

**INFORMATICS INSTITUTE OF TECHNOLOGY**

PROGRAMING PRINCIPAL II

4COSC010C.3

SEMESTER 2 🡪ASSIGNMENT 02

**GYM MANAGEMENT SYSTEM**

Module Leader’s Name – Mr. Poravi Guhanathan

Name - Ashfaaq Ahamed Hilal

UoW No. – 17613343/1

Student ID - 2019394

**INTRODUCTION TO THE APPLICATION**

The application I have created is regarding storing details of members who join to a gym. This is simply known as “Gym Management System”.

**Code Sections Separated**

* **Menu to select the method**

@Override  
public void start(Stage mainStage) throws IOException {  
 Scanner menuScanner = new Scanner(System.*in*);  
 while (true) {  
 System.*out*.println("\n|------------------------------------------------------------------------------------|");  
 System.*out*.println("|------------------ G Y M M A N A G E M E N T S Y S T E M ---------------------|");  
 System.*out*.println("|------------------------------------------------------------------------------------|\n");  
 System.*out*.println(" \t\tEnter 'A' to add new member to the list");  
 System.*out*.println(" \t\tEnter 'D' to delete a member from the list");  
 System.*out*.println(" \t\tEnter 'P' to print a list of members in the system");  
 System.*out*.println(" \t\tEnter 'S' to sort the list of members");  
 System.*out*.println(" \t\tEnter 'C' to search a member in the list");  
 System.*out*.println(" \t\tEnter 'W' to write/Save to a file");  
 System.*out*.println(" \t\tEnter 'O' to open the GUI");  
 System.*out*.println(" \t\tEnter 'X' to end the program\n");  
 System.*out*.print("-->> Select an option : ");  
 String selectMenu = menuScanner.next(); *//Menu option to select which method to start with* switch (selectMenu) {  
 case "A": *//option to add* case "a":  
  
 System.*out*.println("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++");  
 System.*out*.println("+\t\tNumber of members in the gym -> " + *memberArrayList*.size()+"\t\t+");  
 System.*out*.println("+\t\tNumber of members that can be added -> " + (100 - *memberArrayList*.size())+"\t\t+");  
 System.*out*.println("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++");  
 if (*memberArrayList*.size() != 100) {  
 addMember();  
 break;  
 } else {  
 System.*out*.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*LIMIT EXCEEDED\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  
 break;  
 }  
  
 case "D": *//option to delete* case "d":  
 deleteMember();  
 break;  
  
 case "P": *//option to print* case "p":  
 printMember();  
 break;  
  
 case "S": *//option to sort* case "s":  
 sortMember();  
 break;  
  
 case "C": *//option to search* case "c":  
 searchMember();  
 break;  
  
 case "W": *//option to write/save to file* case "w":  
 writeMember();  
 break;  
  
 case "O": *//option to open GUI* case "o":  
 openGUI(mainStage);  
 break;  
  
 case "X": *//option to exit program* case "x":  
 System.*out*.println("======================================================================================");  
 System.*out*.println("= Thank you!!! =");  
 System.*out*.println("======================================================================================");  
 System.*exit*(1);  
  
 default: *//when invalid option entered, it returns back to the menu* System.*out*.println("\n---------------Invalid Option---------------\n");  
 break;  
 }  
 }  
}

* **Selecting the type of member to be added**

public void addMember() { *//select the type of member* System.*out*.println();  
 Scanner addScanner = new Scanner(System.*in*);  
 Menu:  
 while (true) {  
 System.*out*.println(" \t\tEnter '1' to add member to Default Member List");  
 System.*out*.println(" \t\tEnter '2' to add member to Student Member List");  
 System.*out*.println(" \t\tEnter '3' to add member to Over 60 Member List\n");  
 System.*out*.print("-->> Select an option : ");  
 String addMember = addScanner.next(); *//Menu to select the type of the member* switch (addMember) {  
 case "1": *//default member option* defaultMember();  
 break Menu;  
  
 case "2": *//student member option* studentMember();  
 break Menu;  
  
 case "3": *//over 60 member option* over60Member();  
 break Menu;  
  
 default: *//invalid option* System.*out*.println("\n---------------Invalid Option---------------\n");  
 break;  
 }  
 }  
}

* **GymMember Interface created**

interface GymManager {  
 void addMember(); *//adding member method* void defaultMember(); *//adding a default member method* void studentMember(); *//adding a student member method* void over60Member(); *//adding an over 60 member method* void deleteMember(); *//deleting a member method* void printMember(); *//printing a member method* void sortMember(); *//sorting a member method* void searchMember(); *//search a member method* void writeMember() throws IOException; *//write or save members to file method* void openGUI(Stage mainStage); *//open the GUI method*}

* **Validations done to add a member to the list**
* Name validation

while (true){  
 try {  
 Scanner addScanner = new Scanner(System.*in*);  
 System.*out*.print("\nName : "); *//name input and validation* name = addScanner.nextLine().toUpperCase(); *//store name as upper case* while (true){ *//loop if improper name is given* if (!name.matches("[a-zA-Z\\s]+")) { *//validating the name input* System.*out*.println("---------------Please enter valid name---------------\n");  
 System.*out*.print("Name : ");  
 name = addScanner.nextLine().toUpperCase();  
 }else {  
 break;  
 }  
 }

* Membership ID validation

System.*out*.print("Membership Number : "); *//membership ID input and validation* membershipID = addScanner.nextLine();  
  
 boolean checkTaken = true; *//boolean to check if the ID number is repeating* while (checkTaken) { *//loop if ID number is already taken* if (!membershipID.matches("[0-9]+")) { *//validating ID number input* System.*out*.println("---------------Please enter valid ID number---------------\n");  
 System.*out*.print("Membership Number : ");  
 membershipID = addScanner.nextLine();  
 checkTaken = true;  
 } else {  
 checkTaken = false;  
 }  
  
 for (int loop = 0; loop < *memberArrayList*.size(); loop++) {  
 try {  
 if ((*memberArrayList*.get(loop).getMembershipID() == *parseInt*(membershipID))) { *//check if ID number is repeated?* System.*out*.println("---------------Membership ID already taken---------------\n");  
 System.*out*.print("Membership Number : ");  
 membershipID = addScanner.nextLine();  
 checkTaken = true;  
 } else {  
 checkTaken = false;  
 }  
 }catch (Exception e){  
 System.*out*.println("---------------Please enter valid ID number---------------\n"); *//improper ID number entered* System.*out*.print("Membership Number : ");  
 membershipID = addScanner.nextLine();  
 checkTaken = true;  
 }  
 }  
 }

* Date validation

System.*out*.print("Date of join in (DD/MM/YYYY): "); *//Join date input and validation* startMembershipDate = addScanner.nextLine();  
 while (true) {  
 if (startMembershipDate.matches("[0-9/]+")) { *//date can contain only these elements* try {  
 SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy"); *//pattern to enter date* formatter.setLenient(false); *//check date format object* formatter.parse(startMembershipDate); *//parsing input date to formatter* break;  
  
 } catch (ParseException e) {  
 System.*out*.println("Please use the given Date format -> (\"DD/MM/YYYY\")"); *//if entered date is not in the required format* System.*out*.print("Date of join in (DD/MM/YYYY): "); *//Join date input and validation* startMembershipDate = addScanner.nextLine();  
 }  
 } else if (!startMembershipDate.matches("[0-9/]+")){ *//date can contain only these elements* System.*out*.println("---------------Enter valid Date in given Date format -> (\"DD/MM/YYYY\")---------------\n"); *//date validating* System.*out*.print("Date of join in (DD/MM/YYYY): "); *//Join date input and validation* startMembershipDate = addScanner.nextLine();  
 }else {  
 break;  
 }  
 }

* School name validation

System.*out*.print("School : "); *//School name input and validation*schoolName = addScanner.nextLine().toUpperCase();  
while (true){  
 if ((schoolName.matches("[0-9]+")) || (!schoolName.matches("[a-zA-Z0-9-\\s]+"))) { *//validating school name* System.*out*.println("---------------Please enter valid school name---------------\n");  
 System.*out*.print("School : ");  
 schoolName = addScanner.nextLine().toUpperCase(); *//store school name is upperCase* }else {  
 break;  
 }  
}

* Age validation

System.*out*.print("Age : "); *//Age input and validation*age = addScanner.nextLine();  
while (true){  
 if (!age.matches("[0-9]+")){ *//age should contain only these elements* System.*out*.println("---------------Enter valid Age---------------\n");  
 System.*out*.print("Age : ");  
 age = addScanner.nextLine();  
 }else if ((age.matches("[0-9]+")) && (*parseInt*(age)<60)) { *//age should be more than 60* System.*out*.println("---------------The entered age is below 60---------------\n");  
 System.*out*.print("Age : ");  
 age = addScanner.nextLine();  
 }else {  
 break;  
 }  
}

* **Adding the member to the list**

*memberArrayList*.add(new DefaultMember(name,membershipID,new Date(startMembershipDate))); *//adding the default member to array list* System.*out*.println("\n---------------Recruit receipt---------------\n\n\t\* Your name is "+ name+"\n\t\* Your ID is "+membershipID+"\n\t\* Membership date from - "+startMembershipDate+"\n");  
 *//printing the details which was inputted* break;  
 }catch (InputMismatchException e){  
 System.*out*.println("\n---------------Invalid inputs---------------\n");  
 }

* **Deleting a member from the gym list**

public void deleteMember() { *//delete a member by ID* if (*memberArrayList*.size()==0){ *//check if there are any members* System.*out*.print("\n---------------No members registered yet---------------\n");  
 }else {  
 try {  
 while (true) {  
 Scanner deleteScanner = new Scanner(System.*in*);  
 System.*out*.print("\nEnter ID to be deleted OR press \"0\" to cancel: ");  
 int searchDelete = deleteScanner.nextInt();  
 if (searchDelete == 0) { *//exit method by pressing "0"* break;  
 }  
 for (int loop = 0; loop <= *memberArrayList*.size(); loop++)  
 if (*memberArrayList*.get(loop).getMembershipID() == searchDelete) { *//checking the list if the entered ID is there in the list* System.*out*.println("\n---------------Successfully deleted the below member---------------\n\n" + *memberArrayList*.get(loop));  
 *memberArrayList*.remove(loop); *//remove member* System.*out*.println("\n\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++");  
 System.*out*.println("+\t\tNumber of members in the gym -> " + *memberArrayList*.size()+"\t\t+");  
 System.*out*.println("+\t\tNumber of members that can be added -> " + (100 - *memberArrayList*.size())+"\t\t+");  
 System.*out*.println("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++");  
 *//display the number of members* break;  
 }  
 }  
 } catch (Exception e) { *//error msg if ID is not in the list* System.*out*.println("\n---------------Invalid ID---------------\n");  
 deleteMember(); *//iterate again for user to delete a member* }  
 }  
}

* **Printing the list of members in the gym**

public void printMember() { *//print the list of members* if (*memberArrayList*.size()==0){ *//check if there are members in the list* System.*out*.print("\n---------------No members registered yet---------------\n");  
 }else{  
 System.*out*.println();  
 for (int loop = 0 ; loop < *memberArrayList*.size() ; loop++){  
 System.*out*.println(*memberArrayList*.get(loop)); *//print all the members in the list* }  
 }System.*out*.println("\n+++++++++++++++++++++++++++++++++++++++++++++++++++++++++");  
 System.*out*.println("+\t\tNumber of members in the gym -> " + *memberArrayList*.size()+"\t\t+");  
 System.*out*.println("+\t\tNumber of members that can be added -> " + (100 - *memberArrayList*.size())+"\t\t+");  
 System.*out*.println("+++++++++++++++++++++++++++++++++++++++++++++++++++++++++\n");  
 *//display the number of member*}

* **Sort member according to ID numbers in ascending order**

if (sort.equals("1")) { *//sorting by ID <BUBBLE SORT METHOD>* for (int outerLoop = 0; outerLoop < *memberArrayList*.size() - 1; outerLoop++) {  
 for (int innerLoop = 0; innerLoop < *memberArrayList*.size() - (outerLoop + 1); innerLoop++) {  
 if (*memberArrayList*.get(innerLoop).getMembershipID() > *memberArrayList*.get(innerLoop + 1).getMembershipID()) {  
 DefaultMember sortByIDVariable = *memberArrayList*.get(innerLoop + 1); *//store the value in temporary variable  
 memberArrayList*.set(innerLoop + 1, *memberArrayList*.get(innerLoop));  
 *memberArrayList*.set(innerLoop, sortByIDVariable);  
 }  
 }  
 }  
 for (int loop = 0; loop < *memberArrayList*.size(); loop++) {  
 System.*out*.println(*memberArrayList*.get(loop)); *//printing the sorted the list* }System.*out*.println("---------------Sorted according to the ID---------------\n");  
 break;

}

* **Sort member according to names in ascending order**

else if (sort.equals("2")) { *//sorting by ID <BUBBLE SORT METHOD>* for (int outerLoop = 0; outerLoop < *memberArrayList*.size() - 1; outerLoop++) {  
 for (int innerLoop = 0; innerLoop < *memberArrayList*.size() - (outerLoop + 1); innerLoop++) {  
 if (*memberArrayList*.get(innerLoop).getName().compareTo(*memberArrayList*.get(innerLoop + 1).getName()) > 0) {  
 DefaultMember sortByNameVariable = *memberArrayList*.get(innerLoop + 1); *//store the value in temporary variable  
 memberArrayList*.set(innerLoop + 1, *memberArrayList*.get(innerLoop));  
 *memberArrayList*.set(innerLoop, sortByNameVariable);  
 }  
 }  
 }  
 for (int loop = 0; loop < *memberArrayList*.size(); loop++) {  
 System.*out*.println(*memberArrayList*.get(loop)); *//printing the sorted the list* }System.*out*.println("---------------Sorted according to the Name---------------\n");  
 break;  
}

* **Search member according to ID number in ascending order**

if (search.equals("1")) { *//search by ID* System.*out*.print("Enter ID number : "); *//enter ID number* int searchID = searchScanner.nextInt();  
  
 boolean found = false; *//boolean to stop the iteration* for (int loop = 0; loop < *memberArrayList*.size(); loop++) {  
 if (*memberArrayList*.get(loop).getMembershipID() == searchID) { *//check the list if the relevant ID mentioned is available or not* System.*out*.println("\n"+*memberArrayList*.get(loop)); *//print the details of the searched ID* found = true; *//ID is found in the list* break;  
 }  
 }if (!found){ *//ID not found in the list* System.*out*.println("\n---------------ID Number not found in the List---------------\n");  
 break;  
 }break;  
}

* **Search member according to names in ascending order**

else if (search.equals("2")) { *//search by NAME* System.*out*.print("Enter name : "); *//enter name* String searchName = searchScanner.nextLine().toUpperCase();  
  
 boolean found = false; *//boolean to stop the iteration* for (int loop = 0; loop < *memberArrayList*.size(); loop++) {  
 if (*memberArrayList*.get(loop).getName().equals(searchName)) { *//check the list if the relevant name mentioned is available or not* System.*out*.println("\n"+*memberArrayList*.get(loop)); *//print the details of the searched name* found = true; *//name is found in the list* }  
 }if (!found){ *//name not found in the list* System.*out*.println("\n---------------Name not found in the List---------------\n");  
 break;  
 }break;  
}

* **Writing the list of members in the gym to a text file and binary file**

try {  
 *gymFile* = new File("MyGymRecordsFile.txt"); *//check if there is file in that name  
 gymBinaryFile* = new File("GymRecordsBinaryFile.ser"); *//check if there is file in that name  
  
 gymFile*.createNewFile(); *// else create a file  
 gymBinaryFile*.createNewFile(); *// else create a file*} catch (IOException e) {  
 System.*out*.print("Creating file");  
}finally {  
 *//---writing to text file---* FileWriter gymFileWriter = new FileWriter(*gymFile*);  
 PrintWriter gymPrintWriter = new PrintWriter(gymFileWriter, true); *//autoflush enabled* for (int loop = 0; loop < *memberArrayList*.size(); loop++)  
 gymPrintWriter.println(*memberArrayList*.get(loop));  
  
 *//---writing to binary file---* FileOutputStream gymFileOutputStream = new FileOutputStream(*gymBinaryFile*);  
 ObjectOutputStream gymObjectOutputStream = new ObjectOutputStream(gymFileOutputStream);  
 gymObjectOutputStream.writeObject(*memberArrayList*);  
  
 gymFileWriter.close();  
 gymPrintWriter.close();  
  
 gymFileOutputStream.close();  
 gymObjectOutputStream.close();  
  
}

* **Reading the binary file and loading the data to the list**

try {  
 *gymBinaryFile* = new File("GymRecordsBinaryFile.ser"); *//check if this file is available  
 gymBinaryFile*.createNewFile(); *//else create the file*}catch (IOException e) {  
 System.*out*.print("Creating file");  
}finally {  
 try{  
 FileInputStream gymFileInputStream = new FileInputStream(*gymBinaryFile*); *//binary file input stream to read from file* ObjectInputStream gymObjectInputStream = new ObjectInputStream(gymFileInputStream); *//binary file object input stream to read objects  
 memberArrayList* = (ArrayList<DefaultMember>) gymObjectInputStream.readObject(); *//reading* gymFileInputStream.close();  
 gymObjectInputStream.close();  
  
 }catch (EOFException e){ *//prevents error when it reaches the end of file <when reading objects Objects>* System.*out*.println("====");  
 } catch (ClassNotFoundException e) {  
 e.printStackTrace();  
 }  
}

* **Search option used in the GUI**

if (searchTextField.getText().equals("")) { *//if textfield is empty display the whole list* myGymManagerTable.getItems().clear();  
 for(int i = 0; i < *memberArrayList*.size(); i++) {  
 DefaultMember gymManagerTableData = *memberArrayList*.get(i);  
 myGymManagerTable.getItems().addAll(gymManagerTableData);  
 }  
}  
 else if(searchTextField.getText().matches("[0-9]+")){ *//if textfield contains numbers, search according to ID* myGymManagerTable.getItems().clear();  
 for(int i = 0; i < *memberArrayList*.size(); i++){  
 if (*memberArrayList*.get(i).getMembershipID() == *parseInt*(searchTextField.getText())) { *//check the ID list if any matches are there* DefaultMember searchByMembershipID = *memberArrayList*.get(i);  
 myGymManagerTable.getItems().addAll(searchByMembershipID);  
 }  
 }  
 }else if(searchTextField.getText().matches("[a-zA-Z\\s]+")){ *//if textfield contains letters, search according to name* myGymManagerTable.getItems().clear();  
 for(int i = 0; i < *memberArrayList*.size(); i++){  
 if (*memberArrayList*.get(i).getName().contains(searchTextField.getText().toUpperCase())) { *//check the name list if any matches are there* DefaultMember searchByName = *memberArrayList*.get(i);  
 myGymManagerTable.getItems().addAll(searchByName);  
 }  
 }  
}

* **Constructors used DefaultMember class**

public DefaultMember(String name, String membershipID, Date startMembershipDate) { *//constructor for the private variables* this.name = name;  
 this.membershipID = membershipID;  
 this.startMembershipDate = startMembershipDate;  
}

* **Constructors used StudentMember class**

public StudentMember(String name, String membershipID, Date startMembershipDate, String schoolName) { *//constructor for the private variables* super(name, membershipID, startMembershipDate);  
 this.schoolName = schoolName;  
}

* **Constructors used Over60Member class**

public Over60Member(String name, String membershipID, Date startMembershipDate, String age) { *//constructor for the private variables* super(name, membershipID, startMembershipDate);  
 this.age = age;  
}

* **Constructors used Date class**

public Date(String startMembershipDate) { *//constructor for the private variables* this.startMembershipDate = startMembershipDate;  
}

* **Getter methods**

public String getName() { *//method to get name* return name;  
}  
  
public int getMembershipID() { *//method to get ID* return Integer.*parseInt*(membershipID);  
}  
  
public Date getStartMembershipDate() { *//method to get start membership date* return startMembershipDate;  
}

public String getSchoolName() { *//method to get school name* return schoolName;  
}

public String getAge() { *//method to get age* return age;  
}

public String getStartMembershipDate() { *//method to get start membership date* return startMembershipDate;  
}

* **Setter Methods**

public void setName(String name) { *//method to set name* this.name = name;  
}  
  
public void setMembershipID(String membershipdID) { *//method to set ID* this.membershipID = membershipdID;  
}  
  
public void setStartMembershipDate(Date startMembershipDate) { *//method to set start membership date* this.startMembershipDate = startMembershipDate;  
}

public void setSchoolName(String schoolName) { *//method to set school name* this.schoolName = schoolName;  
}

public void setAge(String age) { *//method to set age* this.age = age;  
}

public void setStartMembershipDate(String startMembershipDate) { *//method to set start membership date* this.startMembershipDate = startMembershipDate;  
}

* **toString method used in DefaultMember class**

@Override  
public String toString() { *//converting all the values to string inorder to print it* return "-----Default Member-----\nName : "+getName()+"\nMembership ID : "+getMembershipID()+"\nDate joined : "+getStartMembershipDate()+"\n";  
}

* **toString method used in StudentMember class**

@Override  
public String toString() { *//converting all the values to string inorder to print it* return "-----Student Member-----\nName : "+getName()+"\nMembership ID : "+getMembershipID()+"\nDate joined : "+getStartMembershipDate()+"\nSchool Name : "+getSchoolName()+"\n";  
}

* **toString method used in Over60Member class**

@Override  
public String toString() { *////converting all the values to string inorder to print it* return "-----Over 60 Member-----\nName : "+getName()+"\nMembership ID : "+getMembershipID()+"\nDate joined : "+getStartMembershipDate()+"\nAge : "+getAge()+"\n";  
}

* **toString method used in Date class**

@Override  
public String toString() { *//converting all the values to string inorder to print it* return startMembershipDate;  
}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test case Scenario** | **Purpose of the test case** | **Expected output** | **Actual output** | **Pass / Fail** |
| 01. | Launch the application. | To select the option to start with. | Menu option to select the option. | Menu option to select the option. Loads all the data saved in the file. | Pass |
| 02. | Selecting “A”  (Add a new member). | To add a new member to the gym. | Calculate number of members that could be added in the gym.  Opens a sub menu requesting to which type of member is to be added. | Calculate number of members that could be added in the gym.  Calculates the total members in the gym  Opens a sub menu requesting to which type of member is to be added. | Pass |
| 03. | Select “1”  (In sub-menu) | To add a ‘Default Member’. | Requests inputs for details. | Requests inputs for details. | Pass |
| 04. | Select “2”  (In sub-menu) | To add ‘Student Member’. | Requests inputs for details. | Requests inputs for details. | Pass |
| 05. | Select “3”  (In sub-menu) | To add ‘Over 60 Member’. | Requests inputs for details. | Requests inputs for details. | Pass |
| 06. | Select “D”  (Delete a member) | To delete a member who is already registered in the gym. | Request of membership ID number to check if the relevant member is in the gym.  Display type of member which was removed.  Display number of free spaces left. | Request of membership ID number to check if the relevant member is in the gym.  Display the full information of the member.  Display number of free spaces left and total number of members in gym. | Pass |
| 07. | Select “P”  (Print) | To print the list of members who are available in the gym. | List of members including the type of member gets printed. | List of members including the type of member gets printed.  Display number of free spaces left and total number of members in gym. | Pass |
| 08. | Select “S”  (Sort) | To sort the list of members in ascending order. | Sort the members by either name or membership ID in ascending order. | Request if the user needs to sort according to name or ID number.  If the user selects the ID, it is sorted by ID and printed.  If the user selects the name option, it gets sorted accordingly and prints it. | Pass |
| 09. | Select “C”  (Search) | To search a particular member. | ----- | Request if the user needs to search according to name or ID number.  If the user selects the ID, the relevant member details gets printed.  If the user selects the name option, the members with that name gets printed. | Pass |
| 10. | Select “W”  (Write/Save) | To save the details to the file. | Save all the details of the members that are being added to the list in a file. | Saves the details of the members in text file and also binary file.  Display number of free spaces left and total number of members in gym. | Pass |
| 11. | Select “O”  (Open GUI) | To show a table format view of the list of members in the gym and search a member. | Open GUI and show the list of members in the list.  Search a member according to either name or membership ID. | Open GUI and show the list of members in the list.  Able to search a member by name as well as ID number.  Display number of free spaces left and total number of members in gym. | Pass |
| 12. | Select “X”  (Exit system) | To exit the system. | Exit the system. | Exits the system. | Pass |
| 13. | Anything else pressed other than {A, D, P, S, C, W, O, X} in main menu.  Not case sensitive | Error handling | Prevent error with an error message. | Prints ‘Invalid Option’ and goes back again to the main menu. | Pass |
| 14. | Anything else pressed other than (1, 2, 3} in add member sub menu. | Error handling | Prevent error with an error message. | Prints ‘Invalid Option’ and goes back again to the sub menu. | Pass |
| 15. | Entering a name with anything other than letter when adding a member. | Error handling | Prevent error with an error message. | Prints ‘Enter valid name’ and requests to enter name again. | Pass |
| 16. | Entering ID number with anything other than integers. | Error handling | Prevent error with an error message. | Prints ‘Enter valid ID’ and requests to enter ID again. | Pass |
| 17. | Entering an ID number which is already taken. | Error handling | Prevent error with an error message. | Prints ‘ID number is already taken’ and requests to enter ID again | Pass |
| 18. | Entering date in wrong format. | Error handling | Prevent error with an error message. | The format of the date is provided (DD/MM/YYYY).  Prints ‘Enter valid date’ and request the date again. | Pass |
| 19. | Enter date in out of range. | Error handling | Prevent error with an error message. | Prints ‘Enter valid date’ and request the date again when date entered is not realistic. | Pass |
| 20. | Enter only integers for school name. | Error handling | Prevent error with an error message. | Prints ‘Enter valid school name’ and requests the school name again. | Pass |
| 21. | Enter age without integers. | Error handling | Prevent error with an error message. | Prints ‘Enter valid age’ and requests to enter the age again. | Pass |
| 22. | Enter age which is less than 60. | Error handling | Prevent error with an error message. | Prints ‘Enter valid age’ and requests to enter the age again. | Pass |
| 23. | Check if there are member added yet when delete option is clicked. | Error handling | Prevent error with an error message. | If there are no members in the list |  |
| 23. | Enter an ID which is not in the list to delete a member. | Error handling | Prevent error with an error message. | Prints ‘Enter valid ID’ and requests to enter ID again. | Pass |
| 24. | Entering ID number with anything other than integers in the delete option. | Error handling | Prevent error with an error message. | Prints ‘Enter valid ID’ and requests to enter ID again.  Will be looped till the user clicks “0” to exit. | Pass |
| 25. | Anything else pressed other than (1, 2} in sort member sub menu. | Error handling | Prevent error with an error message. | Prints ‘Invalid Option’ and goes back again to the sub menu. | Pass |
| 26. | Anything else pressed other than (1, 2} in search member sub menu. | Error handling | Prevent error with an error message. | Prints ‘Invalid Option’ and goes back again to the sub menu. | Pass |
| 27. | Enter an ID which is not in the list to search a particular member. | Error handling | Prevent error with an error message. | Prints ‘ID number not found in list’. | Pass |
| 28. | Enter anything other than integers when selecting ID number to be searched. | Error handling | Prevent error with an error message. | Prints “Invalid Inputs”. | Pass |
| 29. | Enter name which is not in the list to search a particular member. | Error handling | Prevent error with an error message. | Prints ‘Name not found in list’. | Pass |
| 30. | Check if the user can add a member. | User can add members only if the number of members in gym is less than 100. | If number of members in the gym is less than 100, then proceeds.  If it is more than 100, it should prevent and error message | If number of members in the gym is less than 100, then proceeds.  If it is more than 100, it will print “Limit exceeded” and direct the user back to main menu. | Pass |
| 31. | Entering valid details when adding member. | When all the details entered doesn’t come up with any exceptions when adding member. | Proceed further. | Prints the details of the recently added member as “Recruit receipt” and then proceed further. | Pass |
| 32. | Entering a valid ID number to be deleted from the list. | Delete the ID number if it matches any ID number from the available list | Remove the member from the list. | Removes the particular member from the list.  Displays the details of the particular member who was deleted.  Prints a message as “Successfully deleted”. | Pass |
| 33. | Select the sort by ID option. | Sort the list of members in ascending order according to their ID numbers. | Sort in ascending order. | Sort in ascending order and leaves a message as “Sorted successfully”. | Pass |
| 33. | Select the sort by name option. | Sort the list of members in ascending order according to their names. | Sort in ascending order. | Sort in ascending order and leaves a message as “Sorted successfully”. | Pass |
|  |  |  |  |  |  |