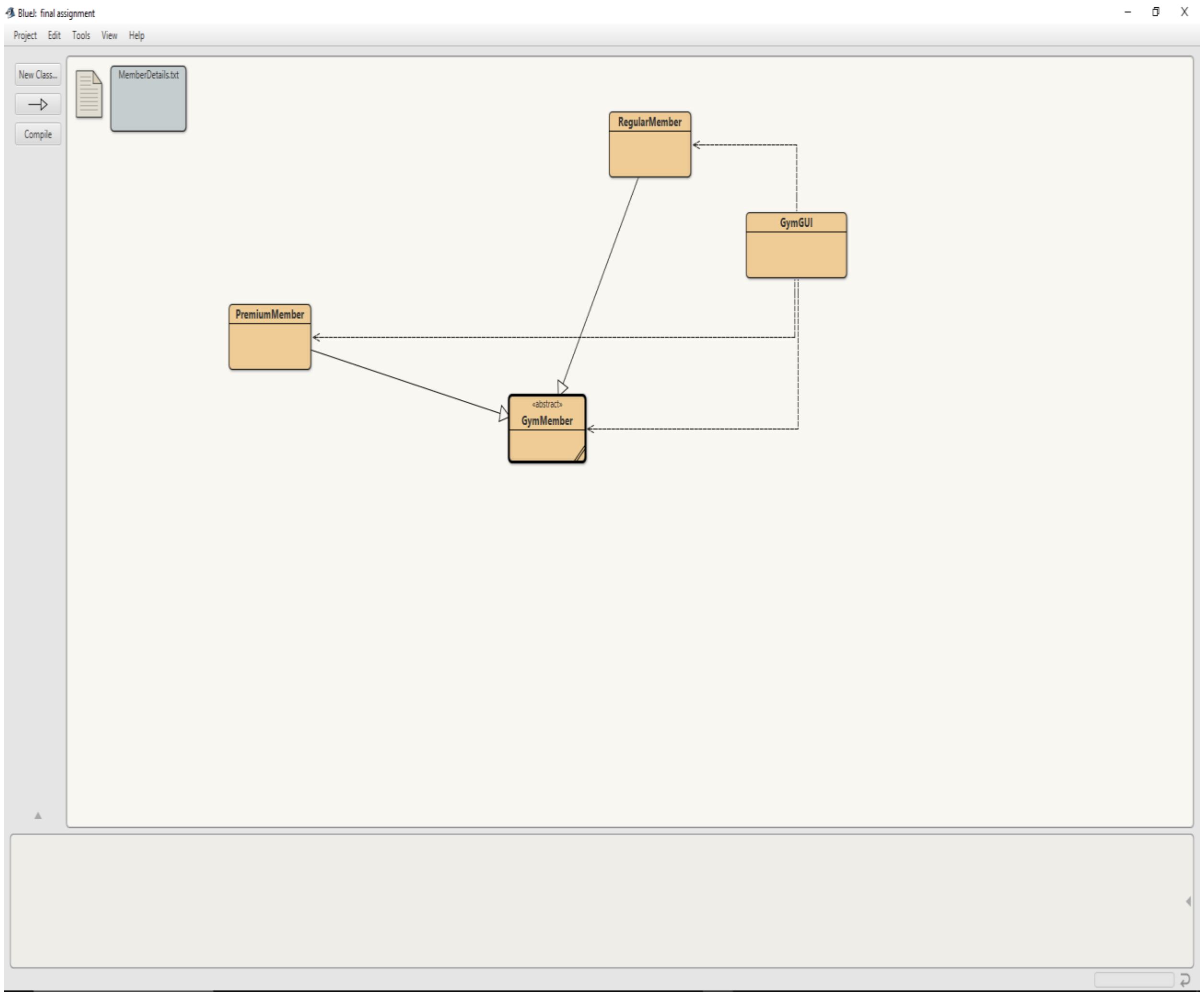


Figure 15: Combined class diagram (actual Relation)

4. Pseudocode

i. Pseudocode for GymMember Class

```
# Constructor
START creating a new GymMember with id, name, location, phone, email, gender, date of birth, membership start date
    ASSIGN the id to this member's id
    ASSIGN the name to this member's name
    ASSIGN the location to this member's location
    ASSIGN the phone to this member's phone
    ASSIGN the email to this member's email
    ASSIGN the gender to this member's gender
    ASSIGN the date of birth to this member's date of birth
    ASSIGN the membership start date to this member's membership start date
    SET this member's active status to false
    SET this member's loyalty points to 0
    SET this member's full payment status to false
FINISH creating

# resetMember
BEGIN resetting the member
    CHANGE loyalty points to 0
    CHANGE active status to false
    CHANGE full payment status to false
END resetting

# activateMembership
START activating membership
    TURN active status to true
```

```
FINISH activating

# deactivateMembership
BEGIN deactivating membership
   IF the member is active THEN
      TURN active status to false
   ELSE
      SHOW message "your membership status is Deactivated."
   END IF
END deactivating

# Abstract methods (to be defined in child classes)
DECLARE makePayment with amount (to be defined later)
DECLARE markAttendance (to be defined later)
```

ii. Pseudocode for RegularMember Class

```
# Pseudocode for RegularMember Class in Plain English

# Constructor
START creating RegularMember with id, name, location, phone, email, gender, dob,
membershipStart, referral, plan, price, isFullPayment
   CALL parent constructor with id, name, location, phone, email, gender, dob,
membershipStart
   IF referral is empty or only spaces THEN
      SHOW error "Referral cannot be empty."
   END IF
   IF plan is empty or not "basic", "standard", or "deluxe" THEN
      SHOW error "Plan must be basic, standard, or deluxe."
   END IF
   IF price is not positive THEN
      SHOW error "Price must be positive."
   END IF
   SET plan to lowercase plan
   SET price to given price
   SET referral to trimmed referral
   SET full payment status to isFullPayment
   SET eligible for upgrade to true
   SET removal reason to empty
   SET paid amount to price if isFullPayment is true, else 0
FINISH creating

# makePayment
START making payment with amount
   IF amount is negative THEN
      SHOW error "Payment amount cannot be negative."
```

```
    END IF
    CALCULATE remaining balance as price minus paid amount
    IF paid amount plus amount is more than price THEN
        SHOW error "Payment exceeds remaining balance of " plus remaining balance
    END IF
    ADD amount to paid amount
    IF paid amount is at least price THEN
        SET full payment status to true
    END IF
FINISH making

# markAttendance
START marking attendance
    IF membership is active THEN
        ADD 5 to loyalty points
    ELSE
        SHOW error "Cannot mark attendance: Membership is not activated."
    END IF
FINISH marking

# upgradePlan
START upgrading plan with newPlan
    IF newPlan is empty or not "basic", "standard", or "deluxe" THEN
        SHOW error "Invalid plan: " plus newPlan
    END IF
    IF not eligible for upgrade THEN
        SHOW error "Member is not eligible for upgrade."
    END IF
    IF loyalty points are less than 150 THEN
        SHOW error "Cannot upgrade: Member must have at least 30 attendances (150
loyalty points). Current points: " plus loyalty points
```

```
    END IF
    SET plan to lowercase newPlan
    IF plan is "basic" THEN
        SET price to 6500
    ELSE IF plan is "standard" THEN
        SET price to 12500
    ELSE IF plan is "deluxe" THEN
        SET price to 18500
    END IF
    SET paid amount to 0
    SET full payment status to false
FINISH upgrading

# revertRegularMember
START reverting regular member with removalReason
    IF removalReason is empty or only spaces THEN
        SHOW error "Removal reason is required for Regular Members."
    END IF
    CALL parent resetMember
    SET eligible for upgrade to false
    SET plan to "basic"
    SET price to 6500
    SET paid amount to 0
    SET removal reason to trimmed removalReason
FINISH reverting

# toString
START converting to string
    CREATE text builder
    ADD parent string to text builder
    ADD newline to text builder
    ADD "Type: Regular" to text builder
    ADD newline to text builder
    ADD "Plan: " plus plan to text builder
    ADD newline to text builder
    ADD "Price: " plus price to text builder
    ADD newline to text builder
    ADD "Referral: " plus referral to text builder
    ADD newline to text builder
    ADD "Paid Amount: " plus paid amount to text builder
    ADD newline to text builder
    ADD "Full Payment: " plus "Yes" if full payment, else "No" to text builder
    ADD newline to text builder
    ADD "Eligible for Upgrade: " plus "Yes" if eligible for upgrade, else "No" to text builder
    IF removal reason is not empty THEN
        ADD newline to text builder
        ADD "Removal Reason: " plus removal reason to text builder
    END IF
    RETURN text builder as string
FINISH converting
```

iii. Pseudocode for PremiumMember Class

```
# Constructor
START creating a new PremiumMember with id, name, location, phone, email, gender, date of birth, membership start date, personal trainer, discount amount, full payment status
   USE GymMember's creation with id, name, location, phone, email, gender, date of birth, membership start date
   IF personal trainer is missing or empty THEN
      SHOW error "Personal trainer name cannot be empty."
   END IF
   IF discount amount is negative or more than premium charge THEN
      SHOW error "Discount must be between 0 and " plus premium charge
   END IF
   ASSIGN trimmed personal trainer to this member's personal trainer
   ASSIGN discount amount to this member's discount amount
   ASSIGN paid amount as (premium charge minus discount amount) if full payment is true, otherwise 0
   ASSIGN full payment status to this member's full payment status
   SET removal reason to empty
FINISH creating

# makePayment
BEGIN making payment with amount
   IF amount is negative THEN
      SHOW error "Payment amount cannot be negative."
   END IF
   ADD amount to paid amount
   CALCULATE total due as premium charge minus discount amount
   IF paid amount is at least total due THEN
      TURN full payment status to true
```

```
    END IF
END making payment

# markAttendance
START marking attendance
    IF member is active THEN
        ADD 10 to loyalty points
    ELSE
        SHOW error "Cannot mark attendance: Membership is not activated."
    END IF
FINISH marking attendance

# revertPremiumMember
START reverting premium member with removal reason
    IF removal reason is missing or empty THEN
        SHOW error "Removal reason is required for Premium Members."
    END IF
    USE GymMember's reset
    SET personal trainer to empty
    TURN full payment status to false
    SET paid amount to 0
    SET discount amount to 0
    ASSIGN trimmed removal reason to this member's removal reason
FINISH reverting

# toString
BEGIN creating string description
    GET GymMember's description
    ADD newline and "Type: Premium"
    ADD newline and "Personal Trainer: " plus personal trainer
    ADD newline and "Discount Amount: " plus discount amount
    ADD newline and "Paid Amount: " plus paid amount
    ADD newline and "Full Payment: " plus "Yes" if full payment is true, else "No"
    IF removal reason is not empty THEN
        ADD newline and "Removal Reason: " plus removal reason
    END IF
    RETURN final description
END creating string
```

iv. Pseudocode for GymGUI Class

```
# Pseudocode for GymGUI Class in Plain English

# Constructor
START creating GymGUI
    CREATE a list to store members
    CREATE main window named "GymFreak"
    SET main window size to 1400 by 900
    SET main window layout to none
    MAKE main window non-resizable
    SET main window to close program when closed
    SET main window background to white
    CREATE header panel
    SET header panel position to 0, 0, 1400, 80
    SET header panel background to teal
    SET header panel layout to none
    ADD header panel to main window
    CREATE title label "GymFreak"
    SET title label position to 580, 15, 240, 50
    SET title label font to Segoe UI, bold, size 36
    SET title label color to cyan
    ADD title label to header panel
    CREATE subtitle label "Membership Management"
    SET subtitle label position to 500, 100, 400, 30
    SET subtitle label font to Segoe UI, bold, size 24
    SET subtitle label color to cyan
    ADD subtitle label to main window
    CREATE Add Member button
    SET Add Member button position to 250, 200, 300, 60
    WHEN Add Member button clicked
        CLOSE main window
    SHOW add member screen
END WHEN
ADD Add Member button to main window
CREATE Activation & Deactivation button
SET Activation & Deactivation button position to 850, 200, 300, 60
WHEN Activation & Deactivation button clicked
    CLOSE main window
    SHOW activation screen
END WHEN
ADD Activation & Deactivation button to main window
CREATE Revert Member button
SET Revert Member button position to 250, 300, 300, 60
WHEN Revert Member button clicked
    CLOSE main window
    SHOW revert member screen
END WHEN
ADD Revert Member button to main window
CREATE Display Members button
SET Display Members button position to 850, 300, 300, 60
WHEN Display Members button clicked
    CLOSE main window
    SHOW display members screen
END WHEN
ADD Display Members button to main window
CREATE Upgrade Plan button
SET Upgrade Plan button position to 250, 400, 300, 60
WHEN Upgrade Plan button clicked
    CLOSE main window
    SHOW upgrade plan screen
END WHEN
ADD Upgrade Plan button to main window
```

```
    CREATE Payment button
    SET Payment button position to 850, 400, 300, 60
    WHEN Payment button clicked
        CLOSE main window
        SHOW payment screen
    END WHEN
    ADD Payment button to main window
    CREATE Read from File button
    SET Read from File button position to 250, 500, 300, 60
    WHEN Read from File button clicked
        READ from file
    END WHEN
    ADD Read from File button to main window
    CREATE Save to File button
    SET Save to File button position to 850, 500, 300, 60
    WHEN Save to File button clicked
        SAVE to file
    END WHEN
    ADD Save to File button to main window
    CREATE main panel
    SET main panel position to 50, 100, 1300, 750
    SET main panel background to white
    SET main panel border to cyan, thickness 1
    SET main panel layout to none
    ADD main panel to main window
    SHOW main window
FINISH creating

# createStyledButton
START creating styled button with text
    CREATE button with given text
    **SET** button to change background to light cyan on hover, else pale cyan
    **SET** button font to Segoe UI, bold, size 20
    **SET** button text color to dark blue
    **SET** button border to cyan, thickness 1
    DISABLE button's default fill
    RETURN button
**FINISH** creating

# readFromFile
**START** reading from file
    FIND file named "MemberDetails.txt"
    **CREATE** display window named "Member Details"
    **SET** display window size to 1400 by 900
    **SET** display window layout to none
    **MAKE** display window non-resizable
    **SET** display window to close when closed
    **SET** display window background to white
    **CREATE** header panel
    **SET** header panel position to 0, 0, 1400, 80
    **SET** header panel background to teal
    **SET** header panel layout to none
    **ADD** header panel to display window
    **CREATE** title label "GymFreak"
    **SET** title label position to 580, 15, 240, 50
    **SET** title label font to Segoe UI, bold, size 36
    **SET** title label color to cyan
    **ADD** title label to header panel
    **CREATE** subtitle label "Member Details"
    **SET** subtitle label position to 550, 100, 300, 30
    **SET** subtitle label font to Segoe UI, bold, size 24
    **SET** subtitle label color to cyan
```

```
ADD subtitle label to display window
CREATE text area for display
SET text area font to Courier New, plain, size 16
MAKE text area non-editable
SET text area border to cyan, thickness 1
TRY
    OPEN file for reading
    CREATE text builder
    READ each line from file
    IF line exists THEN
        ADD line with newline to text builder
    END IF
    CLOSE file
    IF text builder is empty THEN
        ADD "No member details found in the file." to text builder
    END IF
    SET text area content to text builder
CATCH file error
    SET text area content to "Error reading file: " plus error message
END TRY
CREATE scroll pane with text area
SET scroll pane position to 50, 150, 1300, 650
SET scroll pane border to cyan, thickness 1
ADD scroll pane to display window
CREATE Back button
SET Back button position to 600, 820, 200, 40
SET Back button font to Segoe UI, bold, size 16
WHEN Back button clicked
    CLOSE display window
END WHEN
ADD Back button to display window
```

```
    CREATE main panel
    SET main panel position to 50, 100, 1300, 750
    SET main panel background to white
    SET main panel border to cyan, thickness 1
    SET main panel layout to none
    ADD main panel to display window
    SHOW display window
FINISH reading

# saveToFile
START saving to file
    FIND file named "MemberDetails.txt"
    TRY
        OPEN file for writing
        WRITE header with columns ID, Name, Location, Phone, Email, Membership Start, Plan, Price, Attendance, Loyalty Points, Active Status, Full Payment, Discount Amount, Net Amount Paid
        WRITE separator line
        FOR each member in list
            SET plan to empty
            SET price to empty
            SET attendance to member's loyalty points
            SET discount amount to "0"
            SET net amount paid to "0"
            IF member is RegularMember THEN
                GET RegularMember details
                SET plan to member's plan
                SET price to member's price formatted to 2 decimals
                SET discount amount to "N/A"
                SET net amount paid to price if full payment, else "0.00"
                IF removal reason is not empty THEN
```

```
          WRITE "Removal Reason: " plus removal reason
        END IF
      ELSE IF member is PremiumMember THEN
        GET PremiumMember details
        SET plan to "Premium"
        SET price to premium charge formatted to 2 decimals
        SET discount amount to discount amount formatted to 2 decimals
        SET net amount paid to paid amount formatted to 2 decimals
        IF removal reason is not empty THEN
          WRITE "Removal Reason: " plus removal reason
        END IF
      END IF
      WRITE member details formatted with ID, name, location, phone, email,
membership start, plan, price, attendance, loyalty points, active status, full payment,
discount amount, net amount paid
    END FOR
    CLOSE file
    SHOW message "Members saved successfully to MemberDetails.txt"
  CATCH file error
    SHOW error message "Error saving file: " plus error message
  END TRY
FINISH saving

# addMember
START adding member
  CREATE window named "Add Member"
  SET window size to 1400 by 900
  SET window layout to none
  MAKE window non-resizable
  SET window to close program when closed
  SET window background to white
```

```
CREATE header panel
SET header panel position to 0, 0, 1400, 80
SET header panel background to teal
SET header panel layout to none
ADD header panel to window
CREATE title label "GymFreak"
SET title label position to 580, 15, 240, 50
SET title label font to Segoe UI, bold, size 36
SET title label color to cyan
ADD title label to header panel
CREATE subtitle label "Choose Membership Type"
SET subtitle label position to 550, 100, 300, 30
SET subtitle label font to Segoe UI, bold, size 24
SET subtitle label color to cyan
ADD subtitle label to window
CREATE Add Regular Member button
SET Add Regular Member button position to 200, 200, 300, 50
SET Add Regular Member button font to Segoe UI, bold, size 16
WHEN Add Regular Member button clicked
    CLOSE window
    SHOW add regular member screen
END WHEN
ADD Add Regular Member button to window
CREATE Add Premium Member button
SET Add Premium Member button position to 550, 200, 300, 50
SET Add Premium Member button font to Segoe UI, bold, size 16
WHEN Add Premium Member button clicked
    CLOSE window
    SHOW add premium member screen
END WHEN
ADD Add Premium Member button to window
```

```
    **CREATE** Home button
    **SET** Home button position to 900, 200, 300, 50
    **SET** Home button font to Segoe UI, bold, size 16
    **WHEN** Home button clicked
        **CLOSE** window
        **SHOW** main GymGUI screen
    **END WHEN**
    **ADD** Home button to window
    **CREATE** main panel
    **SET** main panel position to 50, 100, 1300, 750
    **SET** main panel background to white
    **SET** main panel border to cyan, thickness 1
    **SET** main panel layout to none
    **ADD** main panel to window
    **SHOW** window
**FINISH** adding

# addRegularMember
**START** adding regular member
    **CREATE** window named "Add Regular Member"
    **SET** window size to 1400 by 900
    **SET** window layout to none
    **MAKE** window non-resizable
    **SET** window to close program when closed
    **SET** window background to white
    **CREATE** header panel
    **SET** header panel position to 0, 0, 1400, 80
    **SET** header panel background to teal
    **SET** header panel layout to none
    **ADD** header panel to window
    **CREATE** title label "GymFreak"
    SET title label position to 580, 15, 240, 50
    SET title label font to Segoe UI, bold, size 36
    SET title label color to cyan
    ADD title label to header panel
    CREATE subtitle label "Add Regular Member"
    SET subtitle label position to 550, 100, 300, 30
    SET subtitle label font to Segoe UI, bold, size 24
    SET subtitle label color to cyan
    ADD subtitle label to window
    CREATE plan prices label "Regular Plans: Basic (6500/-), Standard (12500/-),
Deluxe (18500/-)"
    SET plan prices label position to 200, 140, 800, 30
    SET plan prices label font to Segoe UI, bold, size 20
    SET plan prices label color to dark blue
    ADD plan prices label to window
    CREATE premium plan label "Premium Plan: 50000/-"
    SET premium plan label position to 900, 140, 300, 30
    SET premium plan label font to Segoe UI, bold, size 20
    SET premium plan label color to dark blue
    ADD premium plan label to window
    CREATE ID label "ID:"
    SET ID label position to 100, 200, 100, 30
    SET ID label font to Segoe UI, bold, size 20
    ADD ID label to window
    CREATE ID text field
    SET ID text field position to 200, 200, 200, 30
    SET ID text field font to Segoe UI, plain, size 18
    SET ID text field border to cyan, thickness 1
    ADD ID text field to window
    CREATE Name label "Name:"
    SET Name label position to 450, 200, 100, 30
    SET Name label font to Segoe UI, bold, size 20
    ADD Name label to window
    CREATE Name text field
    SET Name text field position to 550, 200, 200, 30
    SET Name text field font to Segoe UI, plain, size 18
    SET Name text field border to cyan, thickness 1
    ADD Name text field to window
    CREATE Location label "Location:"
    SET Location label position to 800, 200, 150, 30
    SET Location label font to Segoe UI, bold, size 20
    ADD Location label to window
    CREATE Location text field
    SET Location text field position to 950, 200, 200, 30
    SET Location text field font to Segoe UI, plain, size 18
    SET Location text field border to cyan, thickness 1
    ADD Location text field to window
    CREATE Phone label "Phone:"
    SET Phone label position to 100, 270, 100, 30
    SET Phone label font to Segoe UI, bold, size 20
    ADD Phone label to window
    CREATE Phone text field
    SET Phone text field position to 200, 270, 200, 30
    SET Phone text field font to Segoe UI, plain, size 18
    SET Phone text field border to cyan, thickness 1
    ADD Phone text field to window
    CREATE Email label "Email:"
    SET Email label position to 450, 270, 100, 30
    SET Email label font to Segoe UI, bold, size 20
    ADD Email label to window
    CREATE Email text field
    SET Email text field position to 550, 270, 200, 30
```

```
**SET** Email text field font to Segoe UI, plain, size 18
**SET** Email text field border to cyan, thickness 1
**ADD** Email text field to window
**CREATE** Gender label "Gender:"
**SET** Gender label position to 800, 270, 150, 30
**SET** Gender label font to Segoe UI, bold, size 20
**ADD** Gender label to window
**CREATE** Male radio button "Male"
**SET** Male radio button position to 950, 270, 80, 30
**SET** Male radio button font to Segoe UI, plain, size 18
**SET** Male radio button background to white
**ADD** Male radio button to window
**CREATE** Female radio button "Female"
**SET** Female radio button position to 950, 300, 100, 30
**SET** Female radio button font to Segoe UI, plain, size 18
**SET** Female radio button background to white
**ADD** Female radio button to window
**CREATE** Other radio button "Other"
**SET** Other radio button position to 950, 330, 90, 30
**SET** Other radio button font to Segoe UI, plain, size 18
**SET** Other radio button background to white
**ADD** Other radio button to window
**CREATE** gender group
**ADD** Male, Female, Other radio buttons to gender group
**CREATE** DOB label "DOB:"
**SET** DOB label position to 100, 380, 100, 30
**SET** DOB label font to Segoe UI, bold, size 20
**ADD** DOB label to window
**CREATE** DOB day dropdown with numbers 1 to 31
**SET** DOB day dropdown position to 200, 380, 60, 30
**SET** DOB day dropdown font to Segoe UI, plain, size 18
```

```
**ADD** DOB day dropdown to window
**CREATE** DOB month dropdown with months
**SET** DOB month dropdown position to 280, 380, 100, 30
**SET** DOB month dropdown font to Segoe UI, plain, size 18
**ADD** DOB month dropdown to window
**CREATE** DOB year dropdown with years 1950 to 2025
**SET** DOB year dropdown position to 400, 380, 100, 30
**SET** DOB year dropdown font to Segoe UI, plain, size 18
**ADD** DOB year dropdown to window
**CREATE** Membership Start label "Membership Start:"
**SET** Membership Start label position to 550, 380, 200, 30
**SET** Membership Start label font to Segoe UI, bold, size 20
**ADD** Membership Start label to window
**CREATE** Membership Start day dropdown with numbers 1 to 31
**SET** Membership Start day dropdown position to 750, 380, 60, 30
**SET** Membership Start day dropdown font to Segoe UI, plain, size 18
**ADD** Membership Start day dropdown to window
**CREATE** Membership Start month dropdown with months
**SET** Membership Start month dropdown position to 830, 380, 100, 30
**SET** Membership Start month dropdown font to Segoe UI, plain, size 18
**ADD** Membership Start month dropdown to window
**CREATE** Membership Start year dropdown with years 2023 to 2025
**SET** Membership Start year dropdown position to 950, 380, 100, 30
**SET** Membership Start year dropdown font to Segoe UI, plain, size 18
**ADD** Membership Start year dropdown to window
**CREATE** Referral label "Referral:"
**SET** Referral label position to 100, 450, 100, 30
**SET** Referral label font to Segoe UI, bold, size 20
**ADD** Referral label to window
**CREATE** Referral text field
**SET** Referral text field position to 200, 450, 200, 30
```

```
SET Referral text field font to Segoe UI, plain, size 18
SET Referral text field border to cyan, thickness 1
ADD Referral text field to window
CREATE Removal Reason label "Removal Reason:"
SET Removal Reason label position to 450, 450, 200, 30
SET Removal Reason label font to Segoe UI, bold, size 20
ADD Removal Reason label to window
CREATE Removal Reason text area
SET Removal Reason text area position to 650, 450, 200, 60
SET Removal Reason text area font to Segoe UI, plain, size 18
SET Removal Reason text area border to cyan, thickness 1
ADD Removal Reason text area to window
CREATE Plan label "Plan:"
SET Plan label position to 100, 520, 100, 30
SET Plan label font to Segoe UI, bold, size 20
ADD Plan label to window
CREATE Plan dropdown with options "basic", "standard", "deluxe"
SET Plan dropdown position to 200, 520, 200, 30
SET Plan dropdown font to Segoe UI, plain, size 18
ADD Plan dropdown to window
CREATE Clear button
SET Clear button position to 355, 650, 200, 40
SET Clear button font to Segoe UI, bold, size 16
WHEN Clear button clicked
    CLEAR ID text field
    CLEAR Name text field
    CLEAR Location text field
    CLEAR Phone text field
    CLEAR Email text field
    CLEAR Referral text field
    CLEAR Removal Reason text area
```

```
        RESET DOB day dropdown to first option
        RESET DOB month dropdown to first option
        RESET DOB year dropdown to first option
        RESET Membership Start day dropdown to first option
        RESET Membership Start month dropdown to first option
        RESET Membership Start year dropdown to first option
        RESET Plan dropdown to first option
        CLEAR gender selection
    END WHEN
    ADD Clear button to window
    CREATE Add Regular Member button
    SET Add Regular Member button position to 575, 650, 200, 40
    SET Add Regular Member button font to Segoe UI, bold, size 16
    WHEN Add Regular Member button clicked
        TRY
            GET ID from ID text field
            GET name from Name text field
            GET location from Location text field
            GET phone from Phone text field
            GET email from Email text field
            GET referral from Referral text field
            GET removal reason from Removal Reason text area
            GET plan from Plan dropdown
            GET full payment status from checkbox
            IF ID is empty THEN
                SHOW error "ID cannot be empty."
            END IF
            CONVERT ID to number
            IF ID is not a number THEN
                SHOW error "ID must be a numeric value."
            END IF
```

```
IF ID is not positive THEN
    SHOW error "ID must be a positive number."
END IF
FOR each member in list
    IF member ID equals ID THEN
        SHOW error "Member ID already exists."
    END IF
END FOR
IF name is empty or contains non-letters THEN
    SHOW error "Name must contain letters only."
END IF
IF phone is empty or not 10 digits THEN
    SHOW error "Phone must be a 10-digit number."
END IF
IF location is empty THEN
    SHOW error "Location cannot be empty."
END IF
IF email is empty or invalid format THEN
    SHOW error "Invalid email format."
END IF
IF referral is empty THEN
    SHOW error "Referral source cannot be empty."
END IF
IF plan is not selected THEN
    SHOW error "Please select a plan."
END IF
GET gender from radio buttons
IF gender is not selected THEN
    SHOW error "Please select a gender."
END IF
IF DOB day, month, or year is not selected THEN
```

```
            SHOW error "Please select a valid date of birth."
         END IF
         COMBINE DOB as day, month, year
         IF Membership Start day, month, or year is not selected THEN
            SHOW error "Please select a valid membership start date."
         END IF
         COMBINE Membership Start as day, month, year
         SET price based on plan
         IF plan is "basic" THEN
            SET price to 6500
         ELSE IF plan is "standard" THEN
            SET price to 12500
         ELSE IF plan is "deluxe" THEN
            SET price to 18500
         ELSE
            SHOW error "Invalid plan selected."
         END IF
         CREATE new RegularMember with ID, name, location, phone, email, gender,
DOB, Membership Start, referral, plan, price, full payment
         IF full payment is true THEN
            PROCESS payment of price
         END IF
         ADD RegularMember to list
         SHOW message "Regular Member added successfully! Price: " plus price
         TRIGGER Clear button
      CATCH input error
         SHOW error message with error details
      CATCH any error
         SHOW error message "Unexpected error occurred: " plus error details
      END TRY
   END WHEN
```

```
    ADD Add Regular Member button to window
    CREATE Home button
    SET Home button position to 795, 650, 200, 40
    SET Home button font to Segoe UI, bold, size 16
    WHEN Home button clicked
        CLOSE window
        SHOW main GymGUI screen
    END WHEN
    ADD Home button to window
    CREATE main panel
    SET main panel position to 50, 100, 1300, 750
    SET main panel background to white
    SET main panel border to cyan, thickness 1
    SET main panel layout to none
    ADD main panel to window
    SHOW window
FINISH adding

# addPremiumMember
START adding premium member
    CREATE window named "Add Premium Member"
    SET window size to 1400 by 900
    SET window layout to none
    MAKE window non-resizable
    SET window to close program when closed
    SET window background to white
    CREATE header panel
    SET header panel position to 0, 0, 1400, 80
    SET header panel background to teal
    SET header panel layout to none
    ADD header panel to window
    CREATE title label "GymFreak"
    SET title label position to 580, 15, 240, 50
    SET title label font to Segoe UI, bold, size 36
    SET title label color to cyan
    ADD title label to header panel
    CREATE subtitle label "Add Premium Member"
    SET subtitle label position to 550, 100, 300, 30
    SET subtitle label font to Segoe UI, bold, size 24
    SET subtitle label color to cyan
    ADD subtitle label to window
    CREATE plan prices label "Regular Plans: Basic (6500/-), Standard (12500/-), Deluxe (18500/-)"
    SET plan prices label position to 200, 140, 800, 30
    SET plan prices label font to Segoe UI, bold, size 20
    SET plan prices label color to dark blue
    ADD plan prices label to window
    CREATE premium plan label "Premium Plan: 50000/-"
    SET premium plan label position to 900, 140, 300, 30
    SET premium plan label font to Segoe UI, bold, size 20
    SET premium plan label color to dark blue
    ADD premium plan label to window
    CREATE ID label "ID:"
    SET ID label position to 100, 200, 100, 30
    SET ID label font to Segoe UI, bold, size 20
    ADD ID label to window
    CREATE ID text field
    SET ID text field position to 200, 200, 200, 30
    SET ID text field font to Segoe UI, plain, size 18
    SET ID text field border to cyan, thickness 1
    ADD ID text field to window
    CREATE Name label "Name:"
```

```
**SET** Name label position to 450, 200, 100, 30
**SET** Name label font to Segoe UI, bold, size 20
**ADD** Name label to window
**CREATE** Name text field
**SET** Name text field position to 550, 200, 200, 30
**SET** Name text field font to Segoe UI, plain, size 18
**SET** Name text field border to cyan, thickness 1
**ADD** Name text field to window
**CREATE** Location label "Location:"
**SET** Location label position to 800, 200, 150, 30
**SET** Location label font to Segoe UI, bold, size 20
**ADD** Location label to window
**CREATE** Location text field
**SET** Location text field position to 950, 200, 200, 30
**SET** Location text field font to Segoe UI, plain, size 18
**SET** Location text field border to cyan, thickness 1
**ADD** Location text field to window
**CREATE** Phone label "Phone:"
**SET** Phone label position to 100, 270, 100, 30
**SET** Phone label font to Segoe UI, bold, size 20
**ADD** Phone label to window
**CREATE** Phone text field
**SET** Phone text field position to 200, 270, 200, 30
**SET** Phone text field font to Segoe UI, plain, size 18
**SET** Phone text field border to cyan, thickness 1
**ADD** Phone text field to window
**CREATE** Email label "Email:"
**SET** Email label position to 450, 270, 100, 30
**SET** Email label font to Segoe UI, bold, size 20
**ADD** Email label to window
**CREATE** Email text field
```

```
**SET** Email text field position to 550, 270, 200, 30
**SET** Email text field font to Segoe UI, plain, size 18
**SET** Email text field border to cyan, thickness 1
**ADD** Email text field to window
**CREATE** Gender label "Gender:"
**SET** Gender label position to 800, 270, 150, 30
**SET** Gender label font to Segoe UI, bold, size 20
**ADD** Gender label to window
**CREATE** Male radio button "Male"
**SET** Male radio button position to 950, 270, 80, 30
**SET** Male radio button font to Segoe UI, plain, size 18
**SET** Male radio button background to white
**ADD** Male radio button to window
**CREATE** Female radio button "Female"
**SET** Female radio button position to 950, 300, 100, 30
**SET** Female radio button font to Segoe UI, plain, size 18
**SET** Female radio button background to white
**ADD** Female radio button to window
**CREATE** Other radio button "Other"
**SET** Other radio button position to 950, 330, 90, 30
**SET** Other radio button font to Segoe UI, plain, size 18
**SET** Other radio button background to white
**ADD** Other radio button to window
**CREATE** gender group
**ADD** Male, Female, Other radio buttons to gender group
**CREATE** DOB label "DOB:"
**SET** DOB label position to 100, 380, 100, 30
**SET** DOB label font to Segoe UI, bold, size 20
**ADD** DOB label to window
**CREATE** DOB day dropdown with numbers 1 to 31
**SET** DOB day dropdown position to 200, 380, 60, 30
```

```
**SET** DOB day dropdown font to Segoe UI, plain, size 18
**ADD** DOB day dropdown to window
**CREATE** DOB month dropdown with months
**SET** DOB month dropdown position to 280, 380, 100, 30
**SET** DOB month dropdown font to Segoe UI, plain, size 18
**ADD** DOB month dropdown to window
**CREATE** DOB year dropdown with years 1950 to 2025
**SET** DOB year dropdown position to 400, 380, 100, 30
**SET** DOB year dropdown font to Segoe UI, plain, size 18
**ADD** DOB year dropdown to window
**CREATE** Membership Start label "Membership Start:"
**SET** Membership Start label position to 550, 380, 200, 30
**SET** Membership Start label font to Segoe UI, bold, size 20
**ADD** Membership Start label to window
**CREATE** Membership Start day dropdown with numbers 1 to 31
**SET** Membership Start day dropdown position to 750, 380, 60, 30
**SET** Membership Start day dropdown font to Segoe UI, plain, size 18
**ADD** Membership Start day dropdown to window
**CREATE** Membership Start month dropdown with months
**SET** Membership Start month dropdown position to 830, 380, 100, 30
**SET** Membership Start month dropdown font to Segoe UI, plain, size 18
**ADD** Membership Start month dropdown to window
**CREATE** Membership Start year dropdown with years 2023 to 2025
**SET** Membership Start year dropdown position to 950, 380, 100, 30
**SET** Membership Start year dropdown font to Segoe UI, plain, size 18
**ADD** Membership Start year dropdown to window
**CREATE** Personal Trainer label "Personal Trainer:"
**SET** Personal Trainer label position to 100, 450, 200, 30
**SET** Personal Trainer label font to Segoe UI, bold, size 20
**ADD** Personal Trainer label to window
**CREATE** Personal Trainer text field
```

```
SET Personal Trainer text field position to 300, 450, 200, 30
SET Personal Trainer text field font to Segoe UI, plain, size 18
SET Personal Trainer text field border to cyan, thickness 1
ADD Personal Trainer text field to window
CREATE Paid in Full checkbox "Paid in Full"
SET Paid in Full checkbox position to 950, 450, 150, 30
SET Paid in Full checkbox font to Segoe UI, plain, size 18
SET Paid in Full checkbox background to white
ADD Paid in Full checkbox to window
CREATE Clear button
SET Clear button position to 355, 650, 200, 40
SET Clear button font to Segoe UI, bold, size 16
WHEN Clear button clicked
    CLEAR ID text field
    CLEAR Name text field
    CLEAR Location text field
    CLEAR Phone text field
    CLEAR Email text field
    CLEAR Personal Trainer text field
    SET Discount text field to "0"
    UNCHECK Paid in Full checkbox
    RESET DOB day dropdown to first option
    RESET DOB month dropdown to first option
    RESET DOB year dropdown to first option
    RESET Membership Start day dropdown to first option
    RESET Membership Start month dropdown to first option
    RESET Membership Start year dropdown to first option
    CLEAR gender selection
END WHEN
ADD Clear button to window
CREATE Add Premium Member button
```

```
SET Add Premium Member button position to 575, 650, 200, 40
SET Add Premium Member button font to Segoe UI, bold, size 16
WHEN Add Premium Member button clicked
    TRY
        GET ID from ID text field
        GET name from Name text field
        GET location from Location text field
        GET phone from Phone text field
        GET email from Email text field
        GET personal trainer from Personal Trainer text field
        GET discount from Discount text field
        GET full payment status from checkbox
        IF ID is empty THEN
            SHOW error "ID cannot be empty."
        END IF
        CONVERT ID to number
        IF ID is not a number THEN
            SHOW error "ID must be a numeric value."
        END IF
        IF ID is not positive THEN
            SHOW error "ID must be a positive number."
        END IF
        FOR each member in list
            IF member ID equals ID THEN
                SHOW error "Member ID already exists."
            END IF
        END FOR
        IF name is empty or contains non-letters THEN
            SHOW error "Name must contain letters only."
        END IF
        IF phone is empty or not 10 digits THEN
            SHOW error "Phone must be a 10-digit number."
        END IF
        IF location is empty THEN
            SHOW error "Location cannot be empty."
        END IF
        IF email is empty or invalid format THEN
            SHOW error "Invalid email format."
        END IF
        IF personal trainer is empty THEN
            SHOW error "Personal trainer name cannot be empty."
        END IF
        SET discount amount to 0 if discount is empty
        CONVERT discount to number
        IF discount is not a number THEN
            SHOW error "Discount must be a valid number."
        END IF
        IF discount is negative or more than 50000 THEN
            SHOW error "Discount must be between 0 and 50000."
        END IF
        GET gender from radio buttons
        IF gender is not selected THEN
            SHOW error "Please select a gender."
        END IF
        IF DOB day, month, or year is not selected THEN
            SHOW error "Please select a valid date of birth."
        END IF
        COMBINE DOB as day, month, year
        IF Membership Start day, month, or year is not selected THEN
            SHOW error "Please select a valid membership start date."
        END IF
        COMBINE Membership Start as day, month, year
```

```
         CREATE new PremiumMember with ID, name, location, phone, email, gender,
DOB, Membership Start, personal trainer, discount amount, full payment
         ADD PremiumMember to list
         CALCULATE amount due as 50000 minus discount amount
         SHOW message "Premium Member added successfully! Amount Due: " plus
amount due plus " (Paid in Full)" if full payment is true
         TRIGGER Clear button
      CATCH input error
         SHOW error message with error details
      CATCH any error
         SHOW error message "Unexpected error occurred: " plus error details
      END TRY
   END WHEN
   ADD Add Premium Member button to window
   CREATE Home button
   SET Home button position to 795, 650, 200, 40
   SET Home button font to Segoe UI, bold, size 16
   WHEN Home button clicked
      CLOSE window
      SHOW main GymGUI screen
   END WHEN
   ADD Home button to window
   CREATE main panel
   SET main panel position to 50, 100, 1300, 750
   SET main panel background to white
   SET main panel border to cyan, thickness 1
   SET main panel layout to none
   ADD main panel to window
   SHOW window
FINISH adding
```

```
# activation
START managing activation
    CREATE window named "Activation & Deactivation"
    SET window size to 1400 by 900
    SET window layout to none
    MAKE window non-resizable
    SET window to close program when closed
    SET window background to white
    CREATE header panel
    SET header panel position to 0, 0, 1400, 80
    SET header panel background to teal
    SET header panel layout to none
    ADD header panel to window
    CREATE title label "GymFreak"
    SET title label position to 580, 15, 240, 50
    SET title label font to Segoe UI, bold, size 36
    SET title label color to cyan
    ADD title label to header panel
    CREATE subtitle label "Activation & Deactivation"
    SET subtitle label position to 550, 100, 300, 30
    SET subtitle label font to Segoe UI, bold, size 24
    SET subtitle label color to cyan
    ADD subtitle label to window
    CREATE ID label "Enter Member ID:"
    SET ID label position to 50, 190, 150, 30
    SET ID label font to Segoe UI, bold, size 20
    ADD ID label to window
    CREATE ID text field
    SET ID text field position to 200, 190, 200, 30
    SET ID text field font to Segoe UI, plain, size 18
    SET ID text field border to cyan, thickness 1
```

```
ADD ID text field to window
CREATE Activate Membership button
SET Activate Membership button position to 200, 240, 200, 40
SET Activate Membership button font to Segoe UI, bold, size 16
WHEN Activate Membership button clicked
    TRY
        GET ID from ID text field
        IF ID is empty THEN
            SHOW error "ID cannot be empty."
        END IF
        CONVERT ID to number
        IF ID is not a number THEN
            SHOW error "ID must be a numeric value."
        END IF
        IF ID is not positive THEN
            SHOW error "ID must be a positive number."
        END IF
        SET target member to none
        FOR each member in list
            IF member ID equals ID THEN
                SET target member to member
                BREAK
            END IF
        END FOR
        IF target member is none THEN
            SHOW error "Member ID not found."
        END IF
        IF target member is active THEN
            SHOW error "Membership is already active."
        END IF
        ACTIVATE target member's membership
        SHOW message "Membership activated successfully!"
        CLEAR ID text field
    CATCH input error
        SHOW error message with error details
    CATCH state error
        SHOW error message with error details
    CATCH any error
        SHOW error message "Unexpected error occurred: " plus error details
    END TRY
END WHEN
ADD Activate Membership button to window
CREATE Deactivate Membership button
SET Deactivate Membership button position to 450, 240, 200, 40
SET Deactivate Membership button font to Segoe UI, bold, size 16
WHEN Deactivate Membership button clicked
    TRY
        GET ID from ID text field
        IF ID is empty THEN
            SHOW error "ID cannot be empty."
        END IF
        CONVERT ID to number
        IF ID is not a number THEN
            SHOW error "ID must be a numeric value."
        END IF
        IF ID is not positive THEN
            SHOW error "ID must be a positive number."
        END IF
        SET target member to none
        FOR each member in list
            IF member ID equals ID THEN
                SET target member to member
```

```
          BREAK
        END IF
      END FOR
      IF target member is none THEN
        SHOW error "Member ID not found."
      END IF
      IF target member is not active THEN
        SHOW error "Membership is already deactivated."
      END IF
      DEACTIVATE target member's membership
      SHOW message "Membership deactivated successfully!"
      CLEAR ID text field
    CATCH input error
      SHOW |state error
      SHOW error message with error details
    CATCH any error
      SHOW error message "Unexpected error occurred: " plus error details
    END TRY
  END WHEN
  ADD Deactivate Membership button to window
  CREATE Mark Attendance button
  SET Mark Attendance button position to 700, 240, 200, 40
  SET Mark Attendance button font to Segoe UI, bold, size 16
  WHEN Mark Attendance button clicked
    TRY
      GET ID from ID text field
      IF ID is empty THEN
        SHOW error "ID cannot be empty."
      END IF
      CONVERT ID to number
      IF ID is not a number THEN
```

```
            SHOW error "ID must be a numeric value."
          END IF
          IF ID is not positive THEN
            SHOW error "ID must be a positive number."
          END IF
          SET target member to none
          FOR each member in list
            IF member ID equals ID THEN
              SET target member to member
              BREAK
            END IF
          END FOR
          IF target member is none THEN
            SHOW error "Member ID not found."
          END IF
          MARK target member's attendance
          SET member type to "Regular" if target member is RegularMember, else
"Premium"
          SHOW message "Attendance marked for " plus member type plus " Member!
Loyalty Points: " plus target member's loyalty points
          CLEAR ID text field
        CATCH input error
          SHOW error message with error details
        CATCH state error
          SHOW error message "Cannot mark attendance: Membership is not activated."
        CATCH any error
          SHOW error message "Unexpected error occurred: " plus error details
        END TRY
      END WHEN
      ADD Mark Attendance button to window
      CREATE Home button
```

```
    SET Home button position to 950, 240, 200, 40
    SET Home button font to Segoe UI, bold, size 16
    WHEN Home button clicked
        CLOSE window
        SHOW main GymGUI screen
    END WHEN
    ADD Home button to window
    CREATE main panel
    SET main panel position to 50, 100, 1300, 750
    SET main panel background to white
    SET main panel border to cyan, thickness 1
    SET main panel layout to none
    ADD main panel to window
    SHOW window
FINISH managing

# addRevertmember
START reverting member
    CREATE window named "Revert Member"
    SET window size to 1400 by 900
    SET window layout to none
    MAKE window non-resizable
    SET window to close program when closed
    SET window background to white
    CREATE header panel
    SET header panel position to 0, 0, 1400, 80
    SET header panel background to teal
    SET header panel layout to none
    ADD header panel to window
    CREATE title label "GymFreak"
    SET title label position to 580, 15, 240, 50
```

```
**SET** title label font to Segoe UI, bold, size 36
**SET** title label color to cyan
**ADD** title label to header panel
**CREATE** subtitle label "Revert Member"
**SET** subtitle label position to 550, 100, 300, 30
**SET** subtitle label font to Segoe UI, bold, size 24
**SET** subtitle label color to cyan
**ADD** subtitle label to window
**CREATE** ID label "Enter Member ID:"
**SET** ID label position to 50, 190, 150, 30
**SET** ID label font to Segoe UI, bold, size 20
**ADD** ID label to window
**CREATE** ID text field
**SET** ID text field position to 200, 190, 300, 30
**SET** ID text field font to Segoe UI, plain, size 18
**SET** ID text field border to cyan, thickness 1
**ADD** ID text field to window
**CREATE** Removal Reason label "Removal Reason:"
**SET** Removal Reason label position to 50, 240, 250, 30
**SET** Removal Reason label font to Segoe UI, bold, size 20
**ADD** Removal Reason label to window
**CREATE** Removal Reason text area
**SET** Removal Reason text area font to Segoe UI, plain, size 18
**ENABLE** text wrapping for Removal Reason text area
**SET** Removal Reason text area position to 300, 240, 300, 60
**SET** Removal Reason text area border to cyan, thickness 1
**CREATE** scroll pane for Removal Reason text area
**SET** scroll pane position to 300, 240, 300, 60
**SET** scroll pane border to cyan, thickness 1
**ADD** scroll pane to window
**CREATE** note label "Required for all members"
```

```
SET note label position to 300, 300, 300, 20
SET note label font to Segoe UI, italic, size 14
SET note label color to dark blue
ADD note label to window
CREATE Revert Member button
SET Revert Member button position to 200, 330, 200, 40
SET Revert Member button font to Arial, bold, size 16
WHEN Revert Member button clicked
    TRY
        GET ID from ID text field
        GET removal reason from Removal Reason text area
        IF ID is empty THEN
            SHOW error "ID cannot be empty."
        END IF
        CONVERT ID to number
        IF ID is not a number THEN
            SHOW error "ID must be a numeric value."
        END IF
        IF ID is not positive THEN
            SHOW error "ID must be a positive number."
        END IF
        IF removal reason is empty THEN
            SHOW error "Removal reason is required for all members."
        END IF
        SET target member to none
        FOR each member in list
            IF member ID equals ID THEN
                SET target member to member
                BREAK
            END IF
        END FOR
```

```
            IF target member is none THEN
                SHOW error "Member ID not found."
            END IF
            IF target member is RegularMember THEN
                REVERT RegularMember with removal reason
                SHOW message "Regular Member reverted successfully! Plan set to basic,
reason: " plus removal reason
            ELSE IF target member is PremiumMember THEN
                REVERT PremiumMember with removal reason
                SHOW message "Premium Member reverted successfully! Trainer and
payments cleared, reason: " plus removal reason
            END IF
            CLEAR ID text field
            CLEAR Removal Reason text area
        CATCH input error
            SHOW error message with error details
        CATCH any error
            SHOW error message "Unexpected error occurred: " plus error details
        END TRY
    END WHEN
    ADD Revert Member button to window
    CREATE Home button
    SET Home button position to 450, 330, 200, 40
    SET Home button font to Segoe UI, bold, size 16
    WHEN Home button clicked
        CLOSE window
        SHOW main GymGUI screen
    END WHEN
    ADD Home button to window
    CREATE main panel
    SET main panel position to 50, 100, 1300, 750
    SET main panel background to white
    SET main panel border to cyan, thickness 1
    SET main panel layout to none
    ADD main panel to window
    SHOW window
FINISH reverting

# display
START displaying members
    CREATE window named "Display Members"
    SET window size to 1400 by 900
    SET window layout to none
    MAKE window non-resizable
    SET window to close program when closed
    SET window background to white
    CREATE header panel
    SET header panel position to 0, 0, 1400, 80
    SET header panel background to teal
    SET header panel layout to none
    ADD header panel to window
    CREATE title label "GymFreak"
    SET title label position to 580, 15, 240, 50
    SET title label font to Segoe UI, bold, size 36
    SET title label color to cyan
    ADD title label to header panel
    CREATE subtitle label "Display Members"
    SET subtitle label position to 550, 100, 300, 30
    SET subtitle label font to Segoe UI, bold, size 24
    SET subtitle label color to cyan
    ADD subtitle label to window
    CREATE text area for display
```

```
SET text area font to Segoe UI, plain, size 18
MAKE text area non-editable
SET text area border to cyan, thickness 1
TRY
   CREATE text builder
   IF list is empty THEN
      ADD "No members to display." to text builder
   ELSE
      FOR each member in list
         IF member is null THEN
            SKIP to next member
         END IF
         ADD "ID: " plus member ID to text builder
         ADD "Name: " plus member name or "N/A" if null to text builder
         ADD "Location: " plus member location or "N/A" if null to text builder
         ADD "Phone: " plus member phone or "N/A" if null to text builder
         ADD "Email: " plus member email or "N/A" if null to text builder
         ADD "Gender: " plus member gender or "N/A" if null to text builder
         ADD "DOB: " plus member DOB or "N/A" if null to text builder
         ADD "Membership Start: " plus member membership start or "N/A" if null to
text builder
         ADD "Active Status: " plus "Active" if active, else "Inactive" to text builder
         ADD "Loyalty Points: " plus member loyalty points to text builder
         IF member is RegularMember THEN
            ADD "Type: Regular" to text builder
            ADD "Plan: " plus member plan or "N/A" if null to text builder
            ADD "Price: " plus member price to text builder
            ADD "Referral: " plus member referral or "N/A" if null to text builder
            ADD "Full Payment: " plus "Yes" if full payment, else "No" to text builder
            ADD "Eligible for Upgrade: " plus "Yes" if eligible, else "No" to text builder
            IF removal reason is not empty THEN
```

```
                ADD "Removal Reason: " plus removal reason to text builder
            END IF
        ELSE IF member is PremiumMember THEN
            ADD "Type: Premium" to text builder
            ADD "Personal Trainer: " plus member personal trainer or "N/A" if null to
text builder
            ADD "Discount Amount: " plus member discount amount to text builder
            ADD "Paid Amount: " plus member paid amount to text builder
            ADD "Full Payment: " plus "Yes" if full payment, else "No" to text builder
            IF removal reason is not empty THEN
                ADD "Removal Reason: " plus removal reason to text builder
            END IF
        END IF
        ADD separator line to text builder
      END FOR
    END IF
    SET text area content to text builder
  CATCH any error
    SET text area content to "Error displaying members: " plus error message
  END TRY
  CREATE scroll pane with text area
  SET scroll pane position to 50, 190, 1250, 600
  SET scroll pane border to cyan, thickness 1
  ADD scroll pane to window
  CREATE Home button
  SET Home button position to 600, 800, 200, 40
  SET Home button font to Segoe UI, bold, size 16
  WHEN Home button clicked
    CLOSE window
    SHOW main GymGUI screen
  END WHEN
```

```
    **ADD** Home button to window
    **CREATE** main panel
    **SET** main panel position to 50, 100, 1300, 750
    **SET** main panel background to white
    **SET** main panel border to cyan, thickness 1
    **SET** main panel layout to none
    **ADD** main panel to window
    **SHOW** window
**FINISH** displaying

# upgradePlan
**START** upgrading plan
    **CREATE** window named "Upgrade Plan"
    **SET** window size to 1400 by 900
    **SET** window layout to none
    **MAKE** window non-resizable
    **SET** window to close program when closed
    **SET** window background to white
    **CREATE** header panel
    **SET** header panel position to 0, 0, 1400, 80
    **SET** header panel background to teal
    **SET** header panel layout to none
    **ADD** header panel to window
    **CREATE** title label "GymFreak"
    **SET** title label position to 580, 15, 240, 50
    **SET** title label font to Segoe UI, bold, size 36
    **SET** title label color to cyan
    **ADD** title label to header panel
    **CREATE** subtitle label "Upgrade Plan"
    **SET** subtitle label position to 550, 100, 300, 30
    **SET** subtitle label font to Segoe UI, bold, size 24
```

```
SET subtitle label color to cyan
ADD subtitle label to window
CREATE ID label "Enter Member ID:"
SET ID label position to 50, 190, 150, 30
SET ID label font to Segoe UI, bold, size 20
ADD ID label to window
CREATE ID text field
SET ID text field position to 200, 190, 200, 30
SET ID text field font to Segoe UI, plain, size 18
SET ID text field border to cyan, thickness 1
ADD ID text field to window
CREATE New Plan label "New Plan:"
SET New Plan label position to 450, 190, 150, 30
SET New Plan label font to Segoe UI, bold, size 20
ADD New Plan label to window
CREATE Plan dropdown with options "basic", "standard", "deluxe"
SET Plan dropdown position to 600, 190, 200, 30
SET Plan dropdown font to Segoe UI, plain, size 18
ADD Plan dropdown to window
CREATE Upgrade Plan button
SET Upgrade Plan button position to 200, 240, 200, 40
SET Upgrade Plan button font to Segoe UI, bold, size 16
WHEN Upgrade Plan button clicked
    TRY
        GET ID from ID text field
        IF ID is empty THEN
            SHOW error "ID cannot be empty."
        END IF
        CONVERT ID to number
        IF ID is not a number THEN
            SHOW error "ID must be a numeric value."
        END IF
        IF ID is not positive THEN
            SHOW error "ID must be a positive number."
        END IF
        GET new plan from Plan dropdown
        IF new plan is not selected THEN
            SHOW error "Please select a new plan."
        END IF
        SET target member to none
        FOR each member in list
            IF member ID equals ID THEN
                SET target member to member
                BREAK
            END IF
        END FOR
        IF target member is none THEN
            SHOW error "Member ID not found."
        END IF
        IF target member is not RegularMember THEN
            SHOW error "Only Regular Members can upgrade plans."
        END IF
        UPGRADE RegularMember's plan to new plan
        GET new price from RegularMember
        SHOW message "Plan upgraded to " plus new plan plus " successfully! New
Price: " plus new price
        CLEAR ID text field
        RESET Plan dropdown to first option
    CATCH input error
        SHOW error message with error details
    CATCH any error
        SHOW error message "Unexpected error occurred: " plus error details
        END TRY
    END WHEN
    ADD Upgrade Plan button to window
    CREATE Home button
    SET Home button position to 450, 240, 200, 40
    SET Home button font to Segoe UI, bold, size 16
    WHEN Home button clicked
        CLOSE window
        SHOW main GymGUI screen
    END WHEN
    ADD Home button to window
    CREATE main panel
    SET main panel position to 50, 100, 1300, 750
    SET main panel background to white
    SET main panel border to cyan, thickness 1
    SET main panel layout to none
    ADD main panel to window
    SHOW window
FINISH upgrading

# processPayment
START processing payment
    CREATE window named "Process Payment"
    SET window size to 1400 by 900
    SET window layout to none
    MAKE window non-resizable
    SET window to close program when closed
    SET window background to white
    CREATE header panel
    SET header panel position to 0, 0, 1400, 80
    SET header panel background to teal
    SET header panel layout to none
    ADD header panel to window
    CREATE title label "GymFreak"
    SET title label position to 580, 15, 240, 50
    SET title label font to Segoe UI, bold, size 36
    SET title label color to cyan
    ADD title label to header panel
    CREATE subtitle label "Process Payment"
    SET subtitle label position to 550, 100, 300, 30
    SET subtitle label font to Segoe UI, bold, size 24
    SET subtitle label color to cyan
    ADD subtitle label to window
    CREATE ID label "Enter Member ID:"
    SET ID label position to 50, 190, 150, 30
    SET ID label font to Segoe UI, bold, size 20
    ADD ID label to window
    CREATE ID text field
    SET ID text field position to 200, 190, 300, 30
    SET ID text field font to Segoe UI, plain, size 18
    SET ID text field border to cyan, thickness 1
    ADD ID text field to window
    CREATE Amount label "Enter Amount:"
    SET Amount label position to 50, 240, 150, 30
    SET Amount label font to Segoe UI, bold, size 20
    ADD Amount label to window
    CREATE Amount text field
    SET Amount text field position to 200, 240, 300, 30
    SET Amount text field font to Segoe UI, plain, size 18
    SET Amount text field border to cyan, thickness 1
    ADD Amount text field to window
    CREATE status text area
```

```
SET status text area font to Segoe UI, plain, size 18
MAKE status text area non-editable
ENABLE text wrapping for status text area
SET status text area border to cyan, thickness 1
CREATE scroll pane for status text area
SET scroll pane position to 50, 350, 600, 200
SET scroll pane border to cyan, thickness 1
ADD scroll pane to window
CREATE Calculate Discount button
SET Calculate Discount button position to 200, 290, 200, 40
SET Calculate Discount button font to Arial, bold, size 16
WHEN Calculate Discount button clicked
    TRY
        GET ID from ID text field
        IF ID is empty THEN
            SHOW error "ID cannot be empty."
        END IF
        CONVERT ID to number
        IF ID is not a number THEN
            SHOW error "ID must be a numeric value."
        END IF
        IF ID is not positive THEN
            SHOW error "ID must be a positive number."
        END IF
        SET target member to none
        FOR each member in list
            IF member ID equals ID THEN
                SET target member to member
                BREAK
            END IF
        END FOR
```

```
      IF target member is none THEN
        SHOW error "Member ID not found."
      END IF
      IF target member is not PremiumMember THEN
        SHOW error "Discounts are only applicable to Premium Members."
      END IF
      GET premium charge from PremiumMember
      CALCULATE discount as 10% of premium charge
      SET PremiumMember's discount amount to discount
      SET status text to "Discount applied successfully!" plus newline
      ADD "Member ID: " plus ID to status text
      ADD "Discount Amount: " plus discount formatted to 2 decimals to status text
      ADD "Total Due: " plus (premium charge minus discount) formatted to 2
decimals to status text
    CATCH input error
      SET status text to "Error: " plus error message
    CATCH any error
      SET status text to "Unexpected error occurred: " plus error message
    END TRY
  END WHEN
  ADD Calculate Discount button to window
  CREATE Process Payment button
  SET Process Payment button position to 450, 290, 200, 40
  SET Process Payment button font to Arial, bold, size 16
  WHEN Process Payment button clicked
    TRY
      GET ID from ID text field
      GET amount from Amount text field
      IF ID is empty THEN
        SHOW error "ID cannot be empty."
      END IF
```

```
IF amount is empty THEN
   SHOW error "Amount cannot be empty."
END IF
CONVERT ID to number
CONVERT amount to number
IF ID is not a number or amount is not a number THEN
   SHOW error "ID and Amount must be numeric values."
END IF
IF ID is not positive THEN
   SHOW error "ID must be a positive number."
END IF
IF amount is not positive THEN
   SHOW error "Amount must be a positive number."
END IF
SET target member to none
FOR each member in list
   IF member ID equals ID THEN
      SET target member to member
      BREAK
   END IF
END FOR
IF target member is none THEN
   SHOW error "Member ID not found."
END IF
PROCESS target member's payment with amount
CREATE status text
ADD "Payment processed successfully!" plus newline to status text
ADD "Member ID: " plus ID to status text
ADD "Amount Paid: " plus amount formatted to 2 decimals to status text
IF target member is RegularMember THEN
ADD "Total Due: " plus RegularMember's price formatted to 2 decimals to status text
```

```
            ADD "Paid Amount: " plus RegularMember's paid amount formatted to 2
decimals to status text
            ADD "Full Payment: " plus "Yes" if full payment, else "No" to status text
         ELSE IF target member is PremiumMember THEN
            ADD "Total Due: " plus (PremiumMember's premium charge minus discount
amount) formatted to 2 decimals to status text
            ADD "Paid Amount: " plus PremiumMember's paid amount formatted to 2
decimals to status text
            ADD "Full Payment: " plus "Yes" if full payment, else "No" to status text
         END IF
         SET status text area to status text
         CLEAR Amount text field
      CATCH input error
         SET status text to "Error: " plus error message
      CATCH any error
         SET status text to "Unexpected error occurred: " plus error message
      END TRY
   END WHEN
   ADD Process Payment button to window
   CREATE Home button
   SET Home button position to 600, 560, 200, 40
   SET Home button font to Segoe UI, bold, size 16
   WHEN Home button clicked
      CLOSE window
      SHOW main GymGUI screen
   END WHEN
   ADD Home button to window
   CREATE main panel
   SET main panel position to 50, 100, 1300, 750
   SET main panel background to white
   SET main panel border to cyan, thickness 1
   SET main panel layout to none
   ADD main panel to window
   SHOW window
FINISH processing

# generateNumbers
START generating numbers from start to end
   CREATE array of size (end minus start plus 1)
   FOR index from start to end
      SET array position (index minus start) to index as string
   END FOR
   RETURN array
FINISH generating

# generateYears
START generating years from start to end
   CREATE array of size (end minus start plus 1)
   FOR index from start to end
      SET array position (index minus start) to index as string
   END FOR
   RETURN array
FINISH generating

# getMonths
START getting months
   CREATE array with "January", "February", "March", "April", "May", "June", "July",
"August", "September", "October", "November", "December"
   RETURN array
FINISH getting

# main
```

```
**START** program
   **RUN** GymGUI creation on event dispatch thread
**FINISH** program
```

5. Method Description

i. GymGUI methods

- **GymGUI()**
  Initializes the main user interface with the title "GymFreak" and sets up interactive buttons to manage different membership features.

- **createStyledButton(text: String): JButton**
  Returns a visually styled button with custom colors and hover behavior, improving the user interface's look and usability.

- **readFromFile(): void**
  Loads existing member data from a file named MemberDetails.txt. Displays the contents in a scrollable area and handles missing or empty file cases.

- **saveToFile(): void**
  Writes all current member records into MemberDetails.txt in a formatted way. Alerts the user if the save was successful or failed.

- **addMember(): void**
  Shows a selection screen allowing the user to choose between adding a Regular or Premium member by opening the appropriate form.

- **addRegularMember(): void**
  Presents a form for entering regular member details like referral and plan type. Checks all inputs before adding the member to the system.

- **addPremiumMember(): void**
  Presents a form to register a premium member, including fields for trainer name and discount. Performs validations before saving the data.

- **activation(): void**
  Allows activating, deactivating, or tracking attendance of a member using their ID. It reflects changes and notifies the user about results.

- **addRevertmember(): void**
  Lets the user revert a member by ID with a required reason. Resets the member’s internal values and updates the display accordingly.

- **display(): void**
  Outputs all saved member information in a text area, including status and relevant details formatted for easy reading.

- **upgradePlan(): void**
  Opens a window to allow a Regular Member to switch to a different plan after confirming their eligibility and loyalty points.

- **processPayment(): void**
  Opens an interface for handling payments or applying discounts by member ID. Updates the record after processing and shows confirmation.

- **generateNumbers(start: int, end: int): String[]**
  Produces an array of stringified numbers from a start to an end value, often used for dropdown selections like days or years.**generateYears(start: int, end: int): String[]**
  Returns a list of years between two given values, typically used for birth year or start year dropdowns.

- **getMonths(): String[]**
  Returns a 12-element string array containing month names (January to December) used for date selection inputs.

- **truncate(str: String, length: int): String**
  Shortens a given string to the desired length, used for formatting output to fixed-width layouts.

- **main(args: String[]): void**
  Launches the application interface safely using Swing's event-dispatching thread.

ii. GymMember Methods

- **getId()** through **getIsFullPayment()**
  These methods individually return stored personal and membership details for a gym member such as ID, name, contact info, gender, DOB, join date, status, and payment state.

- **activateMembership() / deactivateMembership()**
  These control the member's active status. Activation enables access to features; deactivation restricts use. Proper checks ensure valid state transitions.

- **makePayment(amount: double)**
  Abstract method that subclasses implement to handle payment processing logic based on membership type and pricing.

- **markAttendance()**
  Another abstract method to be defined in subclasses that tracks attendance and adjusts loyalty points accordingly.

- **resetMember()**
  Resets internal fields like loyalty points, payment status, and active status to default values. Mainly used when reverting a member.

- **toString()**
  Returns a complete text summary of the member. Subclasses override it to include more specific details.

iii. Regular Member Methods

- **Constructor**
  Creates a Regular Member instance after validating required fields like referral and plan. Sets the initial price and payment status based on the plan.

- **makePayment(amount: double)**
  Adds a payment to the member’s balance. Checks for overpayment and marks full payment complete if the amount reaches the required price.

- **markAttendance()**
  Rewards 5 loyalty points each time the member is marked present, as long as the membership is active.

- **upgradePlan(newPlan: String)**
  Allows plan changes (basic, standard, deluxe) when the member is eligible and has earned enough loyalty points. Resets payment and adjusts the price.

- **revertRegularMember(removalReason: String)**
  Used when a Regular Member needs to be reverted. Resets their data, sets the plan to "basic", and stores the provided reason for the action.

- **toString()**
  Produces a full textual description of the member including plan, payment status, and if present, the removal reason.

- **setIsFullPayment(isFullPayment: boolean)**
  Updates the flag that indicates if the full fee has been paid. Usually triggered by successful payment processing.

iv. PremiumMember Methods

- **Constructor**
  Instantiates a Premium Member with trainer details and a discount. Ensures the values are within valid limits before setting payment and discount state.

- **makePayment(amount: double)**
  Increases the amount paid toward the premium charge after discount. Once the full adjusted amount is reached, the member is flagged as paid.

- **markAttendance()**
  Increases the member's loyalty points by 10 per session, provided the membership is active.

- **revertPremiumMember(removalReason: String)**
  Used to downgrade or remove a Premium Member. Clears trainer and payment data, and stores a justification for the change.

- **toString()**
  Returns a complete string of the member's information, including trainer name, discount, and payment progress. Also includes removal notes if relevant.

- **setDiscountAmount(discountAmount: double)**
  Adjusts the discount offered to the member. Validates the range to ensure it's within acceptable limits.

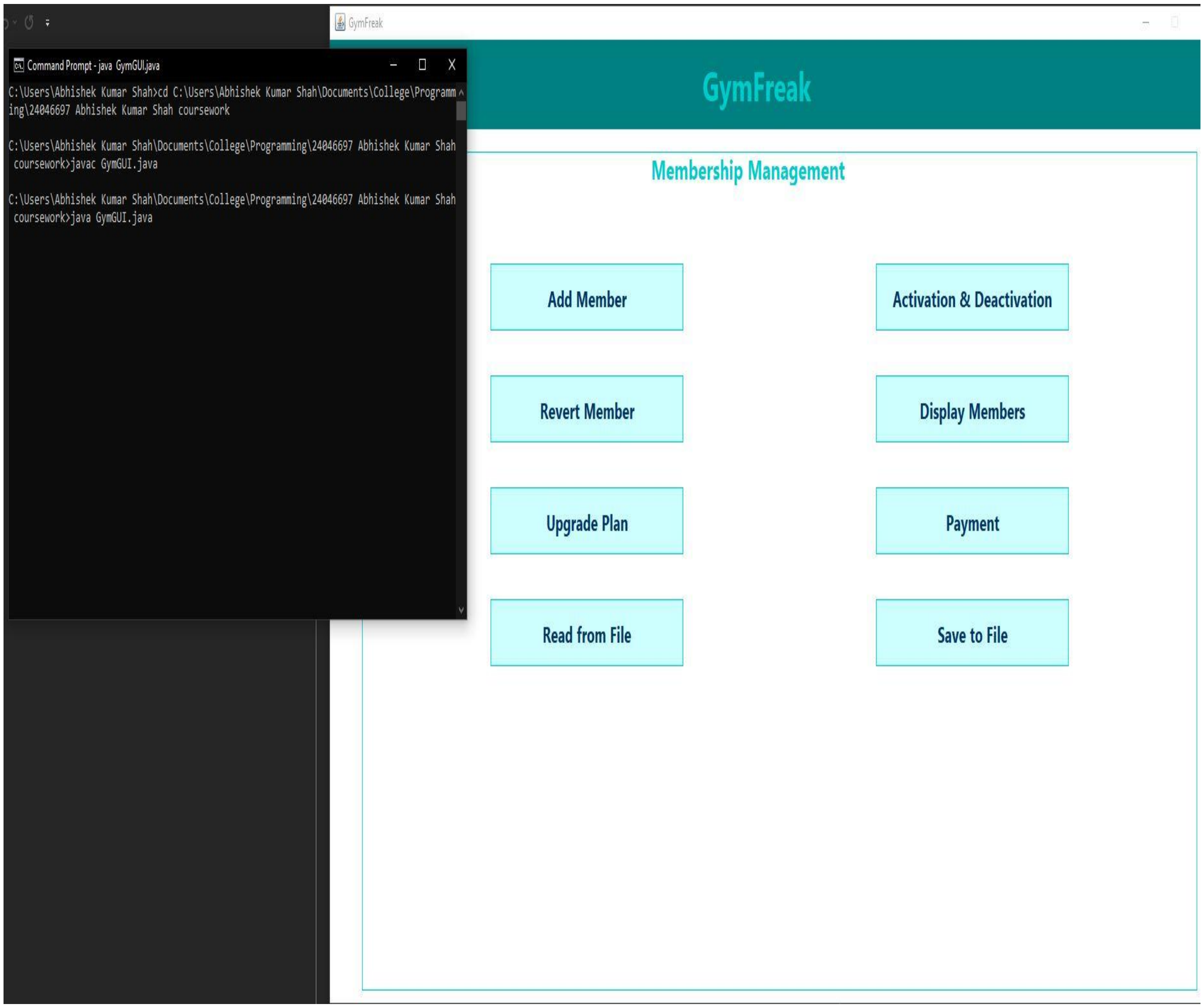
- **setIsFullPayment(isFullPayment: boolean)**
  Directly sets whether the member has paid in full, mostly used internally in response to payment milestones.

6. Testing

Test-1: Compile and run using command prompt/terminal

Table 1: Test 1

Objective	To ensure the application compiles and runs correctly via command line
Action	Opened terminal, navigated to the source folder, compiled all .java files using javac, and ran the main class with java GymGUI.
Expected Result	The program should compile without errors and open the main window with functional UI components.
Actual Result	The application launched successfully. All main buttons and features were displayed properly.
Conclusion	Passed-The system is compiling and executing without issues. Launch process is stable.



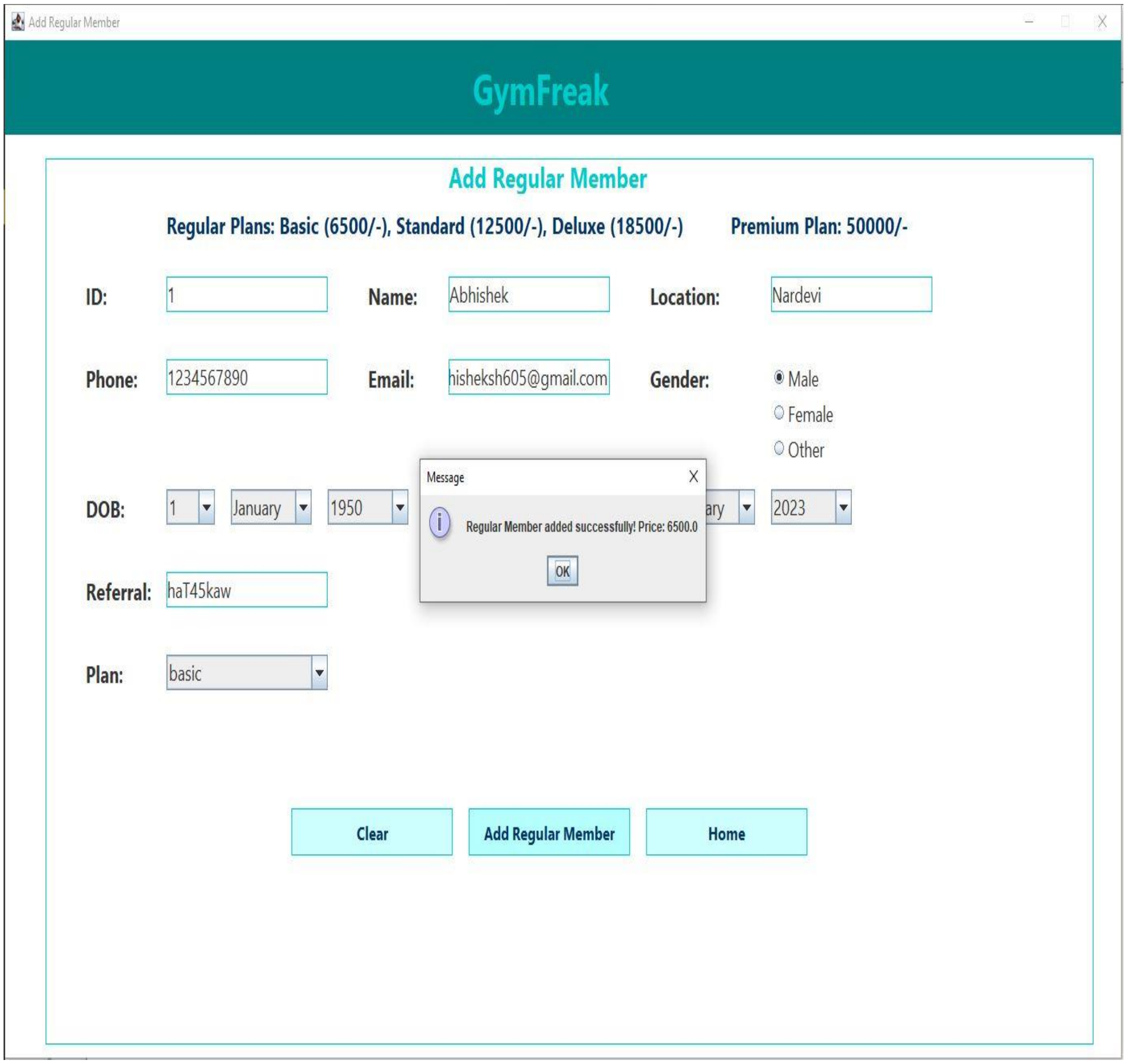
```
C:\Users\Abhishek Kumar Shah>cd C:\Users\Abhishek Kumar Shah\Documents\College\Programm
ing\24046697 Abhishek Kumar Shah coursework

C:\Users\Abhishek Kumar Shah\Documents\College\Programming\24046697 Abhishek Kumar Shah
 coursework>javac GymGUI.java

C:\Users\Abhishek Kumar Shah\Documents\College\Programming\24046697 Abhishek Kumar Shah
 coursework>java GymGUI.java
```

Figure 16: Test 1-Compile and run using command prompt/terminal

Test-2: Add regular and premium members

Table 2: Test 2

Objective	To verify if new Regular and Premium members can be added through the interface.
Action	Used the "Add Member" feature to register both a Regular Member (with referral and plan) and a Premium Member (with trainer).
Expected Result	Member details should be validated and added to the internal list. No errors for valid inputs.
Actual Result	Both members were successfully added. Invalid inputs triggered appropriate error messages.
Conclusion	Passed-Member registration logic works correctly and handles both valid and invalid input scenarios.


Add Regular Member

GymFreak

Add Regular Member

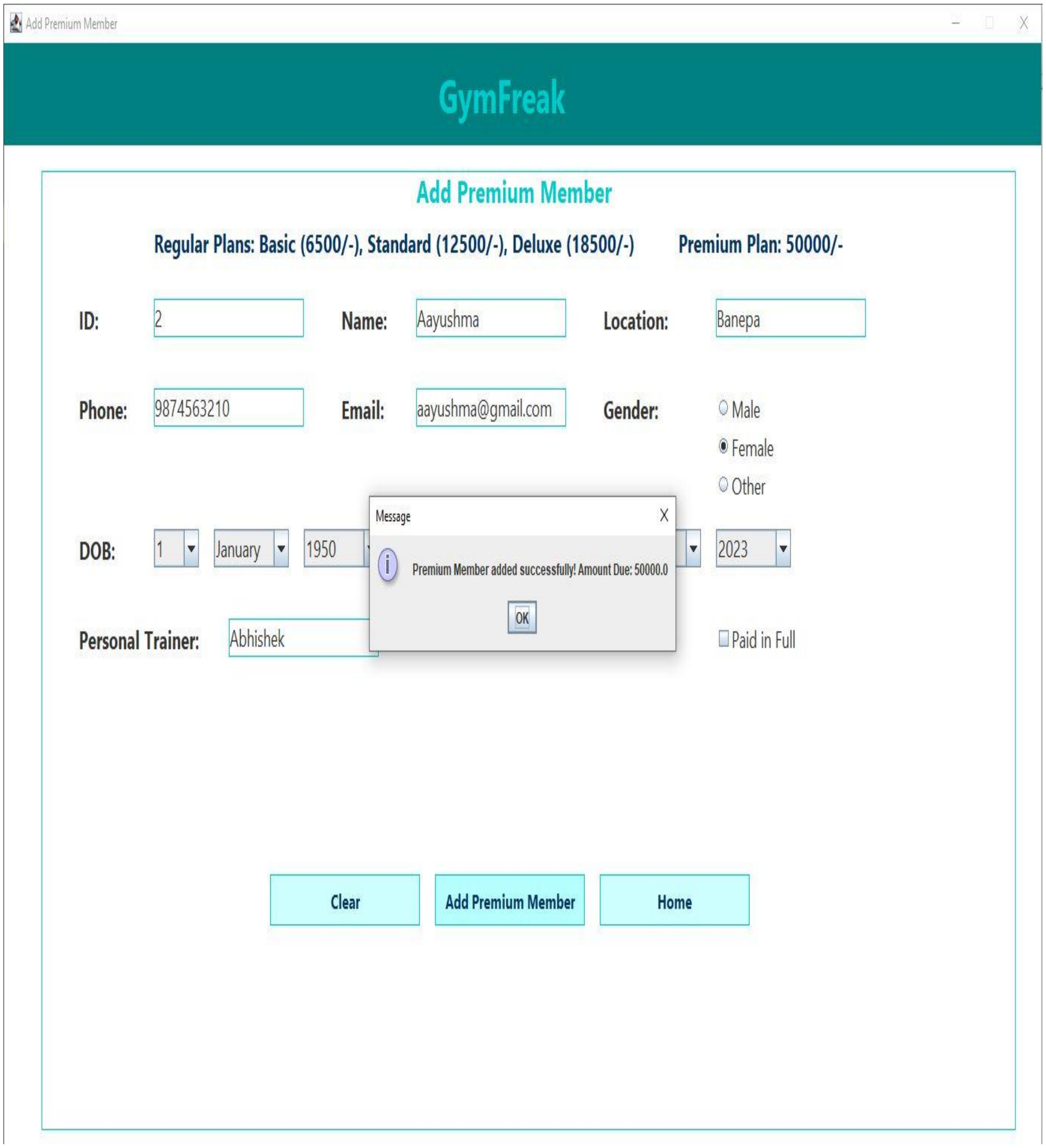
Regular Plans: Basic (6500/-), Standard (12500/-), Deluxe (18500/-) Premium Plan: 50000/-

ID:	1	Name:	Abhishek	Location:	Nardevi
Phone:	1234567890	Email:	hisheksh605@gmail.com	Gender:	● Male ○ Female ○ Other
DOB:	1 January 1950				ary 2023
Referral:	haT45kaw				
Plan:	basic				

Message

Regular Member added successfully! Price: 6500.0

Figure 17: test 2.1-add regular member



Add Premium Member

GymFreak

Add Premium Member

Regular Plans: Basic (6500/-), Standard (12500/-), Deluxe (18500/-) Premium Plan: 50000/-

ID:	2	Name:	Aayushma	Location:	Banepa
Phone:	9874563210	Email:	aayushma@gmail.com	Gender:	Male [✓ Female] Other
DOB:	1 January 1950				2023
Personal Trainer:	Abhishek				☐ Paid in Full

Message

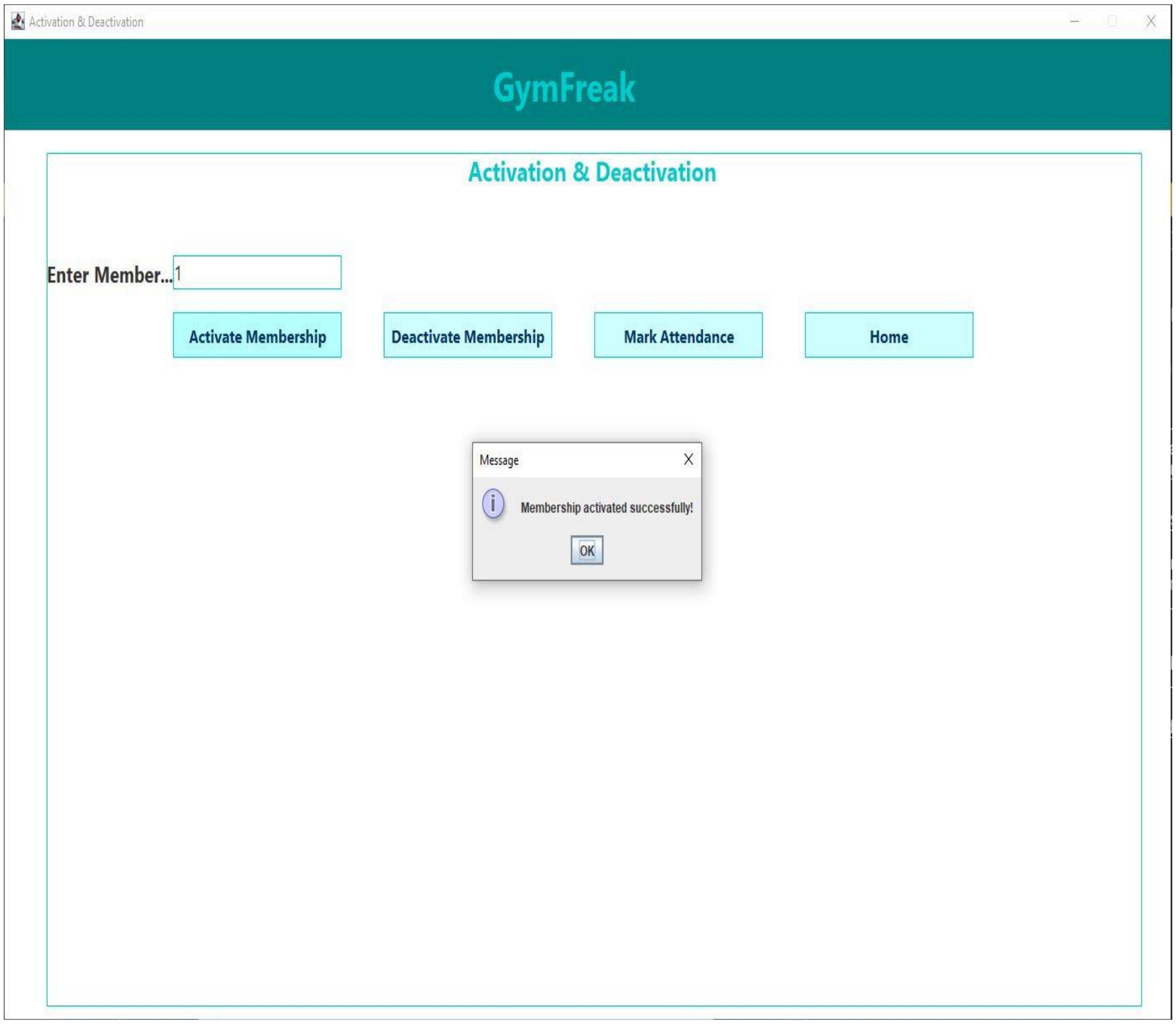
Premium Member added successfully! Amount Due: 50000.0

Figure 18:test 2.2- add premium member

Test-3: Mark attendance and upgrade plan

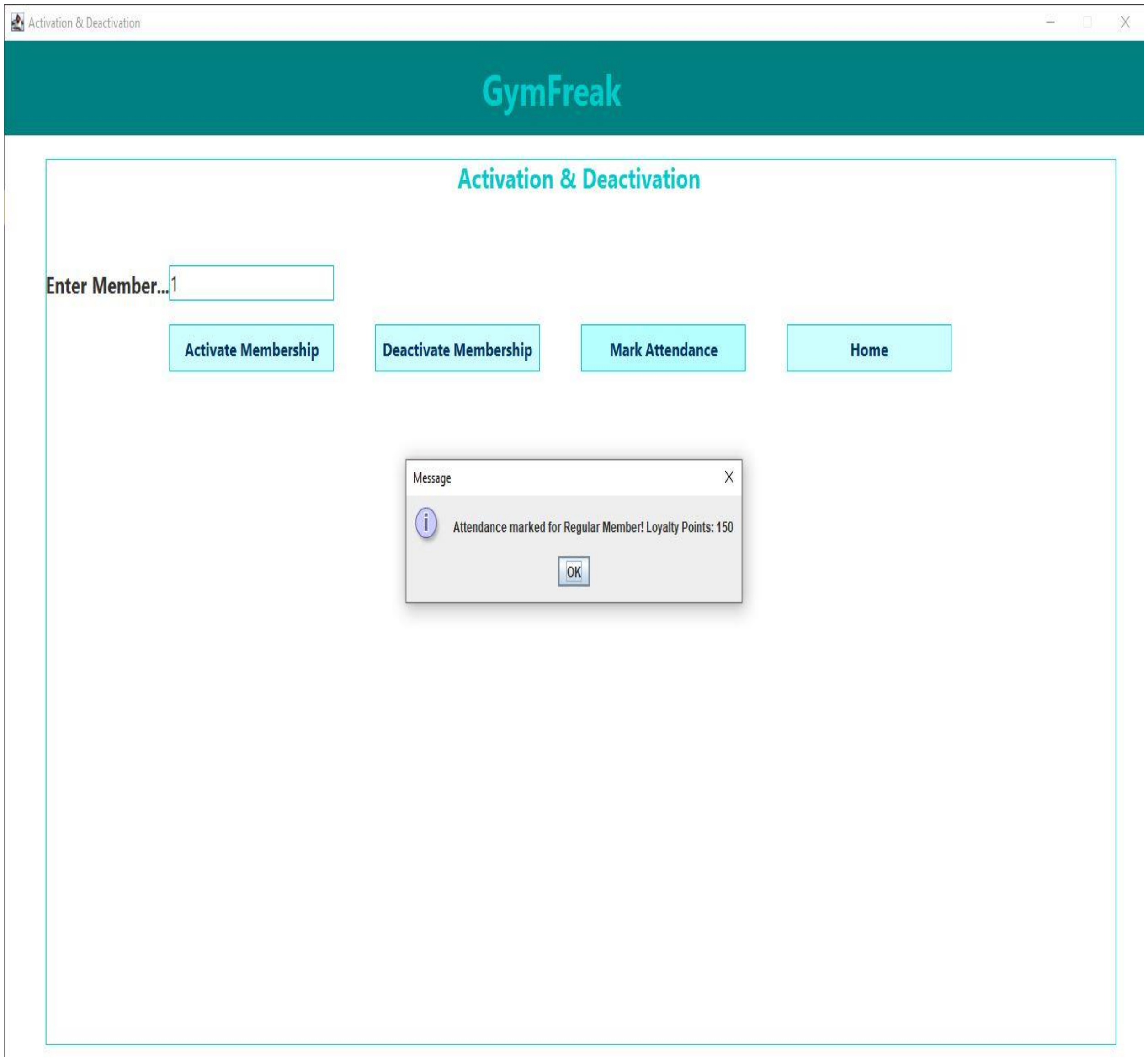
Table 3: Test 3

Objective	To confirm attendance tracking and plan upgrade functionality works for eligible Regular Members.
Action	Marked attendance for a Regular Member until they reached the required loyalty points. Then attempted a plan upgrade.
Expected Result	Loyalty points should increase, and plan upgrade should only be allowed if criteria are met.
Actual Result	Attendance incremented loyalty points accurately. Plan upgraded successfully when eligibility was met.
Conclusion	Passed-Both attendance marking and upgrade plan features function as designed and respect logical conditions.



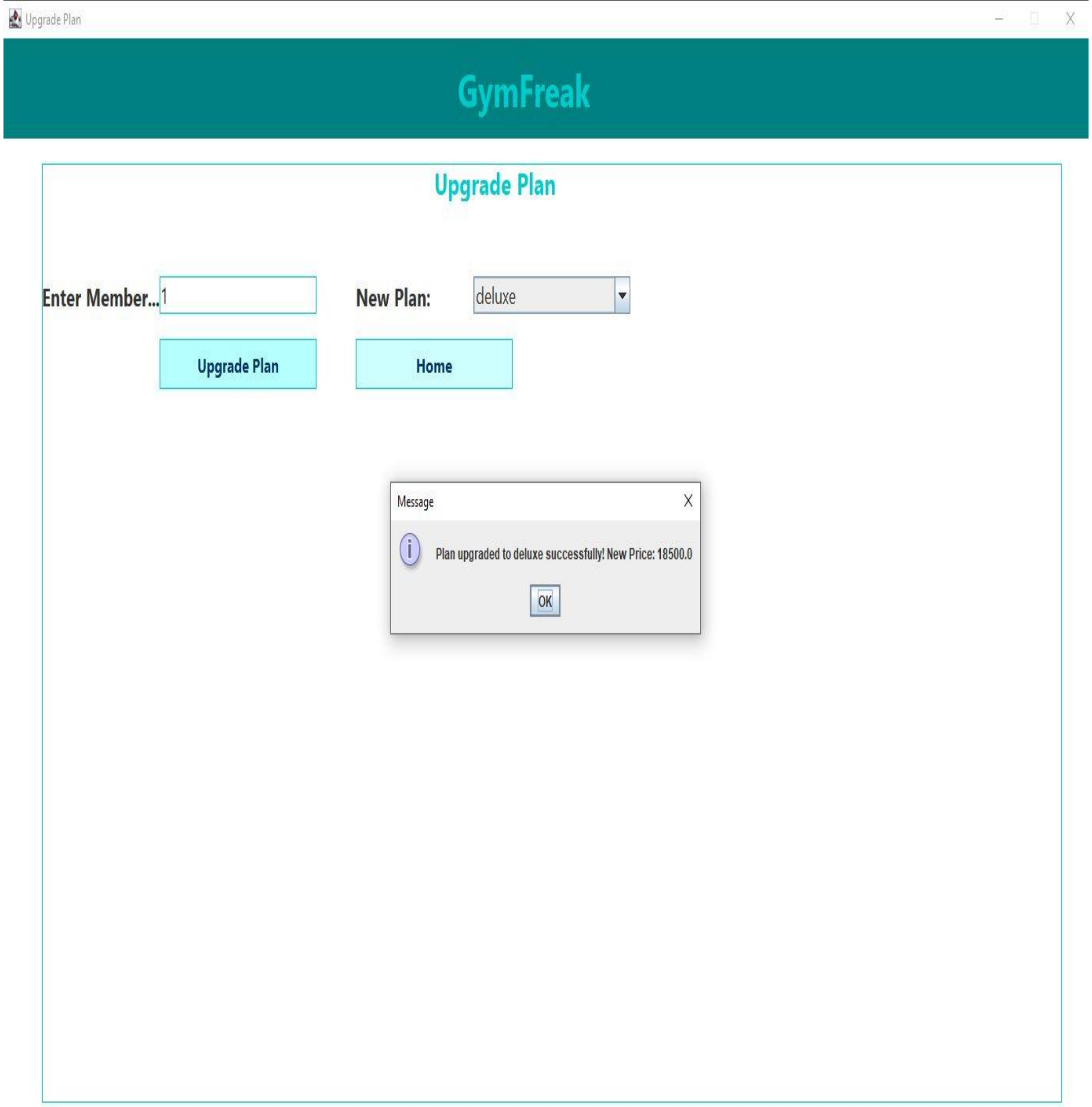
Membership activated successfully!

Figure 19: test 3.1- activating membership



Attendance marked for Regular Member! Loyalty Points: 150

Figure 20: test 3.2- marked attendance 30 times to meet loyalty points at 150



1

deluxe

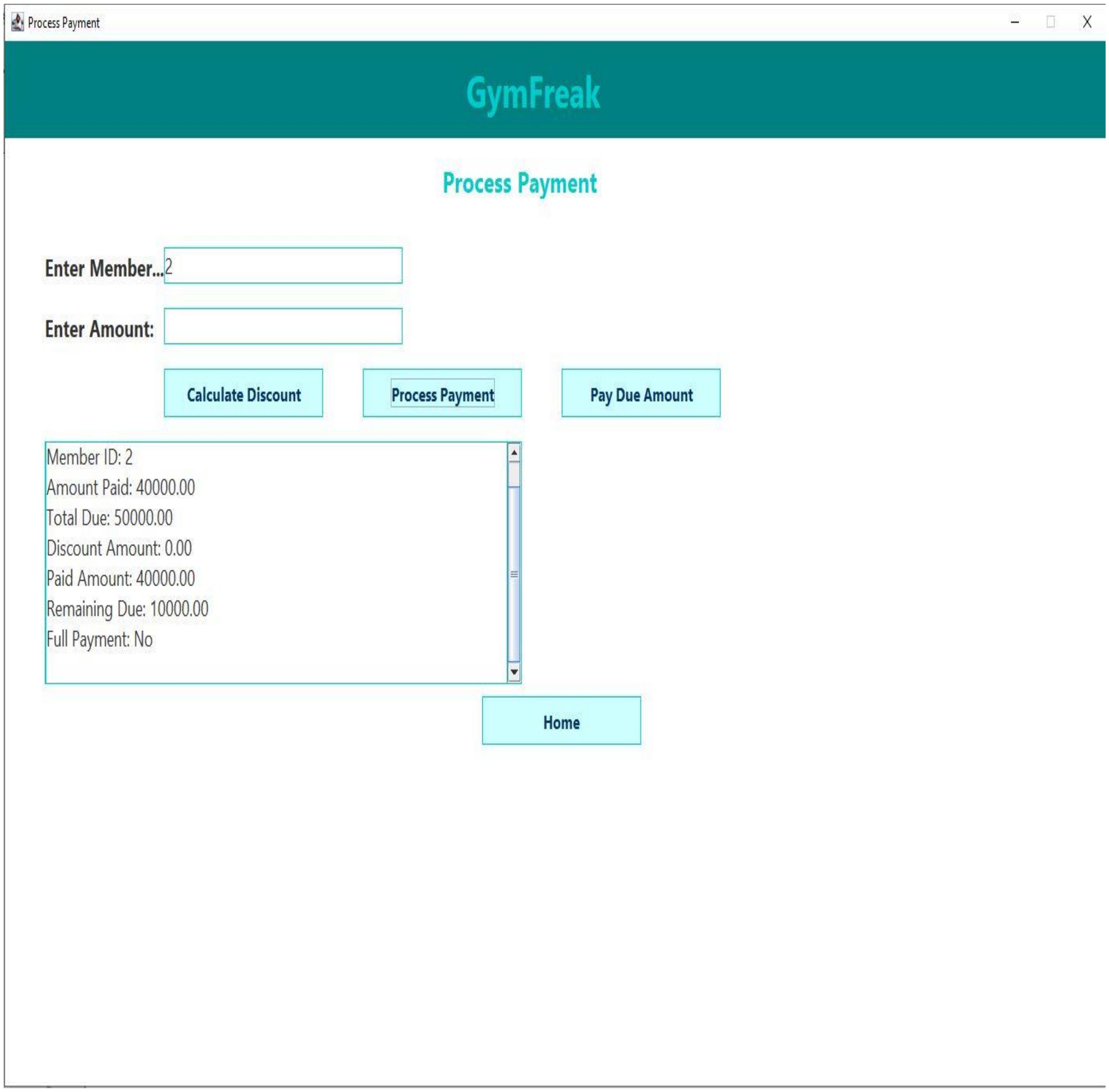
Plan upgraded to deluxe successfully! New Price: 18500.0

Figure 21: test 3.3-plan was upgraded to deluxe

Test-4: Calculate discount, pay due, revert members

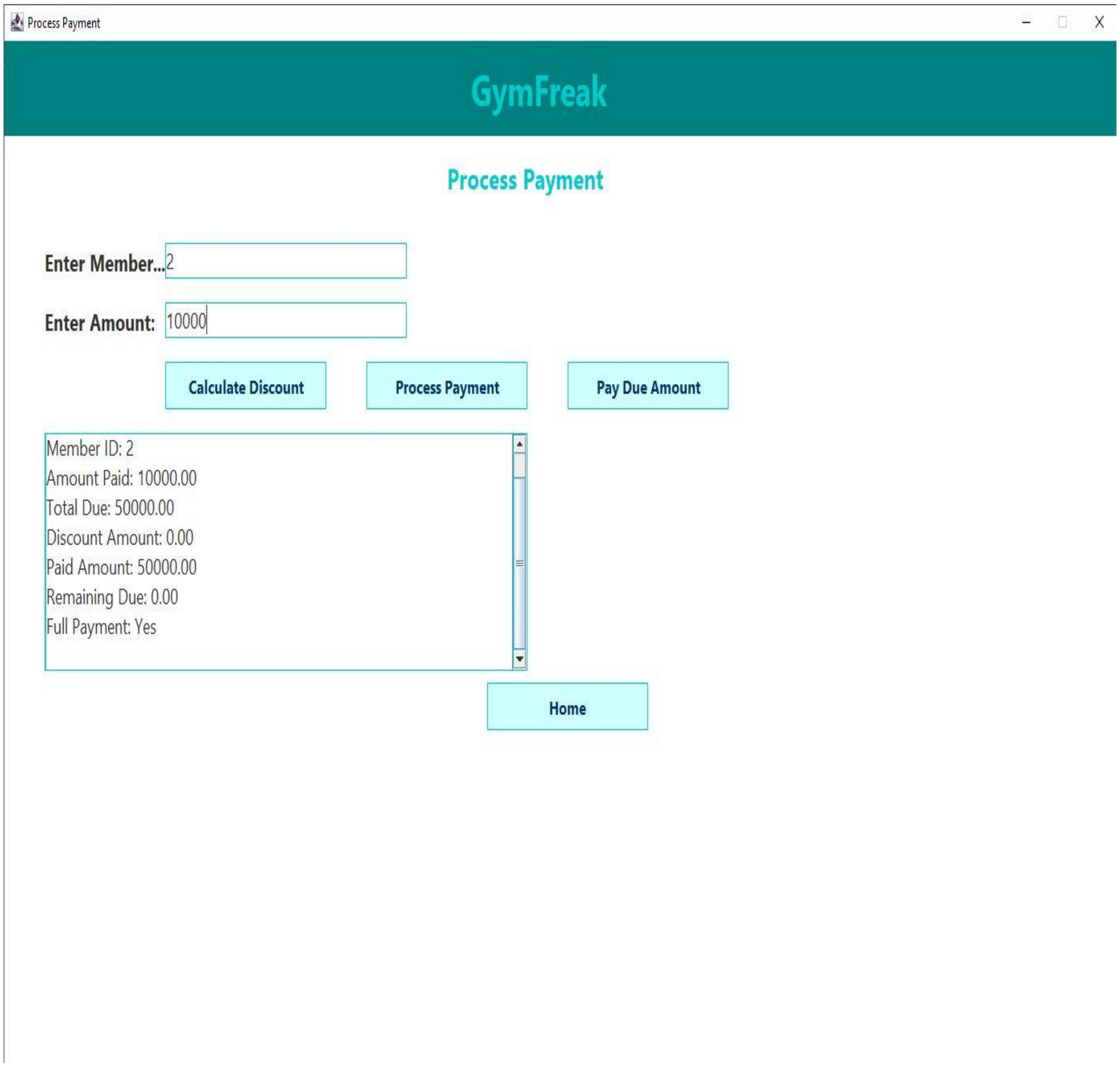
Table 4: Test 4

Objective	To evaluate payment processing, discount application for Premium Members, and the ability to revert members.
Action	Processed a payment for a Premium Member with a discount. Then used the revert option for both member types.
Expected Result	The system should calculate the correct payable amount after discount, update payment status, and reset fields on reversion.
Actual Result	Payment calculations and discount logic were accurate. Revert functionality reset member states and stored the removal reason.
Conclusion	Passed-discount, pay due and revert members functions are behaving reliably with proper validation.



```
Member ID: 2
Amount Paid: 40000.00
Total Due: 50000.00
Discount Amount: 0.00
Paid Amount: 40000.00
Remaining Due: 10000.00
Full Payment: No
```

Figure 22: test4.1-Payment processed

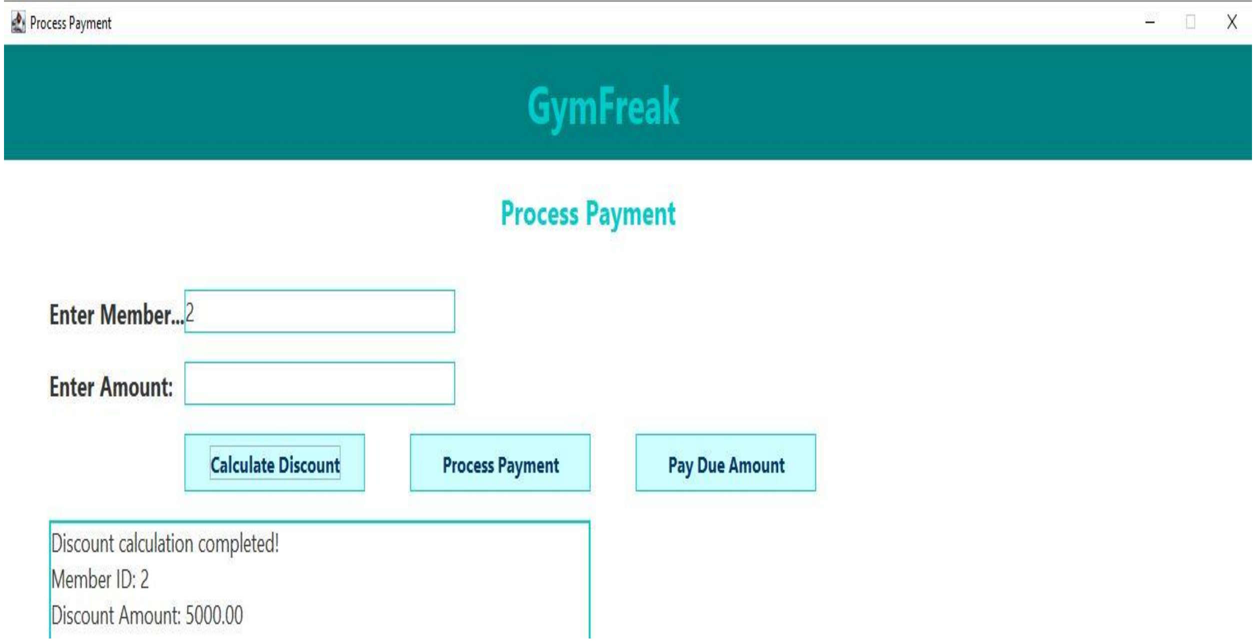


2

10000

```
Member ID: 2
Amount Paid: 10000.00
Total Due: 50000.00
Discount Amount: 0.00
Paid Amount: 50000.00
Remaining Due: 0.00
Full Payment: Yes
```

Figure 23: test 4.2-paid due amount



2

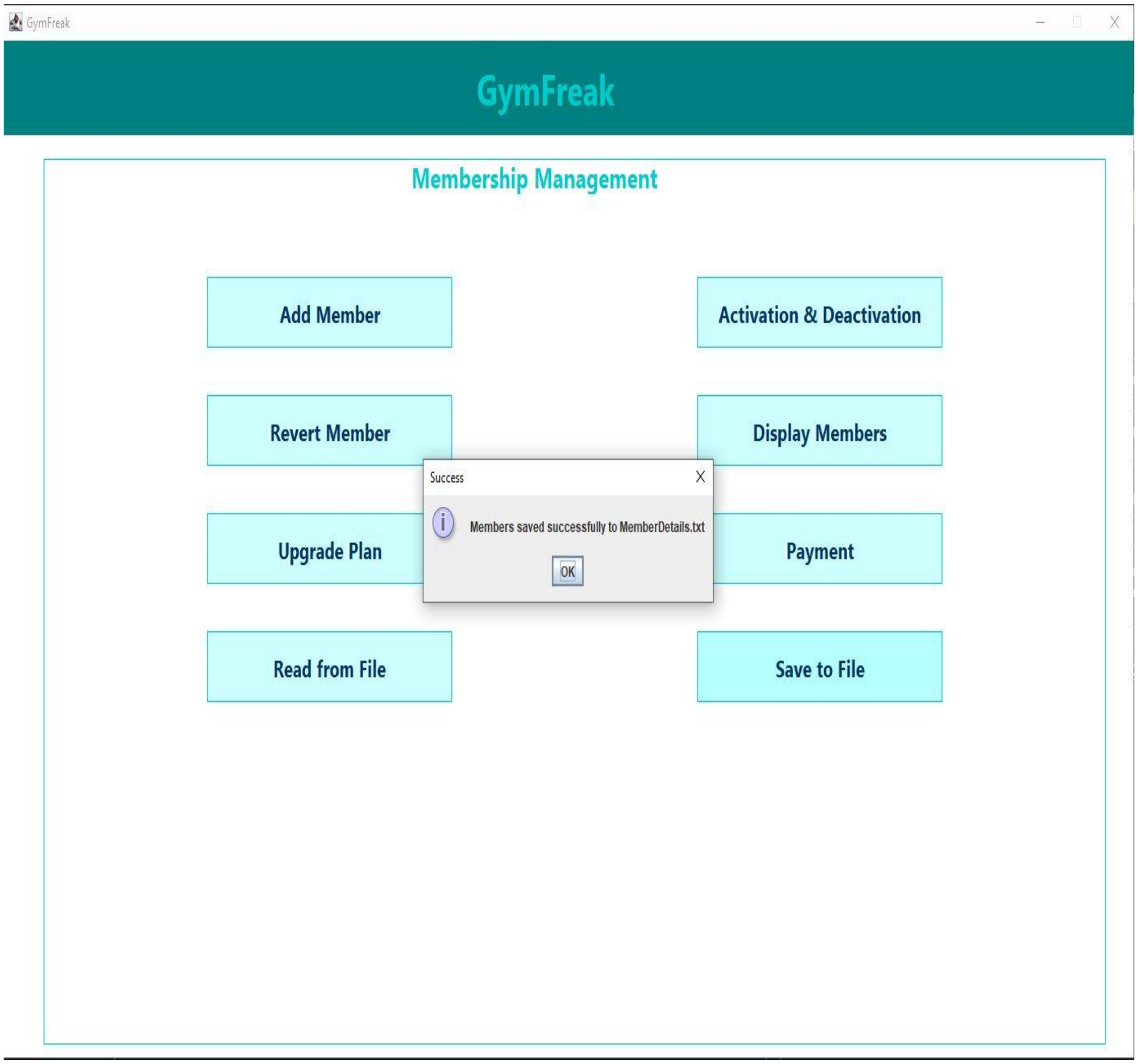
```
Discount calculation completed!
Member ID: 2
Discount Amount: 5000.00
```

Figure 24:test 4.3 calculate discount

Test-5: Save to and read from file

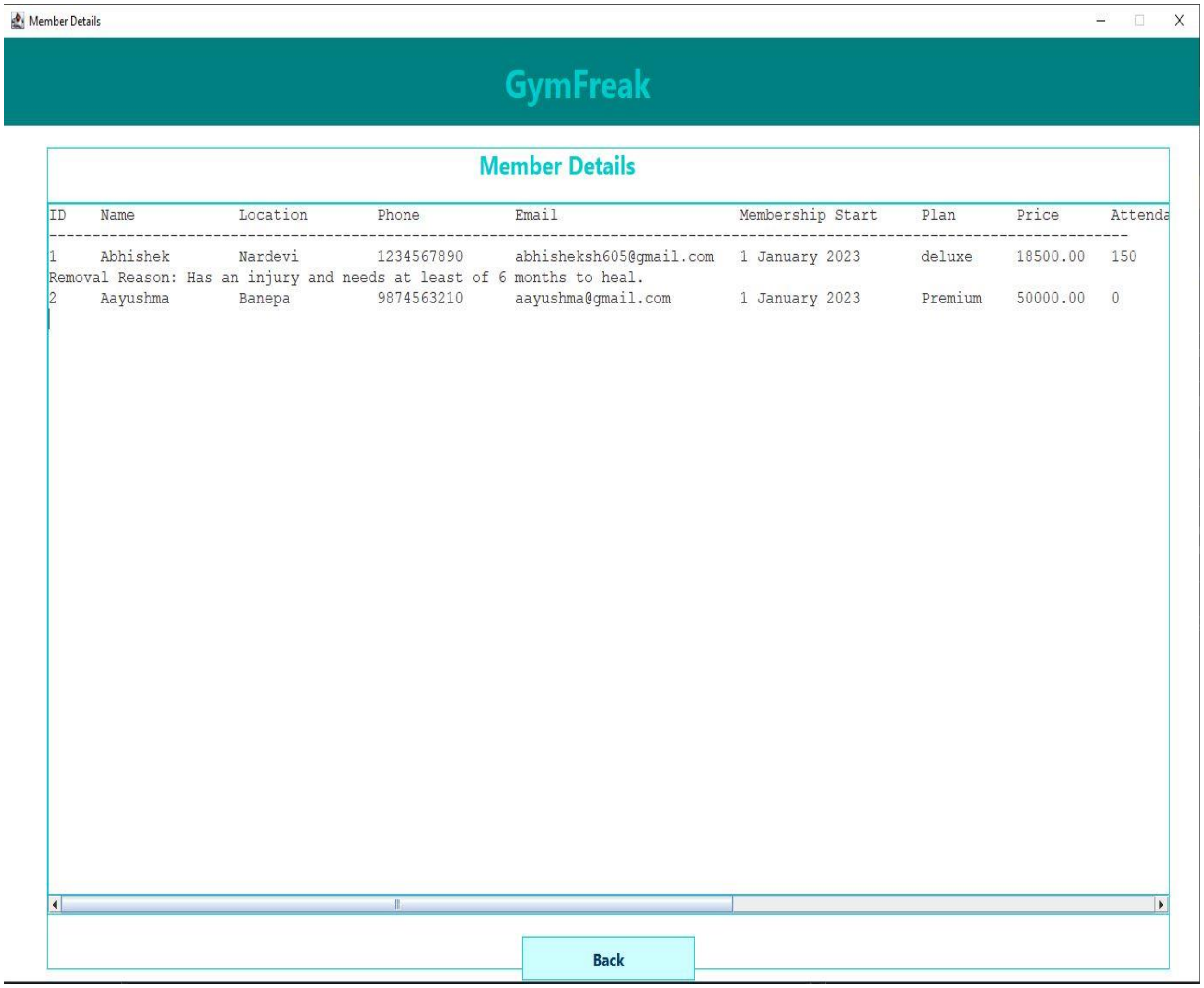
Table 5: Test 5

Objective	To validate that member data can be stored and retrieved from a file.
Action	Used "Save to file" to write member info to a text file, then clicked "Read from file" to display contents.
Expected Result	The saved file should contain formatted member data. Reading should retrieve and display this information.
Actual Result	Member data was successfully saved and correctly displayed when read back into the system.
Conclusion	Passed-File handling is operational. Save to file and Read from file are storing and retrieving data as expected.



Members saved successfully to MemberDetails.txt

Figure 25: test 5.1- save to file



Member Details

```
ID    Name            Location        Phone           Email                     Membership Start     Plan       Price       Attenda
-------------------------------------------------------------------------------------------------------------------------------
1     Abhishek        Nardevi         1234567890      abhisheksh605@gmail.com   1 January 2023       deluxe     18500.00    150
Removal Reason: Has an injury and needs at least of 6 months to heal.
2     Aayushma        Banepa          9874563210      aayushma@gmail.com        1 January 2023       Premium    50000.00    0
```

Figure 26: test 5.2- read from file

Error and Detection

Syntax error

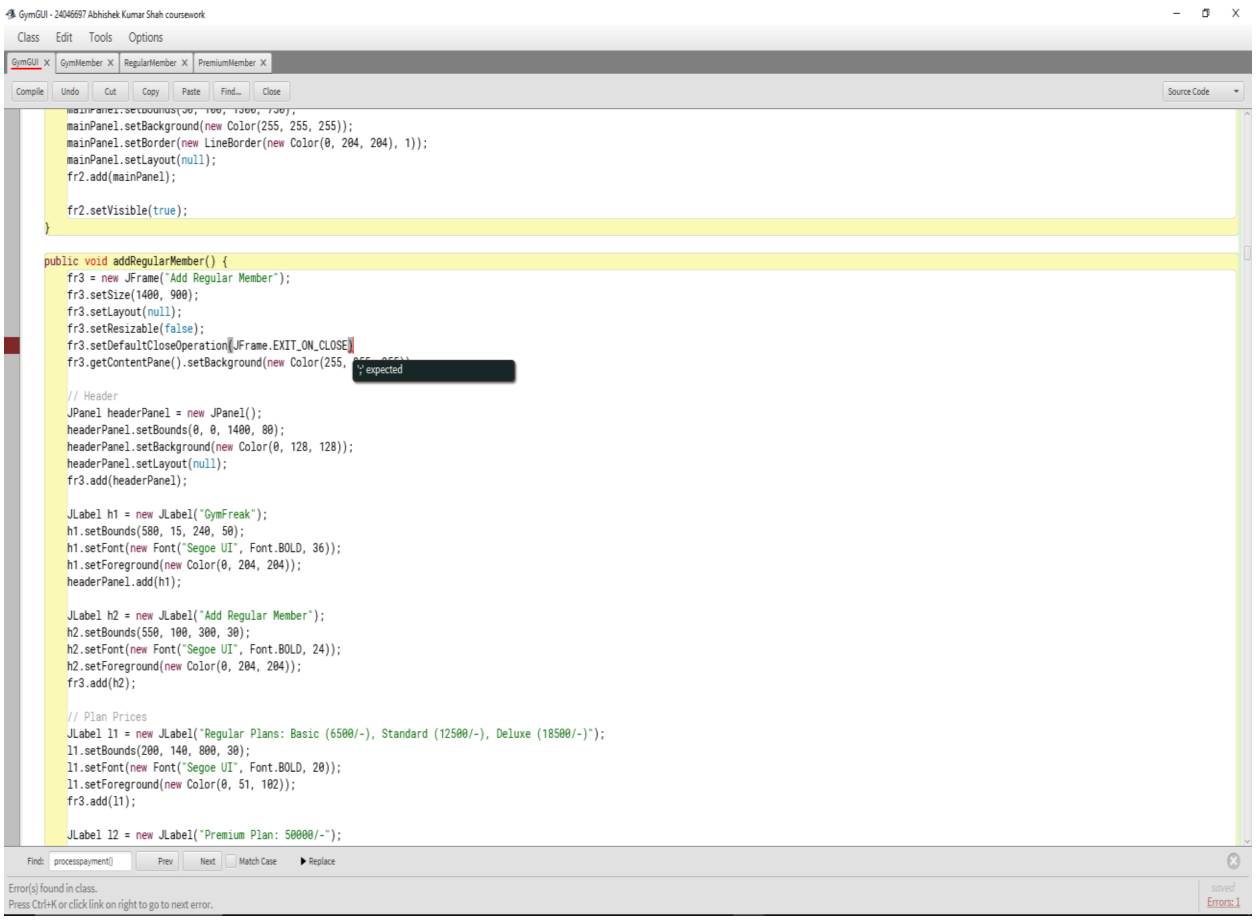
Syntax errors occur when the code doesn’t follow the rules of the Java language. These are usually highlighted during compilation.

Example Encountered:

I forgot to add semicolons at the end of statements .

How I Fixed It:

I used the compiler’s error messages to find the exact line causing the problem. In most cases, it pointed me straight to the missing or incorrect symbol, and I corrected it accordingly.

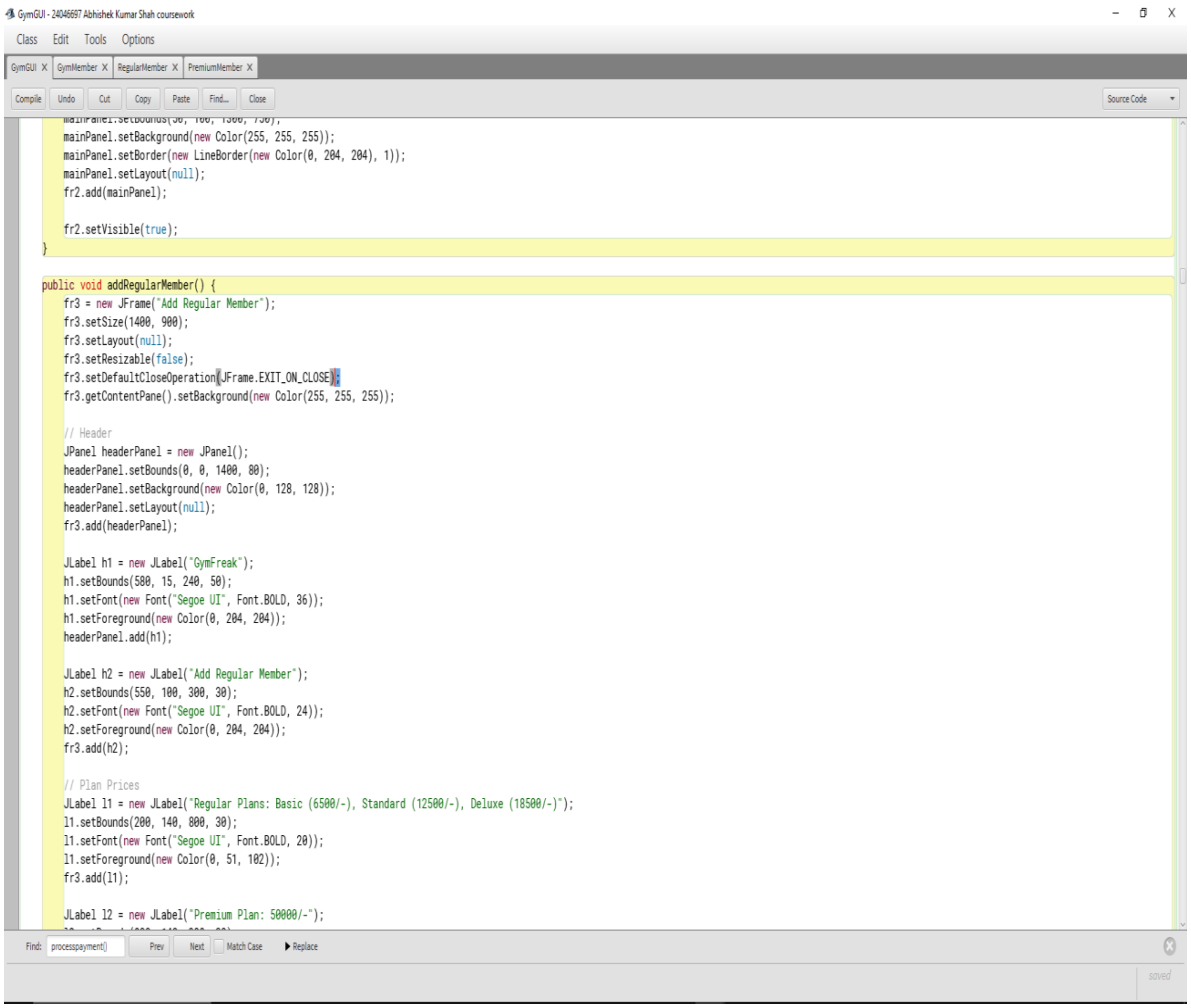

GymGUI - 24046697 Abhishek Kumar Shah coursework
Class Edit Tools Options
GymGUI GymMember RegularMember PremiumMember
Compile Undo Cut Copy Paste Find... Close
Source Code

```
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr2.add(mainPanel);

        fr2.setVisible(true);
    }

    public void addRegularMember() {
        fr3 = new JFrame("Add Regular Member");
        fr3.setSize(1400, 900);
        fr3.setLayout(null);
        fr3.setResizable(false);
        fr3.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE)
        fr3.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
        headerPanel.setBounds(0, 0, 1400, 80);
        headerPanel.setBackground(new Color(0, 128, 128));
        headerPanel.setLayout(null);
        fr3.add(headerPanel);

        JLabel h1 = new JLabel("GymFreak");
        h1.setBounds(580, 15, 240, 50);
        h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
        h1.setForeground(new Color(0, 204, 204));
        headerPanel.add(h1);

        JLabel h2 = new JLabel("Add Regular Member");
        h2.setBounds(550, 100, 300, 30);
        h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
        h2.setForeground(new Color(0, 204, 204));
        fr3.add(h2);

        // Plan Prices
        JLabel l1 = new JLabel("Regular Plans: Basic (6500/-), Standard (12500/-), Deluxe (18500/-)");
        l1.setBounds(200, 140, 800, 30);
        l1.setFont(new Font("Segoe UI", Font.BOLD, 20));
        l1.setForeground(new Color(0, 51, 102));
        fr3.add(l1);

        JLabel l2 = new JLabel("Premium Plan: 50000/-");
```

```
';' expected
```

```
Error(s) found in class.
Press Ctrl+K or click link on right to go to next error.
```

Figure 27: syntax error example



```
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr2.add(mainPanel);

        fr2.setVisible(true);
    }

    public void addRegularMember() {
        fr3 = new JFrame("Add Regular Member");
        fr3.setSize(1400, 900);
        fr3.setLayout(null);
        fr3.setResizable(false);
        fr3.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr3.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
        headerPanel.setBounds(0, 0, 1400, 80);
        headerPanel.setBackground(new Color(0, 128, 128));
        headerPanel.setLayout(null);
        fr3.add(headerPanel);

        JLabel h1 = new JLabel("GymFreak");
        h1.setBounds(580, 15, 240, 50);
        h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
        h1.setForeground(new Color(0, 204, 204));
        headerPanel.add(h1);

        JLabel h2 = new JLabel("Add Regular Member");
        h2.setBounds(550, 100, 300, 30);
        h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
        h2.setForeground(new Color(0, 204, 204));
        fr3.add(h2);

        // Plan Prices
        JLabel l1 = new JLabel("Regular Plans: Basic (6500/-), Standard (12500/-), Deluxe (18500/-)");
        l1.setBounds(200, 140, 800, 30);
        l1.setFont(new Font("Segoe UI", Font.BOLD, 20));
        l1.setForeground(new Color(0, 51, 102));
        fr3.add(l1);

        JLabel l2 = new JLabel("Premium Plan: 50000/-");
```

Figure 28: fixed syntax error

Runtime Error

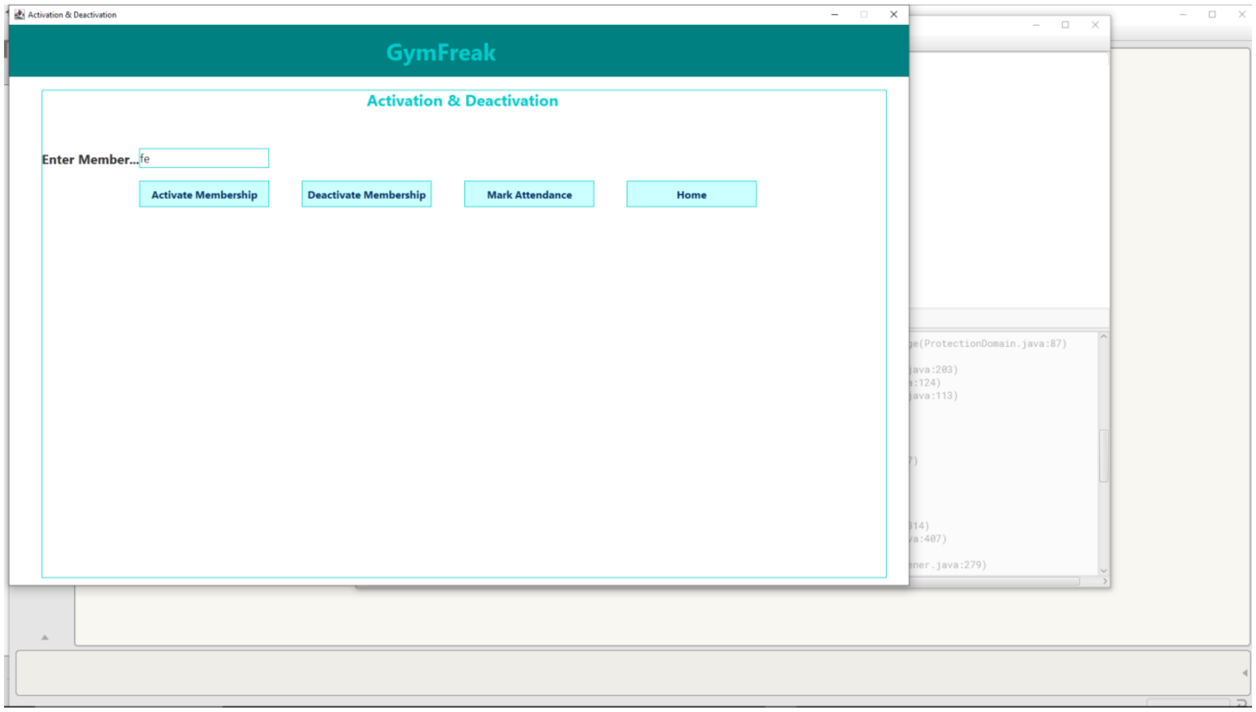
Runtime errors are problems that occur while the program is running. They don't stop the code from compiling but can cause the program to crash when executed.

Example Encountered:

I removed a try-catch block from the file reading method to test something. When I ran the program without it, it crashed immediately because the file MemberDetails.txt didn't exist yet. Java threw a FileNotFoundException.

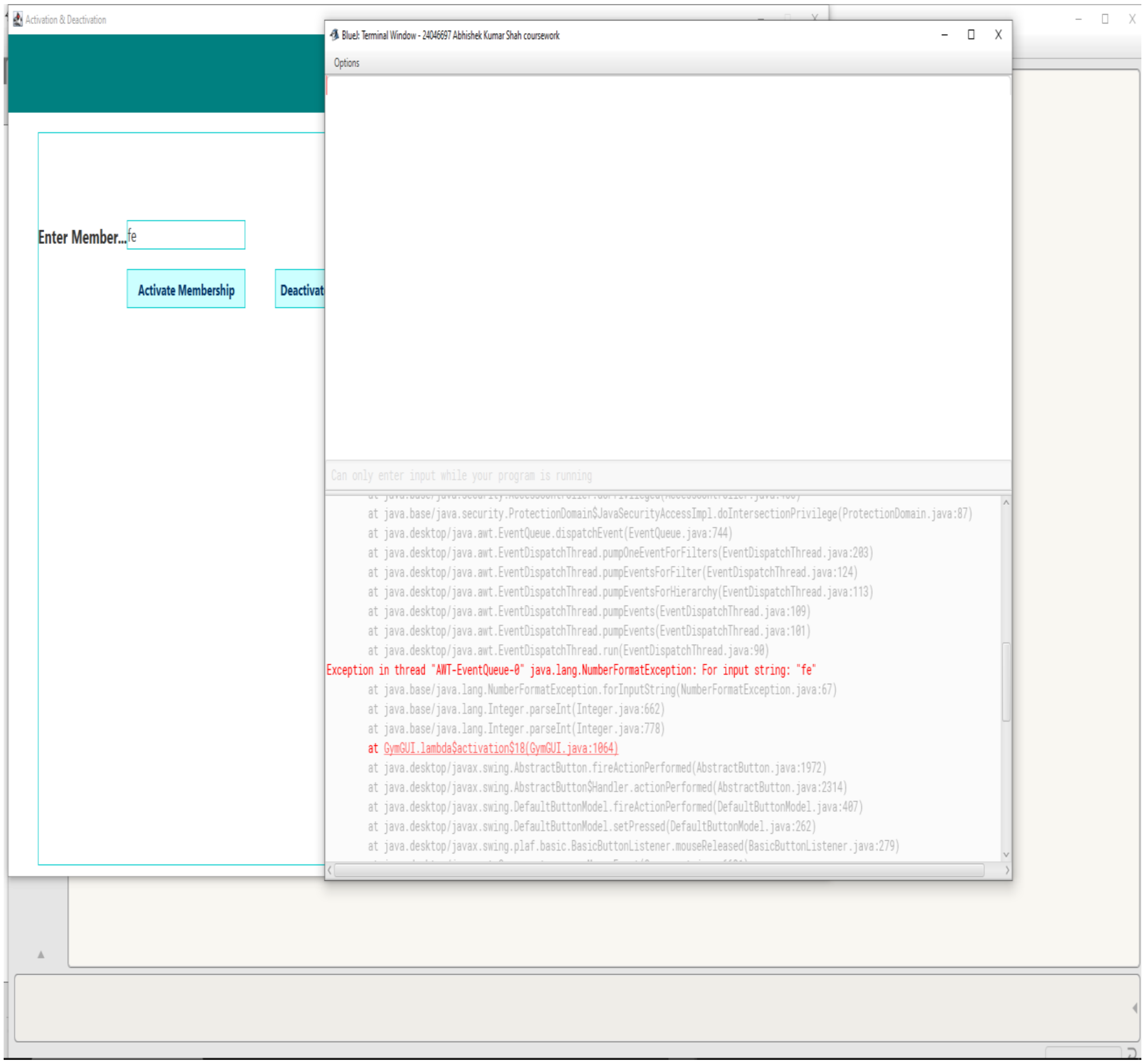
How I Fixed It:

I added the try-catch block back and used it to catch the exception. I also included a simple error message using JOptionPane so that users are informed if the file is missing or empty instead of the program crashing.



```
ge(ProtectionDomain.java:87)

java:203)
a:124)
java:113)




7)





314)
va:407)

ener.java:279)
```

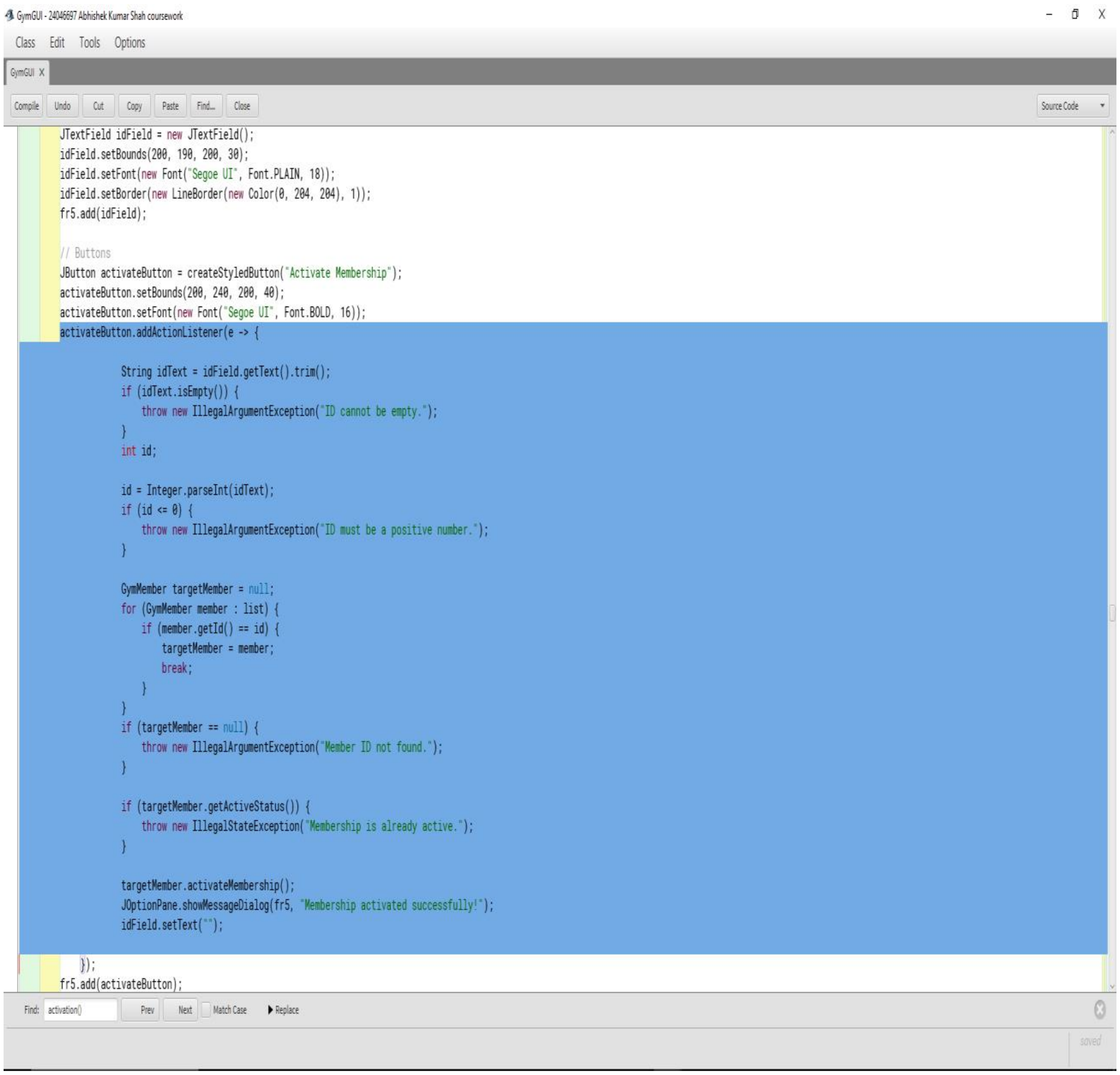
Figure 29: runtime error 1.1


Activation & Deactivation
Enter Member...
fe
Activate Membership
Deactivat
BlueJ: Terminal Window - 24046697 Abhishek Kumar Shah coursework
Options

```
Can only enter input while your program is running
```

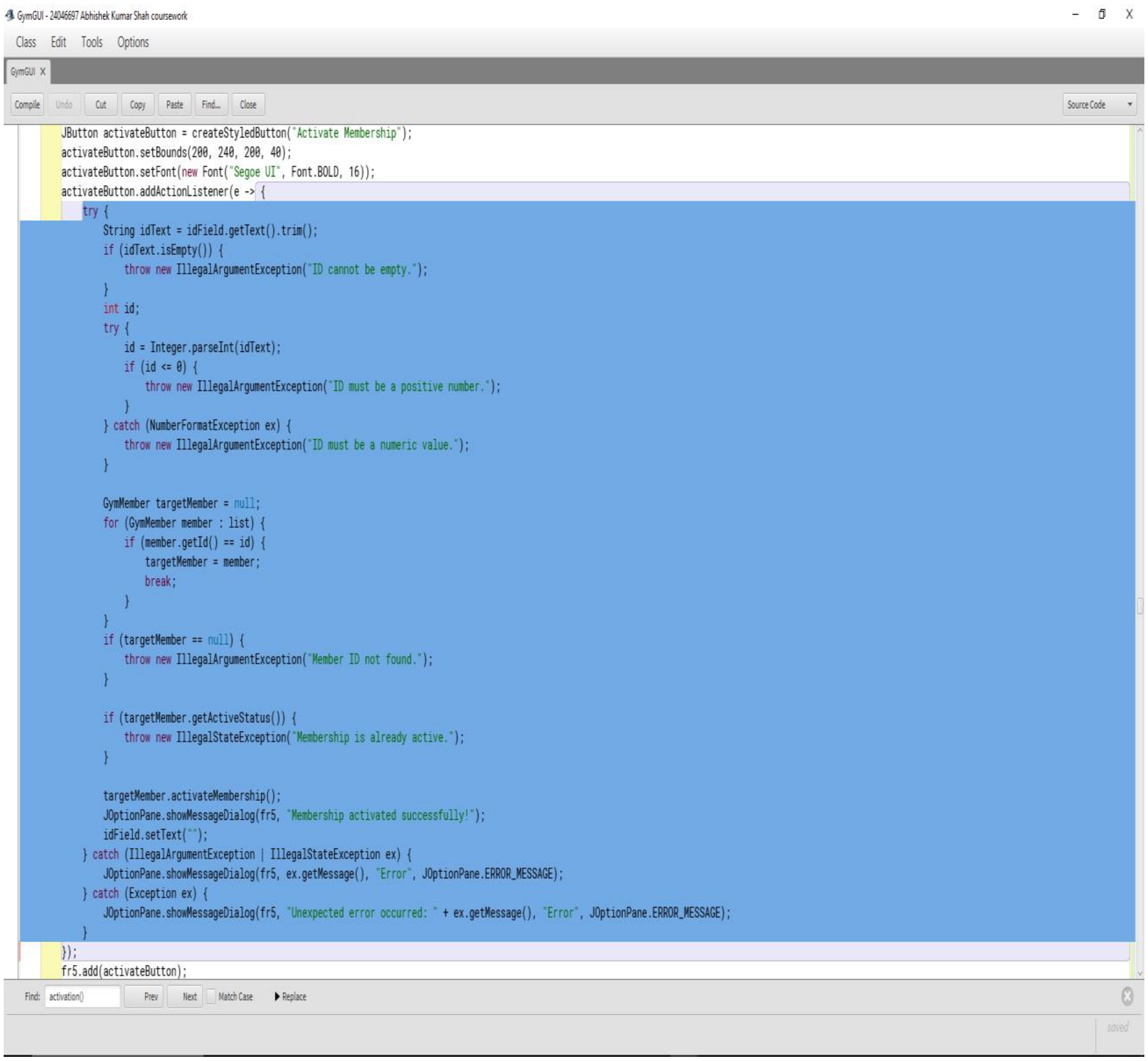
```
        at java.base/java.security.ProtectionDomain$JavaSecurityAccessImpl.doIntersectionPrivilege(ProtectionDomain.java:87)
        at java.desktop/java.awt.EventQueue.dispatchEvent(EventQueue.java:744)
        at java.desktop/java.awt.EventDispatchThread.pumpOneEventForFilters(EventDispatchThread.java:203)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForFilter(EventDispatchThread.java:124)
        at java.desktop/java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:113)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:109)
        at java.desktop/java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:101)
        at java.desktop/java.awt.EventDispatchThread.run(EventDispatchThread.java:90)
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "fe"
        at java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:662)
        at java.base/java.lang.Integer.parseInt(Integer.java:778)
        at GymGUI.lambda$activation$18(GymGUI.java:1064)
        at java.desktop/javax.swing.AbstractButton.fireActionPerformed(AbstractButton.java:1972)
        at java.desktop/javax.swing.AbstractButton$Handler.actionPerformed(AbstractButton.java:2314)
        at java.desktop/javax.swing.DefaultButtonModel.fireActionPerformed(DefaultButtonModel.java:407)
        at java.desktop/javax.swing.DefaultButtonModel.setPressed(DefaultButtonModel.java:262)
        at java.desktop/javax.swing.plaf.basic.BasicButtonListener.mouseReleased(BasicButtonListener.java:279)
```

Figure 30: run time error 1.2


GymGUI - 24046697 Abhishek Kumar Shah coursework
Class Edit Tools Options
GymGUI
Compile Undo Cut Copy Paste Find... Close
Source Code

```
JTextField idField = new JTextField();
idField.setBounds(200, 190, 200, 30);
idField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
idField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr5.add(idField);

// Buttons
JButton activateButton = createStyledButton("Activate Membership");
activateButton.setBounds(200, 240, 200, 40);
activateButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
activateButton.addActionListener(e -> {

            String idText = idField.getText().trim();
            if (idText.isEmpty()) {
                throw new IllegalArgumentException("ID cannot be empty.");
            }
            int id;

            id = Integer.parseInt(idText);
            if (id <= 0) {
                throw new IllegalArgumentException("ID must be a positive number.");
            }

            GymMember targetMember = null;
            for (GymMember member : list) {
                if (member.getId() == id) {
                    targetMember = member;
                    break;
                }
            }
            if (targetMember == null) {
                throw new IllegalArgumentException("Member ID not found.");
            }

            if (targetMember.getActiveStatus()) {
                throw new IllegalStateException("Membership is already active.");
            }

            targetMember.activateMembership();
            JOptionPane.showMessageDialog(fr5, "Membership activated successfully!");
            idField.setText("");

    });
fr5.add(activateButton);
```

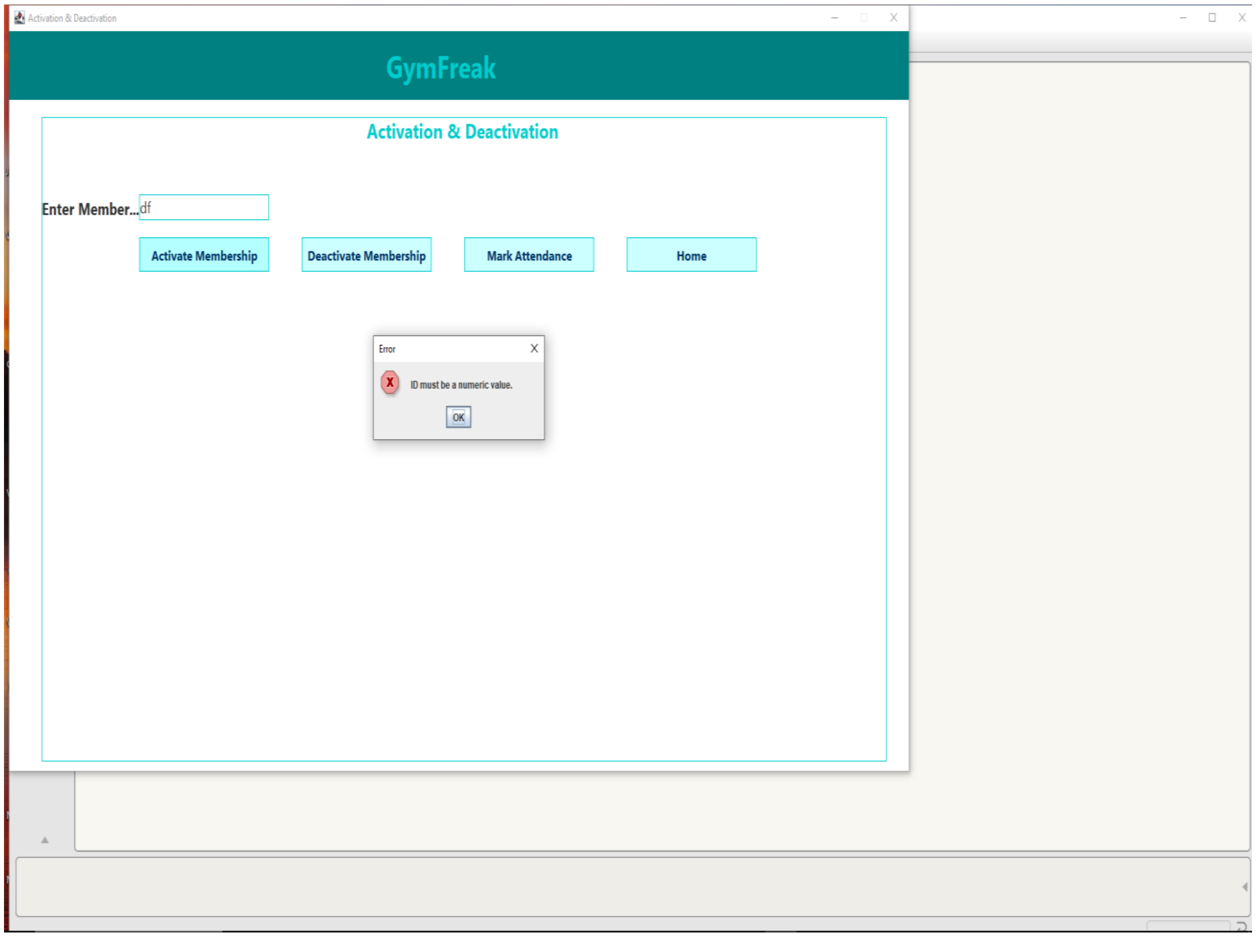
Figure 31: run time error 1.3



```
JButton activateButton = createStyledButton("Activate Membership");
activateButton.setBounds(200, 240, 200, 40);
activateButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
activateButton.addActionListener(e -> {
    try {
        String idText = idField.getText().trim();
        if (idText.isEmpty()) {
            throw new IllegalArgumentException("ID cannot be empty.");
        }
        int id;
        try {
            id = Integer.parseInt(idText);
            if (id <= 0) {
                throw new IllegalArgumentException("ID must be a positive number.");
            }
        } catch (NumberFormatException ex) {
            throw new IllegalArgumentException("ID must be a numeric value.");
        }

        GymMember targetMember = null;
        for (GymMember member : list) {
            if (member.getId() == id) {
                targetMember = member;
                break;
            }
        }
        if (targetMember == null) {
            throw new IllegalArgumentException("Member ID not found.");
        }

        if (targetMember.getActiveStatus()) {
            throw new IllegalStateException("Membership is already active.");
        }

        targetMember.activateMembership();
        JOptionPane.showMessageDialog(fr5, "Membership activated successfully!");
        idField.setText("");
    } catch (IllegalArgumentException | IllegalStateException ex) {
        JOptionPane.showMessageDialog(fr5, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    } catch (Exception ex) {
        JOptionPane.showMessageDialog(fr5, "Unexpected error occurred: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
    }
});
fr5.add(activateButton);
```

GymGUI - 24046697 Abhishek Kumar Shah coursework
Class Edit Tools Options
GymGUI
Compile Undo Cut Copy Paste Find... Close
Source Code
Find: activation()
Prev Next Match Case Replace
saved

Figure 32: runtime error 1.4



ID must be a numeric value.

Figure 33: run time error 1.5

Logical Error

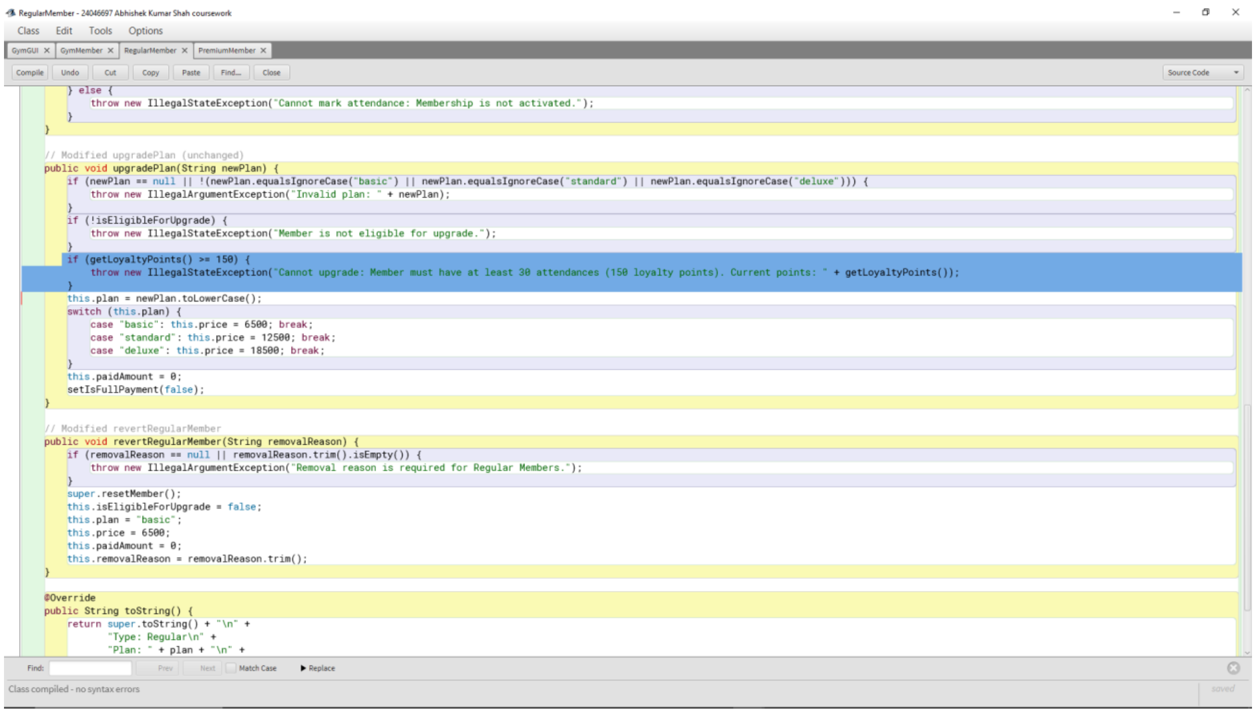
Logical errors occur when the program runs without crashing but doesn't behave the way it should because the logic is incorrect.

Example Encountered:

I initially allowed Regular Members to upgrade plans even if their loyalty points were below the required limit, due to a mistake in the condition used.

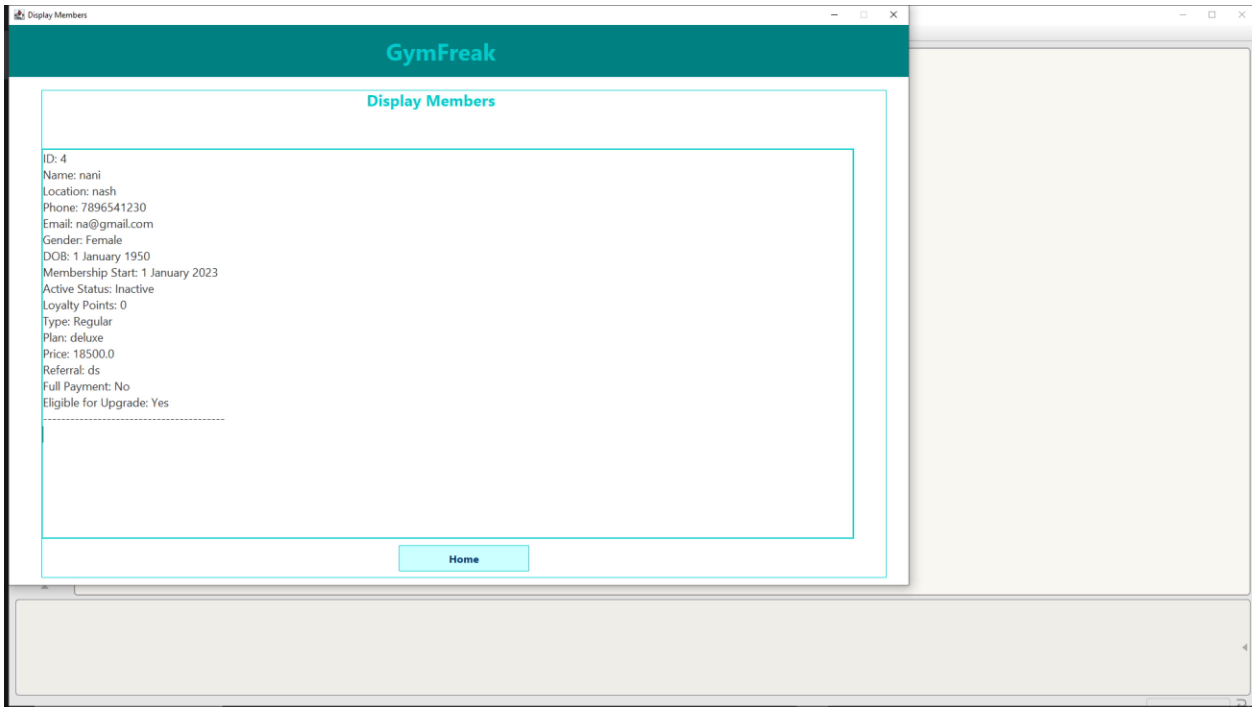
How I Fixed It:

I reviewed the if conditions and compared them with the project requirements. By testing the feature with different loyalty points, I realized the mistake and corrected the logic to check for at least 150 points before allowing an upgrade.

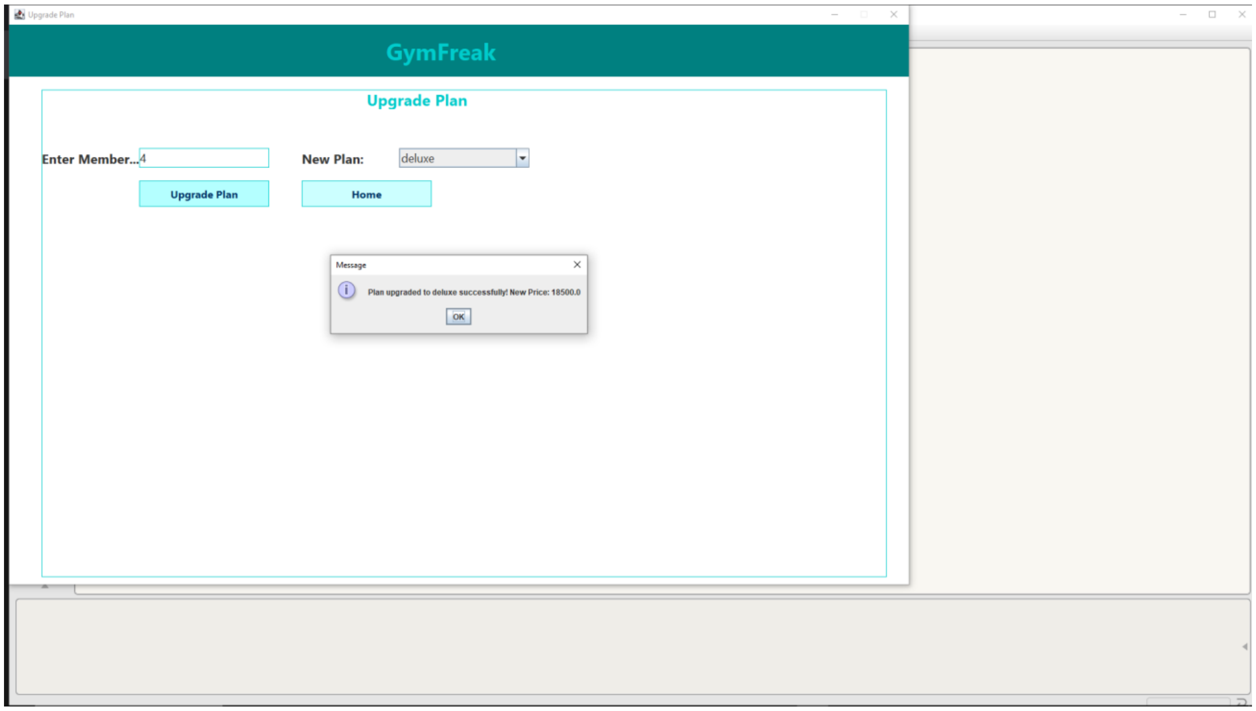

RegularMember - 24046697 Abhishek Kumar Shah coursework
Class Edit Tools Options
GymGUI GymMember RegularMember PremiumMember
Compile Undo Cut Copy Paste Find... Close
Source Code

```java
        } else {
            throw new IllegalStateException("Cannot mark attendance: Membership is not activated.");
        }
    }

    // Modified upgradePlan (unchanged)
    public void upgradePlan(String newPlan) {
        if (newPlan == null || !(newPlan.equalsIgnoreCase("basic") || newPlan.equalsIgnoreCase("standard") || newPlan.equalsIgnoreCase("deluxe"))) {
            throw new IllegalArgumentException("Invalid plan: " + newPlan);
        }
        if (!isEligibleForUpgrade) {
            throw new IllegalStateException("Member is not eligible for upgrade.");
        }
        if (getLoyaltyPoints() >= 150) {
            throw new IllegalStateException("Cannot upgrade: Member must have at least 30 attendances (150 loyalty points). Current points: " + getLoyaltyPoints());
        }
        this.plan = newPlan.toLowerCase();
        switch (this.plan) {
            case "basic": this.price = 6500; break;
            case "standard": this.price = 12500; break;
            case "deluxe": this.price = 18500; break;
        }
        this.paidAmount = 0;
        setIsFullPayment(false);
    }

    // Modified revertRegularMember
    public void revertRegularMember(String removalReason) {
        if (removalReason == null || removalReason.trim().isEmpty()) {
            throw new IllegalArgumentException("Removal reason is required for Regular Members.");
        }
        super.resetMember();
        this.isEligibleForUpgrade = false;
        this.plan = "basic";
        this.price = 6500;
        this.paidAmount = 0;
        this.removalReason = removalReason.trim();
    }

    @Override
    public String toString() {
        return super.toString() + "\n" +
               "Type: Regular\n" +
               "Plan: " + plan + "\n" +
```

Class compiled - no syntax errors



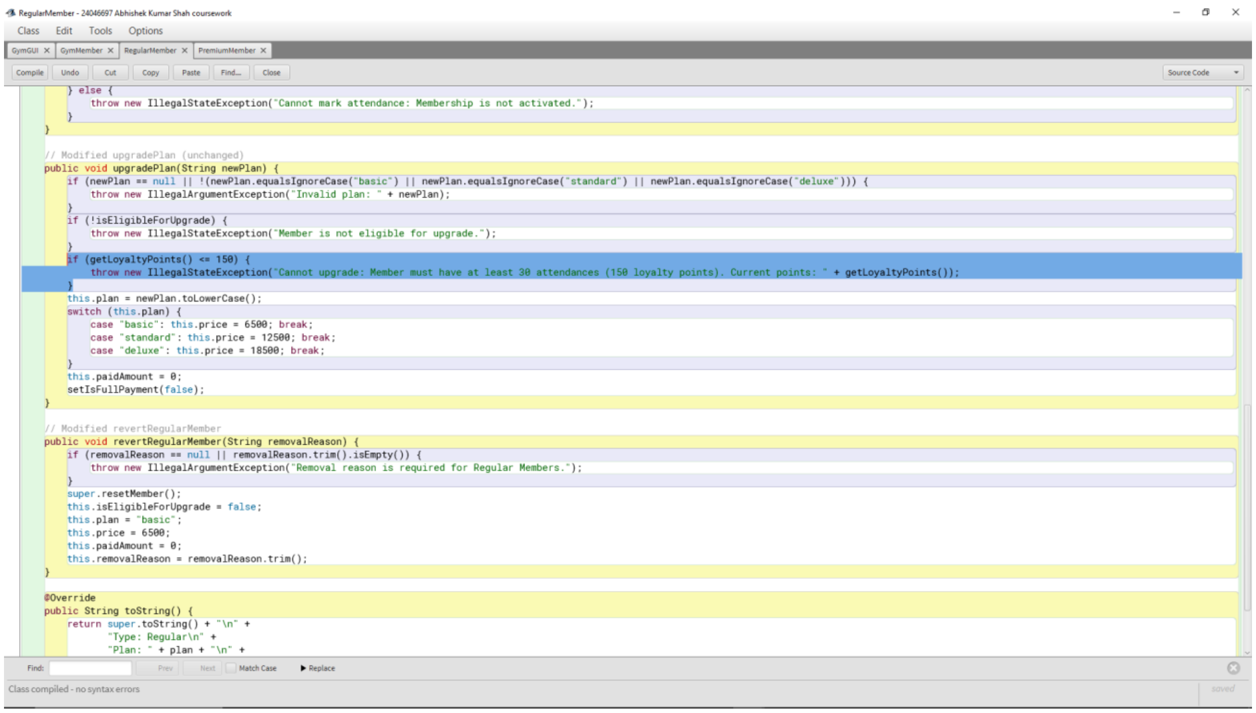
```
ID: 4
Name: nani
Location: nash
Phone: 7896541230
Email: na@gmail.com
Gender: Female
DOB: 1 January 1950
Membership Start: 1 January 2023
Active Status: Inactive
Loyalty Points: 0
Type: Regular
Plan: deluxe
Price: 18500.0
Referral: ds
Full Payment: No
Eligible for Upgrade: Yes
----------------------------------------
```



Enter Member...: 4

New Plan: deluxe

Plan upgraded to deluxe successfully! New Price: 18500.0


RegularMember - 24046697 Abhishek Kumar Shah coursework
Class Edit Tools Options
GymGUI RegularMember GymMember PremiumMember
Compile Undo Cut Copy Paste Find... Close
Source Code
Find: Prev Next Match Case Replace

```java
        } else {
            throw new IllegalStateException("Cannot mark attendance: Membership is not activated.");
        }
    }

    // Modified upgradePlan (unchanged)
    public void upgradePlan(String newPlan) {
        if (newPlan == null || !(newPlan.equalsIgnoreCase("basic") || newPlan.equalsIgnoreCase("standard") || newPlan.equalsIgnoreCase("deluxe"))) {
            throw new IllegalArgumentException("Invalid plan: " + newPlan);
        }
        if (!isEligibleForUpgrade) {
            throw new IllegalStateException("Member is not eligible for upgrade.");
        }
        if (getLoyaltyPoints() <= 150) {
            throw new IllegalStateException("Cannot upgrade: Member must have at least 30 attendances (150 loyalty points). Current points: " + getLoyaltyPoints());
        }
        this.plan = newPlan.toLowerCase();
        switch (this.plan) {
            case "basic": this.price = 6500; break;
            case "standard": this.price = 12500; break;
            case "deluxe": this.price = 18500; break;
        }
        this.paidAmount = 0;
        setIsFullPayment(false);
    }

    // Modified revertRegularMember
    public void revertRegularMember(String removalReason) {
        if (removalReason == null || removalReason.trim().isEmpty()) {
            throw new IllegalArgumentException("Removal reason is required for Regular Members.");
        }
        super.resetMember();
        this.isEligibleForUpgrade = false;
        this.plan = "basic";
        this.price = 6500;
        this.paidAmount = 0;
        this.removalReason = removalReason.trim();
    }

    @Override
    public String toString() {
        return super.toString() + "\n" +
               "Type: Regular\n" +
               "Plan: " + plan + "\n" +
```

Class compiled - no syntax errors

Conclusion

This Gym Management System was a major project that allowed me to apply Java programming skills in a practical way. The application helps manage gym members, both regular and premium, through a user-friendly graphical interface. Users can register new members, manage their membership status, record attendance, handle payments, and store or retrieve data using a text file. The project includes several features commonly needed in real-world systems, and I was able to complete all the main parts successfully.

The system performs the key tasks as intended. I was able to add regular and premium members with proper checks, such as making sure the ID is unique and the phone number has exactly 10 digits. Regular members can upgrade their plans based on how active they are, and premium members can get discounts. I also built features to mark attendance, track loyalty points, and revert members if needed. The save and load functions work smoothly, allowing data to be stored in a file and loaded back later.

The graphical interface, built using Java Swing, is organized into different windows where users can choose what they want to do—like add a member or handle payments. Each part of the program works together to make managing the gym simple and efficient.

Working on this project helped me improve my understanding of object-oriented programming, especially the use of classes, inheritance, and methods. I learned how to create a base class (GymMember) with shared attributes and then extend it to make RegularMember and PremiumMember with their own specific features. This made the code easier to manage and update.

I also got better at designing user interfaces with Swing. I learned how to add buttons, input fields, and menus, and how to switch between different screens based on user actions. Writing clear and organized code became easier as I practiced building and linking different parts of the program.

Another big learning area was validating user input. I had to make sure users enter correct data, like valid phone numbers or proper email addresses, and give clear error messages when something is wrong. I also got more confident using files to save and load data, which taught me how to work with text files in Java.

There were a few problems I ran into, especially with the interface. Switching between windows was confusing at first. Sometimes multiple screens would stay open or not respond properly. I fixed this by making sure to close the current screen before opening the next one.

I also had some trouble with input validation. At first, the program didn't catch things like duplicate IDs or incorrect phone numbers. I solved this by adding extra checks in the code and using pop-up messages to inform the user when input was wrong.

Another tricky part was handling payments, especially applying discounts for premium members and making sure the full amount was updated correctly. I had to carefully test different situations to make sure everything added up and the payment status changed at the right time.

Saving and reading from the file was also a bit hard. I had to make sure the formatting was readable and that the program didn't crash if the file was missing or empty. I added checks to prevent errors and give useful messages instead.

I also created a testing plan that helped me make sure everything worked step-by-step. By going through the features one at a time—like adding members, marking attendance, making payments, and saving to file—I could catch and fix problems early.

This project helped me grow a lot as a Java programmer. I'm proud of how far I've come—from setting up the basic structure to adding full functionality with a user interface. I now feel more confident in handling real-world problems using Java. If I continue working on this project, I'd like to improve the visual design and maybe connect it to a database instead of a text file.

Overall, building this gym management system was a challenging but rewarding experience that taught me valuable skills I can use in future projects.

Appendix

Appendix for GymMember Class

```
/** Abstract class for gym members, providing core attributes and methods. */
public abstract class GymMember {
    // Member details
    protected int id;
    protected String name;
    protected String location;
    protected String phone;
    protected String email;
    protected String gender;
    protected String dob;
    protected String membershipStart;
    protected boolean activeStatus;
    protected int loyaltyPoints;
    protected boolean isFullPayment;

    /** Constructor initializes member with provided details. */
    public GymMember(int id, String name, String location, String phone, String email,
                    String gender, String dob, String membershipStart) {
        this.id = id;
        this.name = name;
        this.location = location;
        this.phone = phone;
        this.email = email;
        this.gender = gender;
        this.dob = dob;
        this.membershipStart = membershipStart;
        this.activeStatus = false; // Default inactive
        this.loyaltyPoints = 0; // Start with zero points
        this.isFullPayment = false; // Payment not completed
    }

    // Getters
    public int getId() { return id; }
    public String getName() { return name; }
    public String getLocation() { return location; }
    public String getPhone() { return phone; }
    public String getEmail() { return email; }
    public String getGender() { return gender; }
    public String getDob() { return dob; }
    public String getMembershipStart() { return membershipStart; }
    public boolean getActiveStatus() { return activeStatus; }
    public int getLoyaltyPoints() { return loyaltyPoints; }
    public boolean getIsFullPayment() { return isFullPayment; }

    // Setter for payment status
    public void setIsFullPayment(boolean isFullPayment) {
        this.isFullPayment = isFullPayment;
    }

    /** Resets member status and clears points. */
    public void resetMember() {
        this.loyaltyPoints = 0;
        this.activeStatus = false;
        this.isFullPayment = false;
    }

    // Abstract methods for subclasses
    public abstract void makePayment(double amount);
    public abstract void markAttendance();
    /** Activates membership. */
    public void activateMembership() {
        this.activeStatus = true;
    }

    /** Deactivates membership if active, else shows message. */
    public void deactivateMembership() {
        if (this.activeStatus) {
            this.activeStatus = false;
        } else {
            System.out.println("Your membership status is already deactivated.");
        }
    }

    // Sets loyalty points for subclasses
    protected void setLoyaltyPoints(int points) {
        this.loyaltyPoints = points;
    }
}
```

Appendix for RegularMember class

```
/** Class for regular gym members with plan-based pricing. */
public class RegularMember extends GymMember {
    protected String plan;
    protected double price;
    protected String referral;
    protected boolean isEligibleForUpgrade;
    protected String removalReason;
    protected double paidAmount;
    /** Constructor initializes regular member with validations. */
    public RegularMember(int id, String name, String location, String phone, String email,
                         String gender, String dob, String membershipStart, String referral,
                         String plan, double price, boolean isFullPayment) {
        super(id, name, location, phone, email, gender, dob, membershipStart);
        if (referral == null || referral.trim().isEmpty()) {
            throw new IllegalArgumentException("Referral cannot be empty.");
        }
        if (plan == null || !(plan.equalsIgnoreCase("basic") ||
plan.equalsIgnoreCase("standard") || plan.equalsIgnoreCase("deluxe"))) {
            throw new IllegalArgumentException("Plan must be basic, standard, or
deluxe.");
        }
        if (price <= 0) {
            throw new IllegalArgumentException("Price must be positive.");
        }
        this.plan = plan.toLowerCase();
        this.price = price;
        this.referral = referral.trim();
        this.isFullPayment = isFullPayment;
        this.isEligibleForUpgrade = true; // Eligible by default
        this.removalReason = ""; // Empty reason initially
        this.paidAmount = isFullPayment ? price : 0; // Set paid amount
    }

    // Getters
    public String getPlan() { return plan; }
    public double getPrice() { return price; }
    public String getReferral() { return referral; }
    public boolean getIsFullPayment() { return isFullPayment; }
    public boolean getIsEligibleForUpgrade() { return isEligibleForUpgrade; }
    public String getRemovalReason() { return removalReason; }
    public double getPaidAmount() { return paidAmount; }

    // Setter for payment status
    public void setIsFullPayment(boolean isFullPayment) {
        this.isFullPayment = isFullPayment;
    }

    /** Processes payment, checks for overpayment. */
    @Override
    public void makePayment(double amount) {
        if (amount < 0) {
            throw new IllegalArgumentException("Payment amount cannot be negative.");
        }
        if (paidAmount + amount > price) {
            throw new IllegalArgumentException("Payment exceeds remaining balance of "
+ (price - paidAmount));
        }
        this.paidAmount += amount;
        if (paidAmount >= price) {
            setIsFullPayment(true);
        }
    }

    /** Marks attendance, adds 5 points if active. */
    @Override
    public void markAttendance() {
        if (getActiveStatus()) {
            setLoyaltyPoints(getLoyaltyPoints() + 5);
        } else {
            throw new IllegalStateException("Cannot mark attendance: Membership is not
activated.");
        }
    }

    /** Upgrades plan if eligible and points are sufficient. */
    public void upgradePlan(String newPlan) {
        if (newPlan == null || !(newPlan.equalsIgnoreCase("basic") ||
newPlan.equalsIgnoreCase("standard") || newPlan.equalsIgnoreCase("deluxe"))) {
            throw new IllegalArgumentException("Invalid plan: " + newPlan);
        }
        if (!isEligibleForUpgrade) {
            throw new IllegalStateException("Member is not eligible for upgrade.");
        }
        if (getLoyaltyPoints() <= 150) {
            throw new IllegalStateException("Cannot upgrade: Member must have at least
30 attendances (150 loyalty points). Current points: " + getLoyaltyPoints());
        }
        this.plan = newPlan.toLowerCase();
        switch (this.plan) {
            case "basic": this.price = 6500; break;
            case "standard": this.price = 12500; break;
            case "deluxe": this.price = 18500; break;
        }
        this.paidAmount = 0; // Reset payment
        setIsFullPayment(false);
    }

    /** Reverts member to basic plan, clears eligibility. */
    public void revertRegularMember(String removalReason) {
        if (removalReason == null || removalReason.trim().isEmpty()) {
            throw new IllegalArgumentException("Removal reason is required for Regular
Members.");
        }
        super.resetMember();
        this.isEligibleForUpgrade = false;
        this.plan = "basic";
        this.price = 6500;
        this.paidAmount = 0;
        this.removalReason = removalReason.trim();
    }

    /** Returns regular member details as string. */
    @Override
    public String toString() {
        return super.toString() + "\n" +
               "Type: Regular\n" +
               "Plan: " + plan + "\n" +
               "Price: " + price + "\n" +
               "Referral: " + referral + "\n" +
               "Paid Amount: " + paidAmount + "\n" +
               "Full Payment: " + (isFullPayment ? "Yes" : "No") + "\n" +
               "Eligible for Upgrade: " + (isEligibleForUpgrade ? "Yes" : "No") +
               (!removalReason.isEmpty() ? "\nRemoval Reason: " + removalReason : "");
    }
}
```

Appendix for PremiumMember class

```
/** Class for premium gym members with extra features. */
public class PremiumMember extends GymMember {
   protected String personalTrainer;
   protected double discountAmount;
   protected double paidAmount;
   protected String removalReason;

   /** Constructor sets up premium member with validations. */
   public PremiumMember(int id, String name, String location, String phone, String
email,
                        String gender, String dob, String membershipStart, String
personalTrainer,
                        double discountAmount, boolean isFullPayment) {
      super(id, name, location, phone, email, gender, dob, membershipStart);
      if (personalTrainer == null || personalTrainer.trim().isEmpty()) {
         throw new IllegalArgumentException("Personal trainer name cannot be empty.");
      }
      if (discountAmount < 0 || discountAmount > getPremiumCharge()) {
         throw new IllegalArgumentException("Discount must be between 0 and " +
getPremiumCharge());
      }
      this.personalTrainer = personalTrainer.trim();
      this.discountAmount = discountAmount;
      this.paidAmount = isFullPayment ? (getPremiumCharge() - discountAmount) : 0;
      this.isFullPayment = isFullPayment;
      this.removalReason = ""; // Empty reason initially
   }
```

```
    // Getters
    public String getPersonalTrainer() { return personalTrainer; }
    public double getDiscountAmount() { return discountAmount; }
    public double getPaidAmount() { return paidAmount; }
    public boolean getIsFullPayment() { return isFullPayment; }
    public double getPremiumCharge() { return 50000; } // Fixed charge
    public String getRemovalReason() { return removalReason; }

    /** Sets valid discount amount. */
    public void setDiscountAmount(double discountAmount) {
        if (discountAmount < 0 || discountAmount > getPremiumCharge()) {
            throw new IllegalArgumentException("Discount must be between 0 and " +
getPremiumCharge());
        }
        this.discountAmount = discountAmount;
    }

    // Setter for payment status
    public void setIsFullPayment(boolean isFullPayment) {
        this.isFullPayment = isFullPayment;
    }

    /** Processes payment and updates status. */
    @Override
    public void makePayment(double amount) {
        if (amount < 0) {
            throw new IllegalArgumentException("Payment amount cannot be negative.");
        }
        this.paidAmount += amount;
        double totalDue = getPremiumCharge() - discountAmount;
        if (paidAmount >= totalDue) {
            setIsFullPayment(true);
        }
    }

    /** Marks attendance, adds 10 points if active. */
    @Override
    public void markAttendance() {
        if (getActiveStatus()) {
            setLoyaltyPoints(getLoyaltyPoints() + 10);
        } else {
            throw new IllegalStateException("Cannot mark attendance: Membership is not
activated.");
        }
    }

    /** Reverts member, clears premium features. */
    public void revertPremiumMember(String removalReason) {
        if (removalReason == null || removalReason.trim().isEmpty()) {
            throw new IllegalArgumentException("Removal reason is required for Premium
Members.");
        }
        super.resetMember();
        this.personalTrainer = "";
        this.isFullPayment = false;
        this.paidAmount = 0;
        this.discountAmount = 0;
        this.removalReason = removalReason.trim();
    }

    /** Returns premium member details as string. */
    @Override
    public String toString() {
        return super.toString() + "\n" +
               "Type: Premium\n" +
               "Personal Trainer: " + personalTrainer + "\n" +
               "Discount Amount: " + discountAmount + "\n" +
               "Paid Amount: " + paidAmount + "\n" +
               "Full Payment: " + (isFullPayment ? "Yes" : "No") +
               (!removalReason.isEmpty() ? "\nRemoval Reason: " + removalReason : "");
    }
}
```

Appendix for GymGUI class

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
import javax.swing.JComboBox;
import javax.swing.JButton;
import javax.swing.JTextArea;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JOptionPane;
import javax.swing.border.LineBorder;
import java.awt.Graphics;
import java.io.BufferedWriter;
import javax.swing.JCheckBox;
import java.awt.Dimension;

import javax.swing.SwingUtilities;
import java.awt.Color;
import java.awt.Font;

import java.awt.BorderLayout;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.io.File;
import java.io.FileWriter;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

public class GymGUI extends JFrame {
    private static ArrayList<GymMember> list = new ArrayList<>();
    private JFrame fr1, fr2, fr3, fr4, fr5, fr6, fr7, fr8, fr9;

    public GymGUI() {
        fr1 = new JFrame("GymFreak");
        fr1.setSize(1400, 900);
        fr1.setLayout(null);
        fr1.setResizable(false);
        fr1.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr1.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
```

```
headerPanel.setBounds(0, 0, 1400, 80);
headerPanel.setBackground(new Color(0, 128, 128));
headerPanel.setLayout(null);
fr1.add(headerPanel);

JLabel h1 = new JLabel("GymFreak");
h1.setBounds(580, 15, 240, 50);
h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
h1.setForeground(new Color(0, 204, 204));
headerPanel.add(h1);

JLabel h2 = new JLabel("Membership Management");
h2.setBounds(500, 100, 400, 30);
h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
h2.setForeground(new Color(0, 204, 204));
fr1.add(h2);

// Buttons
JButton addMember = createStyledButton("Add Member");
addMember.setBounds(250, 200, 300, 60);
addMember.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr1.dispose();
        addMember();
    }
});
fr1.add(addMember);

JButton activation = createStyledButton("Activation & Deactivation");
activation.setBounds(850, 200, 300, 60);
activation.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr1.dispose();
        activation();
    }
});
fr1.add(activation);

JButton revert = createStyledButton("Revert Member");
revert.setBounds(250, 300, 300, 60);
revert.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr1.dispose();
        addRevertmember();
    }
});
fr1.add(revert);

JButton show = createStyledButton("Display Members");
show.setBounds(850, 300, 300, 60);
show.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr1.dispose();
        display();
    }
});
fr1.add(show);
```

```
JButton upgradePlan = createStyledButton("Upgrade Plan");
upgradePlan.setBounds(250, 400, 300, 60);
upgradePlan.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr1.dispose();
        upgradePlan();
    }
});
fr1.add(upgradePlan);

JButton payment = createStyledButton("Payment");
payment.setBounds(850, 400, 300, 60);
payment.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr1.dispose();
        processPayment();
    }
});
fr1.add(payment);

JButton readFile = createStyledButton("Read from File");
readFile.setBounds(250, 500, 300, 60);
readFile.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        readFromFile();
    }
});
fr1.add(readFile);
```

```
        JButton saveFile = createStyledButton("Save to File");
        saveFile.setBounds(850, 500, 300, 60);
        saveFile.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                saveToFile();
            }
        });
        fr1.add(saveFile);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr1.add(mainPanel);

        fr1.setVisible(true);
    }

    private JButton createStyledButton(String text) {
        JButton button = new JButton(text) {
            @Override
            protected void paintComponent(Graphics g) {
                if (getModel().isRollover()) {
                    g.setColor(new Color(180, 255, 255));
                } else {
                    g.setColor(new Color(204, 255, 255));
                }
                g.fillRect(0, 0, getWidth(), getHeight());
                super.paintComponent(g);
            }
        };
        button.setFont(new Font("Segoe UI", Font.BOLD, 20));
        button.setForeground(new Color(0, 51, 102));
        button.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        button.setContentAreaFilled(false);
        return button;
    }

    private void readFromFile() {
        File file = new File("MemberDetails.txt");
        JFrame displayFrame = new JFrame("Member Details");
        displayFrame.setSize(1400, 900);
        displayFrame.setLayout(null);
        displayFrame.setResizable(false);
        displayFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        displayFrame.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
        headerPanel.setBounds(0, 0, 1400, 80);
        headerPanel.setBackground(new Color(0, 128, 128));
        headerPanel.setLayout(null);
        displayFrame.add(headerPanel);

        JLabel h1 = new JLabel("GymFreak");
        h1.setBounds(580, 15, 240, 50);
        h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
        h1.setForeground(new Color(0, 204, 204));
```

```
headerPanel.add(h1);

JLabel h2 = new JLabel("Member Details");
h2.setBounds(550, 100, 300, 30);
h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
h2.setForeground(new Color(0, 204, 204));
displayFrame.add(h2);

// Display Area
JTextArea displayArea = new JTextArea();
displayArea.setFont(new Font("Courier New", Font.PLAIN, 16));
displayArea.setEditable(false);
displayArea.setBorder(new LineBorder(new Color(0, 204, 204), 1));

try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
    StringBuilder sb = new StringBuilder();
    String line;
    while ((line = reader.readLine()) != null) {
        sb.append(line).append("\n");
    }
    if (sb.length() == 0) {
        sb.append("No member details found in the file.\n");
    }
    displayArea.setText(sb.toString());
} catch (IOException ex) {
    displayArea.setText("Error reading file: " + ex.getMessage());
}

JScrollPane scrollPane = new JScrollPane(displayArea);
scrollPane.setBounds(50, 150, 1300, 650);
scrollPane.setBorder(new LineBorder(new Color(0, 204, 204), 1));
```

```
        displayFrame.add(scrollPane);

        // Back Button
        JButton back = createStyledButton("Back");
        back.setBounds(600, 820, 200, 40);
        back.setFont(new Font("Segoe UI", Font.BOLD, 16));
        back.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                displayFrame.dispose();
            }
        });
        displayFrame.add(back);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        displayFrame.add(mainPanel);

        displayFrame.setVisible(true);
    }

    private void saveToFile() {
        File file = new File("MemberDetails.txt");
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(file))) {
            writer.write(String.format("%-5s %-15s %-15s %-15s %-25s %-20s %-10s %-10s
%-10s %15s %-10s %-15s %-15s %-15s\n",
                "ID", "Name", "Location", "Phone", "Email", "Membership Start", "Plan",
"Price", "Attendance", "Loyalty Points",
                "Active Status", "Full Payment", "Discount Amount", "Net Amount Paid"));
            writer.write("-------------------------------------------------------------------------------------
----------------------------------\n");

            for (GymMember member : list) {
                String plan = "";
                String price = "";
                String attendance = String.valueOf(member.getLoyaltyPoints());
                String discountAmount = "0";
                String netAmountPaid = "0";

                if (member instanceof RegularMember) {
                  RegularMember rm = (RegularMember) member;
                  plan = rm.getPlan();
                  price = String.format("%.2f", rm.getPrice());
                  discountAmount = "N/A";
                  netAmountPaid = rm.getIsFullPayment() ? String.format("%.2f",
rm.getPrice()) : "0.00";
                  if (!rm.getRemovalReason().isEmpty()) {
                    writer.write("Removal Reason: " + rm.getRemovalReason() + "\n");
                  }
                } else if (member instanceof PremiumMember) {
                  PremiumMember pm = (PremiumMember) member;
                  plan = "Premium";
                  price = String.format("%.2f", pm.getPremiumCharge());
                  discountAmount = String.format("%.2f", pm.getDiscountAmount());
                  netAmountPaid = String.format("%.2f", pm.getPaidAmount());
                  if (!pm.getRemovalReason().isEmpty()) {
                    writer.write("Removal Reason: " + pm.getRemovalReason() + "\n");
```

```
                }
            }

            writer.write(String.format("%-5d %-15s %-15s %-15s %-25s %-20s %-10s %-
10s %-10s %15d %-10s %-15s %-15s %-15s\n",
                member.getId(),
                truncate(member.getName(), 15),
                truncate(member.getLocation(), 15),
                truncate(member.getPhone(), 15),
                truncate(member.getEmail(), 25),
                truncate(member.getMembershipStart(), 20),
                truncate(plan, 10),
                truncate(price, 10),
                truncate(attendance, 10),
                member.getLoyaltyPoints(),
                member.getActiveStatus() ? "Active" : "Inactive",
                member.getIsFullPayment() ? "Yes" : "No",
                truncate(discountAmount, 15),
                truncate(netAmountPaid, 15)));
          }
          JOptionPane.showMessageDialog(fr1, "Members saved successfully to " +
file.getName(), "Success", JOptionPane.INFORMATION_MESSAGE);
        } catch (IOException ex) {
          JOptionPane.showMessageDialog(fr1, "Error saving file: " + ex.getMessage(),
"File Error", JOptionPane.ERROR_MESSAGE);
        }
    }

    private String truncate(String str, int length) {
        if (str == null) return "";
        return str.length() > length ? str.substring(0, length) : str;
```

```
}

public void addMember() {
    fr2 = new JFrame("Add Member");
    fr2.setSize(1400, 900);
    fr2.setLayout(null);
    fr2.setResizable(false);
    fr2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    fr2.getContentPane().setBackground(new Color(255, 255, 255));

    // Header
    JPanel headerPanel = new JPanel();
    headerPanel.setBounds(0, 0, 1400, 80);
    headerPanel.setBackground(new Color(0, 128, 128));
    headerPanel.setLayout(null);
    fr2.add(headerPanel);

    JLabel h1 = new JLabel("GymFreak");
    h1.setBounds(580, 15, 240, 50);
    h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
    h1.setForeground(new Color(0, 204, 204));
    headerPanel.add(h1);

    JLabel h2 = new JLabel("Choose Membership Type");
    h2.setBounds(550, 100, 300, 30);
    h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
    h2.setForeground(new Color(0, 204, 204));
    fr2.add(h2);

    // Buttons
    JButton addRegularmember = createStyledButton("Add Regular Member");
```

```
addRegularmember.setBounds(200, 200, 300, 50);
addRegularmember.setFont(new Font("Segoe UI", Font.BOLD, 16));
addRegularmember.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr2.dispose();
        addRegularMember();
    }
});
fr2.add(addRegularmember);

JButton addPremiummember = createStyledButton("Add Premium Member");
addPremiummember.setBounds(550, 200, 300, 50);
addPremiummember.setFont(new Font("Segoe UI", Font.BOLD, 16));
addPremiummember.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr2.dispose();
        addPremiumMember();
    }
});
fr2.add(addPremiummember);

JButton back = createStyledButton("Home");
back.setBounds(900, 200, 300, 50);
back.setFont(new Font("Segoe UI", Font.BOLD, 16));
back.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        fr2.dispose();
        new GymGUI();
```

```
            }
        });
        fr2.add(back);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr2.add(mainPanel);

        fr2.setVisible(true);
    }

    public void addRegularMember() {
        fr3 = new JFrame("Add Regular Member");
        fr3.setSize(1400, 900);
        fr3.setLayout(null);
        fr3.setResizable(false);
        fr3.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr3.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
        headerPanel.setBounds(0, 0, 1400, 80);
        headerPanel.setBackground(new Color(0, 128, 128));
        headerPanel.setLayout(null);
        fr3.add(headerPanel);

        JLabel h1 = new JLabel("GymFreak");
        h1.setBounds(580, 15, 240, 50);
        h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
        h1.setForeground(new Color(0, 204, 204));
        headerPanel.add(h1);

        JLabel h2 = new JLabel("Add Regular Member");
        h2.setBounds(550, 100, 300, 30);
        h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
        h2.setForeground(new Color(0, 204, 204));
        fr3.add(h2);

        // Plan Prices
        JLabel l1 = new JLabel("Regular Plans: Basic (6500/-), Standard (12500/-), Deluxe
(18500/-)");
        l1.setBounds(200, 140, 800, 30);
        l1.setFont(new Font("Segoe UI", Font.BOLD, 20));
        l1.setForeground(new Color(0, 51, 102));
        fr3.add(l1);

        JLabel l2 = new JLabel("Premium Plan: 50000/-");
        l2.setBounds(900, 140, 300, 30);
        l2.setFont(new Font("Segoe UI", Font.BOLD, 20));
        l2.setForeground(new Color(0, 51, 102));
        fr3.add(l2);

        // Input Fields - Row 1
        JLabel idLabel = new JLabel("ID:");
        idLabel.setBounds(100, 200, 100, 30);
        idLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
        fr3.add(idLabel);
```

```
JTextField idField = new JTextField();
idField.setBounds(200, 200, 200, 30);
idField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
idField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr3.add(idField);

JLabel nameLabel = new JLabel("Name:");
nameLabel.setBounds(450, 200, 100, 30);
nameLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr3.add(nameLabel);

JTextField nameField = new JTextField();
nameField.setBounds(550, 200, 200, 30);
nameField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
nameField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr3.add(nameField);

JLabel locationLabel = new JLabel("Location:");
locationLabel.setBounds(800, 200, 150, 30);
locationLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr3.add(locationLabel);

JTextField locationField = new JTextField();
locationField.setBounds(950, 200, 200, 30);
locationField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
locationField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr3.add(locationField);

// Input Fields - Row 2
JLabel phoneLabel = new JLabel("Phone:");
phoneLabel.setBounds(100, 270, 100, 30);
phoneLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr3.add(phoneLabel);

JTextField phoneField = new JTextField();
phoneField.setBounds(200, 270, 200, 30);
phoneField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
phoneField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr3.add(phoneField);

JLabel emailLabel = new JLabel("Email:");
emailLabel.setBounds(450, 270, 100, 30);
emailLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr3.add(emailLabel);

JTextField emailField = new JTextField();
emailField.setBounds(550, 270, 200, 30);
emailField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
emailField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr3.add(emailField);

JLabel genderLabel = new JLabel("Gender:");
genderLabel.setBounds(800, 270, 150, 30);
genderLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr3.add(genderLabel);

JRadioButton male = new JRadioButton("Male");
male.setFont(new Font("Segoe UI", Font.PLAIN, 18));
male.setBackground(new Color(255, 255, 255));
male.setBounds(950, 270, 80, 30);
fr3.add(male);
```

```
JRadioButton female = new JRadioButton("Female");
female.setFont(new Font("Segoe UI", Font.PLAIN, 18));
female.setBackground(new Color(255, 255, 255));
female.setBounds(950, 300, 100, 30);
fr3.add(female);

JRadioButton other = new JRadioButton("Other");
other.setFont(new Font("Segoe UI", Font.PLAIN, 18));
other.setBackground(new Color(255, 255, 255));
other.setBounds(950, 330, 90, 30);
fr3.add(other);

ButtonGroup genderGroup = new ButtonGroup();
genderGroup.add(male);
genderGroup.add(female);
genderGroup.add(other);

// Input Fields - Row 3
JLabel dobLabel = new JLabel("DOB:");
dobLabel.setBounds(100, 380, 100, 30);
dobLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr3.add(dobLabel);

JComboBox<String> dobDay = new JComboBox<>(generateNumbers(1, 31));
dobDay.setBounds(200, 380, 60, 30);
dobDay.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr3.add(dobDay);

JComboBox<String> dobMonth = new JComboBox<>(getMonths());
dobMonth.setBounds(280, 380, 100, 30);
dobMonth.setFont(new Font("Segoe UI", Font.PLAIN, 18));
```

```
fr3.add(dobMonth);

JComboBox<String> dobYear = new JComboBox<>(generateYears(1950, 2025));
dobYear.setBounds(400, 380, 100, 30);
dobYear.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr3.add(dobYear);

JLabel mspLabel = new JLabel("Membership Start:");
mspLabel.setBounds(550, 380, 200, 30);
mspLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr3.add(mspLabel);

JComboBox<String> mspDay = new JComboBox<>(generateNumbers(1, 31));
mspDay.setBounds(750, 380, 60, 30);
mspDay.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr3.add(mspDay);

JComboBox<String> mspMonth = new JComboBox<>(getMonths());
mspMonth.setBounds(830, 380, 100, 30);
mspMonth.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr3.add(mspMonth);

JComboBox<String> mspYear = new JComboBox<>(generateYears(2023, 2025));
mspYear.setBounds(950, 380, 100, 30);
mspYear.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr3.add(mspYear);

// Input Fields - Row 4
JLabel referralLabel = new JLabel("Referral:");
referralLabel.setBounds(100, 450, 100, 30);
referralLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
```

```
        fr3.add(referralLabel);

        JTextField referralField = new JTextField();
        referralField.setBounds(200, 450, 200, 30);
        referralField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        referralField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        fr3.add(referralField);

        JLabel removalReasonLabel = new JLabel("Removal Reason:");
        removalReasonLabel.setBounds(450, 450, 200, 30);
        removalReasonLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));

        JTextArea removalReasonArea = new JTextArea();
        removalReasonArea.setBounds(650, 450, 200, 60);
        removalReasonArea.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        removalReasonArea.setBorder(new LineBorder(new Color(0, 204, 204), 1));

        // Input Fields - Row 5
        JLabel planLabel = new JLabel("Plan:");
        planLabel.setBounds(100, 520, 100, 30);
        planLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
        fr3.add(planLabel);

        JComboBox<String> planField = new JComboBox<>(new String[]{"basic",
"standard", "deluxe"});
        planField.setBounds(200, 520, 200, 30);
        planField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        fr3.add(planField);

        JCheckBox fullPaymentCheck = new JCheckBox("Paid in Full");
        fullPaymentCheck.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        fullPaymentCheck.setBackground(new Color(255, 255, 255));
        fullPaymentCheck.setBounds(450, 520, 200, 30);

        // Buttons
        JButton clearButton = createStyledButton("Clear");
        clearButton.setBounds(355, 650, 200, 40);
        clearButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
        clearButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                idField.setText("");
                nameField.setText("");
                locationField.setText("");
                phoneField.setText("");
                emailField.setText("");
                referralField.setText("");
                removalReasonArea.setText("");
                dobDay.setSelectedIndex(0);
                dobMonth.setSelectedIndex(0);
                dobYear.setSelectedIndex(0);
                mspDay.setSelectedIndex(0);
                mspMonth.setSelectedIndex(0);
                mspYear.setSelectedIndex(0);
                planField.setSelectedIndex(0);
                fullPaymentCheck.setSelected(false);
                genderGroup.clearSelection();
            }
        });
        fr3.add(clearButton);

        JButton add = createStyledButton("Add Regular Member");
```

```
add.setBounds(575, 650, 200, 40);
add.setFont(new Font("Segoe UI", Font.BOLD, 16));
add.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String idText = idField.getText().trim();
            String name = nameField.getText().trim();
            String location = locationField.getText().trim();
            String phone = phoneField.getText().trim();
            String email = emailField.getText().trim();
            String referral = referralField.getText().trim();
            String removalReason = removalReasonArea.getText().trim();
            String plan = (String) planField.getSelectedItem();
            boolean isFullPayment = fullPaymentCheck.isSelected();

            if (idText.isEmpty()) {
                throw new IllegalArgumentException("ID cannot be empty.");
            }
            int id;
            try {
                id = Integer.parseInt(idText);
                if (id <= 0) {
                    throw new IllegalArgumentException("ID must be a positive number.");
                }
            } catch (NumberFormatException ex) {
                throw new IllegalArgumentException("ID must be a numeric value.");
            }

            for (GymMember member : list) {
                if (member.getId() == id) {
```

```
                throw new IllegalArgumentException("Member ID already exists.");
            }
        }

        if (name.isEmpty() || !name.matches("[a-zA-Z\\s]+")) {
            throw new IllegalArgumentException("Name must contain letters only.");
        }
        if (phone.isEmpty() || !phone.matches("\\d{10}")) {
            throw new IllegalArgumentException("Phone must be a 10-digit
number.");
        }
        if (location.isEmpty()) {
            throw new IllegalArgumentException("Location cannot be empty.");
        }
        if (email.isEmpty() || !email.matches("^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$")) {
            throw new IllegalArgumentException("Invalid email format.");
        }
        if (referral.isEmpty()) {
            throw new IllegalArgumentException("Referral source cannot be
empty.");
        }
        if (plan == null) {
            throw new IllegalArgumentException("Please select a plan.");
        }

        String gender = male.isSelected() ? "Male" : female.isSelected() ? "Female"
: other.isSelected() ? "Other" : "";
        if (gender.isEmpty()) {
            throw new IllegalArgumentException("Please select a gender.");
        }
```

```
            if (dobDay.getSelectedItem() == null || dobMonth.getSelectedItem() == null
|| dobYear.getSelectedItem() == null) {
                throw new IllegalArgumentException("Please select a valid date of
birth.");
            }
            String dob = dobDay.getSelectedItem() + " " + dobMonth.getSelectedItem()
+ " " + dobYear.getSelectedItem();

            if (mspDay.getSelectedItem() == null || mspMonth.getSelectedItem() == null
|| mspYear.getSelectedItem() == null) {
                throw new IllegalArgumentException("Please select a valid membership
start date.");
            }
            String msp = mspDay.getSelectedItem() + " " +
mspMonth.getSelectedItem() + " " + mspYear.getSelectedItem();

            double price;
            switch (plan.toLowerCase()) {
                case "basic":
                    price = 6500;
                    break;
                case "standard":
                    price = 12500;
                    break;
                case "deluxe":
                    price = 18500;
                    break;
                default:
                    throw new IllegalArgumentException("Invalid plan selected.");
            }
```

```
                    RegularMember rm = new RegularMember(id, name, location, phone,
email, gender, dob, msp, referral, plan, price, isFullPayment);
                    rm.makePayment(isFullPayment ? price : 0);
                    list.add(rm);

                    JOptionPane.showMessageDialog(fr3, "Regular Member added
successfully! Price: " + price);
                    clearButton.doClick();
                } catch (IllegalArgumentException ex) {
                    JOptionPane.showMessageDialog(fr3, ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(fr3, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        fr3.add(add);

        JButton back = createStyledButton("Home");
        back.setBounds(795, 650, 200, 40);
        back.setFont(new Font("Segoe UI", Font.BOLD, 16));
        back.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                fr3.dispose();
                new GymGUI();
            }
        });
        fr3.add(back);
        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr3.add(mainPanel);

        fr3.setVisible(true);
    }

    public void addPremiumMember() {
        fr4 = new JFrame("Add Premium Member");
        fr4.setSize(1400, 900);
        fr4.setLayout(null);
        fr4.setResizable(false);
        fr4.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr4.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
        headerPanel.setBounds(0, 0, 1400, 80);
        headerPanel.setBackground(new Color(0, 128, 128));
        headerPanel.setLayout(null);
        fr4.add(headerPanel);

        JLabel h1 = new JLabel("GymFreak");
        h1.setBounds(580, 15, 240, 50);
        h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
        h1.setForeground(new Color(0, 204, 204));
        headerPanel.add(h1);
        JLabel h2 = new JLabel("Add Premium Member");
        h2.setBounds(550, 100, 300, 30);
        h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
        h2.setForeground(new Color(0, 204, 204));
        fr4.add(h2);

        // Plan Prices
        JLabel l1 = new JLabel("Regular Plans: Basic (6500/-), Standard (12500/-), Deluxe
(18500/-)");
        l1.setBounds(200, 140, 800, 30);
        l1.setFont(new Font("Segoe UI", Font.BOLD, 20));
        l1.setForeground(new Color(0, 51, 102));
        fr4.add(l1);

        JLabel l2 = new JLabel("Premium Plan: 50000/-");
        l2.setBounds(900, 140, 300, 30);
        l2.setFont(new Font("Segoe UI", Font.BOLD, 20));
        l2.setForeground(new Color(0, 51, 102));
        fr4.add(l2);

        // Input Fields - Row 1
        JLabel idLabel = new JLabel("ID:");
        idLabel.setBounds(100, 200, 100, 30);
        idLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
        fr4.add(idLabel);

        JTextField idField = new JTextField();
        idField.setBounds(200, 200, 200, 30);
        idField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        idField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
```

```
fr4.add(idField);

JLabel nameLabel = new JLabel("Name:");
nameLabel.setBounds(450, 200, 100, 30);
nameLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(nameLabel);

JTextField nameField = new JTextField();
nameField.setBounds(550, 200, 200, 30);
nameField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
nameField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr4.add(nameField);

JLabel locationLabel = new JLabel("Location:");
locationLabel.setBounds(800, 200, 150, 30);
locationLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(locationLabel);

JTextField locationField = new JTextField();
locationField.setBounds(950, 200, 200, 30);
locationField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
locationField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr4.add(locationField);

// Input Fields - Row 2
JLabel phoneLabel = new JLabel("Phone:");
phoneLabel.setBounds(100, 270, 100, 30);
phoneLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(phoneLabel);

JTextField phoneField = new JTextField();
```

```
phoneField.setBounds(200, 270, 200, 30);
phoneField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
phoneField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr4.add(phoneField);

JLabel emailLabel = new JLabel("Email:");
emailLabel.setBounds(450, 270, 100, 30);
emailLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(emailLabel);

JTextField emailField = new JTextField();
emailField.setBounds(550, 270, 200, 30);
emailField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
emailField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr4.add(emailField);

JLabel genderLabel = new JLabel("Gender:");
genderLabel.setBounds(800, 270, 150, 30);
genderLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(genderLabel);

JRadioButton male = new JRadioButton("Male");
male.setFont(new Font("Segoe UI", Font.PLAIN, 18));
male.setBackground(new Color(255, 255, 255));
male.setBounds(950, 270, 80, 30);
fr4.add(male);

JRadioButton female = new JRadioButton("Female");
female.setFont(new Font("Segoe UI", Font.PLAIN, 18));
female.setBackground(new Color(255, 255, 255));
female.setBounds(950, 300, 100, 30);
```

```
fr4.add(female);

JRadioButton other = new JRadioButton("Other");
other.setFont(new Font("Segoe UI", Font.PLAIN, 18));
other.setBackground(new Color(255, 255, 255));
other.setBounds(950, 330, 90, 30);
fr4.add(other);

ButtonGroup genderGroup = new ButtonGroup();
genderGroup.add(male);
genderGroup.add(female);
genderGroup.add(other);

// Input Fields - Row 3
JLabel dobLabel = new JLabel("DOB:");
dobLabel.setBounds(100, 380, 100, 30);
dobLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(dobLabel);

JComboBox<String> dobDay = new JComboBox<>(generateNumbers(1, 31));
dobDay.setBounds(200, 380, 60, 30);
dobDay.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr4.add(dobDay);

JComboBox<String> dobMonth = new JComboBox<>(getMonths());
dobMonth.setBounds(280, 380, 100, 30);
dobMonth.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr4.add(dobMonth);

JComboBox<String> dobYear = new JComboBox<>(generateYears(1950, 2025));
dobYear.setBounds(400, 380, 100, 30);
dobYear.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr4.add(dobYear);

JLabel mspLabel = new JLabel("Membership Start:");
mspLabel.setBounds(550, 380, 200, 30);
mspLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(mspLabel);

JComboBox<String> mspDay = new JComboBox<>(generateNumbers(1, 31));
mspDay.setBounds(750, 380, 60, 30);
mspDay.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr4.add(mspDay);

JComboBox<String> mspMonth = new JComboBox<>(getMonths());
mspMonth.setBounds(830, 380, 100, 30);
mspMonth.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr4.add(mspMonth);

JComboBox<String> mspYear = new JComboBox<>(generateYears(2023, 2025));
mspYear.setBounds(950, 380, 100, 30);
mspYear.setFont(new Font("Segoe UI", Font.PLAIN, 18));
fr4.add(mspYear);

// Input Fields - Row 4
JLabel trainerLabel = new JLabel("Personal Trainer:");
trainerLabel.setBounds(100, 450, 200, 30);
trainerLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr4.add(trainerLabel);

JTextField trainerField = new JTextField();
trainerField.setBounds(300, 450, 200, 30);
trainerField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
trainerField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr4.add(trainerField);

JLabel discountLabel = new JLabel("Discount:");
discountLabel.setBounds(550, 450, 150, 30);
discountLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));

JTextField discountField = new JTextField("0");
discountField.setBounds(700, 450, 200, 30);
discountField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
discountField.setBorder(new LineBorder(new Color(0, 204, 204), 1));

// Paid in Full Checkbox
JCheckBox paidInFullCheck = new JCheckBox("Paid in Full");
paidInFullCheck.setFont(new Font("Segoe UI", Font.PLAIN, 18));
paidInFullCheck.setBackground(new Color(255, 255, 255));
paidInFullCheck.setBounds(950, 450, 150, 30);
fr4.add(paidInFullCheck);

// Buttons
JButton clearButton = createStyledButton("Clear");
clearButton.setBounds(355, 650, 200, 40);
clearButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
clearButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        idField.setText("");
        nameField.setText("");
        locationField.setText("");
        phoneField.setText("");
```

```
            emailField.setText("");
            trainerField.setText("");
            discountField.setText("0");
            paidInFullCheck.setSelected(false);
            dobDay.setSelectedIndex(0);
            dobMonth.setSelectedIndex(0);
            dobYear.setSelectedIndex(0);
            mspDay.setSelectedIndex(0);
            mspMonth.setSelectedIndex(0);
            mspYear.setSelectedIndex(0);
            genderGroup.clearSelection();
        }
    });
    fr4.add(clearButton);

    JButton add = createStyledButton("Add Premium Member");
    add.setBounds(575, 650, 200, 40);
    add.setFont(new Font("Segoe UI", Font.BOLD, 16));
    add.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                String idText = idField.getText().trim();
                String name = nameField.getText().trim();
                String location = locationField.getText().trim();
                String phone = phoneField.getText().trim();
                String email = emailField.getText().trim();
                String trainer = trainerField.getText().trim();
                String discountText = discountField.getText().trim();
                boolean isFullPayment = paidInFullCheck.isSelected();
                if (idText.isEmpty()) {
                    throw new IllegalArgumentException("ID cannot be empty.");
                }
                int id;
                try {
                    id = Integer.parseInt(idText);
                    if (id <= 0) {
                        throw new IllegalArgumentException("ID must be a positive number.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("ID must be a numeric value.");
                }

                for (GymMember member : list) {
                    if (member.getId() == id) {
                        throw new IllegalArgumentException("Member ID already exists.");
                    }
                }

                if (name.isEmpty() || !name.matches("[a-zA-Z\\s]+")) {
                    throw new IllegalArgumentException("Name must contain letters only.");
                }
                if (phone.isEmpty() || !phone.matches("\\d{10}")) {
                    throw new IllegalArgumentException("Phone must be a 10-digit
    number.");
                }
                if (location.isEmpty()) {
                    throw new IllegalArgumentException("Location cannot be empty.");
                }
                if (email.isEmpty() || !email.matches("^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$")) {
                    throw new IllegalArgumentException("Invalid email format.");
                }
                if (trainer.isEmpty()) {
                    throw new IllegalArgumentException("Personal trainer name cannot be
    empty.");
                }
                double discountAmount;
                try {
                    discountAmount = discountText.isEmpty() ? 0 :
    Double.parseDouble(discountText);
                    if (discountAmount < 0 || discountAmount > 50000) {
                        throw new IllegalArgumentException("Discount must be between 0
    and 50000.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("Discount must be a valid
    number.");
                }

                String gender = male.isSelected() ? "Male" : female.isSelected() ? "Female"
    : other.isSelected() ? "Other" : "";
                if (gender.isEmpty()) {
                    throw new IllegalArgumentException("Please select a gender.");
                }

                if (dobDay.getSelectedItem() == null || dobMonth.getSelectedItem() == null
    || dobYear.getSelectedItem() == null) {
                    throw new IllegalArgumentException("Please select a valid date of
    birth.");
                }
                String dob = dobDay.getSelectedItem() + " " + dobMonth.getSelectedItem()
    + " " + dobYear.getSelectedItem();
```

```
                    if (mspDay.getSelectedItem() == null || mspMonth.getSelectedItem() == null
|| mspYear.getSelectedItem() == null) {
                        throw new IllegalArgumentException("Please select a valid membership
start date.");
                    }
                    String msp = mspDay.getSelectedItem() + " " +
mspMonth.getSelectedItem() + " " + mspYear.getSelectedItem();

                    PremiumMember pm = new PremiumMember(id, name, location, phone,
email, gender, dob, msp, trainer, discountAmount, isFullPayment);
                    list.add(pm);

                    double amountDue = 50000 - discountAmount;
                    JOptionPane.showMessageDialog(fr4, "Premium Member added
successfully! Amount Due: " + amountDue +
                        (isFullPayment ? " (Paid in Full)" : ""));
                    clearButton.doClick();
                } catch (IllegalArgumentException ex) {
                    JOptionPane.showMessageDialog(fr4, ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(fr4, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        fr4.add(add);

        JButton back = createStyledButton("Home");
        back.setBounds(795, 650, 200, 40);
        back.setFont(new Font("Segoe UI", Font.BOLD, 16));
        back.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                fr4.dispose();
                new GymGUI();
            }
        });
        fr4.add(back);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr4.add(mainPanel);

        fr4.setVisible(true);
    }

    public void activation() {
        fr5 = new JFrame("Activation & Deactivation");
        fr5.setSize(1400, 900);
        fr5.setLayout(null);
        fr5.setResizable(false);
        fr5.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr5.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
```

```
headerPanel.setBounds(0, 0, 1400, 80);
headerPanel.setBackground(new Color(0, 128, 128));
headerPanel.setLayout(null);
fr5.add(headerPanel);

JLabel h1 = new JLabel("GymFreak");
h1.setBounds(580, 15, 240, 50);
h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
h1.setForeground(new Color(0, 204, 204));
headerPanel.add(h1);

JLabel h2 = new JLabel("Activation & Deactivation");
h2.setBounds(550, 100, 300, 30);
h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
h2.setForeground(new Color(0, 204, 204));
fr5.add(h2);

// Input Fields
JLabel idLabel = new JLabel("Enter Member ID:");
idLabel.setBounds(50, 190, 150, 30);
idLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr5.add(idLabel);

JTextField idField = new JTextField();
idField.setBounds(200, 190, 200, 30);
idField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
idField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr5.add(idField);

// Buttons
JButton activateButton = createStyledButton("Activate Membership");
```

```
activateButton.setBounds(200, 240, 200, 40);
activateButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
activateButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String idText = idField.getText().trim();
            if (idText.isEmpty()) {
                throw new IllegalArgumentException("ID cannot be empty.");
            }
            int id;
            try {
                id = Integer.parseInt(idText);
                if (id <= 0) {
                    throw new IllegalArgumentException("ID must be a positive number.");
                }
            } catch (NumberFormatException ex) {
                throw new IllegalArgumentException("ID must be a numeric value.");
            }

            GymMember targetMember = null;
            for (GymMember member : list) {
                if (member.getId() == id) {
                    targetMember = member;
                    break;
                }
            }
            if (targetMember == null) {
                throw new IllegalArgumentException("Member ID not found.");
            }
```

```
                if (targetMember.getActiveStatus()) {
                    throw new IllegalStateException("Membership is already active.");
                }

                targetMember.activateMembership();
                JOptionPane.showMessageDialog(fr5, "Membership activated
successfully!");
                idField.setText("");
            } catch (IllegalArgumentException | IllegalStateException ex) {
                JOptionPane.showMessageDialog(fr5, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(fr5, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
    fr5.add(activateButton);

    JButton deactivateButton = createStyledButton("Deactivate Membership");
    deactivateButton.setBounds(450, 240, 200, 40);
    deactivateButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
    deactivateButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                String idText = idField.getText().trim();
                if (idText.isEmpty()) {
                    throw new IllegalArgumentException("ID cannot be empty.");
                }
                int id;
                try {
                    id = Integer.parseInt(idText);
                    if (id <= 0) {
                        throw new IllegalArgumentException("ID must be a positive number.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("ID must be a numeric value.");
                }

                GymMember targetMember = null;
                for (GymMember member : list) {
                    if (member.getId() == id) {
                        targetMember = member;
                        break;
                    }
                }
                if (targetMember == null) {
                    throw new IllegalArgumentException("Member ID not found.");
                }

                if (!targetMember.getActiveStatus()) {
                    throw new IllegalStateException("Membership is already deactivated.");
                }

                targetMember.deactivateMembership();
                JOptionPane.showMessageDialog(fr5, "Membership deactivated
successfully!");
                idField.setText("");
            } catch (IllegalArgumentException | IllegalStateException ex) {
                JOptionPane.showMessageDialog(fr5, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
```

```
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(fr5, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
        fr5.add(deactivateButton);

        JButton markAttendanceButton = createStyledButton("Mark Attendance");
        markAttendanceButton.setBounds(700, 240, 200, 40);
        markAttendanceButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
        markAttendanceButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    String idText = idField.getText().trim();
                    if (idText.isEmpty()) {
                        throw new IllegalArgumentException("ID cannot be empty.");
                    }
                    int id;
                    try {
                        id = Integer.parseInt(idText);
                        if (id <= 0) {
                            throw new IllegalArgumentException("ID must be a positive number.");
                        }
                    } catch (NumberFormatException ex) {
                        throw new IllegalArgumentException("ID must be a numeric value.");
                    }

                    GymMember targetMember = null;
                    for (GymMember member : list) {
```

```
                    if (member.getId() == id) {
                        targetMember = member;
                        break;
                    }
                }
                if (targetMember == null) {
                    throw new IllegalArgumentException("Member ID not found.");
                }

                targetMember.markAttendance();
                String memberType = (targetMember instanceof RegularMember) ?
"Regular" : "Premium";
                JOptionPane.showMessageDialog(fr5, "Attendance marked for " +
memberType + " Member! Loyalty Points: " + targetMember.getLoyaltyPoints());
                idField.setText("");
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(fr5, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            } catch (IllegalStateException ex) {
                JOptionPane.showMessageDialog(fr5, "Cannot mark attendance:
Membership is not activated.", "Error", JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(fr5, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
    fr5.add(markAttendanceButton);

    JButton back = createStyledButton("Home");
    back.setBounds(950, 240, 200, 40);
```

```
        back.setFont(new Font("Segoe UI", Font.BOLD, 16));
        back.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                fr5.dispose();
                new GymGUI();
            }
        });
        fr5.add(back);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr5.add(mainPanel);

        fr5.setVisible(true);
    }

    public void addRevertmember() {
        fr6 = new JFrame("Revert Member");
        fr6.setSize(1400, 900);
        fr6.setLayout(null);
        fr6.setResizable(false);
        fr6.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr6.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
```

```
headerPanel.setBounds(0, 0, 1400, 80);
headerPanel.setBackground(new Color(0, 128, 128));
headerPanel.setLayout(null);
fr6.add(headerPanel);

JLabel h1 = new JLabel("GymFreak");
h1.setBounds(580, 15, 240, 50);
h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
h1.setForeground(new Color(0, 204, 204));
headerPanel.add(h1);

JLabel h2 = new JLabel("Revert Member");
h2.setBounds(550, 100, 300, 30);
h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
h2.setForeground(new Color(0, 204, 204));
fr6.add(h2);

// Input Fields
JLabel idLabel = new JLabel("Enter Member ID:");
idLabel.setBounds(50, 190, 150, 30);
idLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr6.add(idLabel);

JTextField idField = new JTextField();
idField.setBounds(200, 190, 300, 30);
idField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
idField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr6.add(idField);

JLabel reasonLabel = new JLabel("Removal Reason:");
reasonLabel.setBounds(50, 240, 250, 30);
reasonLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr6.add(reasonLabel);

JTextArea reasonArea = new JTextArea();
reasonArea.setFont(new Font("Segoe UI", Font.PLAIN, 18));
reasonArea.setLineWrap(true);
reasonArea.setWrapStyleWord(true);
reasonArea.setBorder(new LineBorder(new Color(0, 204, 204), 1));

JScrollPane reasonScroll = new JScrollPane(reasonArea);
reasonScroll.setBounds(300, 240, 300, 60);
reasonScroll.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr6.add(reasonScroll);

JLabel reasonNote = new JLabel("Required for all members");
reasonNote.setBounds(300, 300, 300, 20);
reasonNote.setFont(new Font("Segoe UI", Font.ITALIC, 14));
reasonNote.setForeground(new Color(0, 51, 102));
fr6.add(reasonNote);

// Buttons
JButton revertButton = createStyledButton("Revert Member");
revertButton.setBounds(200, 330, 200, 40);
revertButton.setFont(new Font("Arial", Font.BOLD, 16));
revertButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String idText = idField.getText().trim();
            String removalReason = reasonArea.getText().trim();
            if (idText.isEmpty()) {
                throw new IllegalArgumentException("ID cannot be empty.");
            }
            int id;
            try {
                id = Integer.parseInt(idText);
                if (id <= 0) {
                    throw new IllegalArgumentException("ID must be a positive number.");
                }
            } catch (NumberFormatException ex) {
                throw new IllegalArgumentException("ID must be a numeric value.");
            }

            if (removalReason.isEmpty()) {
                throw new IllegalArgumentException("Removal reason is required for all
    members.");
            }

            GymMember targetMember = null;
            for (GymMember member : list) {
                if (member.getId() == id) {
                    targetMember = member;
                    break;
                }
            }
            if (targetMember == null) {
                throw new IllegalArgumentException("Member ID not found.");
            }

            if (targetMember instanceof RegularMember) {
```

```
                    ((RegularMember)
targetMember).revertRegularMember(removalReason);
                    JOptionPane.showMessageDialog(fr6, "Regular Member reverted
successfully! Plan set to basic, reason: " + removalReason,
                        "Success", JOptionPane.INFORMATION_MESSAGE);
                } else if (targetMember instanceof PremiumMember) {
                    ((PremiumMember)
targetMember).revertPremiumMember(removalReason);
                    JOptionPane.showMessageDialog(fr6, "Premium Member reverted
successfully! Trainer and payments cleared, reason: " + removalReason,
                        "Success", JOptionPane.INFORMATION_MESSAGE);
                }

                idField.setText("");
                reasonArea.setText("");
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(fr6, ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(fr6, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
    fr6.add(revertButton);

    JButton back = createStyledButton("Home");
    back.setBounds(450, 330, 200, 40);
    back.setFont(new Font("Segoe UI", Font.BOLD, 16));
    back.addActionListener(new ActionListener() {
        @Override
```

```
            public void actionPerformed(ActionEvent e) {
                fr6.dispose();
                new GymGUI();
            }
        });
        fr6.add(back);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr6.add(mainPanel);

        fr6.setVisible(true);
    }

    public void display() {
        fr9 = new JFrame("Display Members");
        fr9.setSize(1400, 900);
        fr9.setLayout(null);
        fr9.setResizable(false);
        fr9.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr9.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
        JPanel headerPanel = new JPanel();
        headerPanel.setBounds(0, 0, 1400, 80);
        headerPanel.setBackground(new Color(0, 128, 128));
        headerPanel.setLayout(null);
        fr9.add(headerPanel);

        JLabel h1 = new JLabel("GymFreak");
        h1.setBounds(580, 15, 240, 50);
        h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
        h1.setForeground(new Color(0, 204, 204));
        headerPanel.add(h1);

        JLabel h2 = new JLabel("Display Members");
        h2.setBounds(550, 100, 300, 30);
        h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
        h2.setForeground(new Color(0, 204, 204));
        fr9.add(h2);

        // Display Area
        JTextArea displayArea = new JTextArea();
        displayArea.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        displayArea.setEditable(false);
        displayArea.setBorder(new LineBorder(new Color(0, 204, 204), 1));

        try {
            StringBuilder sb = new StringBuilder();
            if (list == null || list.isEmpty()) {
                sb.append("No members to display.\n");
            } else {
                for (GymMember member : list) {
                    if (member == null) continue;
                    sb.append("ID: ").append(member.getId()).append("\n");
                    sb.append("Name: ").append(member.getName() != null ?
member.getName() : "N/A").append("\n");
```

```
                sb.append("Location: ").append(member.getLocation() != null ?
member.getLocation() : "N/A").append("\n");
                sb.append("Phone: ").append(member.getPhone() != null ?
member.getPhone() : "N/A").append("\n");
                sb.append("Email: ").append(member.getEmail() != null ?
member.getEmail() : "N/A").append("\n");
                sb.append("Gender: ").append(member.getGender() != null ?
member.getGender() : "N/A").append("\n");
                sb.append("DOB: ").append(member.getDob() != null ? member.getDob() :
"N/A").append("\n");
                sb.append("Membership Start: ").append(member.getMembershipStart() !=
null ? member.getMembershipStart() : "N/A").append("\n");
                sb.append("Active Status: ").append(member.getActiveStatus() ? "Active" :
"Inactive").append("\n");
                sb.append("Loyalty Points:
").append(member.getLoyaltyPoints()).append("\n");
                if (member instanceof RegularMember) {
                    RegularMember rm = (RegularMember) member;
                    sb.append("Type: Regular\n");
                    sb.append("Plan: ").append(rm.getPlan() != null ? rm.getPlan() :
"N/A").append("\n");
                    sb.append("Price: ").append(rm.getPrice()).append("\n");
                    sb.append("Referral: ").append(rm.getReferral() != null ? rm.getReferral()
: "N/A").append("\n");
                    sb.append("Full Payment: ").append(rm.getIsFullPayment() ? "Yes" :
"No").append("\n");
                    sb.append("Eligible for Upgrade: ").append(rm.getIsEligibleForUpgrade()
? "Yes" : "No").append("\n");
                    if (!rm.getRemovalReason().isEmpty()) {
                        sb.append("Removal Reason:
").append(rm.getRemovalReason()).append("\n");
```

```
                }
            } else if (member instanceof PremiumMember) {
                PremiumMember pm = (PremiumMember) member;
                sb.append("Type: Premium\n");
                sb.append("Personal Trainer: ").append(pm.getPersonalTrainer() != null
? pm.getPersonalTrainer() : "N/A").append("\n");
                sb.append("Discount Amount:
").append(pm.getDiscountAmount()).append("\n");
                sb.append("Paid Amount: ").append(pm.getPaidAmount()).append("\n");
                sb.append("Full Payment: ").append(pm.getIsFullPayment() ? "Yes" :
"No").append("\n");
                if (!pm.getRemovalReason().isEmpty()) {
                    sb.append("Removal Reason:
").append(pm.getRemovalReason()).append("\n");
                }
            }
            sb.append("----------------------------------------\n");
        }
    }
    displayArea.setText(sb.toString());
} catch (Exception ex) {
    displayArea.setText("Error displaying members: " + ex.getMessage());
}

JScrollPane scrollPane = new JScrollPane(displayArea);
scrollPane.setBounds(50, 190, 1250, 600);
scrollPane.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr9.add(scrollPane);

// Button
JButton back = createStyledButton("Home");
```

```
        back.setBounds(600, 800, 200, 40);
        back.setFont(new Font("Segoe UI", Font.BOLD, 16));
        back.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                fr9.dispose();
                new GymGUI();
            }
        });
        fr9.add(back);

        // Main Panel
        JPanel mainPanel = new JPanel();
        mainPanel.setBounds(50, 100, 1300, 750);
        mainPanel.setBackground(new Color(255, 255, 255));
        mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
        mainPanel.setLayout(null);
        fr9.add(mainPanel);

        fr9.setVisible(true);
    }

    public void upgradePlan() {
        fr7 = new JFrame("Upgrade Plan");
        fr7.setSize(1400, 900);
        fr7.setLayout(null);
        fr7.setResizable(false);
        fr7.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fr7.getContentPane().setBackground(new Color(255, 255, 255));

        // Header
```

```
JPanel headerPanel = new JPanel();
headerPanel.setBounds(0, 0, 1400, 80);
headerPanel.setBackground(new Color(0, 128, 128));
headerPanel.setLayout(null);
fr7.add(headerPanel);

JLabel h1 = new JLabel("GymFreak");
h1.setBounds(580, 15, 240, 50);
h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
h1.setForeground(new Color(0, 204, 204));
headerPanel.add(h1);

JLabel h2 = new JLabel("Upgrade Plan");
h2.setBounds(550, 100, 300, 30);
h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
h2.setForeground(new Color(0, 204, 204));
fr7.add(h2);

// Input Fields
JLabel idLabel = new JLabel("Enter Member ID:");
idLabel.setBounds(50, 190, 150, 30);
idLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
fr7.add(idLabel);

JTextField idField = new JTextField();
idField.setBounds(200, 190, 200, 30);
idField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
idField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
fr7.add(idField);

JLabel planLabel = new JLabel("New Plan:");
```

```
        planLabel.setBounds(450, 190, 150, 30);
        planLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
        fr7.add(planLabel);

        JComboBox<String> planField = new JComboBox<>(new String[]{"basic",
"standard", "deluxe"});
        planField.setBounds(600, 190, 200, 30);
        planField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
        fr7.add(planField);

        // Buttons
        JButton upgradeButton = createStyledButton("Upgrade Plan");
        upgradeButton.setBounds(200, 240, 200, 40);
        upgradeButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
        upgradeButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                try {
                    String idText = idField.getText().trim();
                    if (idText.isEmpty()) {
                        throw new IllegalArgumentException("ID cannot be empty.");
                    }
                    int id;
                    try {
                        id = Integer.parseInt(idText);
                        if (id <= 0) {
                            throw new IllegalArgumentException("ID must be a positive number.");
                        }
                    } catch (NumberFormatException ex) {
                        throw new IllegalArgumentException("ID must be a numeric value.");
                    }
                    String newPlan = (String) planField.getSelectedItem();
                    if (newPlan == null) {
                        throw new IllegalArgumentException("Please select a new plan.");
                    }

                    GymMember targetMember = null;
                    for (GymMember member : list) {
                        if (member.getId() == id) {
                            targetMember = member;
                            break;
                        }
                    }
                    if (targetMember == null) {
                        throw new IllegalArgumentException("Member ID not found.");
                    }

                    if (!(targetMember instanceof RegularMember)) {
                        throw new IllegalArgumentException("Only Regular Members can
upgrade plans.");
                    }

                    RegularMember rm = (RegularMember) targetMember;
                    rm.upgradePlan(newPlan);
                    double newPrice = rm.getPrice();
                    JOptionPane.showMessageDialog(fr7, "Plan upgraded to " + newPlan + "
successfully! New Price: " + newPrice);
                    idField.setText("");
                    planField.setSelectedIndex(0);
                } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(fr7, ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(fr7, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
    fr7.add(upgradeButton);

    JButton back = createStyledButton("Home");
    back.setBounds(450, 240, 200, 40);
    back.setFont(new Font("Segoe UI", Font.BOLD, 16));
    back.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            fr7.dispose();
            new GymGUI();
        }
    });
    fr7.add(back);

    // Main Panel
    JPanel mainPanel = new JPanel();
    mainPanel.setBounds(50, 100, 1300, 750);
    mainPanel.setBackground(new Color(255, 255, 255));
    mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
    mainPanel.setLayout(null);
    fr7.add(mainPanel);

    fr7.setVisible(true);
```

```
}

    public void processPayment() {
    fr8 = new JFrame("Process Payment");
    fr8.setSize(1400, 900);
    fr8.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    fr8.setResizable(false);
    fr8.setLayout(new BorderLayout());
    fr8.getContentPane().setBackground(new Color(255, 255, 255));

    // Header Panel
    JPanel headerPanel = new JPanel();
    headerPanel.setPreferredSize(new Dimension(1400, 80));
    headerPanel.setBackground(new Color(0, 128, 128));
    headerPanel.setLayout(null);

    JLabel h1 = new JLabel("GymFreak");
    h1.setBounds(580, 15, 240, 50);
    h1.setFont(new Font("Segoe UI", Font.BOLD, 36));
    h1.setForeground(new Color(0, 204, 204));
    headerPanel.add(h1);

    fr8.add(headerPanel, BorderLayout.NORTH);

    // Main Content Panel
    JPanel contentPanel = new JPanel();
    contentPanel.setBackground(new Color(255, 255, 255));
    contentPanel.setLayout(null);

    JLabel h2 = new JLabel("Process Payment");
    h2.setBounds(550, 20, 300, 30);
```

```
h2.setFont(new Font("Segoe UI", Font.BOLD, 24));
h2.setForeground(new Color(0, 204, 204));
contentPanel.add(h2);

// Input Fields
JLabel idLabel = new JLabel("Enter Member ID:");
idLabel.setBounds(50, 100, 150, 30);
idLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
contentPanel.add(idLabel);

JTextField idField = new JTextField();
idField.setBounds(200, 100, 200, 30);
idField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
idField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
contentPanel.add(idField);

JLabel amountLabel = new JLabel("Payment Amount:");
amountLabel.setBounds(450, 100, 150, 30);
amountLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
contentPanel.add(amountLabel);

JTextField amountField = new JTextField();
amountField.setBounds(600, 100, 200, 30);
amountField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
amountField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
contentPanel.add(amountField);

JLabel discountLabel = new JLabel("Discount Amount:");
discountLabel.setBounds(850, 100, 150, 30);
discountLabel.setFont(new Font("Segoe UI", Font.BOLD, 20));
contentPanel.add(discountLabel);
```

```
JTextField discountField = new JTextField("0.00");
discountField.setBounds(1000, 100, 200, 30);
discountField.setFont(new Font("Segoe UI", Font.PLAIN, 18));
discountField.setBorder(new LineBorder(new Color(0, 204, 204), 1));
discountField.setEditable(false);
contentPanel.add(discountField);

// Buttons
JButton calcDiscountButton = createStyledButton("Calculate Discount");
calcDiscountButton.setBounds(200, 150, 200, 40);
calcDiscountButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
calcDiscountButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        try {
            String idText = idField.getText().trim();
            if (idText.isEmpty()) {
                throw new IllegalArgumentException("ID cannot be empty.");
            }
            int id;
            try {
                id = Integer.parseInt(idText);
                if (id <= 0) {
                    throw new IllegalArgumentException("ID must be a positive number.");
                }
            } catch (NumberFormatException ex) {
                throw new IllegalArgumentException("ID must be a numeric value.");
            }

            GymMember targetMember = null;
```

```
                for (GymMember member : list) {
                    if (member.getId() == id) {
                        targetMember = member;
                        break;
                    }
                }
                if (targetMember == null) {
                    throw new IllegalArgumentException("Member ID not found.");
                }

                if (!(targetMember instanceof PremiumMember)) {
                    throw new IllegalArgumentException("Discounts are only applicable for
Premium Members.");
                }

                PremiumMember pm = (PremiumMember) targetMember;
                int loyaltyPoints = pm.getLoyaltyPoints();
                double discountPercentage = Math.min(loyaltyPoints / 10.0, 50.0); // 1%
per 10 points, max 50%
                double baseAmount = pm.getPremiumCharge(); // 50000
                double discountAmount = (discountPercentage / 100.0) * baseAmount;
                discountField.setText(String.format("%.2f", discountAmount));
                JOptionPane.showMessageDialog(fr8, "Discount calculated: " +
String.format("%.2f", discountAmount) +
                    " based on " + loyaltyPoints + " loyalty points.", "Discount",
JOptionPane.INFORMATION_MESSAGE);
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(fr8, ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(fr8, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
    contentPanel.add(calcDiscountButton);

    JButton payButton = createStyledButton("Process Payment");
    payButton.setBounds(450, 150, 200, 40);
    payButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
    payButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                String idText = idField.getText().trim();
                String amountText = amountField.getText().trim();
                String discountText = discountField.getText().trim();

                if (idText.isEmpty()) {
                    throw new IllegalArgumentException("ID cannot be empty.");
                }
                int id;
                try {
                    id = Integer.parseInt(idText);
                    if (id <= 0) {
                        throw new IllegalArgumentException("ID must be a positive number.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("ID must be a numeric value.");
                }
                if (amountText.isEmpty()) {
                    throw new IllegalArgumentException("Payment amount cannot be
empty.");
                }
                double amount;
                try {
                    amount = Double.parseDouble(amountText);
                    if (amount <= 0) {
                        throw new IllegalArgumentException("Payment amount must be
positive.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("Payment amount must be a valid
number.");
                }

                double discountAmount;
                try {
                    discountAmount = discountText.isEmpty() ? 0 :
Double.parseDouble(discountText);
                    if (discountAmount < 0) {
                        throw new IllegalArgumentException("Discount amount cannot be
negative.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("Discount amount must be a valid
number.");
                }

                GymMember targetMember = null;
                for (GymMember member : list) {
                    if (member.getId() == id) {
                        targetMember = member;
                        break;
                    }
                }
                if (targetMember == null) {
                    throw new IllegalArgumentException("Member ID not found.");
                }

                if (targetMember instanceof PremiumMember && discountAmount > 0) {
                    PremiumMember pm = (PremiumMember) targetMember;
                    pm.setDiscountAmount(discountAmount);
                }

                targetMember.makePayment(amount);
                String memberType = (targetMember instanceof RegularMember) ?
"Regular" : "Premium";
                JOptionPane.showMessageDialog(fr8, "Payment of " + amount + "
processed successfully for " + memberType +
                    " Member!" + (discountAmount > 0 ? " Discount applied: " +
discountAmount : ""));
                idField.setText("");
                amountField.setText("");
                discountField.setText("0.00");
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(fr8, ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);
            } catch (IllegalStateException ex) {
                JOptionPane.showMessageDialog(fr8, "Cannot process payment: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(fr8, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
    });
    contentPanel.add(payButton);

    JButton payDueButton = createStyledButton("Pay Due Amount");
    payDueButton.setBounds(700, 150, 200, 40);
    payDueButton.setFont(new Font("Segoe UI", Font.BOLD, 16));
    payDueButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            try {
                String idText = idField.getText().trim();
                String discountText = discountField.getText().trim();

                if (idText.isEmpty()) {
                    throw new IllegalArgumentException("ID cannot be empty.");
                }
                int id;
                try {
                    id = Integer.parseInt(idText);
                    if (id <= 0) {
                        throw new IllegalArgumentException("ID must be a positive number.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("ID must be a numeric value.");
                }

                double discountAmount;
                try {
                    discountAmount = discountText.isEmpty() ? 0 :
Double.parseDouble(discountText);
                    if (discountAmount < 0) {
                        throw new IllegalArgumentException("Discount amount cannot be
negative.");
                    }
                } catch (NumberFormatException ex) {
                    throw new IllegalArgumentException("Discount amount must be a valid
number.");
                }

                GymMember targetMember = null;
                for (GymMember member : list) {
                    if (member.getId() == id) {
                        targetMember = member;
                        break;
                    }
                }
                if (targetMember == null) {
                    throw new IllegalArgumentException("Member ID not found.");
                }

                double dueAmount = 0;
                String memberType = "";
                if (targetMember instanceof RegularMember) {
                    RegularMember rm = (RegularMember) targetMember;
                    memberType = "Regular";
                    if (!rm.getIsFullPayment()) {
                        dueAmount = rm.getPrice();
                    }
                } else if (targetMember instanceof PremiumMember) {
                    PremiumMember pm = (PremiumMember) targetMember;
                    memberType = "Premium";
                    dueAmount = pm.getPremiumCharge() - pm.getPaidAmount() -
discountAmount;
                    if (discountAmount > 0) {
                        pm.setDiscountAmount(discountAmount);
                    }
                }

                if (dueAmount <= 0) {
                    throw new IllegalStateException("No payment due for this member.");
                }

                targetMember.makePayment(dueAmount);
                JOptionPane.showMessageDialog(fr8, "Due amount of " +
String.format("%.2f", dueAmount) +
                    " paid successfully for " + memberType + " Member!" +
                    (discountAmount > 0 ? " Discount applied: " + discountAmount : ""));
                idField.setText("");
                amountField.setText("");
                discountField.setText("0.00");
            } catch (IllegalArgumentException ex) {
                JOptionPane.showMessageDialog(fr8, ex.getMessage(), "Input Error",
JOptionPane.ERROR_MESSAGE);
            } catch (IllegalStateException ex) {
                JOptionPane.showMessageDialog(fr8, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
            } catch (Exception ex) {
                JOptionPane.showMessageDialog(fr8, "Unexpected error occurred: " +
ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
```

```
            }
        }
    });
    contentPanel.add(payDueButton);

    JButton back = createStyledButton("Home");
    back.setBounds(950, 150, 200, 40);
    back.setFont(new Font("Segoe UI", Font.BOLD, 16));
    back.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            fr8.dispose();
            new GymGUI();
        }
    });
    contentPanel.add(back);

    // Main Panel
    JPanel mainPanel = new JPanel();
    mainPanel.setBounds(50, 70, 1300, 750);
    mainPanel.setBackground(new Color(255, 255, 255));
    mainPanel.setBorder(new LineBorder(new Color(0, 204, 204), 1));
    mainPanel.setLayout(null);
    contentPanel.add(mainPanel);

    fr8.add(contentPanel, BorderLayout.CENTER);
    fr8.setVisible(true);
}

private String[] generateNumbers(int start, int end) {
    String[] numbers = new String[end - start + 1];
```

```
        for (int i = 0; i < numbers.length; i++) {
            numbers[i] = String.valueOf(start + i);
        }
        return numbers;
    }

    private String[] generateYears(int start, int end) {
        String[] years = new String[end - start + 1];
        for (int i = 0; i < years.length; i++) {
            years[i] = String.valueOf(start + i);
        }
        return years;
    }

    private String[] getMonths() {
        return new String[]{"January", "February", "March", "April", "May", "June",
                "July", "August", "September", "October", "November", "December"};
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new GymGUI();
            }
        });
    }
}
```

<u>Bibliography</u>