UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

4COSC010C.3 – PROGRAMMING PRINCIPLES II

Coursework: 02

Simple Gym Management System

Student ID: *2019595*

UOW ID: *w1790135*

Course: *Software Engineering*

Module Leader: Mr. Guhanathan Poravi

Submission Date: 03.08.2020

Youtube URL: https://youtu.be/dppIU75hlcI

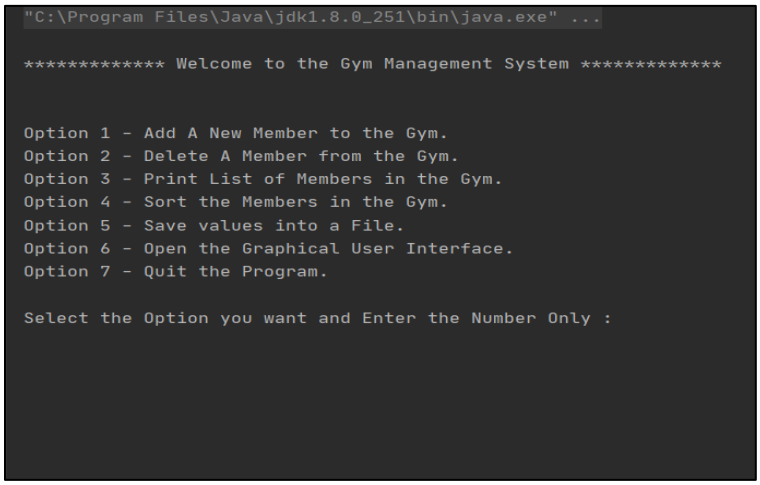
https://www.youtube.com/watch?v=dppIU75hlcI (Alternative Link)

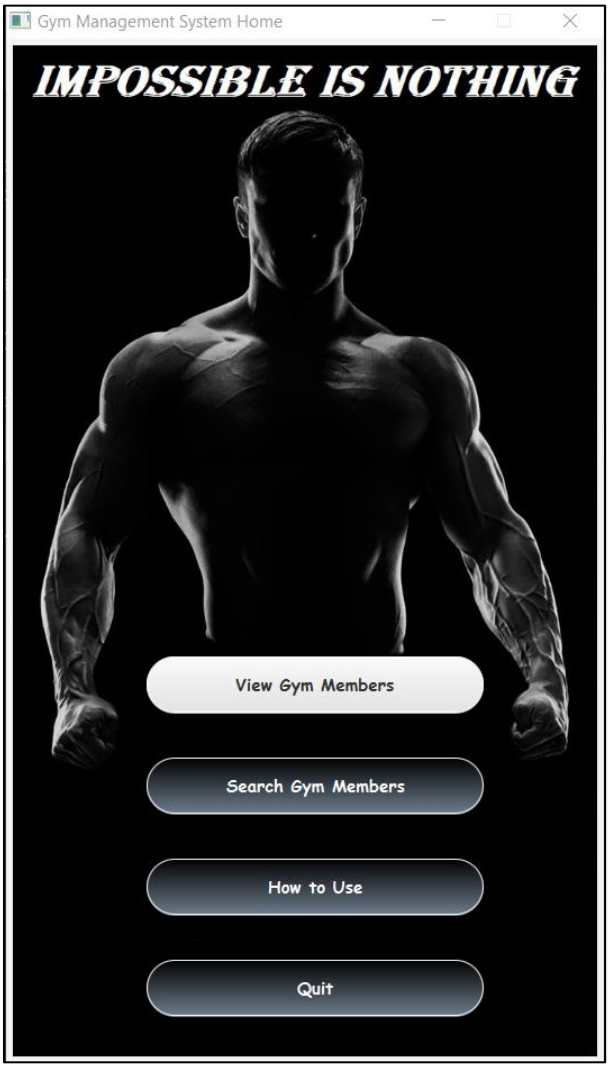
Contents

INTRODUCTION TO THE APPLICATION

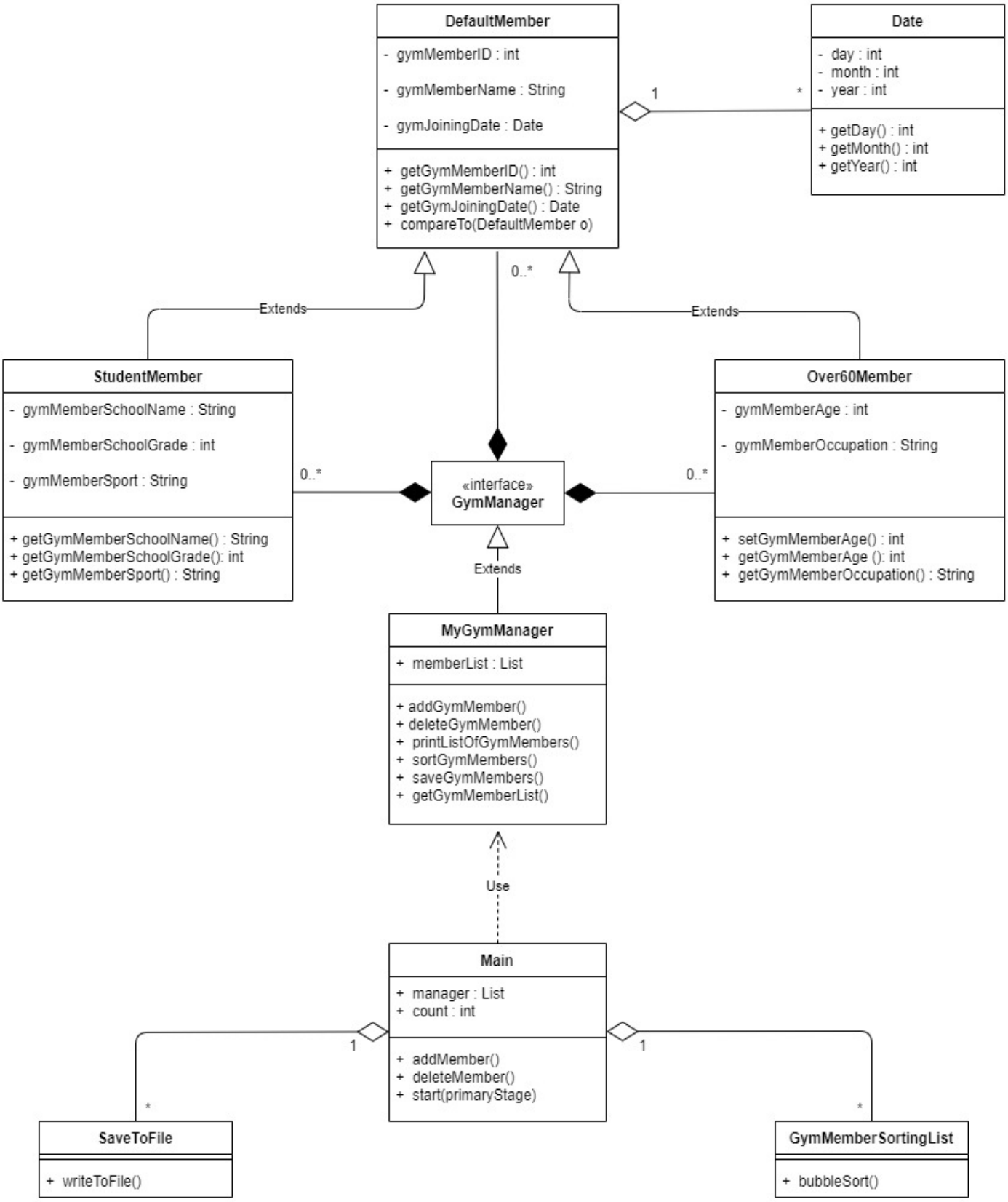
In this coursework, students have to prepare a Simple Gym Management System to use in real life. It is mainly made to test the student's ability to code using Object Oriented Programming Principles (OOP) and the student's ability to successfully build a Graphical User Interface (GUI).

The Gym Management System mainly focuses on six requiremnts. They are; Adding a new member, deleting a member, printing the list of members, sorting the members, wrinting the member details into a file and Displsying the required information in the GUI. All these requirements have to be coded using OOP principles. The fist five requirments are fulfilled in a console system created by the user and the last requirement requires the GUI to view the list of members in the Gym Management system and to search the members in it.

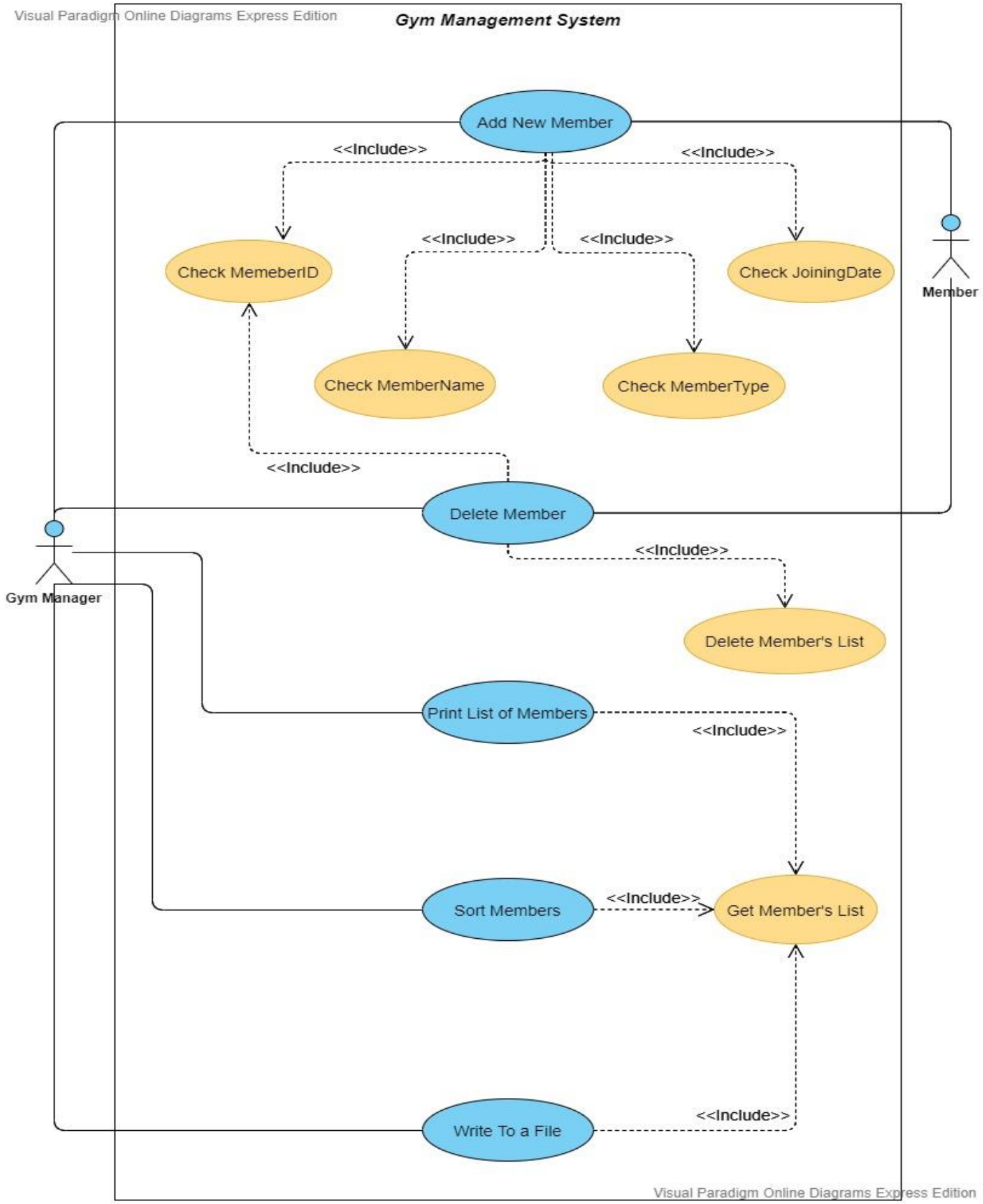




CLASS DIAGRAM FOR THE APPLICATION



USE CASE DIAGRAM FOR THE SYSTEM



REQUIREMENTS FOR THE APPLICATION

R1 – ADD NEW MEMBER

The program prompts the console GUI and from there the first option is to add a new member to the Gym Management System. The user will be asked to enter the values for a default member in the first set of inputs and then the sytem shall prompt the user to choose the type of member. If he/she chooses to add a student member or an over sixty member the system shall ask for another set of inputs depending on the user's choice. Once a new meber is added the system shall give an acknowledgement if the member was added successfully and the counter will reduce by one since the system can only have 100 members. If all the membership slots are full the system shall prompt the user saying that there are no free membership slots available.

Main.java

```
private static void addMember(){
    Scanner userInput = new Scanner(System.in).useDelimiter("\n");
    if (count<100){
        System.out.print("Enter a New Membership ID :\t");
        int membershipNo = userInput.nextInt();
        System.out.print("Enter New Member's Name :\t");
        String membershipName = userInput.next();
        System.out.print("Enter New Member's Weight :\t");
        double memberWeight = userInput.nextDouble();
        System.out.print("Enter New Member's Contact No :\t");
        int contactNo = userInput.nextInt();
        System.out.print("Enter New Member's Joining date :\t");
        String membershipJoinDate = userInput.next();
        System.out.println("\n************* Chooses the Membership Type
*************\n");
        System.out.println("Option 1 - Default Member." +
                "\nOption 2 - Student Member." +
                "\nOption 3 - Over Sixty Members." +
                "\nOption 4 - Exit Program");
        System.out.print("\nSelect the Option you want and Enter the Number Only : ");
        int userIn1 = userInput.nextInt();

        DefaultMember member = null;

        switch (userIn1){
            case 1:
                member = new DefaultMember(membershipNo,membershipName,
membershipJoinDate);
                break;
            case 2:
                System.out.print("Enter the school name : ");
                String schoolName = userInput.next();
                System.out.print("Enter the Grade of the Student :");
                int schoolGrade = userInput.nextInt();
                System.out.print("Enter the Sport played by Student :");
                String sports=userInput.next();
                member = new StudentMember(membershipNo,membershipName,
```

```
membershipJoinDate,schoolName,schoolGrade,sports);
                break;
            case 3:
                System.out.print("Enter the Age of the Member :");
                int age = userInput.nextInt();
                System.out.print("Enter the occupation of the Member :");
                String occupation = userInput.next();
                member = new Over60Members(membershipNo,membershipName,
membershipJoinDate,age,occupation);
                break;
            case 4:
                System.out.println("\n************* Thank You for using the Gym
Management System !! *************");
                System.exit(0);
            default:
                System.out.println("Please check your inputs");
        }
        manager.addGymMember(member);
        count++;
    }else {
        System.out.println("There are No Memberships Available");
    }
}
```

MyGymManager.java

```
@Override
public void addGymMember(DefaultMember member) {
    System.out.println("Add A New Member to the Gym.");
    if(memberList.size() < 100){
        memberList.add(member);
    }else{
        System.out.println("There are no New Memberships Available");
    }
    System.out.println("Number of Members Enrolled in the Gym Currently : " +
memberList.size());
    System.out.println("Number of New Memberships Available : " + (100 -
memberList.size()));
}
```

Output

```
Select the Option you want and Enter the Number Only : 1
Enter a New Membership ID : 1001
Enter New Member's Name :   Thevindu
Enter New Member's Weight : 80
Enter New Member's Contact No : 0761490232
Enter New Member's Joining date :   26-02-2020

************* Chooses the Membership Type *************

Option 1 - Default Member.
Option 2 - Student Member.
Option 3 - Over Sixty Members.
Option 4 - Exit Program

Select the Option you want and Enter the Number Only : 1
Add A New Member to the Gym.
Number of Members Enrolled in the Gym Currently : 1
Number of New Memberships Available : 99
```

R2 – DELETE MEMBER

The second option in the console GUI is to delete a existing member. In order to delete a member, the user have to enter the membership ID of the member and then the system shall find the user and delete their entry. When the member is removed from the system it will prompt a message to the user saying that the member with the relevant ID wad removed. Once the user is deleted from the system the counter of the members will increase by one because the system allows the user to input 100 members into the system.

Main.java

```
private static void deleteMember(){
    Scanner userInput = new Scanner(System.in);
    System.out.print("Enter the membership ID of the member you want deleted : ");
    int deleteGymMember = userInput.nextInt();
    boolean res = manager.deleteGymMember(deleteGymMember);
    if(res){
        count--;
    }
}
```

MyGymManager.java

```
@Override
public boolean deleteGymMember(int membershipNo) {
    boolean flag = false;
    for(DefaultMember member:memberList){
        if (Objects.equals(member.getGymMemberID(), membershipNo)){
            flag = true;
            memberList.remove(member);
            System.out.println("Member with the Membership ID "+membershipNo+" has
been removed.");
            System.out.println("Number of Members Enrolled in the Gym Currently : " +
memberList.size());
            System.out.println("Number of New Memberships Available : " + (100 -
memberList.size()));
            if (member instanceof StudentMember){
                System.out.println("Membership type is : Student Member");
            }else if (member instanceof Over60Members){
                System.out.println("Membership type is : Over60Member");
            }else {
                System.out.println("Membership type is : Default Member");
            }
            break;
        }
    }
    if (!flag){
        System.out.println("There is no such membership ID. Please Check Again!!");
    }
    return flag;
}
```

Output

```
************* Welcome to the Gym Management System *************


Option 1 - Add A New Member to the Gym.
Option 2 - Delete A Member from the Gym.
Option 3 - Print List of Members in the Gym.
Option 4 - Sort the Members in the Gym.
Option 5 - Save values into a File.
Option 6 - Open the Graphical User Interface.
Option 7 - Quit the Program.

Select the Option you want and Enter the Number Only : 2
Enter the membership ID of the member you want deleted : 1001
Member with the Membership ID 1001 has been removed.
Number of Members Enrolled in the Gym Currently : 0
Number of New Memberships Available : 100
Membership type is : Default Member
```

R3 – PRINT THE LIST OF MEMBERS

The third option in the console GUI is printing the members inserted to the Gym Management System. All the members inputted to the system will be accessed through the list and printed on the console GUI. First all the common factors for all three members will be printed and then the system shall check if the member is a default member, student member or an over sixty member. After checking the member type if the member is a student or an over sixty member then the system will print the relevant statements. If the user wants to print the members and if there are no members in the Gym Management system then the system shall prompt a message saying that there are no members in the system.

MyGymManager.java

```
@Override
public void printListOfGymMembers() {
    for(DefaultMember member:memberList){
        System.out.print("( { Membership No : " + member.getGymMemberID()+" }, ");
        System.out.print("{ Name is : "+member.getGymMemberName()+" }, ");
        System.out.print("{ Membership Start Date is : "+member.getGymJoiningDate()+"
}, ");
        if (member instanceof StudentMember){
            System.out.print("{ Student member's School is : "+((StudentMember)
member).getGymMemberSchoolName()+" }, ");
            System.out.print("{ Student member's Grade is : "+((StudentMember)
member).getGymMemberSchoolGrade()+" }, ");
            System.out.println("{ Student member's Sport is : "+((StudentMember)
member).getGymMemberSport()+" }, ");
            System.out.println("{ Membership type is : Student Member } )");
        }else if (member instanceof Over60Members){
            System.out.print("{ Member's Age is : "+((Over60Members)
member).getGymMemberAge()+" }, ");
            System.out.print("{ Member's Occupation : "+((Over60Members)
member).getGymMemberOccupation()+" }, ");
            System.out.println("{ Membership type is : Over60Member } )");
        }else {
            System.out.println("{ Membership type is : Default Member } )");
```

```
            }
        }
        if (memberList.size()==0){
            System.out.println("There are no members in the Gym.");
        }
}
```

Output

```
************* Welcome to the Gym Management System *************


Option 1 - Add A New Member to the Gym.
Option 2 - Delete A Member from the Gym.
Option 3 - Print List of Members in the Gym.
Option 4 - Sort the Members in the Gym.
Option 5 - Save values into a File.
Option 6 - Open the Graphical User Interface.
Option 7 - Quit the Program.

Select the Option you want and Enter the Number Only : 3
( { Membership No : 1001 }, { Name is : Thevindu }, { Membership Start Date is : 26-02-2020 }, { Membership type is : Default Member } )
( { Membership No : 1002 }, { Name is : Hamza }, { Membership Start Date is : 21-03-2020 }, { Student member's School is : STC }, { Student member's Grade is : 14 }, { Student
  member's Sport is : TT },
{ Membership type is : Student Member } )
( { Membership No : 1003 }, { Name is : Pasindu }, { Membership Start Date is : 13-01-2020 }, { Member's Age is : 65 }, { Member's Occupation : SE }, { Membership type is :
 Over60Member } )
```

```
************* Welcome to the Gym Management System *************


Option 1 - Add A New Member to the Gym.
Option 2 - Delete A Member from the Gym.
Option 3 - Print List of Members in the Gym.
Option 4 - Sort the Members in the Gym.
Option 5 - Save values into a File.
Option 6 - Open the Graphical User Interface.
Option 7 - Quit the Program.

Select the Option you want and Enter the Number Only : 3
There are no members in the Gym.
```

R4 – SORT MEMBERS ACCORDING TO NAME

The fourth option given in the console GUI is to sort the members in the system according to their names. When the user selects this option the names of the members will be accessed from the list and then it will be sorted using buble sort and after sorting it the sorted lists will be printed in the terminal.

GymMemberSortingList.java

```
package sample;

public class GymMemberSortingList {
    public static void bubbleSort(DefaultMember[] arr, boolean asc){
        for(int a =0 ; a < arr.length - 1;  a++ ){
            for(int b =0 ; b< arr.length - (a+1); b++){
                if(asc){
                    if
(arr[b].getGymMemberName().compareTo(arr[b+1].getGymMemberName()) > 0){
                        DefaultMember temp = arr[b];
                        arr[b] = arr[b+1];
                        arr[b+1] = temp;
                    }
                }
            }
        }
    }
}
```

MyGymManager.java

```
@Override
public void sortGymMembers() {
    DefaultMember[] arr = memberList.toArray(new DefaultMember[] {});
    GymMemberSortingList.bubbleSort(arr, true);
    for(DefaultMember member: arr){
        System.out.print("( { Membership No : " + member.getGymMemberID()+" }, ");
        System.out.print("{ Name is : "+member.getGymMemberName()+" }, ");
        System.out.print("{ Membership Start Date is : "+member.getGymJoiningDate()+"
}, ");
        if (member instanceof StudentMember){
            System.out.print("{ Student member's School is : "+((StudentMember)
member).getGymMemberSchoolName()+" }, ");
            System.out.print("{ Student member's Grade is : "+((StudentMember)
member).getGymMemberSchoolGrade()+" }, ");
            System.out.println("{ Student member's Sport is : "+((StudentMember)
member).getGymMemberSport()+" }, ");
            System.out.println("{ Membership type is : Student Member } )");
        }else if (member instanceof Over60Members){
            System.out.print("{ Member's Age is : "+((Over60Members)
member).getGymMemberAge()+" }, ");
            System.out.print("{ Member's Occupation : "+((Over60Members)
member).getGymMemberOccupation()+" }, ");
            System.out.println("{ Membership type is : Over60Member } )");
        }else {
            System.out.println("{ Membership type is : Default Member } )");
}}}
```

Output

```
************* Welcome to the Gym Management System *************


Option 1 - Add A New Member to the Gym.
Option 2 - Delete A Member from the Gym.
Option 3 - Print List of Members in the Gym.
Option 4 - Sort the Members in the Gym.
Option 5 - Save values into a File.
Option 6 - Open the Graphical User Interface.
Option 7 - Quit the Program.

Select the Option you want and Enter the Number Only : 4
( { Membership No : 1002 }, { Name is : Hamza }, { Membership Start Date is : 21-03-2020 }, { Student member's School is : STC }, { Student member's Grade is : 14 }, { Student
  member's Sport is : TT },
{ Membership type is : Student Member } )
( { Membership No : 1003 }, { Name is : Pasindu }, { Membership Start Date is : 13-01-2020 }, { Member's Age is : 65 }, { Member's Occupation : SE }, { Membership type is :
 Over60Member } )
( { Membership No : 1001 }, { Name is : Thevindu }, { Membership Start Date is : 26-02-2020 }, { Membership type is : Default Member } )
```

R5 – WRITE/SAVE IN A FILE

The fifth option in the console GUI gives the user the ability to write all members details into a text file. Even when the program is terminated and re-executed again the files will remain in the text file due to the append function used in the system.

Main.java

```
else if (user == 5) {
    List<DefaultMember> memList = manager.getGymMemberList();
    SaveToFile.writeToFile(memList, "Members.txt");
}
```

SaveToFile.java

```
public class SaveToFile{
    public static void writeToFile(List<DefaultMember> membersList, String fileName) {
            try {
                FileWriter myWriter = new FileWriter(fileName,true);
                for (DefaultMember member : membersList) {
                    myWriter.write("( { Membership No : " + member.getGymMemberID() +
" }, " + "{ Name is : " + member.getGymMemberName() + " }, " + "{ Membership Start
Date is : " + member.getGymJoiningDate() +  " }, ");
                    if (member instanceof StudentMember) {
                        myWriter.write("{ Student member's School is : " +
((StudentMember) member).getGymMemberSchoolName() + " }, " + "{ Student member's Grade
is : "+ ((StudentMember) member).getGymMemberSchoolGrade() +" }, " + "{ Student
member's Sport is : "+((StudentMember) member).getGymMemberSport()+" }, " + "{
Membership type is : Student Member } )\n");
                    } else if (member instanceof Over60Members) {
                        myWriter.write("{ Member's Age is : " + ((Over60Members)
member).getGymMemberAge() + " }, " + "{ Member's Occupation : " + ((Over60Members)
```

```
member).getGymMemberOccupation()+" }, " + "{ Membership type is : Over60Member }
)\n");
                    }else{
                        myWriter.write("{ Membership type is : Default Member } )\n");
                    }
                }
                myWriter.close();
                System.out.println("Gym Members have been successfully added to the
Members.txt File.\n");
            } catch (IOException exception) {
                System.out.println("Error!!");
                exception.printStackTrace();
            }
    }
}
```

Output

```
************* Welcome to the Gym Management System *************


 Option 1 - Add A New Member to the Gym.
 Option 2 - Delete A Member from the Gym.
 Option 3 - Print List of Members in the Gym.
 Option 4 - Sort the Members in the Gym.
 Option 5 - Save values into a File.
 Option 6 - Open the Graphical User Interface.
 Option 7 - Quit the Program.

 Select the Option you want and Enter the Number Only : 5
 Gym Members have been successfully added to the Members.txt File.
```

Members.txt - Notepad

File Edit Format View Help

```
( { Membership No : 1001 }, { Name is : Thevindu }, { Membership Start Date is : 26-02-2020 }, { Membership type is : Default Member } )
( { Membership No : 1002 }, { Name is : Hamza }, { Membership Start Date is : 21-03-2020 }, { Student member's School is : STC }, { Student member's Grade is : 14 }, { Student member's Spor
( { Membership No : 1003 }, { Name is : Pasindu }, { Membership Start Date is : 13-01-2020 }, { Member's Age is : 65 }, { Member's Occupation : SE }, { Membership type is : Over60Member } )
```

Ln 1, Col 1 | 100% | Unix (LF) | UTF-8

R6 – OPEN GRAPHICAL USER INTERFACE

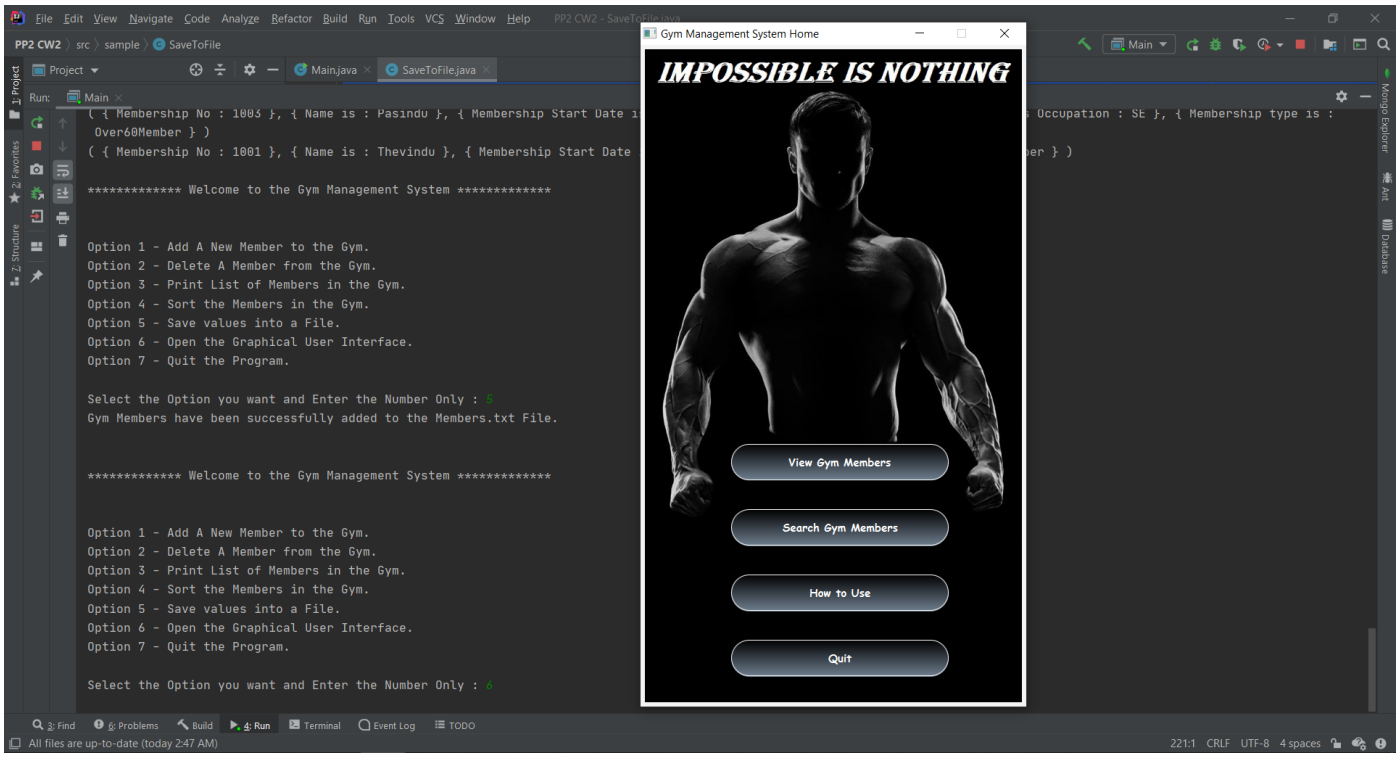
In the sixth option on the console GUI the user has the ability to open their Graphical User Interface created using Javafx. When the GUI is called the MainGui.java class will be loaded in to the Main.java and it will be executed with launch(args).

Main.java

```
public void start(Stage primaryStage){
    GuiMain.home(primaryStage);
}
```

```
else if (user == 6) {
    launch(args);
}
```

Output



CODES

Main.java

```
package sample;

import javafx.application.Application;
import javafx.stage.Stage;
import java.util.*;

public class Main extends Application {

    public void start(Stage primaryStage){
        GuiMain.home(primaryStage);
    }

    private final static MyGymManager manager = new MyGymManager();
    private static int count = 0;

    public static void main(String[] args) {
        try{
            try (Scanner userInput = new Scanner(System.in)) {

                int user;

                while (true) {
                    System.out.println("\n************* Welcome to the Gym Management
System *************" + "\n\n");
                    System.out.println("Option 1 - Add A New Member to the Gym.");
                    System.out.println("Option 2 - Delete A Member from the Gym.");
                    System.out.println("Option 3 - Print List of Members in the
Gym.");
                    System.out.println("Option 4 - Sort the Members in the Gym.");
                    System.out.println("Option 5 - Save values into a File.");
                    System.out.println("Option 6 - Open the Graphical User
Interface.");
                    System.out.println("Option 7 - Quit the Program.\n");
                    System.out.print("Select the Option you want and Enter the Number
Only : ");
                    user = userInput.nextInt();

                    if (user == 1) {
                        addMember();
                    } else if (user == 2) {
                        deleteMember();
                    } else if (user == 3) {
                        manager.printListOfGymMembers();
                    } else if (user == 4) {
                        manager.sortGymMembers();
                    } else if (user == 5) {
                        List<DefaultMember> memList = manager.getGymMemberList();
                        SaveToFile.writeToFile(memList, "Members.txt");
                    } else if (user == 6) {
                        launch(args);
                    } else if (user == 7) {
                        System.out.println("\n************* Thank You for using the
Gym Management System !! *************");
                        System.exit(0);
```

```
                    }
                }
            }
        }catch (InputMismatchException exception){
            System.out.println("\nInvalid Input!! Please Enter an Integer.");
        }
    }




    private static void addMember(){
        Scanner userInput = new Scanner(System.in).useDelimiter("\n");
        if (count<100){
            System.out.print("Enter a New Membership ID :\t");
            int membershipNo = userInput.nextInt();
            System.out.print("Enter New Member's Name :\t");
            String membershipName = userInput.next();
            System.out.print("Enter New Member's Weight :\t");
            double memberWeight = userInput.nextDouble();
            System.out.print("Enter New Member's Contact No :\t");
            int contactNo = userInput.nextInt();
            System.out.print("Enter New Member's Joining date :\t");
            String membershipJoinDate = userInput.next();
            System.out.println("\n************* Chooses the Membership Type
*************\n");
            System.out.println("Option 1 - Default Member." +
                    "\nOption 2 - Student Member." +
                    "\nOption 3 - Over Sixty Members." +
                    "\nOption 4 - Exit Program");
            System.out.print("\nSelect the Option you want and Enter the Number Only :
");
            int userIn1 = userInput.nextInt();

            DefaultMember member = null;

            switch (userIn1){
                case 1:
                    member = new DefaultMember(membershipNo,membershipName,
membershipJoinDate);
                    break;
                case 2:
                    System.out.print("Enter the school name : ");
                    String schoolName = userInput.next();
                    System.out.print("Enter the Grade of the Student :");
                    int schoolGrade = userInput.nextInt();
                    System.out.print("Enter the Sport played by Student :");
                    String sports=userInput.next();
                    member = new StudentMember(membershipNo,membershipName,
membershipJoinDate,schoolName,schoolGrade,sports);
                    break;
                case 3:
                    System.out.print("Enter the Age of the Member :");
                    int age = userInput.nextInt();
                    System.out.print("Enter the occupation of the Member :");
                    String occupation = userInput.next();
                    member = new Over60Members(membershipNo,membershipName,
membershipJoinDate,age,occupation);
                    break;
```

```
                case 4:
                    System.out.println("\n************* Thank You for using the Gym
Management System !! *************");
                    System.exit(0);
                default:
                    System.out.println("Please check your inputs");
            }
            manager.addGymMember(member);
            count++;
        }else {
            System.out.println("There are No Memberships Available");
        }
    }
    private static void deleteMember(){
        Scanner userInput = new Scanner(System.in);
        System.out.print("Enter the membership ID of the member you want deleted : ");
        int deleteGymMember = userInput.nextInt();
        boolean res = manager.deleteGymMember(deleteGymMember);
        if(res){
            count--;
        }
    }

}
```

GymManager.java

```
package sample;
import java.util.List;

public interface GymManager {
    void addGymMember(DefaultMember member);
    boolean deleteGymMember(int membershipNo);
    void printListOfGymMembers();
    void sortGymMembers();
    void saveGymMembers();
    List<DefaultMember> getGymMemberList();
}
```

MyGymManager.java

```
package sample;

import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

public class MyGymManager implements GymManager{

    public static List<DefaultMember> memberList = new ArrayList<>();

    @Override
    public void addGymMember(DefaultMember member) {
        System.out.println("Add A New Member to the Gym.");
        if(memberList.size() < 100){
            memberList.add(member);
        }else{
            System.out.println("There are no New Memberships Available");
        }
        System.out.println("Number of Members Enrolled in the Gym Currently : " +
memberList.size());
        System.out.println("Number of New Memberships Available : " + (100 -
memberList.size()));
    }

    @Override
    public boolean deleteGymMember(int membershipNo) {
        boolean flag = false;
        for(DefaultMember member:memberList){
            if (Objects.equals(member.getGymMemberID(), membershipNo)){
                flag = true;
                memberList.remove(member);
                System.out.println("Member with the Membership ID "+membershipNo+" has
been removed.");
                System.out.println("Number of Members Enrolled in the Gym Currently :
" + memberList.size());
                System.out.println("Number of New Memberships Available : " + (100 -
memberList.size()));
                if (member instanceof StudentMember){
                    System.out.println("Membership type is : Student Member");
                }else if (member instanceof Over60Members){
                    System.out.println("Membership type is : Over60Member");
                }else {
                    System.out.println("Membership type is : Default Member");
                }
                break;
            }
        }
        if (!flag){
            System.out.println("There is no such membership ID. Please Check
Again!!");
        }
        return flag;
    }

    @Override
    public void printListOfGymMembers() {
```

```
        for(DefaultMember member:memberList){
            System.out.print("( { Membership No : " + member.getGymMemberID()+" }, ");
            System.out.print("{ Name is : "+member.getGymMemberName()+" }, ");
            System.out.print("{ Membership Start Date is :
"+member.getGymJoiningDate()+" }, ");
            if (member instanceof StudentMember){
                System.out.print("{ Student member's School is : "+((StudentMember)
member).getGymMemberSchoolName()+" }, ");
                System.out.print("{ Student member's Grade is : "+((StudentMember)
member).getGymMemberSchoolGrade()+" }, ");
                System.out.println("{ Student member's Sport is : "+((StudentMember)
member).getGymMemberSport()+" }, ");
                System.out.println("{ Membership type is : Student Member } )");
            }else if (member instanceof Over60Members){
                System.out.print("{ Member's Age is : "+((Over60Members)
member).getGymMemberAge()+" }, ");
                System.out.print("{ Member's Occupation : "+((Over60Members)
member).getGymMemberOccupation()+" }, ");
                System.out.println("{ Membership type is : Over60Member } )");
            }else {
                System.out.println("{ Membership type is : Default Member } )");
            }
        }
        if (memberList.size()==0){
            System.out.println("There are no members in the Gym.");
        }
    }

    @Override
    public void sortGymMembers() {
        DefaultMember[] arr = memberList.toArray(new DefaultMember[] {});
        GymMemberSortingList.bubbleSort(arr, true);
        for(DefaultMember member: arr){
            System.out.print("( { Membership No : " + member.getGymMemberID()+" }, ");
            System.out.print("{ Name is : "+member.getGymMemberName()+" }, ");
            System.out.print("{ Membership Start Date is :
"+member.getGymJoiningDate()+" }, ");
            if (member instanceof StudentMember){
                System.out.print("{ Student member's School is : "+((StudentMember)
member).getGymMemberSchoolName()+" }, ");
                System.out.print("{ Student member's Grade is : "+((StudentMember)
member).getGymMemberSchoolGrade()+" }, ");
                System.out.println("{ Student member's Sport is : "+((StudentMember)
member).getGymMemberSport()+" }, ");
                System.out.println("{ Membership type is : Student Member } )");
            }else if (member instanceof Over60Members){
                System.out.print("{ Member's Age is : "+((Over60Members)
member).getGymMemberAge()+" }, ");
                System.out.print("{ Member's Occupation : "+((Over60Members)
member).getGymMemberOccupation()+" }, ");
                System.out.println("{ Membership type is : Over60Member } )");
            }else {
                System.out.println("{ Membership type is : Default Member } )");
            }
        }
    }

    @Override
    public void saveGymMembers() { }
```

```
    @Override
    public List<DefaultMember> getGymMemberList() {
        return this.memberList;
    }

}
```

DefaultMember.java

```
package sample;

public class DefaultMember implements Comparable<DefaultMember>{
    private final int gymMemberID;
    private final String gymMemberName;
    private final String gymJoiningDate;

    public DefaultMember(int gymMemberID, String gymMemberName, String gymJoiningDate)
{
        super();
        this.gymMemberID = gymMemberID;
        this.gymMemberName = gymMemberName;
        this.gymJoiningDate = gymJoiningDate;
    }

    public int getGymMemberID() {
        return gymMemberID;
    }

    public String getGymMemberName() {
        return gymMemberName;
    }

    public String getGymJoiningDate() {
        return gymJoiningDate;
    }

    @Override
    public int compareTo(DefaultMember o) {
        return this.gymMemberName.compareTo(o.gymMemberName);
    }
}
```

StudentMember.java

```
package sample;

class StudentMember extends DefaultMember{
    private final String gymMemberSchoolName;
    private final int gymMemberSchoolGrade;
    private final String gymMemberSport;

    public StudentMember(int gymMemberID, String gymMemberName, String gymJoiningDate,
String gymMemberSchoolName, int gymMemberSchoolGrade, String gymMemberSport) {
        super(gymMemberID, gymMemberName, gymJoiningDate);
        this.gymMemberSchoolName = gymMemberSchoolName;
        this.gymMemberSchoolGrade = gymMemberSchoolGrade;
        this.gymMemberSport = gymMemberSport;
    }

    public String getGymMemberSchoolName() {
        return gymMemberSchoolName;
    }

    public int getGymMemberSchoolGrade() {
        return gymMemberSchoolGrade;
    }

    public String getGymMemberSport() {
        return gymMemberSport;
    }

}
```

Over60Members.java

```
package sample;

class Over60Members extends DefaultMember{
    private int gymMemberAge;
    private String gymMemberOccupation;

    public Over60Members(int gymMemberID, String gymMemberName, String gymJoiningDate,
int gymMemberAge, String gymMemberOccupation) {
        super(gymMemberID, gymMemberName, gymJoiningDate);
        setGymMemberAge(gymMemberAge);
        setGymMemberOccupation(gymMemberOccupation);
        this.gymMemberOccupation = gymMemberOccupation;
    }

    public int getGymMemberAge() {
        return gymMemberAge;
    }

    public void setGymMemberAge(int gymMemberAge) {
        if (gymMemberAge >= 60){
            this.gymMemberAge = gymMemberAge;
        }else{
            throw new IllegalArgumentException("Not a Valid Age for this
```

```
Membership.");
        }

    }

    public String getGymMemberOccupation() {
        return gymMemberOccupation;
    }

    public void setGymMemberOccupation(String gymMemberOccupation) {
        this.gymMemberOccupation = gymMemberOccupation;
    }
}
```

GymMemberSortingList.java

```
package sample;

public class GymMemberSortingList {
    public static void bubbleSort(DefaultMember[] arr, boolean asc){
        for(int a =0 ; a < arr.length - 1;  a++ ){
            for(int b =0 ; b< arr.length - (a+1); b++){
                if(asc){
                    if
(arr[b].getGymMemberName().compareTo(arr[b+1].getGymMemberName()) > 0){
                        DefaultMember temp = arr[b];
                        arr[b] = arr[b+1];
                        arr[b+1] = temp;
                    }
                }
            }
        }
    }
}
```

SaveToFile.java

```
package sample;

import java.io.FileWriter;
import java.io.IOException;
import java.util.List;

public class SaveToFile{
    public static void writeToFile(List<DefaultMember> membersList, String fileName) {
            try {
                FileWriter myWriter = new FileWriter(fileName,true);
                for (DefaultMember member : membersList) {
                    myWriter.write("( { Membership No : " + member.getGymMemberID() +
" }, " + "{ Name is : " + member.getGymMemberName() + " }, " + "{ Membership Start
Date is : " + member.getGymJoiningDate() +  " }, ");
                    if (member instanceof StudentMember) {
                        myWriter.write("{ Student member's School is : " +
((StudentMember) member).getGymMemberSchoolName() + " }, " + "{ Student member's Grade
```

```
is : "+ ((StudentMember) member).getGymMemberSchoolGrade() +" }, " + "{ Student
member's Sport is : "+((StudentMember) member).getGymMemberSport()+" }, " + "{
Membership type is : Student Member } )\n");
                    } else if (member instanceof Over60Members) {
                        myWriter.write("{ Member's Age is : " + ((Over60Members)
member).getGymMemberAge() + " }, " + "{ Member's Occupation : " + ((Over60Members)
member).getGymMemberOccupation()+" }, " + "{ Membership type is : Over60Member }
)\n");
                    }else{
                        myWriter.write("{ Membership type is : Default Member } )\n");
                    }
                }
                myWriter.close();
                System.out.println("Gym Members have been successfully added to the
Members.txt File.\n");
            } catch (IOException exception) {
                System.out.println("Error!!");
                exception.printStackTrace();
            }
    }
}
```

GuiMain.java

```
package sample;

import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;

public class GuiMain {
    public static void home(Stage primaryStage){
        Pane root = new Pane();

        Button start = new Button("View Gym Members");
        start.setId("Button");
        start.setLayoutX(125);
        start.setLayoutY(550);
        root.getChildren().add(start);
        start.setOnAction(event -> {
            Stage stage = (Stage) start.getScene().getWindow();
            stage.close();
            GuiGymViewMembers.viewMembers(primaryStage, MyGymManager.memberList);
        });

        Button start2 = new Button("Search Gym Members");
        start2.setId("Button");
        start2.setLayoutX(125);
        start2.setLayoutY(640);
        root.getChildren().add(start2);
        start2.setOnAction(event -> {
            Stage stage = (Stage) start.getScene().getWindow();
            stage.close();
```

```
            GuiGymSearchMembers.searchMembers(primaryStage);
        });

        Button help = new Button("How to Use");
        help.setId("Button");
        help.setLayoutX(125);
        help.setLayoutY(730);
        root.getChildren().add(help);
        help.setOnAction(event -> {
            Stage stage = (Stage) help.getScene().getWindow();
            stage.close();
            GuiGymHelp.GymHowToUse(primaryStage);
        });

        Button quit = new Button("Quit");
        quit.setId("Button");
        quit.setLayoutX(125);
        quit.setLayoutY(820);
        root.getChildren().add(quit);
        quit.setOnAction(event -> {
            Stage stage = (Stage) quit.getScene().getWindow();
            stage.close();
        });

        primaryStage.setTitle("Gym Management System Home");
        root.setId("pane");
        Scene scene = new Scene(root, 520, 900);

scene.getStylesheets().add(Main.class.getResource("style.css").toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setResizable(false);
    }
}
```

GuiGymViewMembers.java

```
package sample;

import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.*;
import javafx.scene.text.Text;
import javafx.stage.Stage;

import java.util.List;

public class GuiGymViewMembers {
```

```
    public static void viewMembers(Stage primaryStage, List<DefaultMember> list){
        Pane root = new Pane();

        Button homeBtn = new Button("Home");
        homeBtn.setId("Button1");
        homeBtn.setOnAction(event -> { Stage stage = (Stage)
homeBtn.getScene().getWindow(); stage.close(); GuiMain.home(primaryStage);});

        Button ViewMembersBtn = new Button("View Members");
        ViewMembersBtn.setId("ButtonSelected");

        Button SearchBtn = new Button("Search");
        SearchBtn.setId("Button1");
        SearchBtn.setOnAction(event -> { Stage stage = (Stage)
SearchBtn.getScene().getWindow(); stage.close();
GuiGymSearchMembers.searchMembers(primaryStage);});

        Button HelpBtn = new Button("How to Use");
        HelpBtn.setId("Button1");
        HelpBtn.setOnAction(event -> { Stage stage = (Stage)
HelpBtn.getScene().getWindow(); stage.close();
GuiGymHelp.GymHowToUse(primaryStage);});

        Button ExitBtn = new Button("Exit");
        ExitBtn.setId("Button1");
        ExitBtn.setOnAction(event -> { Stage stage = (Stage)
ExitBtn.getScene().getWindow(); stage.close();});

        VBox menuButtons = VBoxBuilder.create()
                .spacing(75.0) //In case you are using HBoxBuilder
                .padding(new Insets(50, 10, 5, 25))
                .children(homeBtn,ViewMembersBtn,SearchBtn,HelpBtn,ExitBtn)
                .build();
        menuButtons.setId("Menubar");
        menuButtons.setMinSize(200,950);
        menuButtons.setTranslateX(0);
        menuButtons.setTranslateY(0);

        Label mainHeading = new Label("List of Members in the Gym");
        mainHeading.setId("MainHeading1");

        Label lblNoOfMembers = new Label("Number of Memberships Purchased :");
        lblNoOfMembers.setId("MainLbl");
        Label lblNoOfAvailableMembers = new Label("Number of Available Memberships
:");
        lblNoOfAvailableMembers.setId("MainLbl");
        Label lblMaxNoOfMembers = new Label("Maximum Number of Memberships :");
        lblMaxNoOfMembers.setId("MainLbl");
        Text txtNoOfMembers = new Text("0");
        txtNoOfMembers.setId("MainLbl");
        Text txtNoOfAvailableMembers = new Text("0");
        txtNoOfAvailableMembers.setId("MainLbl");
        Text textMaxNoOfMembers = new Text("100");
        textMaxNoOfMembers.setId("MainLbl");

        GridPane gridPaneMembers = new GridPane();
        gridPaneMembers.setId("Result");
        gridPaneMembers.setHgap(20);
```

```
        gridPaneMembers.setVgap(40);

        gridPaneMembers.add(lblNoOfMembers,0,0);
        gridPaneMembers.add(txtNoOfMembers,1,0);
        gridPaneMembers.add(lblNoOfAvailableMembers,4,0);
        gridPaneMembers.add(txtNoOfAvailableMembers,5,0);
        gridPaneMembers.add(lblMaxNoOfMembers,0,1);
        gridPaneMembers.add(textMaxNoOfMembers,1,1);

        TableView gymMembersList = new TableView();

        TableColumn<Integer, DefaultMember> column1 = new TableColumn<>("ID");
        column1.setCellValueFactory(new PropertyValueFactory<>("ID"));
        column1.setMinWidth(150);

        TableColumn<String, DefaultMember> column2 = new TableColumn<>("Full Name");
        column2.setCellValueFactory(new PropertyValueFactory<>("fullName"));
        column2.setMinWidth(150);

        TableColumn<Integer, DefaultMember> column3 = new TableColumn<>("Age");
        column3.setCellValueFactory(new PropertyValueFactory<>("age"));
        column3.setMinWidth(150);

        TableColumn<String, DefaultMember> column4 = new TableColumn<>("School Name");
        column4.setCellValueFactory(new PropertyValueFactory<>("schoolName"));
        column4.setMinWidth(150);

        TableColumn<Integer, DefaultMember> column5 = new TableColumn<>("Contact No");
        column5.setCellValueFactory(new PropertyValueFactory<>("contactNo"));
        column5.setMinWidth(150);

        for(DefaultMember defaultMember : list) {

gymMembersList.getItems().addAll(defaultMember.getGymMemberID(),defaultMember.getGymMe
mberName(),defaultMember.getGymMemberID(),defaultMember.getGymMemberName(),defaultMemb
er.getGymMemberID());
        }

        gymMembersList.getColumns().addAll(column1,column2,column3,column4,column5);
        gymMembersList.setMaxHeight(700);
        gymMembersList.setId("List");


        VBox content = VBoxBuilder.create()
                .spacing(50.0) //In case you are using HBoxBuilder
                .padding(new Insets(50, 25, 5, 25))
                .children(mainHeading,gridPaneMembers,gymMembersList)
                .build();
        content.setId("Content");
        content.setMinSize(1020,815);
        content.setTranslateX(250);
        content.setTranslateY(50);

        HBox bgImage = new HBox();
        bgImage.setId("BgImage");
        bgImage.setMinSize(1150,950);
        bgImage.setTranslateX(200);
        bgImage.setTranslateY(0);
```

```
        primaryStage.setTitle("Gym Management System View Members");
        Scene scene = new Scene(new Group(menuButtons,bgImage,content,root), 1300,
900);

scene.getStylesheets().add(Main.class.getResource("style.css").toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setResizable(false);
    }
}
```

GuiGymSearchMembers.java

```
package sample;

import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.*;
import javafx.stage.Stage;

public class GuiGymSearchMembers {

    public static void searchMembers(Stage primaryStage){
        Pane root = new Pane();

        Button homeBtn = new Button("Home");
        homeBtn.setId("Button1");
        homeBtn.setOnAction(event -> { Stage stage = (Stage)
homeBtn.getScene().getWindow(); stage.close(); GuiMain.home(primaryStage);});

        Button ViewMembersBtn = new Button("View Members");
        ViewMembersBtn.setId("Button1");
        ViewMembersBtn.setOnAction(event -> { Stage stage = (Stage)
ViewMembersBtn.getScene().getWindow(); stage.close();
GuiGymViewMembers.viewMembers(primaryStage, MyGymManager.memberList);});

        Button SearchBtn = new Button("Search");
        SearchBtn.setId("ButtonSelected");

        Button HelpBtn = new Button("How to Use");
        HelpBtn.setId("Button1");
        HelpBtn.setOnAction(event -> { Stage stage = (Stage)
HelpBtn.getScene().getWindow(); stage.close();
GuiGymHelp.GymHowToUse(primaryStage);});

        Button ExitBtn = new Button("Exit");
        ExitBtn.setId("Button1");
        ExitBtn.setOnAction(event -> { Stage stage = (Stage)
ExitBtn.getScene().getWindow(); stage.close();});

        VBox menuButtons = VBoxBuilder.create()
                .spacing(75.0) //In case you are using HBoxBuilder
                .padding(new Insets(50, 10, 5, 25))
```

```
            .children(homeBtn,ViewMembersBtn,SearchBtn,HelpBtn,ExitBtn)
            .build();
    menuButtons.setId("Menubar");
    menuButtons.setMinSize(200,950);
    menuButtons.setTranslateX(0);
    menuButtons.setTranslateY(0);

    Label mainHeading = new Label("Search for Members in the Gym");
    mainHeading.setId("MainHeading2");

    Label searchOptionsLbl = new Label("Select Search Option : ");
    searchOptionsLbl.setId("MainLbl");
    ChoiceBox<String> searchOptions = new ChoiceBox<>();
    searchOptions.getItems().addAll("ID","Name","Age","SchoolName");
    searchOptions.setValue("ID");

    HBox leftBar = new HBox();
    leftBar.setSpacing(20);
    leftBar.getChildren().addAll(searchOptionsLbl,searchOptions);

    TextField searchValue = new TextField();
    searchValue.setId("SearchBox");
    Button searchBtn = new Button("Search");
    searchBtn.setId("SearchBtn");

    HBox rightBar = new HBox();
    rightBar.setSpacing(20);
    rightBar.getChildren().addAll(searchValue,searchBtn);

    HBox MainBar = new HBox();
    MainBar.setSpacing(200);
    MainBar.getChildren().addAll(leftBar,rightBar);


    TableView gymMembersList = new TableView();
    TableColumn column1 = new TableColumn<>("ID");
    column1.setCellValueFactory(new PropertyValueFactory<>("ID"));
    column1.setMinWidth(150);
    TableColumn column2 = new TableColumn<>("Full Name");
    column2.setCellValueFactory(new PropertyValueFactory<>("fullName"));
    column2.setMinWidth(150);
    TableColumn column3 = new TableColumn<>("Age");
    column3.setCellValueFactory(new PropertyValueFactory<>("age"));
    column3.setMinWidth(150);
    TableColumn column4 = new TableColumn<>("School Name");
    column4.setCellValueFactory(new PropertyValueFactory<>("schoolName"));
    column4.setMinWidth(150);
    TableColumn column5 = new TableColumn<>("Contact No");
    column5.setCellValueFactory(new PropertyValueFactory<>("contactNo"));
    column5.setMinWidth(150);

    gymMembersList.getColumns().addAll(column1,column2,column3,column4,column5);
    gymMembersList.setId("List");


    VBox content = VBoxBuilder.create()
            .spacing(70.0) //In case you are using HBoxBuilder
            .padding(new Insets(50, 25, 20, 25))
            .children(mainHeading,MainBar,gymMembersList)
```

```
                .build();
        content.setId("Content");
        content.setMinSize(1020,815);
        content.setTranslateX(250);
        content.setTranslateY(50);

        HBox bgImage = new HBox();
        bgImage.setId("BgImage");
        bgImage.setMinSize(1150,950);
        bgImage.setTranslateX(200);
        bgImage.setTranslateY(0);


        primaryStage.setTitle("Gym Management System View Members");
        Scene scene = new Scene(new Group(menuButtons,bgImage,content,root), 1300,
900);

scene.getStylesheets().add(Main.class.getResource("style.css").toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setResizable(false);
    }
}
```

GuiGymHelp

```
package sample;

import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.*;
import javafx.stage.Stage;

public class GuiGymHelp {
    public static void GymHowToUse(Stage primaryStage){
        Pane root = new Pane();

        Button homeBtn = new Button("Home");
        homeBtn.setId("Button1");
        homeBtn.setOnAction(event -> { Stage stage = (Stage)
homeBtn.getScene().getWindow(); stage.close(); GuiMain.home(primaryStage);});

        Button ViewMembersBtn = new Button("View Members");
        ViewMembersBtn.setId("Button1");
        ViewMembersBtn.setOnAction(event -> { Stage stage = (Stage)
ViewMembersBtn.getScene().getWindow(); stage.close();
GuiGymViewMembers.viewMembers(primaryStage,MyGymManager.memberList);});

        Button SearchBtn = new Button("Search");
        SearchBtn.setId("Button1");
        SearchBtn.setOnAction(event -> { Stage stage = (Stage)
SearchBtn.getScene().getWindow(); stage.close();
GuiGymSearchMembers.searchMembers(primaryStage);});
```

```
        Button HelpBtn = new Button("How to Use");
        HelpBtn.setId("ButtonSelected");

        Button ExitBtn = new Button("Exit");
        ExitBtn.setId("Button1");
        ExitBtn.setOnAction(event -> { Stage stage = (Stage)
ExitBtn.getScene().getWindow(); stage.close();});

        VBox menuButtons = VBoxBuilder.create()
                .spacing(75.0) //In case you are using HBoxBuilder
                .padding(new Insets(50, 10, 5, 25))
                .children(homeBtn,ViewMembersBtn,SearchBtn,HelpBtn,ExitBtn)
                .build();
        menuButtons.setId("Menubar");
        menuButtons.setMinSize(200,950);
        menuButtons.setTranslateX(0);
        menuButtons.setTranslateY(0);

        Label mainHeading = new Label("How to use the Gym Management System");
        mainHeading.setId("MainHeading3");





        HBox content = HBoxBuilder.create()
                .spacing(75.0) //In case you are using HBoxBuilder
                .padding(new Insets(50, 10, 5, 25))
                .children(mainHeading)
                .build();
        content.setId("Content");
        content.setMinSize(1020,815);
        content.setTranslateX(250);
        content.setTranslateY(50);

        HBox bgImage = new HBox();
        bgImage.setId("BgImage");
        bgImage.setMinSize(1150,950);
        bgImage.setTranslateX(200);
        bgImage.setTranslateY(0);


        primaryStage.setTitle("Gym Management System View Members");
        Scene scene = new Scene(new Group(menuButtons,bgImage,content,root), 1300,
900);

scene.getStylesheets().add(Main.class.getResource("style.css").toExternalForm());
        primaryStage.setScene(scene);
        primaryStage.show();
        primaryStage.setResizable(false);
    }
}
```

style.css

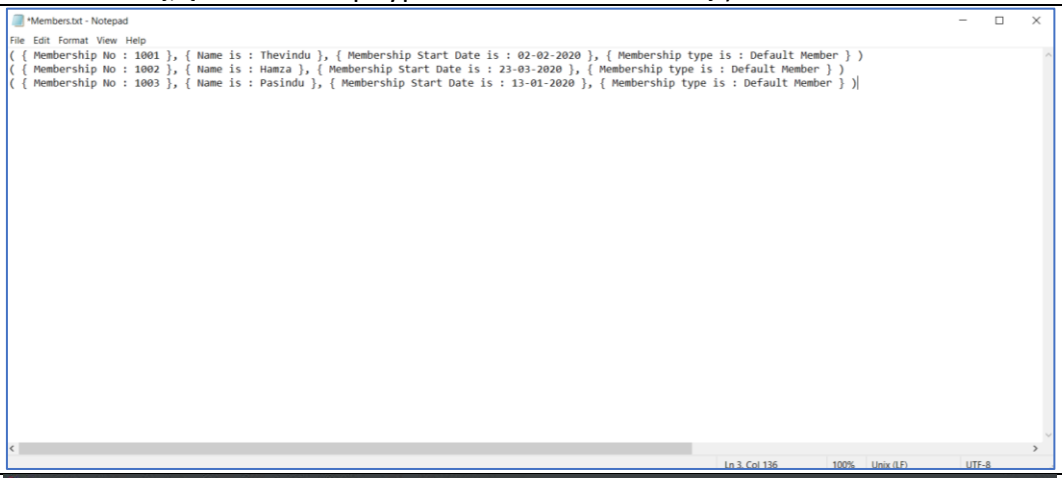
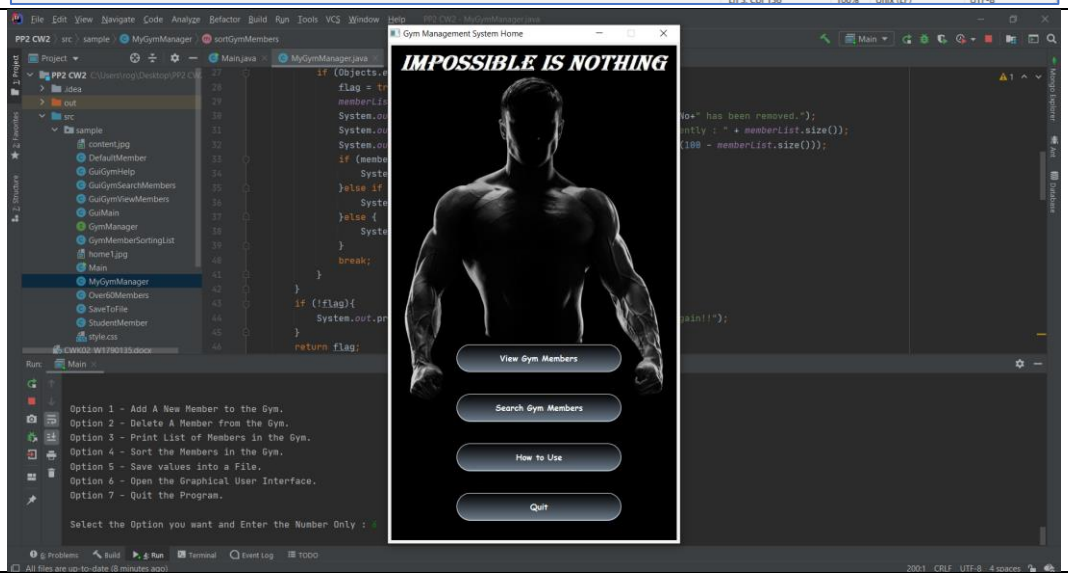
```
#pane{
    -fx-background-image: url('home1.jpg');
    -fx-background-repeat: stretch;
    -fx-background-size: 520 900;
    -fx-background-position: center center;
}
#Button{
    -fx-background-color: linear-gradient(black, #708090);
    -fx-font-family: 'Comic Sans MS';
    -fx-font-weight: bolder;
    -fx-text-fill: white;
    -fx-border-color: white;
    -fx-min-width: 300px;
    -fx-min-height: 50px;
    -fx-border-radius: 100px;
    -fx-background-radius: 100px;
}
#Menubar{
    -fx-background-color: linear-gradient(black,blue);
}
#Button1{
    -fx-background-color: linear-gradient(black,slategrey);
    -fx-font-family: 'Comic Sans MS';
    -fx-font-weight: bolder;
    -fx-text-fill: white;
    -fx-border-color: white;
    -fx-min-width: 150px;
    -fx-min-height: 100px;
    -fx-border-radius: 30px;
    -fx-background-radius: 30px;
}
#ButtonSelected{
    -fx-background-color: linear-gradient(yellow,yellowgreen,green);
    -fx-font-family: 'Comic Sans MS';
    -fx-font-weight: bolder;
    -fx-text-fill: white;
    -fx-border-color: white;
    -fx-min-width: 150px;
    -fx-min-height: 100px;
    -fx-border-radius: 30px;
    -fx-background-radius: 30px;
}
#BgImage{
    -fx-background-image: url('content.jpg');
    -fx-background-repeat: stretch;
    -fx-background-size: 1150 950;
    -fx-background-position: center center;
    -fx-opacity: 0.9;
}
#Content{
    -fx-background-color: linear-gradient(lightslategrey,dimgrey);
    -fx-opacity: 0.9;
    -fx-background-radius: 50px;
}
#Button1:hover,#Button:hover{
    -fx-background-color: -fx-shadow-highlight-color, -fx-outer-border, -fx-inner-
```

```
border, -fx-body-color;
    -fx-background-insets: 0 0 -1 0, 0, 1, 2;
    -fx-padding: 0.333333em 0.666667em 0.333333em 0.666667em;
    -fx-text-fill: -fx-text-base-color;
    -fx-alignment: CENTER;
    -fx-content-display: LEFT;
}
#MainHeading1,#MainHeading2,#MainHeading3{
    -fx-text-fill: linear-gradient(orange,orangered,darkred) ;
    -fx-font-family: 'Comic Sans MS';
    -fx-font-weight: bolder;
    -fx-font-size: 40px;
}
#MainHeading1{-fx-label-padding: 0px 0px 0px 200px;}
#MainHeading2{-fx-label-padding: 0px 0px 0px 180px;}
#MainHeading3{-fx-label-padding: 0px 0px 0px 100px;}
#SearchBox{
    -fx-min-width: 350px;
    -fx-background-radius: 30px;
}
#SearchBtn{
    -fx-background-color: linear-gradient(yellow,yellowgreen,green);
    -fx-background-radius: 30px;
}
#MainLbl{
    -fx-font-size: 20px;
    -fx-font-weight: bolder;
}
```

TEST PLAN

TestID	***Test Case Name***	***Description***	***Expected Result***
TC01	Input Validation	Entering and wrong input to the sytem and checking if the error is caught by the System.	Invalid Input!! Please enter an integer
TC02	Add Member	Adding a member into the system.	Add A New Member to the Gym. Number of Members Enrolled in the Gym Currently: 1 Number of New Memberships Available: 99
TC03	Delete Member	Deleting a member from the system.	Member with the Membership ID 1001 has been removed. Number of Members Enrolled in the Gym Currently: 0 Number of New Memberships Available: 100 Membership type is: Default Member
TC04	Print Members	Printing the Members in the Terminal.	Printing the List of members.
TC05	SortMembers	Sorting the Members in the List.	Printing the Sorted list of members
TC06	Write/Save to a File	Writing the values into a file.	Writing the members details into a text file.
TC07	Open GUI	Open the GUI.	Open the GUI.
TC08	Exit	Exit the program Succesfully.	Exit the Program.

TestID	***Actual Output***	***Pass/Fail***
TC01	Invalid Input!! Please Enter an Integer.	***Pass***
TC02	Add A New Member to the Gym. Number of Members Enrolled in the Gym Currently: 1 Number of New Memberships Available: 99	***Pass***
TC03	Member with the Membership ID 1001 has been removed. Number of Members Enrolled in the Gym Currently: 0 Number of New Memberships Available: 100 Membership type is: Default Member	***Pass***
TC04	({ Membership No : 1001 }, { Name is : Thevindu }, { Membership Start Date is : 02-02-2020 }, { Membership type is : Default Member })	***Pass***

	({ Membership No : 1002 }, { Name is : Hamza }, { Membership Start Date is : 23-03-2020 }, { Membership type is : Default Member }) ({ Membership No : 1003 }, { Name is : Pasindu }, { Membership Start Date is : 13-01-2020 }, { Membership type is : Default Member })	
TC05	({ Membership No : 1002 }, { Name is : Hamza }, { Membership Start Date is : 23-03-2020 }, { Membership type is : Default Member }) ({ Membership No : 1003 }, { Name is : Pasindu }, { Membership Start Date is : 13-01-2020 }, { Membership type is : Default Member }) ({ Membership No : 1001 }, { Name is : Thevindu }, { Membership Start Date is : 02-02-2020 }, { Membership type is : Default Member })	***Pass***
TC06		***Pass***
TC07		***Pass***
TC08	********** Thank You for using the Gym Management System!! ********** Process finished with exit code 0	***Pass***

CONCLUSION

In attending to Gym Management system coursework, I have learnt Java Object Oriented Principles and this coursework extended my knowledge on Javafx. I came up with minor problems when connecting the Graphical User Interface with Java. But with the immense support of my lecturer and the tutors I was able to manage them. Overall, this coursework extended my knowledge and I think it is a success.

REFERENCES

Mainly the code was refered from the lectures and the session done for the PP2 coursework.

https://stackoverflow.com/

https://www.w3schools.com/

http://tutorials.jenkov.com/java/index.html

http://tutorials.jenkov.com/javafx/index.html