TARUMT
TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

BACS2063 Data Structures and Algorithms

ASSIGNMENT 202305

University Management System

Declaration

- I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University of Management and Technology's plagiarism policy.
- I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

Student Name	Student ID	Prog / Tut.Grp	Signature
Lee Weng Yi	2309332	RSD G3	
Choo Shi Yi	2309309	RSD G3	
Bok Cheong Roy	2309298	RSD G3	
Pua Jia Qian	2309358	RSD G3	

Table of Contents

A. TEAM REPORT

1. Abstract Data Type (ADT)

1.1 ADT Specification

Description:

This ADT represents a generic list data structure that stores elements of type T. It provides various operations for managing and manipulating the elements within the list.

Operations:

1. **boolean add(T newEntry)**

   Description :Adds newEntry at position newPosition within the list.

   the list.

   Precondition:newPosition must be between1 to total entries + 1.

   Postcondition:newEntry has been added to the indicated position of the list.

   Returns:true if newEntry was successfully added to the list; false otherwise.

2. **int getPosition(T anEntry)**

   Description: Retrieves the position (index) of the first occurrence of anEntry in the list.

   Postcondition: The list remains unchanged.

   Return: The index (position) of anEntry in the list, or -1 if not found.

3. **T getEntry(int givenPosition)**

   Description: Retrieves the entry at position givenPosition in the list.

   Precondition:newPosition must be between1 to total entries.

   Postcondition: The list remains unchanged.

   Return: The element at the givenPosition, or null if the position is invalid.

4**. T remove(int givenPosition)**

   Description: Removes the entry at position givenPosition within the list.

   Precondition:givenPosition must be between 1 to total entries.

   Postcondition: The entry at position givenPosition has been removed from the list.

Return: The entry that was removed from the list.

5. **boolean remove(T anEntry)**

Description: Removes the first occurrence of anEntry from the list, if present.

Postcondition: The first occurrence of anEntry is removed from the list, and subsequent elements are shifted to fill the gap.

Return: true if anEntry is found and removed, false otherwise.

6. **boolean contains(T anEntry)**

Description: Checks if anEntry is present in the list.

Postcondition: The list remains unchanged.

Return: true if anEntry is found in the list, false otherwise.

7. **void clear()**

Description: Clears all elements from the list.

Postcondition: The list becomes empty.

Return: None.

8. **int getNumberOfEntries**()

Description: Retrieves the number of elements in the list.

Precondition: The elements in the list are comparable

Postcondition: The list remains unchanged.

Return: The number of elements in the list.

9. **void selectionSortAscending()**

Description: Sorts the elements in ascending order using a selection sort algorithm.

Precondition: The elements in the list are comparable

Postcondition: The elements in the list are sorted in ascending order.

Return: None.

10. **void selectionSortDescending()**

Description: Sorts the elements in descending order using a selection sort algorithm.

Postcondition: The elements in the list are sorted in descending order.

Return: None.

11. **boolean isEmpty()**

    Description: Checks if the list is empty.

    Postcondition: The list remains unchanged.

    Return: true if the list is empty, false otherwise.

1.2 ADT Implementation

a. Java Interface

```
 1 package adt;
 2
 3 /** taken from sample code, except selectionSortAscending and
selectionSortDescending
 4  * @author Frank M. Carrano
 5  * @version 2.0
 6  * @param <T>
 7  */
 8 public interface SortedListInterface<T extends Comparable<T>> {
 9
10     public boolean add(T newEntry);
11
12     public int getPosition(T anEntry);
13
14     public T getEntry(int givenPosition);
15
16     public T remove(int givenPosition);
17
18     public boolean remove(T anEntry);
19
20     public boolean contains(T anEntry);
21
22     public void clear();
23
24     public int getNumberOfEntries();
25
26     public void selectionSortAscending();
27
28     public void selectionSortDescending();
29
30     public boolean isEmpty();
31
32 }
```

b. Java Implementation

```
 1 package adt;
 2 import java.io.Serializable;
 3
 4 /**taken from sample code, except selectionSortAscending and
selectionSortDescending
 5  * @author Frank M. Carrano
 6  * @version 2.0
 7  * @param <T>
 8  */
 9
10 public class SortedArrayList<T extends Comparable<T>> implements
SortedListInterface<T>, Serializable {
11
12     private T[] array;
13     private int numberOfEntries;
14     private static final int DEFAULT_CAPACITY = 25;
15
16     public SortedArrayList() {
17         this(DEFAULT_CAPACITY);
18     }
19
20     public SortedArrayList(int initialCapacity) {
21         numberOfEntries = 0;
22         array = (T[]) new Comparable[initialCapacity];
23     }
24
25     @Override
26     public boolean add(T newEntry) {
27         int i = 0;
28         while (i < numberOfEntries && newEntry.compareTo(array[i]) > 0) {
29             i++;
30         }
31         makeRoom(i + 1);
32         array[i] = newEntry;
33         numberOfEntries++;
34         return true;
35     }
36
37     @Override
38     public boolean remove(T anEntry) {
39         boolean found = false;
40
41         if (!isEmpty()) {
```

```
42              int position = getPosition(anEntry);
43              if (position >= 0) {
44                  removeGap(position);
45                  numberOfEntries--;
46                  found = true;
47              } // end if
48          } // end if
49
50          return found;
51      }
52
53      @Override
54      public int getPosition(T anEntry) {
55          int position = 1;
56
57          while ((position <= numberOfEntries) &&
(anEntry.compareTo(getEntry(position - 1)) > 0)) {
58              position++;
59          } // end while
60
61          if ((position > numberOfEntries) ||
!anEntry.equals(getEntry(position - 1))) {
62              position = -position;
63          } // end if
64
65          return position;
66      } // end getPosition
67
68      // list operations
69      @Override
70      public T remove(int givenPosition) {
71          T result = null; // return value
72
73          if ((givenPosition >= 1) && (givenPosition <= numberOfEntries)) {
// test catches empty list
74
75              result = array[givenPosition - 1]; // get entry to be removed
76
77              // move subsequent entries towards entry to be removed,
78              // unless it is last in list
79              if (givenPosition < numberOfEntries) {
80                  removeGap(givenPosition);
81              }
```

```
82
83              numberOfEntries--;
84          } // end if
85
86          return result; // return reference to removed entry,
87          // or null if list is empty
88      } // end remove
89
90      @Override
91      public void clear() {
92          numberOfEntries = 0;
93      }
94
95      @Override
96      public boolean contains(T anEntry) {
97          boolean found = false;
98          for (int index = 0; !found && (index < numberOfEntries); index++)
{
99              if (anEntry.equals(array[index])) {
100                 found = true;
101             }
102         }
103         return found;
104     }
105
106     @Override
107     public int getNumberOfEntries() {
108         return numberOfEntries;
109     }
110
111     @Override
112     public T getEntry(int givenPosition) {
113         T result = null; // result to return
114
115         if (givenPosition >= 0 && givenPosition < numberOfEntries) {
116             result = array[givenPosition];
117         } // end if
118
119         return result;
120     } // end getEntry
121
122     @Override
123     public boolean isEmpty() {
```

```
124         return numberOfEntries == 0;
125     }
126 
127     @Override
128     public String toString() {
129         String outputStr = "";
130         for (int index = 0; index < numberOfEntries; ++index) {
131             outputStr += array[index] + "\n";
132         }
133 
134         return outputStr;
135     }
136 
137     private void makeRoom(int newPosition) {
138         int newIndex = newPosition - 1;
139         int lastIndex = numberOfEntries - 1;
140 
141         for (int index = lastIndex; index >= newIndex; index--) {
142             array[index + 1] = array[index];
143         }
144     }
145 
146     private void removeGap(int givenPosition) {
147         int removedIndex = givenPosition - 1;
148         int lastIndex = numberOfEntries - 1;
149 
150         for (int index = removedIndex; index < lastIndex; index++) {
151             array[index] = array[index + 1];
152         }
153     }
154 
155     @Override
156     public void selectionSortAscending() {
157         int n = numberOfEntries;
158 
159         for (int i = 0; i < n - 1; i++) {
160             int minIndex = i;
161             for (int j = i + 1; j < n; j++) {
162                 if (array[j].compareTo(array[minIndex]) < 0) {
163                     minIndex = j;
164                 }
165             }
166 
```

```
167             // Swap elements if needed
168             if (minIndex != i) {
169                 T temp = array[i];
170                 array[i] = array[minIndex];
171                 array[minIndex] = temp;
172             }
173         }
174     }
175 
176     @Override
177     public void selectionSortDescending() {
178         int n = numberOfEntries;
179 
180         for (int i = 0; i < n - 1; i++) {
181             int maxIndex = i;
182             for (int j = i + 1; j < n; j++) {
183                 if (array[j].compareTo(array[maxIndex]) > 0) {
184                     maxIndex = j;
185                 }
186             }
187 
188             // Swap elements if needed (reversed comparison for descending
order)
189             if (maxIndex != i) {
190                 T temp = array[i];
191                 array[i] = array[maxIndex];
192                 array[maxIndex] = temp;
193             }
194         }
195     }
196 
197 
198 
199 }
```

B. INDIVIDUAL REPORTS

2. Use of ADTs
2.1 Course Management

Student Name	Student ID	Prog / Tut.Grp	Signature
Pua Jia Qian	2309358	RSD2G3	

2.1.1 Source Code
MaintainCourse.java

```
 1 package control;
 2
 3 import adt.*;
 4 import dao.*;
 5 import entity.*;
 6 import boundary.*;
 7 import java.util.Scanner;
 8 import utility.*;
 9
10 /**
11  *
12  * @author Pua Jia Qian
13  */
14 public class MaintainCourse {
15
16     Scanner scanner = new Scanner(System.in);
17
18     private SortedListInterface<Course> courseList = new
SortedArrayList<>();
19     private SortedListInterface<Programme> programList = new
SortedArrayList<>();
20     private final CourseDAO cd = new CourseDAO();
21     private final ProgrammeDAO pd = new ProgrammeDAO();
22     private final CourseManagementUI courseUI = new CourseManagementUI();
23     private ProgrammeManagement pm = new ProgrammeManagement();
24     private ProgrammeManagementUI pmUI = new ProgrammeManagementUI();
25
26     public MaintainCourse() {
27         try {
28             courseList = cd.retrieveFromFile();
29             if (courseList == null) {
30                 System.out.println("The course list could not be loaded
from the file or is empty.");
31             }
32         } catch (ClassNotFoundException ex) {
33             System.out.println("Error: Something Wrong when reading
file");
34         }
35     }
36
37     public void runCourseManagement() {
38         int choice;
39         do {
```

```
40              choice = courseUI.courseMenu();
41              switch (choice) {
42
43                  case 1:
44                      courseUI.addTitle();
45                      addNewCourse();
46                      break;
47                  case 2:
48                      courseUI.removeTitle();
49                      removeCourse(courseUI.inputCourseCode());
50                      break;
51                  case 3:
52                      courseUI.findTitle();
53                      findCourseByCode(courseUI.inputCourseCode());
54                      break;
55
56                  case 4:
57                      courseUI.modifyTitle();
58                      modifyCourse(courseUI.inputCourseCode());
59                      break;
60                  case 5:
61                      courseUI.displayTitle();
62                      displayCourses();
63                      break;
64                  case 6:
65                      courseUI.assignProgramTitle();
66                      assignProgrammeToCourse(courseUI.inputCourseCode());
67                      break;
68                  case 7:
69                      courseUI.removeProgramTitle();
70                      removeProgrammeFromCourse(courseUI.inputCourseCode());
71                      break;
72                  case 8:
73                      courseUI.reportTitle();
74                      generateReport();
75                      break;
76                  case 0:
77                      MessageUI.displayExitSubsystemMessage();
78                      //System.exit(0); // Successful termination
79                      break;
80                  default:
81                      MessageUI.displayInvalidChoiceMessage();
82              }
```

```
 83          } while (choice != 0);
 84      }
 85
 86      public void initialCourses() {
 87
 88          courseList = new SortedArrayList<>();
 89          courseList.add(new Course("BACS2234", "Human Computer
Interaction", 3));
 90          courseList.add(new Course("ABCD1234", "Discrete Mathematics", 2));
 91          courseList.add(new Course("AACD3034", "Data Structure &
Algorithms", 3));
 92      }
 93
 94      public void addNewCourse() {
 95
 96          String newCourseCode = courseUI.inputCourseCode();
 97
 98          // Check for empty fields
 99          if (newCourseCode.isEmpty()) {
100              System.out.println("Please fill in Course Code...");
101              MessageUI.displayExitSubsystemMessage();
102              return;
103
104              //check course code format
105          } else if (!newCourseCode.matches("^[A-Z]{4}\\d{4}$")) {
106              System.out.println("Invalid Course Code format, Please follow
correct format...");
107              MessageUI.displayExitSubsystemMessage();
108              return;
109
110              // Check for duplicate course code
111          } else if (checkCourseCodeDuplicate(newCourseCode)) {
112              System.out.println("This course code " + newCourseCode + "
already exists...");
113              MessageUI.displayExitSubsystemMessage();
114              return;
115          }
116          String newCourseName = courseUI.inputCourseName();
117          if (newCourseName.isEmpty()) {
118              System.out.println("Please fill in Course Name...");
119              MessageUI.displayExitSubsystemMessage();
120              return;
121
```

```
122         } else if (newCourseName.length() > 40) {
123             System.out.println("Course Name cannot exceeds 40 words...");
124             MessageUI.displayExitSubsystemMessage();
125             return;
126         }
127 
128         int newCreditHours = courseUI.inputCreditHours();
129         if (newCreditHours <= 0) {
130             System.out.println("Please fill in Credit Hour...");
131             MessageUI.displayExitSubsystemMessage();
132             return;
133         }
134 
135         Course newCourse = new Course(newCourseCode, newCourseName,
newCreditHours);
136 
137         courseUI.printCourseDetails(newCourse);
138 
139         // confirmation
140         if (courseUI.inputConfirmation().equals("y")) {
141             courseList.add(newCourse);
142             cd.saveToFile(courseList);
143             MessageUI.displaySuccessfulMessage();
144 
145         } else {
146             System.out.println("Cancel Add Course...");
147             MessageUI.displayExitSubsystemMessage();
148         }
149 
150     }
151 
152     private boolean checkCourseCodeDuplicate(String courseCode) {
153         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
154             Course course = courseList.getEntry(i);
155             if (course.getCourseCode().equals(courseCode)) {
156                 return true;
157             }
158         }
159         return false;
160     }
161 
162     public void removeCourse(String cCode) {
163 
```

```
164         // check empty course code
165         if (cCode.isEmpty()) {
166             System.out.println("Invalid course code. Please provide a
valid course code...");
167             MessageUI.displayExitSubsystemMessage();
168             return;
169         }
170 
171         boolean courseFound = false; // Flag to check if the course is
found
172 
173         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
174             Course currentCourse = courseList.getEntry(i); // get course
details
175 
176             if (currentCourse.getCourseCode().equals(cCode)) { // compare
course code
177                 courseUI.printCourseDetails(currentCourse);
178 
179                 if (courseUI.inputConfirmation().equals("y")) {
180                     boolean removed = courseList.remove(currentCourse);
181                     if (removed) {
182                         cd.saveToFile(courseList);
183                         MessageUI.displaySuccessfulMessage();
184                     } else {
185                         System.out.println("-- Course could not be
removed. --");
186                     }
187                 } else {
188                     System.out.println("Cancel Remove Course...");
189                 }
190                 courseFound = true;
191                 break; // Exit the loop when the course is found and
modified
192 
193             }
194         }
195 
196         if (!courseFound) {
197             System.out.println("This '" + cCode + "' Course not found...
");
198         }
199     }
```

```
200
201     public boolean findCourseByCode(String courseCode) {
202
203         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
204             Course currentCourse = courseList.getEntry(i);
205             if (currentCourse.getCourseCode().equals(courseCode)) {
206                 courseUI.printCourseDetails(currentCourse);
207                 return true;
208             }
209         }
210
211         System.out.println("This '" + courseCode + "' Course not found...
");
212         return false;
213     }
214
215     public void modifyCourse(String cCode) {
216
217         // check empty course code
218         if (cCode.isEmpty()) {
219             System.out.println("Invalid course code. Please provide a
valid course code...");
220             MessageUI.displayExitSubsystemMessage();
221             return;
222         }
223
224         boolean courseFound = false; // Flag to check if the course is
found
225
226         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
227             Course currentCourse = courseList.getEntry(i);
228             if (currentCourse.getCourseCode().equals(cCode)) {
229                 courseUI.printCourseDetails(currentCourse);
230
231                 String newCourseName = currentCourse.getCourseName();
232                 int newCreditHours = currentCourse.getCreditHours();
233
234                 switch (courseUI.inputModifyOption()) {
235                     case 1:
236                         System.out.print("Enter new Course Name: ");
237                         newCourseName = scanner.nextLine();
238                         if (newCourseName.isEmpty()) {
239                             System.out.println("Please fill in Course
```

```
Name...");
                            MessageUI.displayExitSubsystemMessage();
                            return;

                        } else if (newCourseName.length() > 40) {
                            System.out.println("Course Name cannot exceeds
40 words...");
                            MessageUI.displayExitSubsystemMessage();
                            return;
                        }
                        break;
                    case 2:
                        while (true) {
                            System.out.print("Enter Credit Hours: ");
                            if (scanner.hasNextInt()) {
                                newCreditHours = scanner.nextInt();
                                if (newCreditHours <= 0) {
                                    System.out.println("Credit Hours must
be a positive integer.");
                                } else {
                                    break;
                                }
                            } else {
                                scanner.nextLine(); // Consume the invalid
input
                                System.out.println("Invalid input. Please
enter a valid positive integer for Credit Hours.");
                            }
                        }
                        break;

                    default:
                        MessageUI.displayInvalidChoiceMessage();

                }

                if (courseUI.inputConfirmation().equals("y")) {
                    currentCourse.setCourseName(newCourseName);
                    currentCourse.setCreditHours(newCreditHours);

                    cd.saveToFile(courseList);
                    MessageUI.displaySuccessfulMessage();
                } else {
```

```
278                     return;
279                 }
280                 courseFound = true; // Set the flag to true
281                 break; // Exit the loop when the course is found and
modified
282             }
283         }
284         if (!courseFound) {
285             System.out.println("This '" + cCode + "' Course not found...
");
286         }
287     }
288
289     public void assignProgrammeToCourse(String cCode) {
290         boolean courseFound = false; // Flag to check if the course is
found
291
292         retrieveProgramme();
293
294         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
295             Course currentCourse = courseList.getEntry(i);
296             if (currentCourse.getCourseCode().equals(cCode)) {
297                 courseUI.printCourseDetails(currentCourse);
298
299                 //pmUI.displayAllProgrammes(programList);
300                 pmUI.displayAllProgrammes(programList);
301                 System.out.print("\nChoose Number of Programme to
assign:");
302                 int currentProgramNo;
303
304                 try {
305                     currentProgramNo =
Integer.parseInt(scanner.nextLine());
306                 } catch (NumberFormatException e) {
307                     System.out.println("Invalid input. Please enter a
valid number.");
308                     return;
309                 }
310
311                 if (currentProgramNo < 1 || currentProgramNo >
programList.getNumberOfEntries()) {
312                     MessageUI.displayInvalidChoiceMessage();
313                     return;
```

```
314                 }
315
316                 Programme currentProgramme =
programList.getEntry(currentProgramNo - 1);
317                 if (courseUI.inputConfirmation().equals("y")) {
318                     currentCourse.addProgramme(currentProgramme);
319                     cd.saveToFile(courseList);
320                     System.out.println(" '" +
currentProgramme.getProgrammeName() + "' Programme is added to '" +
currentCourse.getCourseName() + "' Course !");
321                 } else {
322                     System.out.println("Assign programme canceled...");
323                 }
324                 courseFound = true; // Set the flag to true
325                 break; // Exit the loop when the course is found and
modified
326             }
327         }
328         if (!courseFound) {
329             System.out.println("This '" + cCode + "' Course not found...
");
330         }
331     }
332
333     public void removeProgrammeFromCourse(String cCode) {
334         boolean courseFound = false; // Flag to check if the course is
found
335
336         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
337             Course currentCourse = courseList.getEntry(i);
338             if (currentCourse.getCourseCode().equals(cCode)) {
339                 courseUI.printCourseDetails(currentCourse);
340
341                 programList = currentCourse.getProgramme();
342
343                 if (!programList.isEmpty()) {
344                     // display programme list
345                     pmUI.displayAllProgrammes(programList);
346
347                     System.out.print("\nChoose Number of Programme to
remove:");
348                     int currentProgramNo;
349                     try {
```

```
350                         currentProgramNo =
Integer.parseInt(scanner.nextLine());
351                     } catch (NumberFormatException e) {
352                         System.out.println("Invalid input. Please enter a
valid number.");
353                         return;
354                     }
355                     if (currentProgramNo < 1 || currentProgramNo >
programList.getNumberOfEntries()) {
356                         MessageUI.displayInvalidChoiceMessage();
357                         return;
358                     }
359
360                     Programme currentProgramme =
programList.getEntry(currentProgramNo - 1);
361
362                     if (courseUI.inputConfirmation().equals("y")) {
363
364                         // Remove the selected programme
365                         programList.remove(currentProgramme);
366                         cd.saveToFile(courseList);
367                         System.out.println("\n '" +
currentProgramme.getProgrammeName() + "' Programme removed successfully!");
368                     } else {
369                         System.out.println("\nCancel Remove
Programme...");
370                     }
371                 } else {
372                     System.out.println("\nNo programme found for this
course...");
373                 }
374                 courseFound = true;
375                 break; // Exit the loop when the course is found and
modified
376             }
377         }
378
379         if (!courseFound) {
380             System.out.println("This '" + cCode + "' Course not found...
");
381         }
382
383     }
```

```
384
385     //retrieve programme from the file
386     public void retrieveProgramme() {
387
388         try {
389             programList = pd.retrieveFromFile();
390         } catch (ClassNotFoundException ex) {
391             System.out.println("Something Wrong in File");
392         }
393     }
394
395     // choose 1 type of report
396     public void generateReport() {
397         switch (courseUI.inputReportOption()) {
398             case 1:
399                 courseUI.courseDetailsReportTitle();
400                 allCoursesReport();
401                 break;
402             case 2:
403                 courseUI.courseNameReportTitle();
404                 courseNameReport(courseUI.inputCourseKeyword());
405                 break;
406             default:
407                 MessageUI.displayInvalidChoiceMessage();
408                 break;
409
410         }
411     }
412
413     public void allCoursesReport() {
414
415         courseUI.reportsubTitle();
416         int totalCreditHours = 0;
417
418         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
419             Course currentCourse = courseList.getEntry(i);
420
421             programList = currentCourse.getProgramme();
422
423             if (!programList.isEmpty()) {
424                 for (int j = 0; j < programList.getNumberOfEntries(); j++)
{
425                     Programme programme = programList.getEntry(j);
```

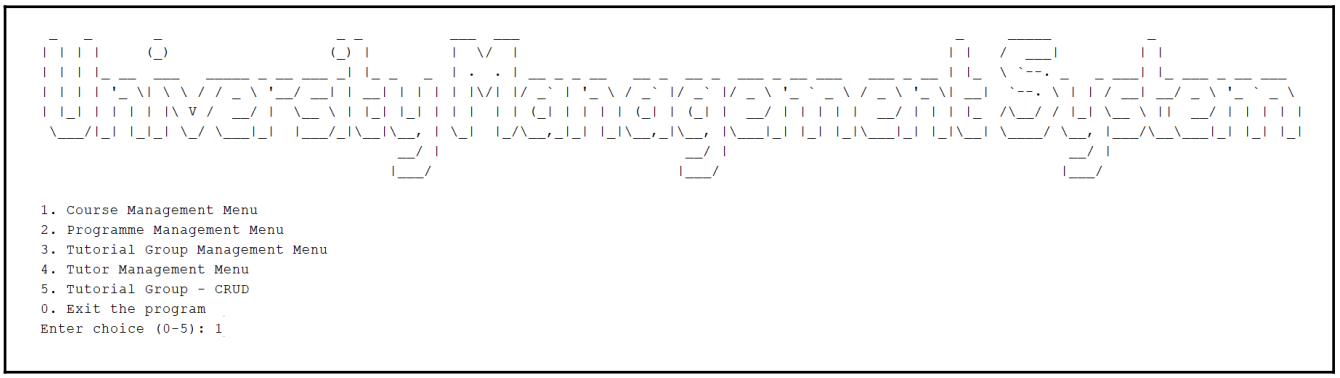
```
426                     System.out.printf("| %-11s | %-30s |     %-4d     |
%-14s | %-30s |      %-7d      |\n",
427                             currentCourse.getCourseCode(),
428                             currentCourse.getCourseName(),
429                             currentCourse.getCreditHours(),
430                             programme.getProgrammeCode(),
431                             programme.getProgrammeName(),
432                             programme.getDurationOfYear()
433                     );
434                     totalCreditHours += currentCourse.getCreditHours();
435
436                 }
437                 courseUI.separateLine();
438
439             } else {
440                 System.out.printf("| %-11s | %-30s |     %-4d     | %-67s
|\n",
441                         currentCourse.getCourseCode(),
442                         currentCourse.getCourseName(),
443                         currentCourse.getCreditHours(),
444                         "No associated Programme");
445                 courseUI.separateLine();
446
447             }
448
449         }
450         System.out.printf("\n\t\t\tTotal Credit Hours(s):   %-12d",
totalCreditHours);
451         System.out.printf("\tTotal Programme(s):   %-12d\n",
getTotalProgramme((SortedArrayList<Course>) courseList));
452
453     }
454
455     public static int getTotalProgramme(SortedArrayList<Course>
courseList) {
456         int totalProgramme = 0;
457
458         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
459             Course currentCourse = courseList.getEntry(i);
460             totalProgramme +=
currentCourse.getProgramme().getNumberOfEntries();
461         }
462
```

```
463         return totalProgramme;
464     }
465
466     public void courseNameReport(String keyword) {
467
468         boolean courseFound = false; // Flag to check if any course
matches the keyword
469         int totalCreditHours = 0;
470
471         for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
472             Course currentCourse = courseList.getEntry(i);
473             String courseName = currentCourse.getCourseName();
474
475             if (courseName.toLowerCase().contains(keyword.toLowerCase()))
{
476                 if (!courseFound) {
477                     courseUI.reportsubTitle();
478
479                     programList = currentCourse.getProgramme();
480
481                     if (!programList.isEmpty()) {
482                         for (int j = 0; j <
programList.getNumberOfEntries(); j++) {
483                             Programme programme = programList.getEntry(j);
484                             System.out.printf("| %-11s | %-30s |     %-4d
| %-14s | %-30s |      %-7d      |\n",
485                                     currentCourse.getCourseCode(),
486                                     courseName,
487                                     currentCourse.getCreditHours(),
488                                     programme.getProgrammeCode(),
489                                     programme.getProgrammeName(),
490                                     programme.getDurationOfYear()
491                             );
492
493                             totalCreditHours +=
currentCourse.getCreditHours();
494                         }
495                         courseUI.separateLine();
496
497                     } else {
498                         System.out.printf("| %-11s | %-30s |     %-4d
| %-67s |\n",
499                                 currentCourse.getCourseCode(),
```

```
500                                 currentCourse.getCourseName(),
501                                 currentCourse.getCreditHours(),
502                                 "No associated Programme");
503                         courseUI.separateLine();
504                     }
505                     courseFound = true;
506                     break;
507                 }
508             }
509         }
510
511         System.out.printf("\n\t\t\tTotal Credit Hours(s):   %-12d",
totalCreditHours);
512         System.out.printf("\tTotal Programme(s):   %-12d\n",
programList.getNumberOfEntries());
513         if (!courseFound) {
514             System.out.println("No courses matching the keyword were
found...");
515         }
516
517     }
518
519     public void displayCourses() {
520
521         //check empty list
522         if (courseList.isEmpty()) {
523             System.out.println("No courses found.");
524         } else {
525             courseUI.courseSubTitle();
526             for (int i = 0; i < courseList.getNumberOfEntries(); i++) {
527                 Course course = courseList.getEntry(i);
528                 System.out.printf("| %2d. | %-11s | %-30s |     %-4d
|\n", i + 1, course.getCourseCode(), course.getCourseName(),
course.getCreditHours());
529
System.out.println("+-------------------------------------------------------
----------+");
530             }
531         }
532
533     }
534
535 }
```

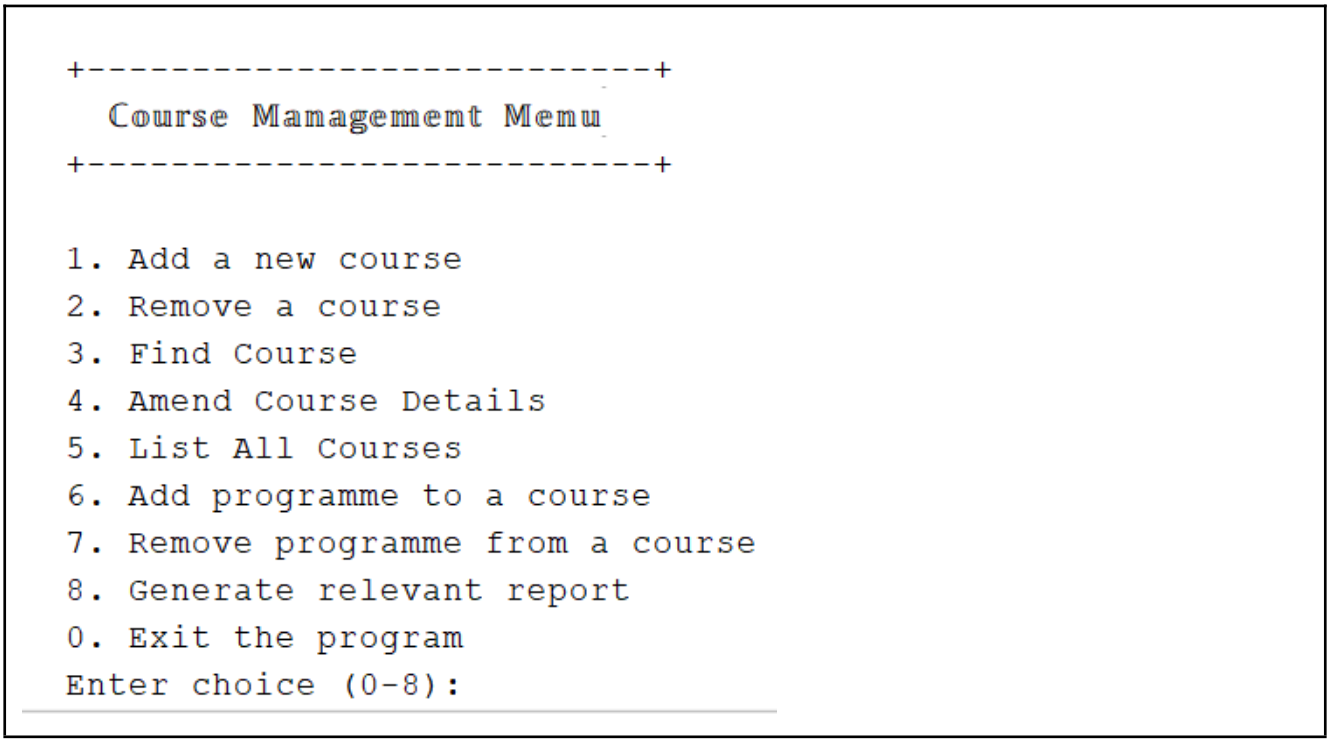
2.1.2 Screenshot of sample user interface

University Management System Main Menu



First, the university management system main menu will be displayed on the screen. Users can input 1 to redirect to the course management menu.

Course Management Menu



After input 1, the course management menu will be displayed on the screen. The menu consists of a list of functions and it is allowed for users to enter a number to select the option.
Otherwise, if users enter letters or negative integers, it will prompt an ‘invalid choice’ message.

1. Add a new Course

```
+---------------------------+
  Course Management Menu
+---------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 1
====================
  Add New Course
====================

Enter Course Code(eg.ABCD1234): abc
Invalid Course Code format, Please follow correct format...

------ Exiting Subsystem --------


====================
  Add New Course
====================

Enter Course Code(eg.ABCD1234): AACS2233
Enter Course Name: Operating System
Enter Credit Hours: d
Invalid input. Please enter a valid positive integer for Credit Hours.
Enter Credit Hours: 2

====================
  Course Details
====================
+-------------------------------------------------------------+
| Course Code | Couse Name                    | Credit Hours |
+-------------------------------------------------------------+
| AACS2233    | Operating System              |     2        |
+-------------------------------------------------------------+
Confirm (y/n)? y

---- Successful! ------
```

Users can choose 1 to add a new course. For adding a new course, it must enter a valid Course Code format with 4 uppercase letters and 4 numbers. When inputting a Course Name, it cannot exceed 40 characters, else it will prompt an error message. When inputting a Credit Hour, it must enter a positive and more than 0 integer, otherwise it will let users reenter. After entering all course details, it will be shown and ask for a confirmation message.If choose ‘y’, users will be confirmed to add a new course, it will show a successful message, else it will return to the course management menu.

2. Remove a Course

```
+--------------------------+
  Course Management Menu
+--------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 2
====================
  Remove a Course
====================

Enter Course Code(eg.ABCD1234): AACS2233

====================
  Course Details
====================
+-------------------------------------------------------------+
| Course Code | Couse Name                      | Credit Hours |
+-------------------------------------------------------------+
| AACS2233    | Operating System                |     3        |
+-------------------------------------------------------------+
Confirm (y/n)? Y

---- Successful! ------
```

Users can choose 2 to remove a course by entering a course code. If they enter an invalid course code, it will prompt an error message ‘Course not found’. Otherwise, if entering a valid course code, it will show the selected course details and then ask for a confirmation.If input ‘y’, it will be confirmed to remove the selected course, else it will not be removed. After that, it will prompt a successful message.

3. Find Course

```
+--------------------------+
  Course Management Menu
+--------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 3
====================
  Find a Course
====================

Enter Course Code(eg.ABCD1234): abcd
This 'abcd' Course not found...

====================
  Find a Course
====================

Enter Course Code(eg.ABCD1234): AACS2233

====================
  Course Details
====================
+-------------------------------------------------------------+
| Course Code | Couse Name                    | Credit Hours |
+-------------------------------------------------------------+
| AACS2233    | Operating System              |     2        |
+-------------------------------------------------------------+
```

Users can choose 3 to find a course by entering a course code.If they enter a course code that does not exist in the file, it will display a ‘Course Not Found’ message. Otherwise, it will display the available course code with course details.

4. Amend Course Details

```
+--------------------------+
  Course Management Menu
+--------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 4
====================
  Modify a Course
====================

Enter Course Code(eg.ABCD1234): AACS2233

====================
  Course Details
====================
+-------------------------------------------------------------+
| Course Code | Couse Name                    | Credit Hours |
+-------------------------------------------------------------+
| AACS2233    | Operating System              |     2        |
+-------------------------------------------------------------+

Select which part you want to modify:
1. Course Name
2. Credit Hours
Choose (1/2): 2
Enter Credit Hours: FSD
Invalid input. Please enter a valid positive integer for Credit Hours.
Enter Credit Hours: 3
Confirm (y/n)? Y

---- Successful! ------
```

Users can choose 4 to amend a course details by entering a course code.Then, it will display the selected course details. It will display a small menu to let users modify course name or credit hour. If they enter an invalid input, it will prompt an error message to reenter.After entering a valid credit hour, it will ask for a confirmation. If they type ‘y’, it will confirm to modify course details and then display a successful message.

5. List All Courses

```
+--------------------------+
  Course Management Menu
+--------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 5
========================
  Display Course List
========================

+-----------------------------------------------------------------------+
| No. | Course Code | Couse Name                      | Credit Hours |
+-----------------------------------------------------------------------+
|  1. | AACD3034    | Data Structure & Algorithms     |     3        |
+-----------------------------------------------------------------------+
|  2. | AACS2233    | Operating System                |     3        |
+-----------------------------------------------------------------------+
|  3. | AACS4014    | Introduction to Data Science    |     3        |
+-----------------------------------------------------------------------+
|  4. | AMIT2033    | Cloud Computing                 |     3        |
+-----------------------------------------------------------------------+
|  5. | BACS2032    | Research Method                 |     3        |
+-----------------------------------------------------------------------+
|  6. | BACS2234    | Human Computer Interaction      |     3        |
+-----------------------------------------------------------------------+
```

Users can choose 5 to display all courses in the course management menu. It will list all courses that have been saved in the file.

6. Add a programme to a course

```
+---------------------------+
  Course Management Menu
+---------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 6
==================================
 Assign a Programme to a Course
==================================

Enter Course Code(eg.ABCD1234): AACS2233

====================
  Course Details
====================
+-------------------------------------------------------------+
| Course Code | Couse Name                      | Credit Hours |
+-------------------------------------------------------------+
| AACS2233    | Operating System                |      3       |
+-------------------------------------------------------------+
+-----+-----------------+---------------------------------+------------+
| No. | Programme Code  | Programme Name                  | Duration   |
+-----+-----------------+---------------------------------+------------+
|  1. | DCS             | Degree in Computer Science      |      3     |
+-----+-----------------+---------------------------------+------------+
|  2. | DHM             | Degree in Hotel Management      |      3     |
+-----+-----------------+---------------------------------+------------+
|  3. | DIM             | Degree in Marketing             |      3     |
+-----+-----------------+---------------------------------+------------+
|  4. | DIN             | Degree in Networking            |      3     |
+-----+-----------------+---------------------------------+------------+

Choose Number of Programme to assign:1
Confirm (y/n)? y
 'Degree in Computer Science' Programme is added to 'Operating System' Course !
```

Users can choose 6 to add a programme to a course. First, users must enter a valid available course code and then it will display the selected course details and a list of available programmes. After that, they can choose a valid number in the programme list that they want to assign to the selected course. It will ask for a confirmation.If inputting ‘y’, it will confirm to add a programme to the selected course.As a result, the successful message will be shown as they choose 1 that is ‘Degree in Computer Science’ to add to the ‘Operating System’ Course.

7. Remove a programme from a course

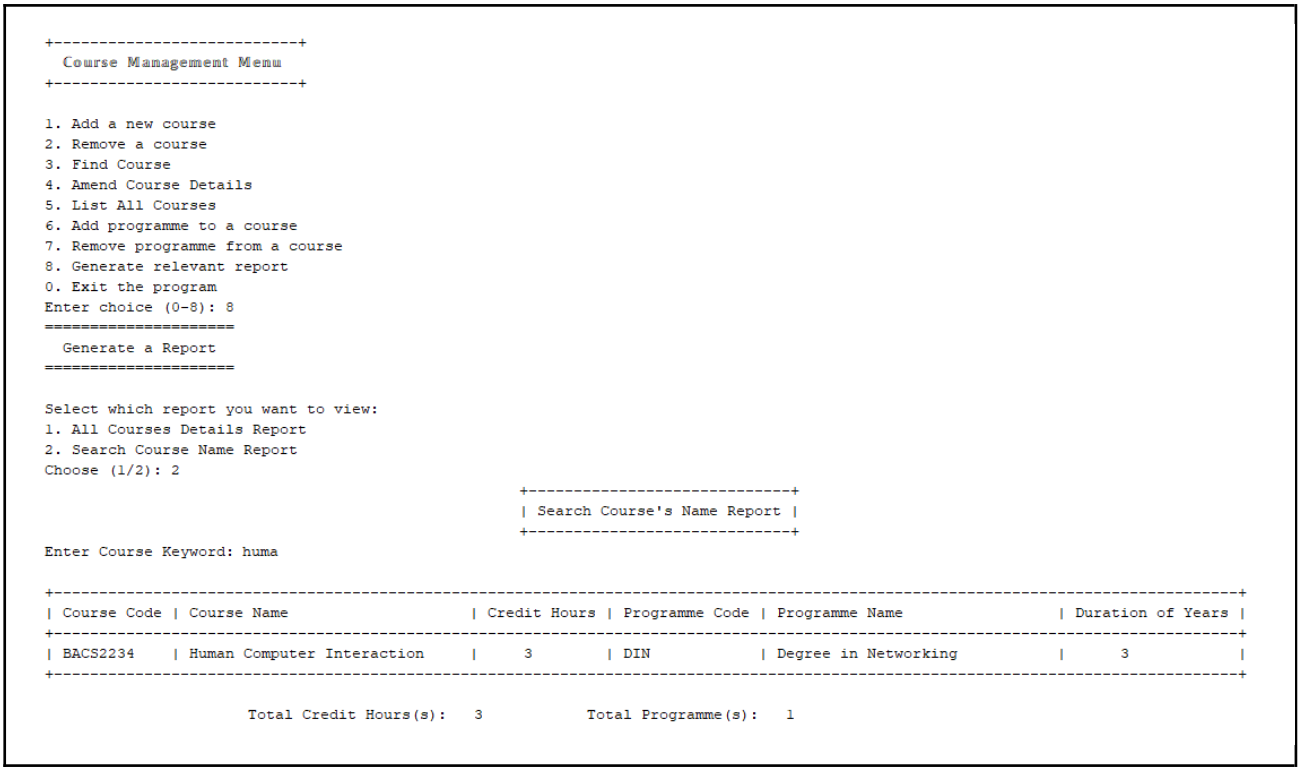
```
+---------------------------+
  Course Management Menu
+---------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 7
====================================
 Remove a Programme from a Course
====================================

Enter Course Code(eg.ABCD1234): AACS2233

====================
  Course Details
====================
+-------------------------------------------------------------------+
| Course Code | Couse Name                      | Credit Hours |
+-------------------------------------------------------------------+
| AACS2233    | Operating System                |     3        |
+-------------------------------------------------------------------+
+-----+-----------------+---------------------------------+------------+
| No. | Programme Code  | Programme Name                  | Duration   |
+-----+-----------------+---------------------------------+------------+
|  1. | DCS             | Degree in Computer Science      |      3     |
+-----+-----------------+---------------------------------+------------+

Choose Number of Programme to remove:1
Confirm (y/n)? y

 'Degree in Computer Science' Programme removed successfully!
```

Users can choose 7 to remove a programme from a course. First, users must enter a valid available course code and then it will check whether the programme has been assigned to the course or not. If yes, it will show a list of programmes that have been assigned to the selected course and then they must enter a valid number to remove the programme from the course.It will ask for a confirmation. As they input ‘y’, it will be confirmed and prompt a successful message that is ‘Degree in Computer Science Programme removed successfully’. If the selected course has not been assigned any programme, it will prompt a ‘No Programme found in this course’ message.

8. Generate relevant report

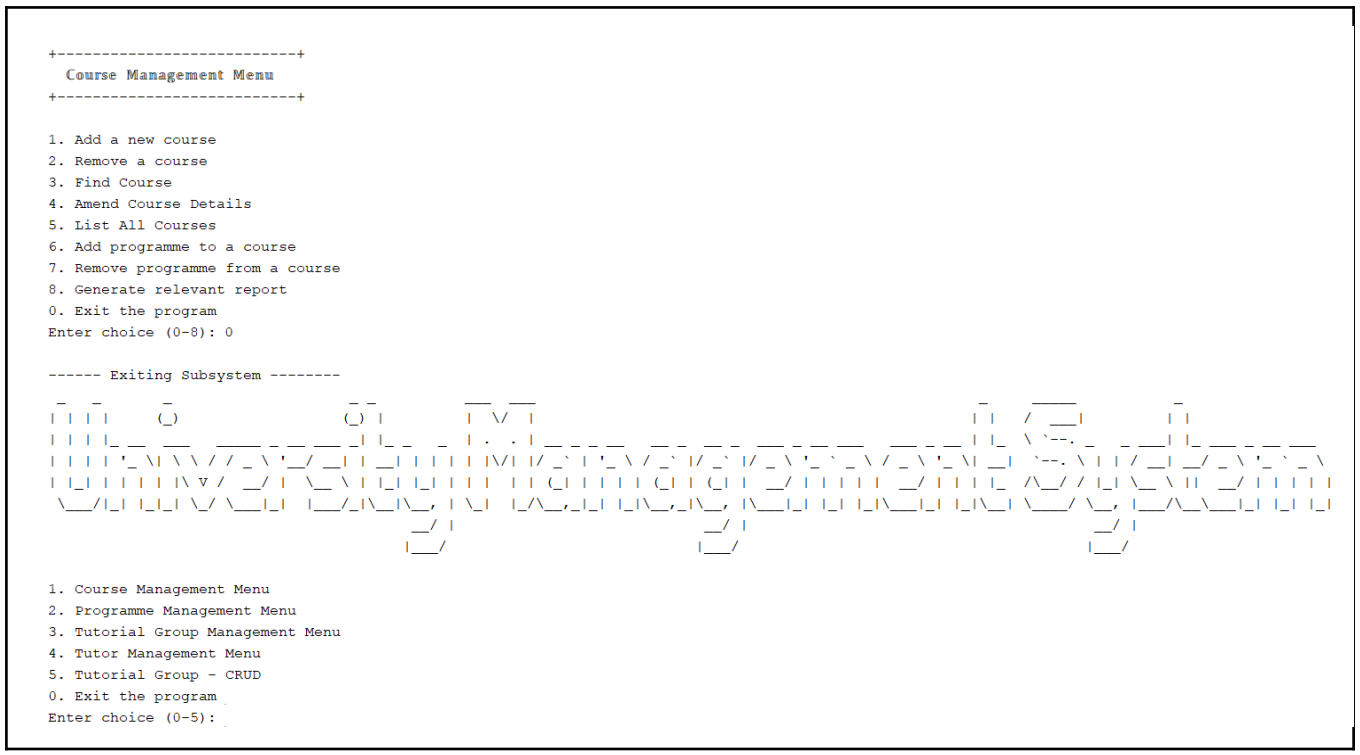
```
+---------------------------+
  Course Management Menu
+---------------------------+

1. Add a new course
2. Remove a course
3. Find Course
4. Amend Course Details
5. List All Courses
6. Add programme to a course
7. Remove programme from a course
8. Generate relevant report
0. Exit the program
Enter choice (0-8): 8
====================
  Generate a Report
====================

Select which report you want to view:
1. All Courses Details Report
2. Search Course Name Report
Choose (1/2): 1
                                              +-----------------------+
                                              |Courses Details Report |
                                              +-----------------------+

+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+
| Course Code | Course Name               | Credit Hours | Programme Code | Programme Name            | Duration of Years |
+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+
| AACD3034    | Data Structure & Algorithms |     3        | DCS            | Degree in Computer Science |      3          |
+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+
| AACS2233    | Operating System          |     3        | No associated Programme                                              |
+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+
| AACS4014    | Introduction to Data Science |     3        | No associated Programme                                              |
+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+
| AMIT2033    | Cloud Computing           |     3        | No associated Programme                                              |
+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+
| BACS2032    | Research Method           |     3        | No associated Programme                                              |
+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+
| BACS2234    | Human Computer Interaction |     3        | DIN            | Degree in Networking      |      3          |
+-------------+---------------------------+--------------+--------------+---------------------------+-----------------+

                    Total Credit Hours(s):   6          Total Programme(s):   2
```

Users can choose 8 to generate relevant reports. First, it will show a small menu which has 2 types of report including all course details report and search course name report. If they enter option 1, the report will display a list of all the courses with the associated programmes. If the course has not been assigned any programme, it will display ‘no associated programme’. It will also show the total credit hours and total programmes.



If users select option 2 , it allows users to generate a report by searching a course name. They can enter the course keyword and then it will display the report of course name that matches with the keyword they type.For example, if they type ‘huma’, it will view the ‘Human Computer Interaction’ Course report with the associated programme, total credit hour and total programme.

9. Exit Course Management Menu



Users can choose 0 to exit the Course Management Menu and then return to the main menu. If they want to exit the system, they will input 0 in the main menu.

2.2 Programme Management

Student Name	Student ID	Prog / Tut.Grp	Signature
Lee Weng Yi	2309332	RSD2G3	

2.2.1 Source Code

ProgrammeManagementUI.java

```
 1 package boundary;
 2
 3 import adt.*;
 4 import entity.*;
 5 import java.util.Scanner;
 6
 7 /**
 8  *
 9  * @author Lee Weng Yi
10  */
11 public class ProgrammeManagementUI {
12
13     private Scanner scanner = new Scanner(System.in);
14
15     public int getProgrammeMenu() {
16
System.out.println("\n\n\n\n\n+----------------------------------------+"
);
17         System.out.println("|         Programme Management Menu
|");
18
System.out.println("+----------------------------------------+\n");
19         System.out.println("1. Add a new programmes");
20         System.out.println("2. Remove a programme");
21         System.out.println("3. Find programme");
22         System.out.println("4. Amend programme details");
23         System.out.println("5. List all programme");
24         System.out.println("6. Add tutorial group to a programme");
25         System.out.println("7. Remove tutorial group from a
programme");
26         System.out.println("8. List all tutorial groups for a
programme");
27         System.out.println("9. Generate relevant report");
28         System.out.println("0. Exit programme management");
29         int choice;
30         while (true) {
31             System.out.print("Enter choice (0-9): ");
32             if (scanner.hasNextInt()) {
33                 choice = scanner.nextInt();
34                 scanner.nextLine();
35                 return choice;
36             } else {
37                 scanner.nextLine();
38                 System.out.println("Invalid input. Please enter an
integer.");
39             }
```

```
 40         }
 41     }
 42
 43     public String inputProgrammeCode(SortedListInterface<Programme>
programmeList) {
 44         String code;
 45         while (true) {
 46             System.out.print("Enter Programme Code (3 letters): ");
 47             code = scanner.nextLine().toUpperCase();
 48             if (!code.isEmpty()) {
 49                 if (code.length() == 3) {
 50                     boolean duplicate = false;
 51                     for (int i = 0; i <
programmeList.getNumberOfEntries(); i++) {
 52                         Programme currentProgramme =
programmeList.getEntry(i);
 53                         if
(currentProgramme.getProgrammeCode().equals(code)) {
 54                             System.out.println("Programme code is
duplicated in programme list.\n\n");
 55                             duplicate = true;
 56                             break;
 57                         }
 58                     }
 59                     if (!duplicate) {
 60                         return code;
 61                     }
 62                 } else {
 63                     System.out.println("Programme code must be 3
letters");
 64                 }
 65             } else {
 66                 System.out.println("Programme code cannot be empty.");
 67             }
 68         }
 69     }
 70
 71     public String inputProgrammeCode() {
 72         System.out.print("Enter Programme Code: ");
 73         String code = scanner.nextLine().toUpperCase();
 74         return code;
 75     }
 76
 77     public String inputProgrammeName(SortedListInterface<Programme>
programmeList) {
 78         String name;
```

```
 79         while (true) {
 80             System.out.print("Enter Programme Name (max 30 characters):
");
 81             name = scanner.nextLine().trim();
 82             if (!name.isEmpty()) {
 83                 if (name.length() <= 30) {
 84                     boolean duplicate = false;
 85                     for (int i = 0; i <
programmeList.getNumberOfEntries(); i++) {
 86                         Programme currentProgramme =
programmeList.getEntry(i);
 87                         if
(currentProgramme.getProgrammeName().equals(name)) {
 88                             System.out.println("Programme name is
duplicated in programme list.\n\n");
 89                             duplicate = true;
 90                             break;
 91                         }
 92                     }
 93                     if (!duplicate) {
 94                         return name;
 95                     }
 96                 } else {
 97                     System.out.println("Please enter a name with 30
characters or fewer.\n\n");
 98                 }
 99             } else {
100                 System.out.println("Invalid name, Programme name cannot
be empty.");
101             }
102         }
103     }
104
105     public int inputDurationOfYear() {
106         int duration;
107         while (true) {
108             System.out.print("Enter Duration of Year: ");
109             if (scanner.hasNextInt()) {
110                 duration = scanner.nextInt();
111                 if (duration == 0) {
112                     System.out.println("Duration of year cannot be
0.");
113                 } else {
114                     return duration;
115                 }
116             } else {
```

```
117                 scanner.nextLine();
118                 System.out.println("Invalid duration. Please enter an
integer.");
119             }
120         }
121     }
122
123     public Programme
inputProgrammeDetails(SortedListInterface<Programme> programmeList) {
124         String programmeCode = inputProgrammeCode(programmeList);
125         String programmeName = inputProgrammeName(programmeList);
126         int duration = inputDurationOfYear();
127         return new Programme(programmeCode, programmeName, duration);
128     }
129
130     public int getModifySelection() {
131
132         System.out.println("\nModify Programme Detail\n");
133
134         System.out.println("1. Programme Name");
135         System.out.println("2. Duration of Year");
136         System.out.println("0. Done");
137         int choice;
138         while (true) {
139             System.out.print("Enter choice (0-2): ");
140             if (scanner.hasNextInt()) {
141                 choice = scanner.nextInt();
142                 scanner.nextLine();
143                 return choice;
144             } else {
145                 scanner.nextLine();
146                 System.out.println("\nInvalid input. Please enter an
integer.");
147             }
148         }
149     }
150
151     public String inputConfirmation(String question, String action) {
152         String confirm;
153         while (true) {
154             System.out.print(question + " " + action + "? (Y/N): ");
155             confirm = scanner.next().toUpperCase();
156             scanner.nextLine();
157             if (confirm.equals("Y") || confirm.equals("N")) {
158                 return confirm;
159             } else {
```

```
160                 System.out.println("Invalid input. Please enter 'Y' or
'N'.");
161             }
162         }
163     }
164
165     public int getSelectedGroup(SortedListInterface<TutorialGroup>
group) {
166         int selectedGroupNumber = 0;
167         displayGroup(group);
168
169         while (true) {
170             System.out.print("Enter the number of the tutorial group:
");
171
172             if (scanner.hasNextInt()) {
173                 selectedGroupNumber = scanner.nextInt();
174                 scanner.nextLine();
175
176                 if (selectedGroupNumber >= 1 && selectedGroupNumber <=
group.getNumberOfEntries()) {
177                     break;
178                 } else {
179                     System.out.println("Invalid selection, please try
again.");
180                 }
181             } else {
182                 scanner.nextLine();
183                 System.out.println("Invalid input. Please enter an
integer.");
184             }
185         }
186
187         return selectedGroupNumber;
188     }
189
190     public void displayActionTitle(int action) {
191         if (action != 0) {
192
System.out.println("\n\n+----------------------------------------+");
193         }
194         switch (action) {
195             case 1:
196                 System.out.println("|            Add a new programme
|");
197                 break;
```

```
198             case 2:
199                 System.out.println("|            Remove a programme
|");
200                 break;
201             case 3:
202                 System.out.println("|              Find programme
|");
203                 break;
204             case 4:
205                 System.out.println("|          Amend programme details
|");
206                 break;
207             case 5:
208                 System.out.println("|             List all programme
|");
209                 break;
210             case 6:
211                 System.out.println("|    Add tutorial group to a
programme     |");
212                 break;
213             case 7:
214                 System.out.println("|  Remove tutorial group from a
programme  |");
215                 break;
216             case 8:
217                 System.out.println("| List all tutorial groups for a
programme |");
218                 break;
219             case 9:
220                 System.out.println("|                  Report
|");
221                 break;
222         }
223         if (action != 0) {
224
System.out.println("+----------------------------------------+\n");
225         }
226     }
227
228     public void displayProgramme(Programme programme) {
229
System.out.println("\n\n\n+-----------------+------------------------------
--+------------+");
230         System.out.printf("| %-15s | %-30s | %-10s |\n", "Programme
Code", "Programme Name", "Duration");
231
```

```
System.out.println("+-----------------+-------------------------------+---
---------+");
232         System.out.printf("| %-15s | %-30s | %6d     |\n",
programme.getProgrammeCode(), programme.getProgrammeName(),
programme.getDurationOfYear());
233
System.out.println("+-----------------+-------------------------------+---
---------+");
234     }
235
236     public void displayAllProgrammes(SortedListInterface<Programme>
programmeList) {
237
System.out.println("+-----+-----------------+-------------------------------
--+------------+");
238         System.out.printf("| %3s | %-15s | %-30s | %-10s |\n", "No.",
"Programme Code", "Programme Name", "Duration");
239
System.out.println("+-----+-----------------+-------------------------------
--+------------+");
240         if (!programmeList.isEmpty()) {
241             for (int i = 0; i < programmeList.getNumberOfEntries();
i++) {
242                 Programme currentProgramme = programmeList.getEntry(i);
243                 System.out.printf("| %2d. | %-15s | %-30s | %6d
|\n", i + 1, currentProgramme.getProgrammeCode(),
currentProgramme.getProgrammeName(), currentProgramme.getDurationOfYear());
244
System.out.println("+-----+-----------------+-------------------------------
--+------------+");
245             }
246         } else {
247             System.out.printf("| %-67s |\n", "Programme list is
empty");
248
System.out.println("+-----+-----------------+-------------------------------
--+------------+");
249         }
250
251     }
252
253     public void displayMessage(String message) {
254         switch (message) {
255             case "addSuccess":
256                 System.out.println("Programme successfully added.");
257                 break;
```

```
258             case "addCancel":
259                 System.out.println("Addition cancelled.");
260                 break;
261             case "modifySuccess":
262                 System.out.println("Programme successfully modified.");
263                 break;
264             case "modifyCancel":
265                 System.out.println("Modification cancelled");
266                 break;
267             case "deleteSuccess":
268                 System.out.println("Programme successfully deleted.");
269                 break;
270             case "deleteCancel":
271                 System.out.println("Deletion cancelled.");
272                 break;
273             case "unableDelete":
274                 System.out.println("Cannot delete the programme"
275                         + " because it has associated tutorial groups.
");
276                 break;
277
278         }
279     }
280
281     public void displayGroup(SortedListInterface<TutorialGroup> group)
{
282         if (!group.isEmpty()) {
283
System.out.println("+-----+-------------------------------------------------
---------+");
284             System.out.printf("| %3s | %-55s |\n", "No.", "Tutorial
Group");
285
System.out.println("+-----+-------------------------------------------------
---------+");
286             for (int i = 0; i < group.getNumberOfEntries(); i++) {
287                 TutorialGroup currentGroup = group.getEntry(i);
288                 System.out.printf("| %2d. | %-55s |\n", i + 1,
currentGroup.getTgName());
289
System.out.println("+-----+-------------------------------------------------
---------+");
290             }
291         } else {
292
System.out.println("+-----------------------------------------------------
```

```
---------+");
293             System.out.println("|             No tutorial group in this
programme               |");
294
System.out.println("+-----------------------------------------------------
---------+");
295         }
296     }
297
298     public void displayReport(SortedListInterface<Programme>
programmeList, ListInterface<Student> studentList) {
299
System.out.println("+-----+-------------------------------+---------------
--+-----------------+");
300         System.out.printf("| %3s | %-30s | %-15s |      %-10s |\n",
"No.", "Programme", "Tutorial Group", "Student");
301
System.out.println("+-----+-------------------------------+---------------
--+-----------------+");
302         for (int i = 0; i < programmeList.getNumberOfEntries(); i++) {
303             Programme currentProgramme = programmeList.getEntry(i);
304             SortedListInterface<TutorialGroup> programGroups =
currentProgramme.getTutorialGroup();
305
306             int numberStudent = 0;
307             for (int j = 1; j <= studentList.getNumberOfEntries(); j++)
{
308                 Student currentStudent = studentList.getEntry(j);
309                 if
(currentStudent.getProgramme().equals(currentProgramme.getProgrammeName()))
{
310                     ++numberStudent;
311                 }
312             }
313             System.out.printf("| %2d. | %-30s |  %8d       |  %8d
|\n", i + 1, currentProgramme.getProgrammeName(),
programGroups.getNumberOfEntries(), numberStudent);
314
System.out.println("+-----+-------------------------------+---------------
--+-----------------+");
315         }
316     }
317
318     public void displayNotFound(String programmeCode) {
319         System.out.println("Programmme " + programmeCode + " not
found.");
```

```
320     }
321
322     public void displayGroupMessage(String action, String group, String
programme) {
323         switch (action) {
324             case "add":
325                 System.out.println("Tutorial group " + group + " added
to programme " + programme + ".");
326                 break;
327             case "remove":
328                 System.out.println("Tutorial group " + group + "
removed from programme " + programme + ".");
329                 break;
330         }
331     }
332
333     public void displayAllAssociated() {
334         System.out.println("All tutorial groups are already associated
with programme.");
335     }
336 }
```

ProgrammeManagement.java

```
 1 package control;
 2
 3 import adt.*;
 4 import dao.*;
 5 import entity.*;
 6 import boundary.ProgrammeManagementUI;
 7 import utility.*;
 8
 9 /**
10  *
11  * @author Lee Weng Yi
12  */
13 public class ProgrammeManagement {
14
15     private SortedListInterface<Programme> programmeList = new
SortedArrayList<>();
16     private SortedListInterface<TutorialGroup> groupList = new
SortedArrayList<>();
17     private final ProgrammeDAO pd = new ProgrammeDAO();
18     private ProgrammeManagementUI programmeUI = new
ProgrammeManagementUI();
19     private TutorialGroupManagement tgm = new
TutorialGroupManagement();
20     private final TutorialGroupDAO tgd = new TutorialGroupDAO();
21     private ListInterface<Student> studentList = new ArrayList<>();
22     private final StudentDAO sd = new StudentDAO();
23
24     public ProgrammeManagement() {
25         try {
26             programmeList = pd.retrieveFromFile();
27         } catch (ClassNotFoundException ex) {
28             System.out.println("Something Wrong in File");
29         }
30     }
31
32     public void runProgrammeManagement() {
33         int choice = 0;
34         do {
35             choice = programmeUI.getProgrammeMenu();
36             programmeUI.displayActionTitle(choice);
37             switch (choice) {
38                 case 1:
39                     addNewProgramme();
40                     programmeUI.displayAllProgrammes(programmeList);
41                     break;
42                 case 2:
```

```
 43
deleteProgrammeByCode(programmeUI.inputProgrammeCode());
 44                     break;
 45                 case 3:
 46
searchProgrammeByCode(programmeUI.inputProgrammeCode());
 47                     break;
 48                 case 4:
 49
modifyProgrammeByCode(programmeUI.inputProgrammeCode());
 50                     break;
 51                 case 5:
 52                     programmeUI.displayAllProgrammes(programmeList);
 53                     break;
 54                 case 6:
 55
addTutorialGroupToProgramme(programmeUI.inputProgrammeCode());
 56                     break;
 57                 case 7:
 58
removeTutorialGroupFromProgramme(programmeUI.inputProgrammeCode());
 59                     break;
 60                 case 8:
 61
listTutorialGroupsForProgramme(programmeUI.inputProgrammeCode());
 62                     break;
 63                 case 9:
 64                     generateReport();
 65                     break;
 66                 case 0:
 67                     MessageUI.displayExitSubsystemMessage();
 68                     break;
 69                 default:
 70                     MessageUI.displayInvalidChoiceMessage();
 71             }
 72         } while (choice != 0);
 73     }
 74
 75     public void addNewProgramme() {
 76         Programme newProgramme =
programmeUI.inputProgrammeDetails(programmeList);
 77         programmeUI.displayProgramme(newProgramme);
 78
 79         if (!programmeUI.inputConfirmation("Confirm",
"add").equals("Y")) {
 80             programmeUI.displayMessage("addCancel");
```

```
 81          } else {
 82              programmeList.add(newProgramme);
 83              pd.saveToFile(programmeList);
 84              programmeUI.displayMessage("addSuccess");
 85              if (programmeUI.inputConfirmation("Continue",
"add").equals("Y")) {
 86                  addNewProgramme();
 87              }
 88          }
 89          runProgrammeManagement();
 90      }
 91
 92      public boolean searchProgrammeByCode(String programCode) {
 93
 94          Programme searchKey = new Programme(programCode);
 95          int position = programmeList.getPosition(searchKey);
 96
 97          if (position > 0) {
 98              Programme foundProgramme = programmeList.getEntry(position
- 1);
 99              programmeUI.displayProgramme(foundProgramme);
100              return true;
101          } else {
102              programmeUI.displayNotFound(programCode);
103              return false;
104          }
105      }
106
107      public void deleteProgrammeByCode(String programmeCode) {
108          if (searchProgrammeByCode(programmeCode)) {
109              Programme selectedProgram =
programmeList.getEntry(programmeList.getPosition(new
Programme(programmeCode)) - 1);
110              if (!programmeUI.inputConfirmation("Confirm",
"delete").equals("Y")) {
111                  programmeUI.displayMessage("deleteCancel");
112              } else {
113                  if (!selectedProgram.getTutorialGroup().isEmpty()) {
114                      programmeUI.displayMessage("unableDelete");
115                      return;
116                  }
117                  Programme searchKey = new Programme(programmeCode);
118                  programmeList.remove(searchKey);
119                  pd.saveToFile(programmeList);
120                  programmeUI.displayMessage("deleteSuccess");
121              }
```

```
122         }
123     }
124
125     public void modifyProgrammeByCode(String programmeCode) {
126         if (searchProgrammeByCode(programmeCode)) {
127             Programme selectedProgram =
programmeList.getEntry(programmeList.getPosition(new
Programme(programmeCode)) - 1);
128             //get Modify Selection
129             int choice;
130             String modifiedProgramName =
selectedProgram.getProgrammeName();
131             int modifiedDuration = selectedProgram.getDurationOfYear();
132             // Display the modified details
133             Programme modifiedProgramme = new
Programme(selectedProgram.getProgrammeCode(), modifiedProgramName,
modifiedDuration);
134             do {
135                 choice = programmeUI.getModifySelection();
136                 switch (choice) {
137                     case 1:
138                         modifiedProgramName =
programmeUI.inputProgrammeName(programmeList);
139
modifiedProgramme.setProgrammeName(modifiedProgramName);
140
programmeUI.displayProgramme(modifiedProgramme);
141                         break;
142                     case 2:
143                         modifiedDuration =
programmeUI.inputDurationOfYear();
144
modifiedProgramme.setDurationOfYear(modifiedDuration);
145
programmeUI.displayProgramme(modifiedProgramme);
146                         break;
147                     case 0:
148                         break;
149                 }
150             } while (choice != 0);
151             // Display the modified details
152             programmeUI.displayProgramme(modifiedProgramme);
153             if (!programmeUI.inputConfirmation("Confirm",
"modify").equals("Y")) {
154                 programmeUI.displayMessage("modifyCancel");
155             } else {
```

```
156                 // Replace the existing program with the modified
program
157                 selectedProgram.setProgrammeName(modifiedProgramName);
158                 selectedProgram.setDurationOfYear(modifiedDuration);
159                 pd.saveToFile(programmeList);
160                 programmeUI.displayMessage("modifySuccess");
161             }
162         }
163     }
164
165     public void addTutorialGroupToProgramme(String programmeCode) {
166
167         if (searchProgrammeByCode(programmeCode)) {
168             Programme selectedProgramme =
programmeList.getEntry(programmeList.getPosition(new
Programme(programmeCode)) - 1);
169
170             // Load all tutorial groups
171             loadTutorialGroups();
172
173             // Get the tutorial groups that haven't been associated
with the selected program
174             SortedListInterface<TutorialGroup> unassociatedGroups =
getUnassociatedTutorialGroup();
175             // Display unassociated tutorial groups
176             if (!unassociatedGroups.isEmpty()) {
177                 int selectedGroupNumber =
programmeUI.getSelectedGroup(unassociatedGroups);
178                 TutorialGroup selectedGroup =
unassociatedGroups.getEntry(selectedGroupNumber - 1);
179                 if (!programmeUI.inputConfirmation("Confirm",
"add").equals("Y")) {
180                     programmeUI.displayMessage("addCancel");
181                 } else {
182                     selectedProgramme.addTutorialGroup(selectedGroup);
183                     pd.saveToFile(programmeList);
184                     programmeUI.displayGroupMessage("add",
selectedGroup.getTgName(), selectedProgramme.getProgrammeName());
185                 }
186             } else {
187                 programmeUI.displayAllAssociated();
188             }
189         }
190     }
191
192     public void loadTutorialGroups() {
```

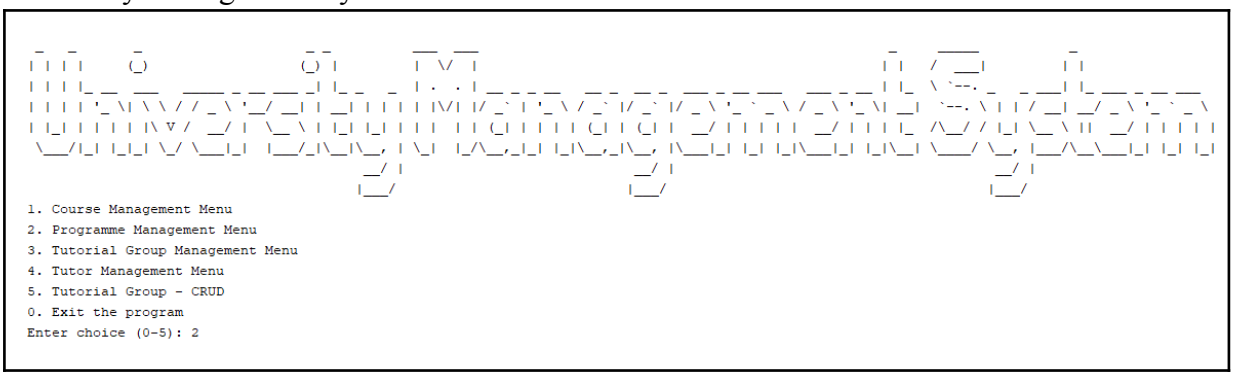
```
193         try {
194             // Retrieve tutorial groups from the file
195             groupList = tgd.retrieveFromFile();
196         } catch (ClassNotFoundException ex) {
197             System.out.println("\nClass not found while loading
tutorial groups.");
198         }
199     }
200
201     public void loadStudent() {
202         try {
203             // Retrieve tutorial groups from the file
204             studentList = sd.retrieveFromFile();
205         } catch (ClassNotFoundException ex) {
206             System.out.println("\nClass not found while loading
tutorial groups.");
207         }
208     }
209
210     public void removeTutorialGroupFromProgramme(String programmeCode)
{
211         if (searchProgrammeByCode(programmeCode)) {
212             Programme selectedProgram =
programmeList.getEntry(programmeList.getPosition(new
Programme(programmeCode)) - 1);
213             SortedListInterface<TutorialGroup> tgList =
selectedProgram.getTutorialGroup();
214
215             if (!tgList.isEmpty()) {
216                 int selectedGroupNumber =
programmeUI.getSelectedGroup(tgList);
217                 TutorialGroup selectedGroup =
tgList.getEntry(selectedGroupNumber - 1);
218
219                 if (programmeUI.inputConfirmation("Confirm",
"delete").equals("Y")) {
220                     tgList.remove(selectedGroup);
221                     pd.saveToFile(programmeList);
222                     programmeUI.displayGroupMessage("remove",
selectedGroup.getTgName(), selectedProgram.getProgrammeName());
223                 }
224             } else {
225                 programmeUI.displayGroup(tgList);
226             }
227         }
228     }
```

```
229
230     public void listTutorialGroupsForProgramme(String programmeCode) {
231         if (searchProgrammeByCode(programmeCode)) {
232             Programme selectedProgram =
programmeList.getEntry(programmeList.getPosition(new
Programme(programmeCode)) - 1);
233             SortedListInterface<TutorialGroup> tgList =
selectedProgram.getTutorialGroup();
234             programmeUI.displayGroup(tgList);
235         }
236     }
237
238     public SortedListInterface<TutorialGroup>
getUnassociatedTutorialGroup() {
239         SortedListInterface<TutorialGroup> associatedGroup = new
SortedArrayList<>();
240         SortedListInterface<TutorialGroup> unassociatedGroup = new
SortedArrayList<>();
241         // Get associated tutorial groups
242         for (int i = 0; i < programmeList.getNumberOfEntries(); i++) {
243             Programme selectedProgram = programmeList.getEntry(i);
244             SortedListInterface<TutorialGroup> programGroups =
selectedProgram.getTutorialGroup();
245             if (!programGroups.isEmpty()) {
246                 for (int j = 0; j < programGroups.getNumberOfEntries();
j++) {
247                     TutorialGroup group = programGroups.getEntry(j);
248                     associatedGroup.add(group);
249                 }
250             }
251         }
252         // Compare All groups and associated groups to get Unassociated
253         for (int i = 0; i < groupList.getNumberOfEntries(); i++) {
254             TutorialGroup currentTutorialGroup = groupList.getEntry(i);
255             boolean isAssociated = false;
256             for (int j = 0; j < associatedGroup.getNumberOfEntries();
j++) {
257                 TutorialGroup currentAssociatedGroup =
associatedGroup.getEntry(j);
258                 if
(currentTutorialGroup.getTgCode().equals(currentAssociatedGroup.getTgCode()
)) {
259                     isAssociated = true;
260                     break;
261                 }
262             }
```

```
263             if (!isAssociated) {
264                 unassociatedGroup.add(currentTutorialGroup);
265             }
266         }
267         return unassociatedGroup;
268     }
269
270     public void generateReport() {
271         loadStudent();
272         programmeUI.displayReport(programmeList, studentList);
273     }
274
275 }
```

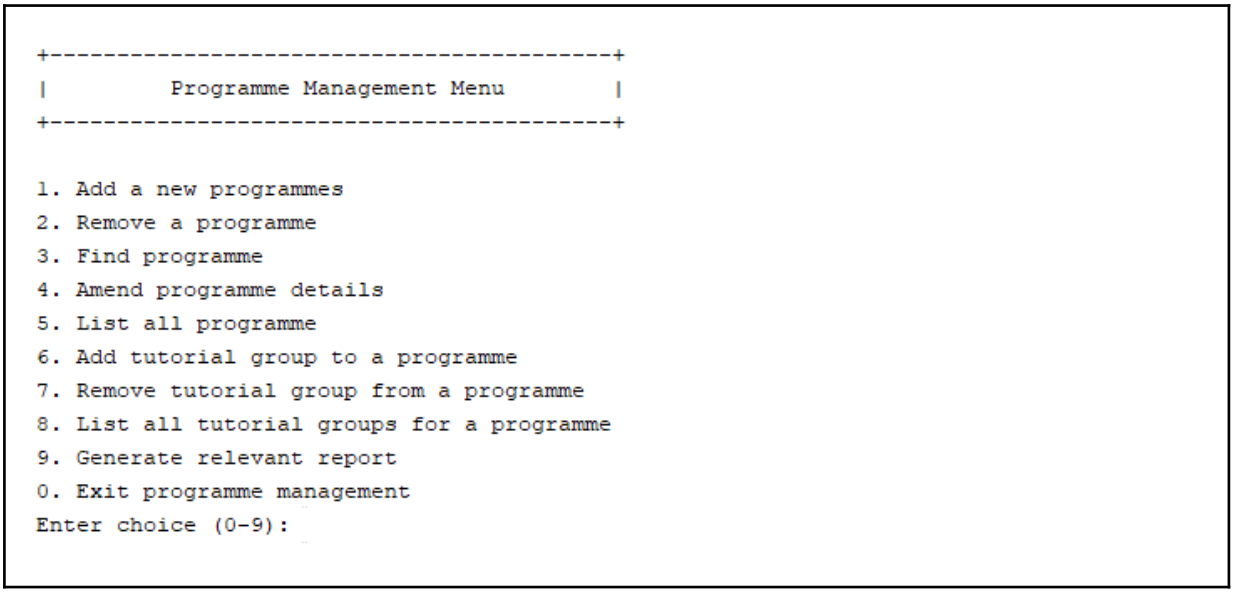
2.2.2 Screenshot of sample user interface

University Management System Main Menu



User entered 2 in the main menu will redirect the user to the programme management menu.

Programme Management Menu



A list of actions can be done in the programme management menu and users can select which function they want to proceed with. It only accepts user input integers.

1. Add a new programmes

```
+-----------------------------------------+
|            Add a new programme          |
+-----------------------------------------+

Enter Programme Code (3 letters): A
Programme code must be 3 letters
Enter Programme Code (3 letters): DIS
Enter Programme Name (max 30 characters): Diploma In Science
Enter Duration of Year: a
Invalid duration. Please enter an integer.
Enter Duration of Year: 3



+-----------------+---------------------------------+------------+
| Programme Code  | Programme Name                  | Duration   |
+-----------------+---------------------------------+------------+
| DIS             | Diploma In Science              |      3     |
+-----------------+---------------------------------+------------+
Confirm add? (Y/N): y
Programme successfully added.
Continue add? (Y/N): n
```

After user input 1 from the programme management menu, it will redirect users to “Add a new programme”. It will prompt the user to enter Program Code, if the program code entered is duplicated or the length of the program code is not 3, it will prompt the user to enter again. If a valid programme code is entered, it will prompt the user to enter the program name, the length of the program name must be less than 30. The system will prompt the user to enter the duration of the year, the input must be an integer. After that, it will display the user’s input and require the user’s confirmation to proceed. If the user confirms to add, it will prompt “Programme successfully added”. Users can also continue adding after one.

2. Remove a programme

After user input 2 in the programme management menu, it will redirect the user to “Remove a programme”

```
+----------------------------------------+
|            Remove a programme          |
+----------------------------------------+

Enter Programme Code: DIN



+-----------------+--------------------------------+------------+
| Programme Code  | Programme Name                 | Duration   |
+-----------------+--------------------------------+------------+
| DIN             | Degree in Networking           |      3     |
+-----------------+--------------------------------+------------+
Confirm delete? (Y/N): y
Cannot delete the programme because it has associated tutorial groups.
```

It will prompt the user to enter programme code, if programme code is found, it will display the found programme details for the user. The system will request confirmation from the user before proceeding to delete. If the programme contains a tutorial group, it will prompt a message “Cannot delete the programme because it has associated tutorial groups” to inform users the programme is unable to delete.

```
+----------------------------------------+
|            Remove a programme          |
+----------------------------------------+

Enter Programme Code: DIS



+-----------------+--------------------------------+------------+
| Programme Code  | Programme Name                 | Duration   |
+-----------------+--------------------------------+------------+
| DIS             | Diploma In Science             |      3     |
+-----------------+--------------------------------+------------+
Confirm delete? (Y/N): y
Programme successfully deleted.
```

If the programme does not contain any tutorial groups, the system will prompt successful deleted messages.

3. Find programme

After user input 3 in the programme management menu, it will redirect the user to “Find programme”

```
+-----------------------------------------+
|               Find programme            |
+-----------------------------------------+

Enter Programme Code: DIS
Programmme DIS not found.
```

This function allows the user to find the selected programme. Users will need to enter programme code to search it. If programme code is not available, it will prompt a message “Programme ‘Input Program Code’ not found”.

```
+-----------------------------------------+
|               Find programme            |
+-----------------------------------------+

Enter Programme Code: DIM



+-----------------+---------------------------------+------------+
| Programme Code  | Programme Name                  | Duration   |
+-----------------+---------------------------------+------------+
| DIM             | Degree in Marketing             |      3     |
+-----------------+---------------------------------+------------+
```

If the input programme code is available, it will display the programme details.

4. Amend programme details

```
+-----------------------------------------+
|          Amend programme details        |
+-----------------------------------------+

Enter Programme Code: DIN



+-----------------+---------------------------------+------------+
| Programme Code  | Programme Name                  | Duration   |
+-----------------+---------------------------------+------------+
| DIN             | Degree in Networking            |      2     |
+-----------------+---------------------------------+------------+

Modify Programme Detail

1. Programme Name
2. Duration of Year
0. Done
Enter choice (0-2): 2
Enter Duration of Year: 3



+-----------------+---------------------------------+------------+
| Programme Code  | Programme Name                  | Duration   |
+-----------------+---------------------------------+------------+
| DIN             | Degree in Networking            |      3     |
+-----------------+---------------------------------+------------+

Modify Programme Detail

1. Programme Name
2. Duration of Year
0. Done
Enter choice (0-2): 0

 +-----------------+---------------------------------+------------+
 | Programme Code  | Programme Name                  | Duration   |
 +-----------------+---------------------------------+------------+
 | DIN             | Degree in Networking            |      3     |
 +-----------------+---------------------------------+------------+
Confirm modify? (Y/N): y
Programme successfully modified.
```

After user input 4 in the programme management menu, it will redirect the user to “Amend programme details”. It will prompt the user to enter programme code, if the system found the programme, it will display the programme details. The system will display a modify programme detail menu for users to select which section to modify. It will only allow the user to enter an integer to select. If user enter 1 it will prompt user to enter programme name, else if

enter 2 will prompt user to enter duration of year. After user input, it will display the modified details and return to modify the programme detail menu. If the user enters 0, it will end the modify menu and display again the modified programme detail. System will request user confirmation to amend the programme, if confirmed, the programme detail will modify to new details, else the programme detail will remain.

5. List all programme

```
+-----------------------------------------+
|             List all programme          |
+-----------------------------------------+

+-----+-----------------+---------------------------------+------------+
| No. | Programme Code  | Programme Name                  | Duration   |
+-----+-----------------+---------------------------------+------------+
|  1. | DCS             | Degree in Computer Science      |      3     |
+-----+-----------------+---------------------------------+------------+
|  2. | DHM             | Degree in Hotel Management      |      3     |
+-----+-----------------+---------------------------------+------------+
|  3. | DIM             | Degree in Marketing             |      3     |
+-----+-----------------+---------------------------------+------------+
|  4. | DIN             | Degree in Networking            |      2     |
+-----+-----------------+---------------------------------+------------+
```

After user input 5 in the programme management menu, it will redirect the user to “List all programmes”. It will display all programmes in the programme list.

6. Add tutorial group to a programme

```
+-----------------------------------------+
|    Add tutorial group to a programme    |
+-----------------------------------------+

Enter Programme Code: DCS



+-----------------+---------------------------------+------------+
| Programme Code  | Programme Name                  | Duration   |
+-----------------+---------------------------------+------------+
| DCS             | Degree in Computer Science      |      3     |
+-----------------+---------------------------------+------------+
+-----+-------------------------------------------------------------+
| No. | Tutorial Group                                              |
+-----+-------------------------------------------------------------+
|  1. | DCS Group 3                                                 |
+-----+-------------------------------------------------------------+
|  2. | DCS Group 4                                                 |
+-----+-------------------------------------------------------------+
|  3. | DIM Group 3                                                 |
+-----+-------------------------------------------------------------+
|  4. | DIM Group 4                                                 |
+-----+-------------------------------------------------------------+
|  5. | DIN Group 3                                                 |
+-----+-------------------------------------------------------------+
|  6. | DIN Group 4                                                 |
+-----+-------------------------------------------------------------+
Enter the number of the tutorial group: a
Invalid input. Please enter an integer.
Enter the number of the tutorial group: 3
Confirm add? (Y/N): y
Tutorial group DIM Group 3 added to programme Degree in Computer Science.
```

After user input 6 in the programme management menu, it will redirect the user to “Add tutorial group to a programme”. It will prompt users to enter programme code, if programme code is available, it will display the programme details and display an unassociated tutorial group list for users to select. Users are only allowed to input integers and the integer should be in the range of the list. The system will request user confirmation to add the tutorial group into the programme. If users confirm, the tutorial group will be added into the programme and prompt a successful message.

7. Remove tutorial group from a programme

After user input 7 in the programme management menu, it will redirect the user to “Remove tutorial group from a programme”.

```
+-----------------------------------------+
|  Remove tutorial group from a programme  |
+-----------------------------------------+

Enter Programme Code: DCS



+-----------------+---------------------------------+------------+
| Programme Code  | Programme Name                  | Duration   |
+-----------------+---------------------------------+------------+
| DCS             | Degree in Computer Science      |      3     |
+-----------------+---------------------------------+------------+
+-----+-------------------------------------------------------------+
| No. | Tutorial Group                                              |
+-----+-------------------------------------------------------------+
|  1. | DCS Group 1                                                 |
+-----+-------------------------------------------------------------+
|  2. | DCS Group 2                                                 |
+-----+-------------------------------------------------------------+
|  3. | DIM Group 3                                                 |
+-----+-------------------------------------------------------------+
Enter the number of the tutorial group: a
Invalid input. Please enter an integer.
Enter the number of the tutorial group: 3
Confirm delete? (Y/N): y
Tutorial group DIM Group 3 removed from programme Degree in Computer Science.
```

It will prompt the user to enter the programme code, if the programme is found, it will display the programme details with the tutorial group in the programme. After that , the user will need to input an integer within the range of the tutorial group. After confirmation, the selected tutorial group will be removed and a message will display showing which group is deleted from what programme.

```
+-----------------------------------------+
|  Remove tutorial group from a programme  |
+-----------------------------------------+

Enter Programme Code: dhm



+-----------------+---------------------------------+-------------+
| Programme Code  | Programme Name                  | Duration    |
+-----------------+---------------------------------+-------------+
| DHM             | Degree in Hotel Management      |      3      |
+-----------------+---------------------------------+-------------+
+-----------------------------------------------------------------+
|             No tutorial group in this programme                 |
+-----------------------------------------------------------------+
```

If the programme does not contain any tutorial group, it will display “No tutorial group in this programme”.

8. List all tutorial groups for a programme

After user input 8 in the programme management menu, it will redirect the user to “List all tutorial groups for a programme”.

```
+----------------------------------------+
| List all tutorial groups for a programme |
+----------------------------------------+

Enter Programme Code: DCS



+-----------------+--------------------------------+------------+
| Programme Code  | Programme Name                 | Duration   |
+-----------------+--------------------------------+------------+
| DCS             | Degree in Computer Science     |      3     |
+-----------------+--------------------------------+------------+
+-----+-------------------------------------------------------+
| No. | Tutorial Group                                        |
+-----+-------------------------------------------------------+
|  1. | DCS Group 1                                           |
+-----+-------------------------------------------------------+
|  2. | DCS Group 2                                           |
+-----+-------------------------------------------------------+
```

It will prompt the user to enter the programme code, if the programme is found, it will display the programme details with the tutorial group in the programme.

```
+----------------------------------------+
| List all tutorial groups for a programme |
+----------------------------------------+

Enter Programme Code: DHM



+-----------------+--------------------------------+------------+
| Programme Code  | Programme Name                 | Duration   |
+-----------------+--------------------------------+------------+
| DHM             | Degree in Hotel Management     |      3     |
+-----------------+--------------------------------+------------+
+-------------------------------------------------------------+
|             No tutorial group in this programme             |
+-------------------------------------------------------------+
```

If the programme does not contain any tutorial group, it will display “No tutorial group in this programme”.

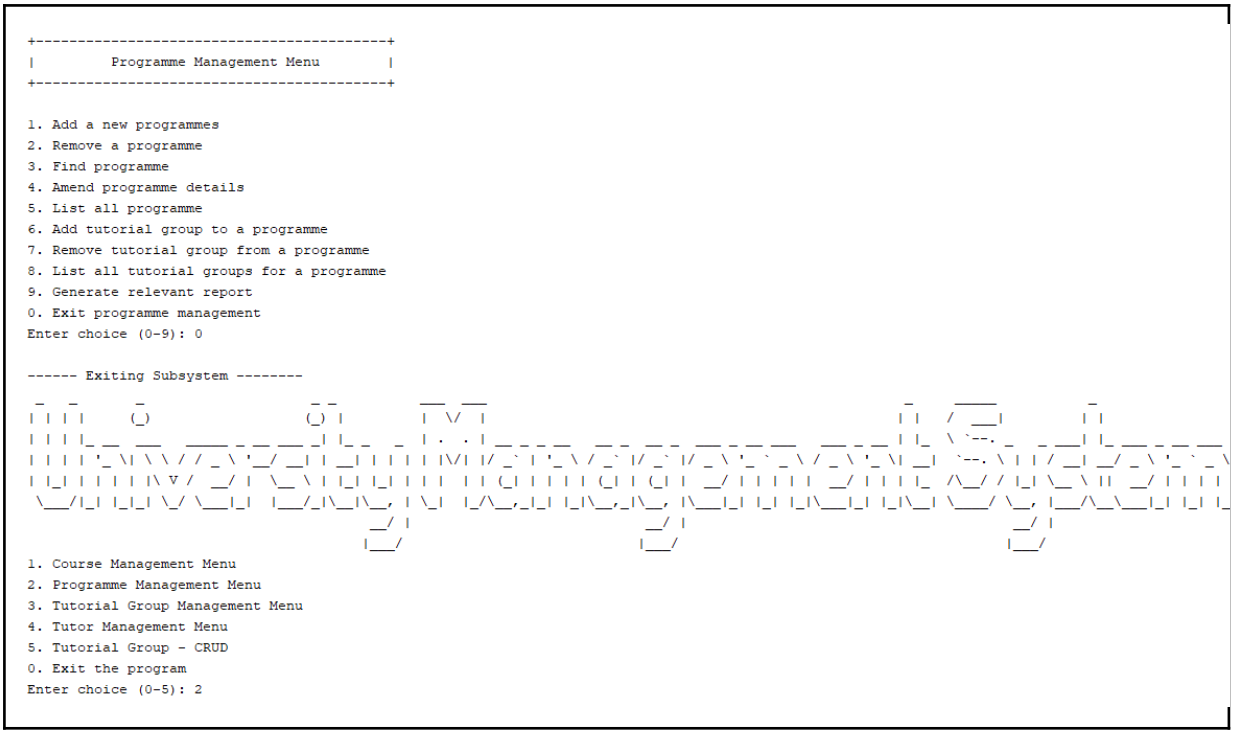
9. Generate relevant report

After user input 9 in the programme management menu, it will redirect the user to “Report”.

```
+-----------------------------------------+
|                  Report                 |
+-----------------------------------------+

+-----+---------------------------------+-----------------+-----------------+
| No. | Programme                       | Tutorial Group  |      Student    |
+-----+---------------------------------+-----------------+-----------------+
|  1. | Degree in Computer Science      |         2       |         4       |
+-----+---------------------------------+-----------------+-----------------+
|  2. | Degree in Hotel Management      |         0       |         0       |
+-----+---------------------------------+-----------------+-----------------+
|  3. | Degree in Marketing             |         2       |         1       |
+-----+---------------------------------+-----------------+-----------------+
|  4. | Degree in Networking            |         2       |         6       |
+-----+---------------------------------+-----------------+-----------------+
```

The system will display the report after calculating the total tutorial in each group and total student in the programme.

10. Exit programme management



```
+-----------------------------------------+
|         Programme Management Menu       |
+-----------------------------------------+

1. Add a new programmes
2. Remove a programme
3. Find programme
4. Amend programme details
5. List all programme
6. Add tutorial group to a programme
7. Remove tutorial group from a programme
8. List all tutorial groups for a programme
9. Generate relevant report
0. Exit programme management
Enter choice (0-9): 0

------ Exiting Subsystem --------
```


After the user enters 0, it will return to the University Management System main menu.

2.3 Tutorial Group Management

Student Name	Student ID	Prog / Tut.Grp	Signature
Bok Cheong Roy	2309298	RSD2G3	Roy

2.3.1 Source Code

TutorialGroupManagementUI.java

```
1 package boundary;
2
3 import adt.*;
4 import dao.ProgrammeDAO;
5 import entity.*;
6 import java.util.Scanner;
7
8 /**
9  *
10  * @author Bok Cheong Roy
11  */
12 public class TutorialGroupManagementUI {
13
14     private SortedListInterface<Programme> programmeList = new
SortedArrayList<>();
15     private final ProgrammeDAO programmeDAO = new ProgrammeDAO();
16
17     public TutorialGroupManagementUI() {
18         try {
19             programmeList = programmeDAO.retrieveFromFile();
20         } catch (ClassNotFoundException ex) {
21             System.out.println("Something Wrong in File");
22         }
23     }
24
25     Scanner scanner = new Scanner(System.in);
26
27     public int getMenuChoice() {
28         System.out.println("\nTUTORIAL GROUP MANAGEMENT MENU");
29         System.out.println("1. Add a Student to a Tutorial Group");
30         System.out.println("2. Remove Student from Tutorial Group");
31         System.out.println("3. Change the Tutorial Group for a Student");
32         System.out.println("4. Find a Student in a Tutorial Group");
33         System.out.println("5. List all Students in a Tutorial Group");
34         System.out.println("6. Filter tutorial groups based on criteria");
35         System.out.println("7. Generate relevant reports");
36         System.out.println("0. Quit");
37         int choice;
38         while (true) {
39             System.out.print("Enter choice (0-7): ");
40             if (scanner.hasNextInt()) {
41                 choice = scanner.nextInt();
42                 scanner.nextLine();
```

```
43                  return choice;
44              } else {
45                  scanner.nextLine();
46                  System.out.println("Invalid input. Please enter an
integer.");
47              }
48          }
49      }
50
51      public int findStudentMenu() {
52          System.out.println("\nFind Student");
53          System.out.println("1.By Name");
54          System.out.println("2.By ID");
55          System.out.println("0. Quit to TUTORIAL GROUP MANAGEMENT MENU");
56          int option;
57          while (true) {
58              System.out.print("Enter choice: ");
59              if (scanner.hasNextInt()) {
60                  option = scanner.nextInt();
61                  scanner.nextLine();
62                  return option;
63              } else {
64                  scanner.nextLine();
65                  System.out.println("Invalid input. Please enter an
integer.");
66              }
67          }
68      }
69
70      public int filterMenu() {
71          System.out.println("\nFilter Student");
72          System.out.println("1.By Name");
73          System.out.println("2.By ID");
74          System.out.println("3.By Programme");
75          System.out.println("0. Quit to TUTORIAL GROUP MANAGEMENT MENU");
76          int option;
77          while (true) {
78              System.out.print("Enter choice: ");
79              if (scanner.hasNextInt()) {
80                  option = scanner.nextInt();
81                  scanner.nextLine();
82                  return option;
83              } else {
```

```
                scanner.nextLine();
                System.out.println("Invalid input. Please enter an
integer.");
            }
        }
    }

    public void listAllStudents(String outputStr) {
        System.out.println("\nList of Students:\n" + outputStr);
    }

    public void printStudentDetails(Student student) {
        System.out.println("Student Details");
        System.out.println("Student Name:" + student.getName());
        System.out.println("Student ID: " + student.getStudentID());
        System.out.println("Programme: " + student.getProgramme());
        System.out.println("TutorialGroup: " +
student.getTutorialGroup());
    }

    public String inputStudentName() {
        String name;
        do {
            System.out.print("Enter Student Name: ");
            name = scanner.nextLine().trim();
            if (name.isEmpty()) {
                System.out.println("Name cannot be empty. Please try
again.");
            }
        } while (name.isEmpty());
        return name;
    }

    public String inputStudentID() {
        String studentID;
        do {
            System.out.print("Enter Student ID (7-digit number): ");
            studentID = scanner.nextLine().trim();

            // Check if the input is empty
            if (studentID.isEmpty()) {
                System.out.println("Student ID cannot be empty. Please try
again.");
```

```
123                 continue; // Skip the validation below and re-prompt for
input
124             }
125
126             // Check if the input consists of exactly 7 digits
127             if (!studentID.matches("\\d{7}")) {
128                 System.out.println("Student ID must be a 7-digit number.
Please try again.");
129                 studentID = ""; // Reset studentID to trigger another
iteration
130             }
131         } while (studentID.isEmpty());
132
133         return studentID;
134     }
135
136     public String inputProgramme() {
137         int programCount = programmeList.getNumberOfEntries();
138         if (programCount == 0) {
139             System.out.println("No programme available.");
140             return null; // Handle the case when there are no programs
141         }
142
143         String programme = null;
144         int choice;
145
146         do {
147             System.out.println("\nProgramme\n" + "---------");
148             for (int i = 0; i < programCount; i++) {
149                 System.out.println((i + 1) + ". " +
programmeList.getEntry(i).getProgrammeName());
150             }
151             System.out.print("Enter Programme: ");
152             String input = scanner.nextLine().trim(); // Read a line and
trim leading/trailing spaces
153
154             if (input.isEmpty()) {
155                 System.out.println("Please enter a valid programme.");
156             } else {
157                 try {
158                     choice = Integer.parseInt(input);
159                     if (choice >= 1 && choice <= programCount) {
160                         programme = programmeList.getEntry(choice -
```

```
1).getProgrammeName();
161                     } else {
162                         System.out.println("Invalid choice. Please select
a valid programme.");
163                     }
164                 } catch (NumberFormatException e) {
165                     System.out.println("Invalid input. Please enter a
valid programme number.");
166                 }
167             }
168         } while (programme == null);
169
170         return programme;
171     }
172
173     public String inputTutorialGroup(String programme) {
174         String tutorialGroup;
175         int programmeNum = -1;
176
177         for (int i = 0; i < programmeList.getNumberOfEntries(); i++) {
178             if
(programmeList.getEntry(i).getProgrammeName().equals(programme)) {
179                 programmeNum = i;
180                 int tutorialCount =
programmeList.getEntry(i).getTutorialGroup().getNumberOfEntries();
181
182                 if (tutorialCount == 0) {
183                     System.out.println("No tutorial groups available for
this program.\n");
184                     return null; // Handle the case when there are no
tutorial groups for the selected program
185                 }
186
187                 String input;
188                 int choice;
189
190                 do {
191                     System.out.println("Tutorial Groups for " + programme
+ "\n" + "-------------------");
192                     for (int j = 0; j < tutorialCount; j++) {
193                         int num = j + 1;
194                         System.out.println(num + ". " +
programmeList.getEntry(i).getTutorialGroup().getEntry(j).getTgName());
```

```
195                     }
196                     System.out.print("Enter Tutorial Group: ");
197                     input = scanner.nextLine().trim(); // Read a line and
trim leading/trailing spaces
198
199                     if (input.isEmpty()) {
200                         System.out.println("Please enter a valid tutorial
group.\n");
201                     } else {
202                         try {
203                             choice = Integer.parseInt(input);
204                             if (choice >= 1 && choice <= tutorialCount) {
205                                 tutorialGroup =
programmeList.getEntry(programmeNum).getTutorialGroup().getEntry(choice -
1).getTgName();
206                                 return tutorialGroup;
207                             } else {
208                                 System.out.println("Invalid choice. Please
select a valid tutorial group.\n");
209                             }
210                         } catch (NumberFormatException e) {
211                             System.out.println("Invalid input. Please
enter a valid tutorial group number.\n");
212                         }
213                     }
214                 } while (true);
215             }
216         }
217
218         System.out.println("Invalid program name.");
219         return null; // Handle the case when the program name is not found
220     }
221
222     public Student inputStudentDetails() {
223         String studentName = inputStudentName();
224         String studentID = inputStudentID();
225         String programme = inputProgramme();
226         String tutorialGroup = inputTutorialGroup(programme);
227         System.out.println();
228         return new Student(studentName, studentID, programme,
tutorialGroup);
229     }
230
```

```
231     public int inputPosition() {
232         int position = 0;
233         boolean isValid = false;
234 
235         do {
236             System.out.print("Enter Position by using number: ");
237             String input = scanner.nextLine().trim(); // Read a line and
trim leading/trailing spaces
238 
239             if (input.isEmpty()) {
240                 System.out.println("Please enter a valid integer.\n");
241             } else {
242                 try {
243                     position = Integer.parseInt(input);
244                     isValid = true;
245                 } catch (NumberFormatException e) {
246                     System.out.println("Invalid input. Please enter a
valid integer.\n");
247                 }
248             }
249         } while (!isValid);
250 
251         return position;
252     }
253 
254     public boolean getConfirmation(String message) {
255         String userInput = "";
256         boolean valid = false;
257         do {
258             System.out.print(message + " (y/n): ");
259             userInput = scanner.nextLine().toLowerCase().trim();
260             if (userInput.isEmpty()) {
261                 System.out.println("Cannot be empty. Please try again.");
262             }else if (!userInput.equals("y") && !userInput.equals("n")) {
263                 System.out.println("Cannot be empty. Please try again.");
264             }
265         } while (!userInput.equals("y") && !userInput.equals("n"));
266 
267         if (userInput.equals("y")) {
268             valid = true;
269         }
270         return valid;
271     }
```

```
272
273     public String inputProgrammeForListStudentInGroup() {
274         int programCount = programmeList.getNumberOfEntries();
275         if (programCount == 0) {
276             System.out.println("No programme available.");
277             return null; // Handle the case when there are no programs
278         }
279
280         String programme = null;
281         int choice;
282
283         do {
284             System.out.println("\nProgramme\n" + "---------");
285             for (int i = 0; i < programCount; i++) {
286                 System.out.println((i + 1) + ". " +
programmeList.getEntry(i).getProgrammeName());
287             }
288             System.out.println("0. Quit to Tutorial Group Management");
289             System.out.print("Enter Programme: ");
290             String input = scanner.nextLine().trim(); // Read a line and
trim leading/trailing spaces
291
292             if (input.isEmpty()) {
293                 System.out.println("Please enter a valid programme.");
294             } else {
295                 try {
296                     choice = Integer.parseInt(input);
297                     if (choice == 0) {
298                         programme = "0";
299                         return programme;
300                     }
301                     if (choice >= 1 && choice <= programCount) {
302                         programme = programmeList.getEntry(choice -
1).getProgrammeName();
303                     } else {
304                         System.out.println("Invalid choice. Please select
a valid programme.");
305                     }
306                 } catch (NumberFormatException e) {
307                     System.out.println("Invalid input. Please enter a
valid programme number.");
308                 }
309             }
```

```
310         } while (programme == null);
311
312         return programme;
313     }
314
315     public String inputTutorialGroupForListStudentInGroup(String
programme) {
316         String tutorialGroup;
317         int programmeNum = -1;
318
319         for (int i = 0; i < programmeList.getNumberOfEntries(); i++) {
320             if
(programmeList.getEntry(i).getProgrammeName().equals(programme)) {
321                 programmeNum = i;
322                 int tutorialCount =
programmeList.getEntry(i).getTutorialGroup().getNumberOfEntries();
323
324                 if (tutorialCount == 0) {
325                     System.out.println("No tutorial groups available for
this program.\n");
326                     return null; // Handle the case when there are no
tutorial groups for the selected program
327                 }
328
329                 String input;
330                 int choice;
331
332                 do {
333                     System.out.println("Tutorial Groups for " + programme
+ "\n" + "--------------------");
334                     for (int j = 0; j < tutorialCount; j++) {
335                         int num = j + 1;
336                         System.out.println(num + ". " +
programmeList.getEntry(i).getTutorialGroup().getEntry(j).getTgName());
337                     }
338                     System.out.println("0. Quit to Programme.");
339                     System.out.print("Enter Tutorial Group: ");
340                     input = scanner.nextLine().trim(); // Read a line and
trim leading/trailing spaces
341
342                     if (input.isEmpty()) {
343                         System.out.println("Please enter a valid tutorial
group.\n");
```

```
344                     } else {
345                         try {
346                             choice = Integer.parseInt(input);
347                             if (choice == 0) {
348                                 tutorialGroup = "0";
349                                 return tutorialGroup;
350                             }
351                             if (choice >= 1 && choice <= tutorialCount) {
352                                 tutorialGroup =
programmeList.getEntry(programmeNum).getTutorialGroup().getEntry(choice -
1).getTgName();
353                                 return tutorialGroup;
354                             } else {
355                                 System.out.println("Invalid choice. Please
select a valid tutorial group.\n");
356                             }
357                         } catch (NumberFormatException e) {
358                             System.out.println("Invalid input. Please
enter a valid tutorial group number.\n");
359                         }
360                     }
361                 } while (true);
362             }
363         }
364
365         System.out.println("Invalid program name.");
366         return null; // Handle the case when the program name is not found
367     }
368 }
```

TutorialGroupManagement.java

```
 1 package control;
 2
 3 import adt.*;
 4 import boundary.TutorialGroupManagementUI;
 5 import dao.*;
 6 import entity.*;
 7 import utility.MessageUI;
 8
 9 /**
10  *
11  * @author Bok Cheong Roy
12  */
13 public class TutorialGroupManagement {
14
15     private ListInterface<Student> studentList = new ArrayList<>();
16     private final StudentDAO studentDAO = new StudentDAO();
17     private final TutorialGroupManagementUI tutorialUI = new
TutorialGroupManagementUI();
18     private SortedListInterface<Programme> programmeList = new
SortedArrayList<>();
19     private ProgrammeDAO programmeDAO = new ProgrammeDAO();
20
21     public TutorialGroupManagement() {
22         try {
23             studentList = studentDAO.retrieveFromFile();
24         } catch (ClassNotFoundException ex) {
25             System.out.println("Something Wrong in File");
26         }
27         try {
28             programmeList = programmeDAO.retrieveFromFile();
29         } catch (ClassNotFoundException ex) {
30             System.out.println("Something Wrong in File");
31         }
32     }
33
34     public void runStudentMaintenance() {
35         int choice = 0;
36         do {
37             choice = tutorialUI.getMenuChoice();
38             switch (choice) {
39                 case 0:
40                     MessageUI.displayExitSubsystemMessage();
41                     break;
```

```
42                  case 1:
43                      addNewStudent();
44                      break;
45                  case 2:
46                      displayStudents();
47                      removeStudent();
48                      break;
49                  case 3:
50                      displayStudents();
51                      changeTutorialGroup();
52                      break;
53                  case 4:
54                      findStudent();
55                      break;
56                  case 5:
57                      displayStudentsOfGroup();
58                      break;
59                  case 6:
60                      filterStudent();
61                      break;
62                  case 7:
63                      displayStudentReport();
64                      break;
65                  default:
66                      MessageUI.displayInvalid();
67              }
68          } while (choice != 0);
69      }
70
71      public void addNewStudent() {
72          Student newStudent;
73          boolean isUnique;
74
75          do {
76              newStudent = tutorialUI.inputStudentDetails();
77              isUnique = isStudentIDUnique(newStudent.getStudentID());
78
79              if (!isUnique) {
80                  System.out.println("Student ID already exists. Please
enter a unique ID.");
81              }
82          } while (!isUnique);
83          if (newStudent.getTutorialGroup() == null) {
```

```
 84             return;
 85         }
 86         boolean confirmation = tutorialUI.getConfirmation("Are you
Confirm to Add?");
 87         if (confirmation == true) {
 88             studentList.add(newStudent);
 89             studentDAO.saveToFile(studentList);
 90             displayStudents();
 91             System.out.println("Succesful added Student:" +
newStudent.getName());
 92             return;
 93         }
 94         System.out.println("Action Cancelled");
 95     }
 96
 97     public boolean isStudentIDUnique(String studentIDToCheck) {
 98         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
 99             Student student = studentList.getEntry(i);
100             if (student != null &&
student.getStudentID().equalsIgnoreCase(studentIDToCheck)) {
101                 return false; // The studentID already exists in the list
102             }
103         }
104         return true; // The studentID is unique
105     }
106
107     public void removeStudent() {
108         int position;
109
110         do {
111             position = tutorialUI.inputPosition();
112             if (position < 1 || position >
studentList.getNumberOfEntries()) {
113                 System.out.println("Invalid Choice. Please Choose a valid
position between 1 and " + studentList.getNumberOfEntries() + "\n");
114             }
115         } while (position < 1 || position >
studentList.getNumberOfEntries());
116
117         boolean confirmation = tutorialUI.getConfirmation("Are you
Confirm to Remove?");
118         if (confirmation == true) {
119             System.out.println("Successful Removed Student:" +
```

```
studentList.getEntry(position).getName());
120             studentList.remove(position);
121             studentDAO.saveToFile(studentList);
122             return;
123         }
124         System.out.println("Action Cancelled");
125     }
126
127     public void changeTutorialGroup() {
128         int position;
129
130         do {
131             position = tutorialUI.inputPosition();
132             if (position < 1 || position >
studentList.getNumberOfEntries()) {
133                 System.out.println("Invalid Choice. Please Choose a valid
position between 1 and " + studentList.getNumberOfEntries() + "\n");
134             }
135         } while (position < 1 || position >
studentList.getNumberOfEntries());
136
137         Student modifyStudent = studentList.getEntry(position);
138         String newTutorialGroup =
tutorialUI.inputTutorialGroup(studentList.getEntry(position).getProgramme());
139         if (modifyStudent.getTutorialGroup() == null) {
140             return;
141         }
142         boolean confirmation = tutorialUI.getConfirmation("Are you
Confirm to Edit?");
143         if (confirmation == true) {
144             modifyStudent.setTutorialGroup(newTutorialGroup);
145             studentList.replace(position, modifyStudent);
146             studentDAO.saveToFile(studentList);
147             System.out.println("Successful Edited Student:" +
studentList.getEntry(position).getName());
148             return;
149         }
150         System.out.println("Action Cancelled");
151
152     }
153
154     public void findStudent() {
155         int choice = 0;
```

```
156         do {
157             choice = tutorialUI.findStudentMenu();
158             switch (choice) {
159                 case 0:
160                     System.out.println("Exit");
161                     break;
162                 case 1:
163                     findStudentByName();
164                     break;
165                 case 2:
166                     findStudentByID();
167                     break;
168                 default:
169                     System.out.println("Invalid Choice Please Choose
Again");
170             }
171 
172         } while (choice != 0);
173     }
174 
175     public void findStudentByName() {
176         String studentName = tutorialUI.inputStudentName().toLowerCase();
// Convert the input to lowercase
177         Student foundStudent = null;
178 
179         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
180             String currentName =
studentList.getEntry(i).getName().toLowerCase(); // Convert the current
student's name to lowercase
181             if (currentName.equals(studentName)) {
182                 foundStudent = studentList.getEntry(i);
183                 tutorialUI.printStudentDetails(foundStudent);
184                 System.out.println("");
185             }
186         }
187 
188         if (foundStudent == null) {
189             System.out.println("Student not found");
190         }
191     }
192 
193     public void findStudentByID() {
194         String studentID = tutorialUI.inputStudentID().toLowerCase(); //
```

```
Convert the input to lowercase
195         Student foundStudent = null;
196
197         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
198             String currentID =
studentList.getEntry(i).getStudentID().toLowerCase(); // Convert the current
student's ID to lowercase
199             if (currentID.equals(studentID)) {
200                 foundStudent = studentList.getEntry(i);
201                 tutorialUI.printStudentDetails(foundStudent);
202                 System.out.println("");
203             }
204         }
205
206         if (foundStudent == null) {
207             System.out.println("Student not found");
208         }
209     }
210
211     public void filterStudent() {
212         int choice = 0;
213         do {
214             choice = tutorialUI.filterMenu();
215             switch (choice) {
216                 case 0:
217                     System.out.println("Exit");
218                     break;
219                 case 1:
220                     tutorialUI.listAllStudents(filterStudentByName());
221                     break;
222                 case 2:
223                     tutorialUI.listAllStudents(filterStudentByID());
224
225                     break;
226                 case 3:
227
tutorialUI.listAllStudents(filterStudentByProgramme());
228                     break;
229                 default:
230                     System.out.println("Invalid Choice Please Choose
Again");
231             }
232
```

```
233         } while (choice != 0);
234     }
235 
236     public String filterStudentByProgramme() {
237         ListInterface<Student> filterStudent = new ArrayList<>();
238         int numberOfEntries = studentList.getNumberOfEntries();
239 
240         for (int i = 1; i <= numberOfEntries; i++) {
241             Student sortStudent = studentList.getEntry(i);
242 
243             // Check if sortStudent is not null
244             if (sortStudent != null) {
245                 int j = 0;
246 
247                 while (j < filterStudent.getNumberOfEntries()) {
248                     Student existingStudent = filterStudent.getEntry(j +
1);
249 
250                     // Check if existingStudent is not null and has
non-null Programme and TutorialGroup
251                     if (existingStudent != null
252                             && existingStudent.getProgramme() != null
253                             && existingStudent.getTutorialGroup() != null
254                             && sortStudent.getProgramme() != null
255                             && sortStudent.getTutorialGroup() != null) {
256 
257                         int programmeComparison =
sortStudent.compareToProgramme(existingStudent.getProgramme());
258                         int tutorialGroupComparison =
sortStudent.compareToTutorial(existingStudent.getTutorialGroup());
259 
260                         if (programmeComparison > 0 ||
(programmeComparison == 0 && tutorialGroupComparison > 0)) {
261                             j++;
262                         } else {
263                             break; // Stop comparing once the correct
position is found
264                         }
265                     } else {
266                         // Handle null values or invalid data as needed
267                         // You may choose to skip or log such entries
268                         j++;
269                     }
```

```
270                 }
271                 filterStudent.add(j + 1, sortStudent);
272             }
273         }
274 
275         String outputStr = String.format(" %4s %-20s %-10s %-30s %-12s\n",
"No.", "Name", "ID", "Programme", "Group")
276                 +
"-------------------------------------------------------------------------
-----\n";
277         for (int i = 1; i <= filterStudent.getNumberOfEntries(); i++) {
278             Student currentStudent = filterStudent.getEntry(i);
279             outputStr = outputStr + String.format(" %3d. %-20s %-10s %-30s
%-12s\n",
280                     i, currentStudent.getName(),
281                     currentStudent.getStudentID(),
282                     currentStudent.getProgramme(),
283                     currentStudent.getTutorialGroup());
284         }
285         return outputStr;
286     }
287 
288     public String filterStudentByName() {
289         ListInterface<Student> filterStudent = new ArrayList<>();
290         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
291             Student sortStudent = studentList.getEntry(i);
292             if (filterStudent.isEmpty()) {
293                 filterStudent.add(sortStudent);
294             } else {
295                 int j = 0;
296                 while (j < filterStudent.getNumberOfEntries() &&
sortStudent.compareTo(filterStudent.getEntry(j + 1).getName()) > 0) {
297                     j++;
298                 }
299                 filterStudent.add(j + 1, sortStudent);
300             }
301         }
302 
303         String outputStr = String.format(" %4s %-20s %-10s %-30s %-12s\n",
"No.", "Name", "ID", "Programme", "Group")
304                 +
"-------------------------------------------------------------------------
-----\n";
```

```
305         for (int i = 1; i <= filterStudent.getNumberOfEntries(); i++) {
306             Student currentStudent = filterStudent.getEntry(i);
307             outputStr = outputStr + String.format(" %3d. %-20s %-10s %-30s
%-12s\n",
308                     i, currentStudent.getName(),
309                     currentStudent.getStudentID(),
310                     currentStudent.getProgramme(),
311                     currentStudent.getTutorialGroup());
312         }
313
314         return outputStr;
315     }
316
317     public String filterStudentByID() {
318         ListInterface<Student> filterStudent = new ArrayList<>();
319         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
320             Student sortStudent = studentList.getEntry(i);
321             if (filterStudent.isEmpty()) {
322                 filterStudent.add(sortStudent);
323             } else {
324                 int j = 0;
325                 while (j < filterStudent.getNumberOfEntries() &&
sortStudent.compareToIgnoreCase(filterStudent.getEntry(j + 1).getStudentID())
> 0) {
326                     j++;
327                 }
328                 filterStudent.add(j + 1, sortStudent);
329             }
330         }
331         String outputStr = String.format(" %4s %-20s %-10s %-30s %-12s\n",
"No.", "Name", "ID", "Programme", "Group")
332                 +
"-------------------------------------------------------------------------
-----\n";
333         for (int i = 1; i <= filterStudent.getNumberOfEntries(); i++) {
334             Student currentStudent = filterStudent.getEntry(i);
335             outputStr = outputStr + String.format(" %3d. %-20s %-10s %-30s
%-12s\n",
336                     i, currentStudent.getName(),
337                     currentStudent.getStudentID(),
338                     currentStudent.getProgramme(),
339                     currentStudent.getTutorialGroup());
340         }
```

```
341
342         return outputStr;
343     }
344
345     public String getAllStudents() {
346         String outputStr = String.format(" %4s %-20s %-10s %-30s %-12s\n",
"No.", "Name", "ID", "Programme", "Group")
347                 +
"-------------------------------------------------------------------------
-----\n";
348         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
349             Student currentStudent = studentList.getEntry(i);
350             outputStr = outputStr + String.format(" %3d. %-20s %-10s %-30s
%-12s\n",
351                     i, currentStudent.getName(),
352                     currentStudent.getStudentID(),
353                     currentStudent.getProgramme(),
354                     currentStudent.getTutorialGroup());
355         }
356         return outputStr;
357     }
358
359     public void displayStudents() {
360         tutorialUI.listAllStudents(getAllStudents());
361     }
362
363     public String getAllStudentsOfGroup(String TutorialGroup) {
364         String outputStr = String.format(" %4s %-20s %-10s %-30s %-12s\n",
"No.", "Name", "ID", "Programme", "Group")
365                 +
"-------------------------------------------------------------------------
-----\n";
366         int j = 1;
367         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
368             if
(TutorialGroup.equals(studentList.getEntry(i).getTutorialGroup())) {
369                 Student currentStudent = studentList.getEntry(i);
370                 outputStr = outputStr + String.format(" %3d. %-20s %-10s
%-30s %-12s\n",
371                         j, currentStudent.getName(),
372                         currentStudent.getStudentID(),
373                         currentStudent.getProgramme(),
374                         currentStudent.getTutorialGroup());
```

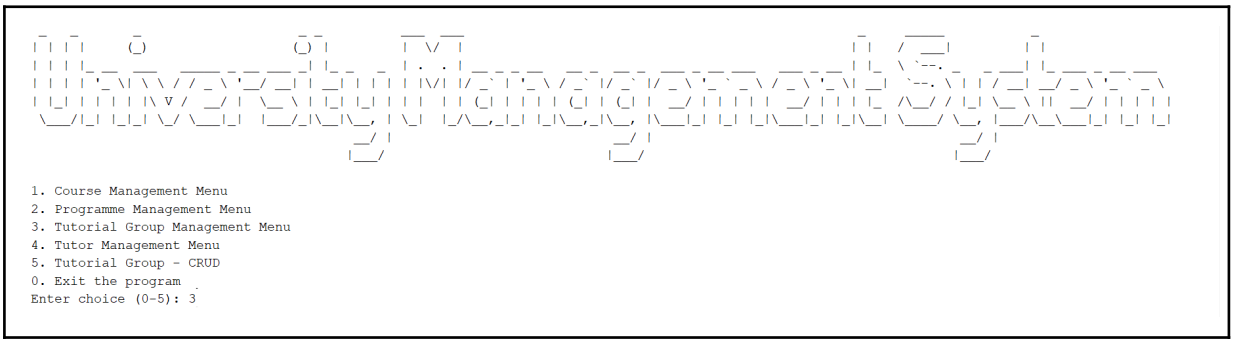
```
375                 j++;
376             }
377         }
378         return outputStr;
379     }
380
381     public void displayStudentsOfGroup() {
382     String repeat="";
383     do {
384         String programme =
tutorialUI.inputProgrammeForListStudentInGroup();
385         if (programme.equals("0")) {
386             return;
387         }
388         String tutorialGroup =
tutorialUI.inputTutorialGroupForListStudentInGroup(programme);
389         if (tutorialGroup == null) {
390             System.out.println("No Tutorial Group Found");
391         } else if (tutorialGroup.equals("0")) {
392             repeat = "repeat";
393         } else {
394             String output = getAllStudentsOfGroup(tutorialGroup);
395             tutorialUI.listAllStudents(output);
396         }
397     } while (repeat.equals("repeat"));
398 }
399
400     public void displayStudentReport() {
401         int totalPrograms = programmeList.getNumberOfEntries();
402         int totalTutorialGroups = 0;
403         int totalStudents = studentList.getNumberOfEntries();
404
405         System.out.println("Report:");
406         System.out.println(String.format("%-20s%-30s%-30s%-30s", "",
"Programme", "Tutorial Group", "Students"));
407
System.out.println("-----------------------------------------------------------
-----------------------------\n");
408
409         for (int i = 0; i < totalPrograms; i++) {
410             Programme program = programmeList.getEntry(i);
411             String programName = program.getProgrammeName();
412             int subtotalTutorialGroups = 0;
```

```
413             int subtotalStudents = 0;
414
415             System.out.println(String.format("%20s%-30s%-30s%-30s", "",
programName, "", ""));
416
System.out.println("------------------------------------------------------------
------------------------------");
417             // Iterate through each tutorial group
418             for (int j = 0; j <
program.getTutorialGroup().getNumberOfEntries(); j++) {
419                 String tutorialGroupName = "";
420                 if (program.getTutorialGroup().getEntry(j) != null) {
421                     tutorialGroupName =
program.getTutorialGroup().getEntry(j).getTgName();
422                 }
423                 int tutorialGroupStudents =
countStudentsInTutorialGroup(programName, tutorialGroupName);
424
425                 System.out.println(String.format("%-20s%-30s%-30s%-30s",
"", programName, tutorialGroupName, tutorialGroupStudents));
426                 subtotalTutorialGroups = j+1;
427                 subtotalStudents += tutorialGroupStudents;
428             }
429             if (program.getTutorialGroup().getNumberOfEntries() == 0) {
430                 System.out.println(String.format("%-20s%-30s%-30s%-30s",
"", "", "No Tutorial Group", ""));
431             }
432             // Subtotal for each program
433
System.out.println("------------------------------------------------------------
------------------------------");
434             System.out.println(String.format("%-20s%-30s%-30s%-30s",
"Subtotal:", "", subtotalTutorialGroups, subtotalStudents));
435             System.out.println("\n");
436             totalTutorialGroups += subtotalTutorialGroups;
437             subtotalTutorialGroups = 0;
438             subtotalStudents = 0;
439         }
440
441         // Total number of students in all programs
442
System.out.println("------------------------------------------------------------
------------------------------");
```

```
443         System.out.println(String.format("%-20s%-30s%-30s%-30s", "
Total:", totalPrograms, totalTutorialGroups, totalStudents));
444
System.out.println("----------------------------------------------------------
------------------------------");
445     }
446
447 // Helper method to count students in a specific tutorial group of a
program
448     private int countStudentsInTutorialGroup(String programName, String
tutorialGroupName) {
449         int count = 0;
450         for (int i = 1; i <= studentList.getNumberOfEntries(); i++) {
451             Student student = studentList.getEntry(i);
452             if (programName.equals(student.getProgramme()) &&
tutorialGroupName.equals(student.getTutorialGroup())) {
453                 count++;
454             }
455         }
456         return count;
457     }
458
459 }
```

2.3.2 Screenshot of sample user interface

Main Menu



User entered 3 in the main menu will redirect the user to the Tutorial management menu.

Tutorial Group Management Menu

```
TUTORIAL GROUP MANAGEMENT MENU
1. Add a Student to a Tutorial Group
2. Remove Student from Tutorial Group
3. Change the Tutorial Group for a Student
4. Find a Student in a Tutorial Group
5. List all Students in a Tutorial Group
6. Filter tutorial groups based on criteria
7. Generate relevant reports
0. Quit
Enter choice (0-7): sad
Invalid input. Please enter an integer.
Enter choice (0-7):
```

It will first show the tutorial group management menu after from the university main menu.It allow user to access to 7 different function by entering the number from 1-7 and enter 0 for exit to university main menu.If user enter a non integer choice,it will display user a error message and ask user to renter.

Add New Student

```
TUTORIAL GROUP MANAGEMENT MENU
1. Add a Student to a Tutorial Group
2. Remove Student from Tutorial Group
3. Change the Tutorial Group for a Student
4. Find a Student in a Tutorial Group
5. List all Students in a Tutorial Group
6. Filter tutorial groups based on criteria
7. Generate relevant reports
0. Quit
Enter choice (0-7): 1
Enter Student Name: Choo Fan Chi
Enter Student ID (7-digit number): 2309298

Programme
---------
1. Degree in Computer Science
2. Degree in Hotel Management
3. Degree in Marketing
4. Degree in Networking
Enter Programme: 1
Tutorial Groups for Degree in Computer Science
------------------
1. DCS Group 1
2. DCS Group 2
Enter Tutorial Group: 2

Student ID already exists. Please enter a unique ID.
```

When the user comes to add a student into a tutorial group function,it will first ask the user to enter the details of student which is student name and student id.Then,the user are required to choose the programme which the student study with by entering the number for the respective programme.Last,the user need to choose which class the student need to be add in.

The system will prompt the user with an error message and ask for student details if the students are already existing in the tutorial group.

```
Enter Student Name: Choo Fan Chi
Enter Student ID (7-digit number): 2309309

Programme
---------
1. Degree in Computer Science
2. Degree in Hotel Management
3. Degree in Marketing
4. Degree in Networking
Enter Programme: 1
Tutorial Groups for Degree in Computer Science
------------------
1. DCS Group 1
2. DCS Group 2
Enter Tutorial Group: 2

Are you Confirm to Add? (y/n): y
```

If there are no errors with the process of adding students into a tutorials group,it will ask user confirmation by entering y for yes or n for no.

```
List of Students:
  No. Name                ID         Programme                       Group
-------------------------------------------------------------------------------
   1. Angeline            2103000    Degree in Computer Science      DCS Group 2
   2. Bok Bok             2104120    Degree in Computer Science      DCS Group 1
   3. Choo Choo           2103821    Degree in Computer Science      DCS Group 1
   4. Kyan                2104850    Degree in Networking            DIN Group 1
   5. Qian Qian           2104127    Degree in Computer Science      DCS Group 2
   6. Louis               2108744    Degree in Networking            DIN Group 2
   7. Junwei              2104875    Degree in Networking            DIN Group 1
   8. May Teng            2104551    Degree in Marketing             DIM Group 1
   9. Wei Xian            2108451    Degree in Networking            DIN Group 2
  10. Shun                2103801    Degree in Networking            DIN Group 1
  11. Rain                2103820    Degree in Networking            DIN Group 1
  12. Mei Fan Chi         2309298    Degree in Marketing             DIM Group 1
  13. Choo Fan Chi        2309309    Degree in Computer Science      DCS Group 2

Succesful added Student:Choo Fan Chi
```

If the user confirms ,it will show a latest list of students with a tutorials group and show the successful message.

Remove Student

```
TUTORIAL GROUP MANAGEMENT MENU
1. Add a Student to a Tutorial Group
2. Remove Student from Tutorial Group
3. Change the Tutorial Group for a Student
4. Find a Student in a Tutorial Group
5. List all Students in a Tutorial Group
6. Filter tutorial groups based on criteria
7. Generate relevant reports
0. Quit
Enter choice (0-7): 2
```

when the user enters choice 2 in the tutorial group management,it allows the user to remove a student from a tutorial group.

```
List of Students:
  No. Name                ID         Programme                       Group
-------------------------------------------------------------------------------
   1. Angeline            2103000    Degree in Computer Science      DCS Group 2
   2. Bok Bok             2104120    Degree in Computer Science      DCS Group 1
   3. Choo Choo           2103821    Degree in Computer Science      DCS Group 1
   4. Kyan                2104850    Degree in Networking            DIN Group 1
   5. Qian Qian           2104127    Degree in Computer Science      DCS Group 2
   6. Louis               2108744    Degree in Networking            DIN Group 2
   7. Junwei              2104875    Degree in Networking            DIN Group 1
   8. May Teng            2104551    Degree in Marketing             DIM Group 1
   9. Wei Xian            2108451    Degree in Networking            DIN Group 2
  10. Shun                2103801    Degree in Networking            DIN Group 1
  11. Rain                2103820    Degree in Networking            DIN Group 1
  12. Mei Fan Chi         2309298    Degree in Marketing             DIM Group 1
  13. Choo Fan Chi        2309309    Degree in Computer Science      DCS Group 2

Enter Position by using number: 14
Invalid Choice. Please Choose a valid position between 1 and 13
```

It will first show the user an existing student with a tutorial group list and ask the user to enter the student representative number to delete that particular student.If the number are no exist or invalid,it will prompt an error message and ask for reenter.

```
Enter Position by using number: 12
Are you Confirm to Remove? (y/n): y
Successful Removed Student:Mei Fan Chi
```

If the number entered is existing and valid,it will ask the user for a confirmation.Then,it will show the user the successful message of removing a student from the tutorial group.

Change Tutorial Group For A Student

```
TUTORIAL GROUP MANAGEMENT MENU
1. Add a Student to a Tutorial Group
2. Remove Student from Tutorial Group
3. Change the Tutorial Group for a Student
4. Find a Student in a Tutorial Group
5. List all Students in a Tutorial Group
6. Filter tutorial groups based on criteria
7. Generate relevant reports
0. Quit
Enter choice (0-7): 3
```

Users can access the function of changing the tutorial group for a student by entering 3 in the tutorial group management menu.

```
List of Students:
  No. Name                ID         Programme                       Group
-------------------------------------------------------------------------------
   1. Angeline            2103000    Degree in Computer Science      DCS Group 2
   2. Bok Bok             2104120    Degree in Computer Science      DCS Group 1
   3. Choo Choo           2103821    Degree in Computer Science      DCS Group 1
   4. Kyan                2104850    Degree in Networking            DIN Group 1
   5. Qian Qian           2104127    Degree in Computer Science      DCS Group 2
   6. Louis               2108744    Degree in Networking            DIN Group 2
   7. Junwei              2104875    Degree in Networking            DIN Group 1
   8. May Teng            2104551    Degree in Marketing             DIM Group 1
   9. Wei Xian            2108451    Degree in Networking            DIN Group 2
  10. Shun                2103801    Degree in Networking            DIN Group 1
  11. Rain                2103820    Degree in Networking            DIN Group 1
  12. Choo Fan Chi        2309309    Degree in Computer Science      DCS Group 2

Enter Position by using number: 12
```

It will first show the user an existing student with a tutorial group list and ask the user to enter the student representative number to change that particular student tutorial group.

```
Tutorial Groups for Degree in Computer Science
------------------
1. DCS Group 1
2. DCS Group 2
Enter Tutorial Group: 1
Are you Confirm to Edit? (y/n): y
Successful Edited Student:Choo Fan Chi
```

After choose the student that would like to change the tutorial group,it will ask the users to enter the tutorial group that would like to change for that student.Then, a confirmation will be asked .If the confirmation is yes,it will show the users a successful message.

Find A Student In Tutorial Group

```
TUTORIAL GROUP MANAGEMENT MENU
1. Add a Student to a Tutorial Group
2. Remove Student from Tutorial Group
3. Change the Tutorial Group for a Student
4. Find a Student in a Tutorial Group
5. List all Students in a Tutorial Group
6. Filter tutorial groups based on criteria
7. Generate relevant reports
0. Quit
Enter choice (0-7): 4

Find Student
1.By Name
2.By ID
0. Quit to TUTORIAL GROUP MANAGEMENT MENU
Enter choice: 1
Enter Student Name: Mei Fan Chi
Student not found
```

Users can enter to find a student in a tutorial group function by entering choice 4 in the tutorial group management menu.
First,the user is needed to enter the choice to choose what they want to find by name or id.It will prompt an error message if the student is not existing and ask to enter again.

```
Find Student
1.By Name
2.By ID
0. Quit to TUTORIAL GROUP MANAGEMENT MENU
Enter choice: 1
Enter Student Name: Mei Fan Chi
Student not found

Find Student
1.By Name
2.By ID
0. Quit to TUTORIAL GROUP MANAGEMENT MENU
Enter choice: 1
Enter Student Name: Choo Fan Chi
Student Details
Student Name:Choo Fan Chi
Student ID: 2309309
Programme: Degree in Computer Science
TutorialGroup: DCS Group 1
```

The user is required to enter the name of the student if they are choosing using student name to find and the student is existing.Then,it will display the user with the respective student details.

```
Find Student
1.By Name
2.By ID
0. Quit to TUTORIAL GROUP MANAGEMENT MENU
Enter choice: 2
Enter Student ID (7-digit number): 2309309
Student Details
Student Name:Choo Fan Chi
Student ID: 2309309
Programme: Degree in Computer Science
TutorialGroup: DCS Group 1
```

Else,the user is required to enter the id of the student if they are choosing using student id to find and the student is existing.Then,it will display the user with the respective student details.

List All Students In A Tutorial Group

```
TUTORIAL GROUP MANAGEMENT MENU
1. Add a Student to a Tutorial Group
2. Remove Student from Tutorial Group
3. Change the Tutorial Group for a Student
4. Find a Student in a Tutorial Group
5. List all Students in a Tutorial Group
6. Filter tutorial groups based on criteria
7. Generate relevant reports
0. Quit
Enter choice (0-7): 5
```

Users can enter into list all students in a tutorial group function by entering choice 5 in the tutorial group mangement menu.

```
Programme
---------
1. Degree in Computer Science
2. Degree in Hotel Management
3. Degree in Marketing
4. Degree in Networking
0. Quit to Tutorial Group Management
Enter Programme: 1
```

when come into list student in a tutorial group,it will first ask the user to enter which programme of tutorial group they are looking for by enter the choice from 1-4 and 0 for exits.

```
Tutorial Groups for Degree in Computer Science
----------------------
1. DCS Group 1
2. DCS Group 2
0. Quit to Programme.
Enter Tutorial Group: 1
```

Then it will ask which of the tutorial groups that would like to list out.

```
List of Students:
  No. Name                ID         Programme                        Group
-------------------------------------------------------------------------------
   1. Bok Bok             2104120    Degree in Computer Science       DCS Group 1
   2. Choo Choo           2103821    Degree in Computer Science       DCS Group 1
   3. Choo Fan Chi        2309309    Degree in Computer Science       DCS Group 1
```

It will list out all the students in the particular tutorial group if the tutorial is existing.

Filter Tutorial Groups Based On Criteria

```
TUTORIAL GROUP MANAGEMENT MENU
1. Add a Student to a Tutorial Group
2. Remove Student from Tutorial Group
3. Change the Tutorial Group for a Student
4. Find a Student in a Tutorial Group
5. List all Students in a Tutorial Group
6. Filter tutorial groups based on criteria
7. Generate relevant reports
0. Quit
Enter choice (0-7): 6
```

Users can enter into filter tutorial groups based on criteria function by entering choice 6 in the tutorial group management menu.

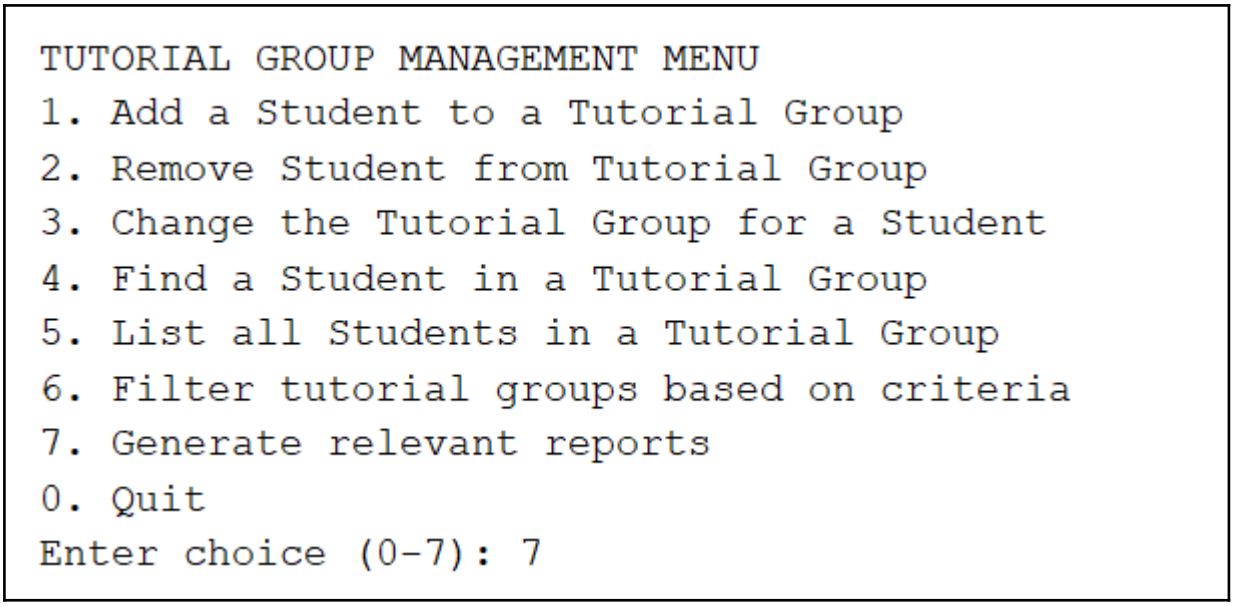
```
Filter Student
1.By Name
2.By ID
3.By Programme
0. Quit to TUTORIAL GROUP MANAGEMENT MENU
Enter choice: 1
```

Then ,it will prompt a filter menu to let the user enter the choice to choose filter by name ,id,or programme .

```
List of Students:
  No. Name                ID         Programme                      Group
--------------------------------------------------------------------------------
   1. Angeline            2103000    Degree in Computer Science     DCS Group 2
   2. Bok Bok             2104120    Degree in Computer Science     DCS Group 1
   3. Choo Choo           2103821    Degree in Computer Science     DCS Group 1
   4. Choo Fan Chi        2309309    Degree in Computer Science     DCS Group 1
   5. Junwei              2104875    Degree in Networking           DIN Group 1
   6. Kyan                2104850    Degree in Networking           DIN Group 1
   7. Louis               2108744    Degree in Networking           DIN Group 2
   8. May Teng            2104551    Degree in Marketing            DIM Group 1
   9. Qian Qian           2104127    Degree in Computer Science     DCS Group 2
  10. Rain                2103820    Degree in Networking           DIN Group 1
  11. Shun                2103801    Degree in Networking           DIN Group 1
  12. Wei Xian            2108451    Degree in Networking           DIN Group 2
```

After that,it will display a list of students based on the user selected filter method.

Generate Relevant Reports



Users can enter to generate relevant report functions by entering choice 7 in the tutorial group management menu.

```
Report:
                    Programme                     Tutorial Group                Students
-------------------------------------------------------------------------------------------

                    Degree in Computer Science
-------------------------------------------------------------------------------------------
                    Degree in Computer Science    DCS Group 1                   3
                    Degree in Computer Science    DCS Group 2                   2
-------------------------------------------------------------------------------------------
Subtotal:                                         2                             5


                    Degree in Hotel Management
-------------------------------------------------------------------------------------------
                                                  No Tutorial Group
-------------------------------------------------------------------------------------------
Subtotal:                                         0                             0


                    Degree in Marketing
-------------------------------------------------------------------------------------------
                    Degree in Marketing           DIM Group 1                   1
                    Degree in Marketing           DIM Group 2                   0
-------------------------------------------------------------------------------------------
Subtotal:                                         2                             1


                    Degree in Networking
-------------------------------------------------------------------------------------------
                    Degree in Networking          DIN Group 1                   4
                    Degree in Networking          DIN Group 2                   2
-------------------------------------------------------------------------------------------
Subtotal:                                         2                             6


-------------------------------------------------------------------------------------------
   Total:           4                             6                             12
-------------------------------------------------------------------------------------------
```

Then,it will display users the report which is programme summary report with total number in programme and tutorial group.

2.4 Tutor Management

Student Name	Student ID	Prog / Tut.Grp	Signature
Choo Shi Yi	2309309	RSD2G3	

2.4.1 Source Code

TutorUI.java

```
 1 package boundary;
 2
 3 import adt.*;
 4 import entity.Tutor;
 5 import java.util.Scanner;
 6 import utility.MessageUI;
 7
 8 /**
 9  *
10  * @author Choo Shi Yi
11  */
12 public class TutorUI {
13
14     Scanner scanner = new Scanner(System.in);
15
16     public int getMenuChoice() {
17         System.out.println("[illegible]\n"
18                 + "[illegible]");
19
20         System.out.println("\n----------------------------------------");
21         System.out.println("Tutor Menu");
22         System.out.println("----------------------------------------");
23         System.out.println("1. Add new tutor ");
24         System.out.println("2. Remove a tutor");
25         System.out.println("3. Find a tutor");
26         System.out.println("4. Amend tutor details");
27         System.out.println("5. List all tutors");
28         System.out.println("6. Filter tutors based on criteria");
29         System.out.println("7. Generate relevant reports");
30         System.out.println("0. Exit\n");
31         int choice;
32         while (true) {
33             System.out.print("Enter choice (0-7): ");
34             if (scanner.hasNextInt()) {
35                 choice = scanner.nextInt();
36                 scanner.nextLine();
37                 return choice;
38             } else {
39                 scanner.nextLine();
40                 System.out.println("Invalid input. Please enter an
integer.");
```

```
41              }
42          }
43      }
44
45      public void listAllTutor(String outputStr) {
46          System.out.println("------------------------");
47          System.out.println("List Of Tutor");
48          System.out.println("------------------------");
49
50          if (outputStr.isEmpty()) {
51              MessageUI.displayNoTutors();
52          } else {
53              System.out.println(
54
String.format("+--------------------+------------------+--------------------
-+--------------------+------------------+--------------------+-----------
---------+\n"
55                                + "| %-19s | %-17s | %-19s | %-19s | %-17s |
%-19s | %-19s |",
56                                "Tutor ID", "Tutor Name", "Contact Number",
"Email", "Specialization", "Experience Year", "Tutor type"));
57
System.out.println("+====================+==================+===============
======+====================+==================+====================+=======
==============+");
58              System.out.print(outputStr);
59          }
60
61
System.out.println("+--------------------+------------------+---------------
------+--------------------+------------------+--------------------+-------
--------------+");
62      }
63
64      public void printTutorDetails(Tutor tutor) {
65
66          System.out.println(
67
String.format("+--------------------+------------------+--------------------
-+--------------------+------------------+--------------------+-----------
---------+\n"
68                            + "| %-19s | %-17s | %-19s | %-19s | %-17s | %-19s
| %-19s |",
```

```
 69                          "Tutor ID", "Tutor Name", "Contact Number",
"Email", "Specialization", "Experience Year", "Tutor type"));
 70
System.out.println("+====================+==================+==============
======+====================+==================+====================+=======
==============+");
 71          System.out.println(
 72                  String.format("| %-19s | %-17s | %-19s | %-19s | %-17s |
%-19s | %-19s |",
 73                          tutor.getTutorId(), tutor.getTutorName(),
tutor.getTutorContactNo(), tutor.getTutorEmail(),
 74                          tutor.getTutorSpecialization(),
tutor.getTutorExpYear(), tutor.getTutorType()));
 75
 76
System.out.println("+--------------------+------------------+--------------
------+--------------------+------------------+--------------------+-------
--------------+");
 77      }
 78
 79      public String inputTutorId() {
 80          System.out.print("Enter tutor ID: ");
 81          String tutorId = scanner.nextLine().toUpperCase();
 82
 83          while (tutorId.isEmpty() || !tutorId.matches("^T\\d{4}$")) {
 84              if (tutorId.isEmpty()) {
 85                  MessageUI.displayEmpty();
 86              } else {
 87                  MessageUI.displayIdInvalid();
 88              }
 89              System.out.print("Enter tutor ID: ");
 90              tutorId = scanner.nextLine().toUpperCase();
 91          }
 92          return tutorId;
 93      }
 94
 95      public String inputTutorName() {
 96          System.out.print("Enter tutor name: ");
 97          String tutorName = scanner.nextLine().toUpperCase();
 98
 99          while (tutorName.isEmpty()) {
100              MessageUI.displayEmpty();
101              System.out.print("Enter tutor name: ");
```

```
102             tutorName = scanner.nextLine();
103         }
104         return tutorName;
105     }
106 
107     public String inputTutorContactNo() {
108         System.out.print("Enter tutor contact no: ");
109         String tutorContactNo = scanner.nextLine();
110 
111         while (tutorContactNo.isEmpty() ||
!tutorContactNo.matches("^\\d{3}-\\d{7,8}$")) {
112             if (tutorContactNo.isEmpty()) {
113                 MessageUI.displayEmpty();
114             } else {
115                 MessageUI.displayCnInvalid();
116             }
117             System.out.print("Enter tutor contact no: ");
118             tutorContactNo = scanner.nextLine();
119         }
120 
121         return tutorContactNo;
122     }
123 
124     public String inputTutorEmail() {
125         System.out.print("Enter tutor email: ");
126         String tutorEmail = scanner.nextLine();
127 
128         while (tutorEmail.isEmpty() || !tutorEmail.contains("@")) {
129             if (tutorEmail.isEmpty()) {
130                 MessageUI.displayEmpty();
131             } else {
132                 MessageUI.displayEmailInvalid();
133             }
134             System.out.print("Enter tutor email: ");
135             tutorEmail = scanner.nextLine();
136         }
137         return tutorEmail;
138     }
139 
140     public String inputTutorSpecialization() {
141         System.out.print("Enter tutor specialization: ");
142         String tutorSubject = scanner.nextLine();
143 
```

```
144         while (tutorSubject.isEmpty()) {
145             MessageUI.displayEmpty();
146             System.out.print("Enter tutor subject: ");
147             tutorSubject = scanner.nextLine();
148         }
149         return tutorSubject;
150     }
151 
152     public int inputTutorExpYear() {
153         int tutorExpYear = 0;
154 
155         do {
156             System.out.print("Enter tutor experience year(Exp:1,2,5): ");
157             String expYearInput = scanner.nextLine();
158             if (expYearInput.isEmpty()) {
159                 MessageUI.displayEmpty();
160             } else if (expYearInput.contains(".")) {
161                 MessageUI.displayExpInvalid();
162             } else {
163                 try {
164                     tutorExpYear = Integer.parseInt(expYearInput);
165                     if (tutorExpYear <= 0) {
166                         MessageUI.displayPositiveInt();
167                     }
168                 } catch (NumberFormatException e) {
169                     MessageUI.displayExpFormat();
170                 }
171             }
172         } while (tutorExpYear <= 0);
173         return tutorExpYear;
174     }
175 
176     public String inputTutorType() {
177         System.out.print("Enter tutor type(FT/PT): ");
178         String tutorType = scanner.nextLine().toUpperCase();
179 
180         while (tutorType.isEmpty() || (!tutorType.equals("FT") &&
!tutorType.equals("PT"))) {
181             if (tutorType.isEmpty()) {
182                 MessageUI.displayEmpty();
183             } else {
184                 MessageUI.displayTypeInvalid();
185             }
```

```
186             System.out.print("Enter tutor type(FT/PT): ");
187             tutorType = scanner.nextLine().toUpperCase();
188         }
189 
190         return tutorType;
191     }
192 
193     public Tutor inputTutorDetails() {
194         String tutorId;
195         String tutorName;
196         String tutorContactNo;
197         String tutorEmail;
198         String tutorSpecialization;
199         int tutorExpYear = 0;
200         String tutorType;
201 
202         boolean isValidTutorId = false;
203         boolean isValidTutorName = false;
204         boolean isValidTutorContactNo = false;
205         boolean isValidTutorEmail = false;
206         boolean isValidTutorSpecialization = false;
207         boolean isValidTutorExpYear = false;
208         boolean isValidTutorType = false;
209 
210         do {
211             tutorId = inputTutorId().toUpperCase();
212 
213             if (tutorId.isEmpty()) {
214                 MessageUI.displayEmpty();
215             } else if (!tutorId.matches("^T\\d{4}$")) {
216                 MessageUI.displayIdInvalid();
217             } else {
218                 isValidTutorId = true;
219             }
220         } while (!isValidTutorId);
221 
222         do {
223             tutorName = inputTutorName().toUpperCase();
224 
225             if (tutorName.isEmpty()) {
226                 MessageUI.displayEmpty();
227             } else {
228                 isValidTutorName = true;
```

```
229             }
230         } while (!isValidTutorName);
231 
232         do {
233             tutorContactNo = inputTutorContactNo();
234 
235             if (tutorContactNo.isEmpty()) {
236                 MessageUI.displayEmpty();
237             } else if (!tutorContactNo.matches("^\\d{3}-\\d{7,8}$")) {
238                 MessageUI.displayCnInvalid();
239             } else {
240                 isValidTutorContactNo = true;
241             }
242         } while (!isValidTutorContactNo);
243 
244         do {
245             tutorEmail = inputTutorEmail();
246 
247             if (tutorEmail.isEmpty()) {
248                 MessageUI.displayEmpty();
249             } else if (!tutorEmail.contains("@")) {
250                 MessageUI.displayEmailInvalid();
251             } else {
252                 isValidTutorEmail = true;
253             }
254         } while (!isValidTutorEmail);
255 
256         do {
257             tutorSpecialization = inputTutorSpecialization();
258 
259             if (tutorSpecialization.isEmpty()) {
260                 MessageUI.displayEmpty();
261             } else {
262                 isValidTutorSpecialization = true;
263             }
264         } while (!isValidTutorSpecialization);
265 
266         do {
267 
268             tutorExpYear = inputTutorExpYear();
269             if (tutorExpYear > 0) {
270                 isValidTutorExpYear = true;
271             } else {
```

```
272                 MessageUI.displayTypeInvalid();
273             }
274
275         } while (!isValidTutorExpYear);
276
277         do {
278             tutorType = inputTutorType().toUpperCase();
279
280             if (tutorType.isEmpty()) {
281                 MessageUI.displayEmpty();
282             } else if (!tutorType.equalsIgnoreCase("pt") &&
!tutorType.equalsIgnoreCase("ft")) {
283                 MessageUI.displayTypeInvalid();
284             } else {
285                 isValidTutorType = true;
286             }
287         } while (!isValidTutorType);
288
289         System.out.println();
290
291         return new Tutor(tutorId, tutorName, tutorContactNo, tutorEmail,
tutorSpecialization, tutorExpYear, tutorType);
292     }
293
294     public int inputModifyOption() {
295         System.out.println("Select which information you want to
modify:");
296         System.out.println("1. Tutor Name");
297         System.out.println("2. Tutor Conatct No");
298         System.out.println("3. Tutor Email");
299         System.out.println("4. Tutor Specialization");
300         System.out.println("5. Tutor Experience Year");
301         System.out.println("6. Tutor Type(FT/PT)");
302         System.out.println("7. Exit\n");
303         int option;
304         while (true) {
305             System.out.print("Enter(1-7): ");
306             if (scanner.hasNextInt()) {
307                 option = scanner.nextInt();
308                 scanner.nextLine();
309                 return option;
310             } else {
311                 scanner.nextLine();
```

```
312                 System.out.println("Invalid input. Please enter an
integer.");
313             }
314         }
315     }
316
317     public void printAdd() {
318         System.out.println("----------------------------------------");
319         System.out.println("Add Tutor");
320         System.out.println("----------------------------------------");
321     }
322
323     public void printModify() {
324         System.out.println("----------------------------------------");
325         System.out.println("Modify Tutor");
326         System.out.println("----------------------------------------");
327         System.out.println("Enter the tutor id that want to modify.");
328     }
329
330     public void printRemove() {
331         System.out.println("----------------------------------------");
332         System.out.println("Delete Tutor");
333         System.out.println("----------------------------------------");
334         System.out.println("Enter the tutor Id that want to delete.");
335
336     }
337
338     public int printFind() {
339         System.out.println("----------------------------------------");
340         System.out.println("Find Tutor");
341         System.out.println("----------------------------------------");
342         System.out.println("Search tutor by id or name");
343         System.out.println("1.ID");
344         System.out.println("2.Name");
345         int option;
346         while (true) {
347             System.out.print("Enter(1-2): ");
348             if (scanner.hasNextInt()) {
349                 option = scanner.nextInt();
350                 scanner.nextLine();
351                 return option;
352             } else {
353                 scanner.nextLine();
```

```
354                 System.out.println("Invalid input. Please enter an
integer.");
355             }
356         }
357
358     }
359
360     public int printFilter() {
361         System.out.println("----------------------------------------");
362         System.out.println("Sorting Tutor");
363         System.out.println("----------------------------------------");
364         System.out.println("Select the meyhod of sorting tutor");
365         System.out.println("1.Ascending sorting tutor name");
366         System.out.println("2.Descending sorting tutor name");
367         System.out.println("3.Exit to tutor menu");
368         int option;
369         while (true) {
370             System.out.print("Enter(1-3): ");
371             if (scanner.hasNextInt()) {
372                 option = scanner.nextInt();
373                 return option;
374             } else {
375                 scanner.nextLine();
376                 System.out.println("Invalid input. Please enter an
integer.");
377             }
378         }
379     }
380
381     public static String printSortedList(SortedListInterface<Tutor>
tutors, String sortingMethod) {
382         StringBuilder outputStr = new StringBuilder();
383         String sortType = sortingMethod.equals("ascending") ? "Ascending"
: "Descending";
384
385         outputStr.append(sortType).append(" Sorted List\n");
386         outputStr.append(
387
String.format("+--------------------+------------------+--------------------
-+--------------------+------------------+--------------------+------------
---------+\n"
388                         + "| %-19s | %-17s | %-19s | %-19s | %-17s | %-19s
| %-19s |\n",
```

```
389                         "Tutor ID", "Tutor Name", "Contact Number",
"Email", "Specialization", "Experience Year", "Tutor type"));
390
outputStr.append("+====================+==================+================
====+====================+==================+====================+========
============+\n");
391
392         for (int i = 0; i < tutors.getNumberOfEntries(); i++) {
393             Tutor tutor = tutors.getEntry(i);
394             outputStr.append(
395                     String.format("| %-19s | %-17s | %-19s | %-19s | %-17s
| %-19s | %-19s |\n",
396                             tutor.getTutorId(), tutor.getTutorName(),
tutor.getTutorContactNo(), tutor.getTutorEmail(),
397                             tutor.getTutorSpecialization(),
tutor.getTutorExpYear(), tutor.getTutorType()));
398         }
399
outputStr.append("+--------------------+------------------+----------------
----+--------------------+------------------+--------------------+---------
------------+\n");
400
401         return outputStr.toString();
402     }
403
404     public int printReport() {
405         System.out.println("-----------------------------------------");
406         System.out.println("Tutor Report");
407         System.out.println("-----------------------------------------");
408         System.out.println("Select the type of report");
409         System.out.println("1.Part-Time/Full-Time Tutor Report");
410         System.out.println("2.Experienced Tutor Report");
411         System.out.println("3.Exit to tutor menu");
412         int option;
413         while (true) {
414             System.out.print("Enter(1-3): ");
415             if (scanner.hasNextInt()) {
416                 option = scanner.nextInt();
417                 return option;
418             } else {
419                 scanner.nextLine();
420                 System.out.println("Invalid input. Please enter an
integer.");
```

```
421             }
422         }
423
424     }
425
426     public void printFtTutors(ListInterface<Tutor> tutors) {
427         int totalCount = tutors.getNumberOfEntries(); // Count the total
number of full-time tutors
428
429         System.out.println("----------------------------------------");
430         System.out.println("Full Time Tutor");
431         System.out.println("----------------------------------------");
432         System.out.println(
433
String.format("+--------------------+------------------+--------------------
-+--------------------+------------------+--------------------+------------
---------+\n"
434                         + "| %-19s | %-17s | %-19s | %-19s | %-17s | %-19s
| %-19s |",
435                         "Tutor ID", "Tutor Name", "Contact Number",
"Email", "Specialization", "Experience Year", "Tutor type"));
436
System.out.println("+====================+==================+===============
======+====================+==================+====================+=======
==============+");
437
438         for (int i = 1; i <= tutors.getNumberOfEntries(); i++) {
439             Tutor tutor = tutors.getEntry(i);
440             System.out.println(
441                     String.format("| %-19s | %-17s | %-19s | %-19s | %-17s
| %-19s | %-19s |",
442                             tutor.getTutorId(), tutor.getTutorName(),
tutor.getTutorContactNo(), tutor.getTutorEmail(),
443                             tutor.getTutorSpecialization(),
tutor.getTutorExpYear(), tutor.getTutorType()));
444         }
445
446         // Display the total count at the bottom of the table
447
System.out.println("+--------------------+------------------+--------------
------+--------------------+------------------+--------------------+-------
--------------+");
448         System.out.println(String.format("| Total Tutor: %-134d |",
```

```
totalCount));
449
System.out.println("+---------------------+------------------+--------------
------+---------------------+------------------+--------------------+-------
--------------+");
450     }
451
452     public void printPtTutors(ListInterface<Tutor> tutors) {
453         int totalCount = tutors.getNumberOfEntries();
454         System.out.println("----------------------------------------");
455         System.out.println("Part-Time Tutor");
456         System.out.println("----------------------------------------");
457         System.out.println(
458
String.format("+---------------------+------------------+--------------------
-+---------------------+------------------+--------------------+------------
---------+\n"
459                         + "| %-19s | %-17s | %-19s | %-19s | %-17s | %-19s
| %-19s |",
460                         "Tutor ID", "Tutor Name", "Contact Number",
"Email", "Specialization", "Experience Year", "Tutor type"));
461
System.out.println("+=====================+==================+==============
======+=====================+==================+====================+=======
==============+");
462
463         //for (Tutor tutor : tutors) {
464         for (int i = 1; i <= tutors.getNumberOfEntries(); i++) {
465             Tutor tutor = tutors.getEntry(i);
466             System.out.println(
467                     String.format("| %-19s | %-17s | %-19s | %-19s | %-17s
| %-19s | %-19s |",
468                             tutor.getTutorId(), tutor.getTutorName(),
tutor.getTutorContactNo(), tutor.getTutorEmail(),
469                             tutor.getTutorSpecialization(),
tutor.getTutorExpYear(), tutor.getTutorType()));
470         }
471
System.out.println("+---------------------+------------------+--------------
------+---------------------+------------------+--------------------+-------
--------------+");
472
473         System.out.println(String.format("| Total Tutor: %-134d |",
```

```
totalCount));
474
System.out.println("+--------------------+------------------+--------------
------+--------------------+------------------+--------------------+-------
--------------+");
475     }
476
477     public void displayTutorCategory(String categoryHeader,
ListInterface<Tutor> tutors) {
478         int totalCount = tutors.getNumberOfEntries();
479
480         System.out.println("----------------------------------------");
481         System.out.println(categoryHeader);
482         System.out.println("----------------------------------------");
483         System.out.println(
484
String.format("+--------------------+------------------+--------------------
-+--------------------+------------------+--------------------+------------
---------+\n"
485                         + "| %-19s | %-17s | %-19s | %-19s | %-17s | %-19s
| %-19s |",
486                         "Tutor ID", "Tutor Name", "Contact Number",
"Email", "Specialization", "Experience Year", "Tutor type"));
487
System.out.println("+====================+==================+==============
======+====================+==================+====================+=======
==============+");
488
489         for (int i = 1; i <= tutors.getNumberOfEntries(); i++) {
490             Tutor tutor = tutors.getEntry(i);
491             System.out.println(
492                     String.format("| %-19s | %-17s | %-19s | %-19s | %-17s
| %-19s | %-19s |",
493                             tutor.getTutorId(), tutor.getTutorName(),
tutor.getTutorContactNo(), tutor.getTutorEmail(),
494                             tutor.getTutorSpecialization(),
tutor.getTutorExpYear(), tutor.getTutorType()));
495         }
496
497         // Display the total count at the bottom of the table
498
System.out.println("+--------------------+------------------+--------------
------+--------------------+------------------+--------------------+-------
```

```
--------------+");
499         System.out.println(String.format("| Total Count: %-134d |",
totalCount));
500
System.out.println("+--------------------+------------------+--------------
------+--------------------+------------------+--------------------+-------
--------------+");
501     }
502
503     public String inputContinue() {
504         System.out.print("Do you want to continue to manage tutor?(Y/N):
");
505         String inputContinue = scanner.nextLine().toUpperCase();
506         return inputContinue;
507     }
508
509 }
```

TutorMaintenance.java

```
 1 package control;
 2
 3 import adt.*;
 4 import dao.TutorDAO;
 5 import boundary.TutorUI;
 6 import entity.Tutor;
 7 import java.util.Scanner;
 8 import utility.MessageUI;
 9
10 /**
11  *
12  * @author Choo Shi Yi
13  */
14 public class TutorMaintenance {
15
16     Scanner scanner = new Scanner(System.in);
17     private SortedListInterface<Tutor> tutorList = new
SortedArrayList<>();
18     private final TutorDAO tutorDAO = new TutorDAO();
19     private final TutorUI tutorUI = new TutorUI();
20     UniversityManagement um = new UniversityManagement();
21
22     public TutorMaintenance() {
23         tutorList = tutorDAO.retrieveFromFile();
24     }
25
26     public void runTutorMaintenance() {
27         int choice = 0;
28         do {
29             choice = tutorUI.getMenuChoice();
30             switch (choice) {
31                 case 0:
32                     MessageUI.displayExitProgram();
33                     //System.exit(0);
34                     break;
35                 case 1:
36                     tutorUI.printAdd();
37                     addNewTutor();
38                     break;
39                 case 2:
40                     tutorUI.listAllTutor(getAllTutor(tutorList));
41                     tutorUI.printRemove();
42                     removeTutor();
```

```
43                      break;
44                  case 3:
45                      findTutor();
46                      break;
47                  case 4:
48                      tutorUI.listAllTutor(getAllTutor(tutorList));
49                      tutorUI.printModify();
50                      amendTutor();
51                      break;
52                  case 5:
53                      tutorUI.listAllTutor(getAllTutor(tutorList));
54                      break;
55                  case 6:
56                      filterTutor(tutorList);
57                      break;
58                  case 7:
59                      tutorReport();
60                      break;
61                  default:
62                      MessageUI.displayInvalid();
63              }
64          } while (choice != 0);
65      }
66
67      public void addNewTutor() {
68          // Input details for the new tutor
69          Tutor newTutor = tutorUI.inputTutorDetails();
70          String tutorId = newTutor.getTutorId(); // Get the tutor ID of the
new tutor
71
72          while (isTutorIdExists(tutorId)) {
73              System.out.println("A tutor with the same ID already exists.
Please choose a different tutor ID.");
74              newTutor = tutorUI.inputTutorDetails();
75              tutorId = newTutor.getTutorId();
76          }
77
78          tutorUI.printTutorDetails(newTutor);
79          String addConfirm;
80          String choice = "";
81
82          do {
83              System.out.print("Are you sure to add this tutor (Y/N)? ");
```

```
84              addConfirm = scanner.nextLine().trim().toUpperCase();
85
86              if (addConfirm.equals("Y")) {
87                  tutorList.add(newTutor);
88                  tutorDAO.saveToFile(tutorList);
89                  MessageUI.displaySuccessfulAdd();
90                  tutorUI.listAllTutor(getAllTutor(tutorList));
91                  choice = tutorUI.inputContinue();
92                  if (!choice.equals("Y")) {
93                      MessageUI.displayExitProgram();
94                      um.runMenu();
95                  }
96              } else if (addConfirm.equals("N")) {
97                  MessageUI.displayExitAdd();
98              } else if (addConfirm.isEmpty()) {
99                  MessageUI.displayEmpty();
100             } else {
101                 MessageUI.displayInvalid();
102             }
103         } while (!addConfirm.equals("N") && !addConfirm.equals("Y") &&
addConfirm.isEmpty());
104     }
105
106     private boolean isTutorIdExists(String tutorId) {
107         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
108             Tutor existingTutor = tutorList.getEntry(i);
109             if (existingTutor.getTutorId().equalsIgnoreCase(tutorId)) {
110                 return true;
111             }
112         }
113         return false;
114     }
115
116     public void removeTutor() {
117         String removeTutorId = tutorUI.inputTutorId();
118         boolean tutorFound = false;
119
120         for (int i = 1; i <= tutorList.getNumberOfEntries(); i++) {
121             if (removeTutorId.equals(tutorList.getEntry(i).getTutorId()))
{
122                 Tutor removeTutor = tutorList.getEntry(i);
123                 tutorUI.printTutorDetails(removeTutor);
124                 tutorFound = true;
```

```
125                 break;
126             }
127         }
128
129         if (!tutorFound) {
130             System.out.println("Sorry, tutor could not be found or
removed");
131         } else {
132             String deleteConfirm;
133
134             do {
135                 System.out.print("\nAre you sure to remove this tutor
(Y/N)? ");
136                 deleteConfirm = scanner.next().toUpperCase();
137
138                 if (deleteConfirm.isEmpty()) {
139                     MessageUI.displayEmpty();
140                 } else if (!deleteConfirm.equals("Y") &&
!deleteConfirm.equals("N")) {
141                     MessageUI.displayInvalid();
142                 }
143             } while (deleteConfirm.isEmpty() ||
(!deleteConfirm.equals("Y") && !deleteConfirm.equals("N")));
144
145             if (deleteConfirm.equals("Y")) {
146                 for (int i = 1; i <= tutorList.getNumberOfEntries(); i++)
{
147                     if
(removeTutorId.equals(tutorList.getEntry(i).getTutorId())) {
148                         Tutor tutorRemove = tutorList.getEntry(i);
149                         tutorList.remove(tutorRemove);
150                         tutorDAO.saveToFile(tutorList);
151                         MessageUI.displaySuccessfulDelete();
152                         String choice = " ";
153                         choice = tutorUI.inputContinue();
154                         if (!choice.equals("Y")) {
155                             MessageUI.displayExitProgram();
156                             um.runMenu();
157                         }
158                     }
159                 }
160             } else {
161                 MessageUI.displayExitDelete();
```

```
162             }
163         }
164     }
165
166     public void amendTutor() {
167         String modifyTutor = tutorUI.inputTutorId();
168
169         Tutor modifyTutor1 = new Tutor();
170         String newTutorName = "";
171         String newTutorEmail = "";
172         String newTutorContactNo = "";
173         String newTutorType = "";
174         int newTutorExpYear = 0;
175         String newTutorSpecialization = "";
176
177         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
178             if (modifyTutor.equals(tutorList.getEntry(i).getTutorId())) {
179                 modifyTutor1 = tutorList.getEntry(i);
180                 tutorUI.printTutorDetails(modifyTutor1);
181                 newTutorName = modifyTutor1.getTutorName();
182                 newTutorEmail = modifyTutor1.getTutorEmail();
183                 newTutorContactNo = modifyTutor1.getTutorContactNo();
184                 newTutorType = modifyTutor1.getTutorType();
185                 newTutorExpYear = modifyTutor1.getTutorExpYear();
186                 newTutorSpecialization =
modifyTutor1.getTutorSpecialization();
187             }
188         }
189
190         int choice = 8;
191         do {
192             choice = tutorUI.inputModifyOption();
193             switch (choice) {
194                 case 0:
195                     break;
196                 case 1:
197                     newTutorName = tutorUI.inputTutorName();
198                     break;
199                 case 2:
200                     newTutorContactNo = tutorUI.inputTutorContactNo();
201                     break;
202                 case 3:
203                     newTutorEmail = tutorUI.inputTutorEmail();
```

```
204                     break;
205                 case 4:
206                     newTutorSpecialization =
tutorUI.inputTutorSpecialization();
207                     break;
208                 case 5:
209                     newTutorExpYear = tutorUI.inputTutorExpYear();
210                     break;
211                 case 6:
212                     newTutorType = tutorUI.inputTutorType();
213                     break;
214                 case 7:
215                     MessageUI.displayExitEdit();
216                     break;
217                 default:
218                     MessageUI.displayInvalid();
219             }
220         } while (choice > 0 && choice < 7);
221 
222         String modifyConfirm;
223         do {
224             System.out.print("Are you sure to modify this tutor (Y/N)? ");
225             modifyConfirm = scanner.next().toUpperCase();
226 
227             if (modifyConfirm.isEmpty()) {
228                 MessageUI.displayEmpty();
229             } else if (!modifyConfirm.equals("Y") &&
!modifyConfirm.equals("N")) {
230                 MessageUI.displayInvalid();
231             }
232         } while (modifyConfirm.isEmpty() || (!modifyConfirm.equals("Y") &&
!modifyConfirm.equals("N")));
233 
234         String choice1 = "";
235         if (modifyConfirm.equals("Y")) {
236             modifyTutor1.setTutorName(newTutorName);
237             modifyTutor1.setTutorContactNo(newTutorContactNo);
238             modifyTutor1.setTutorEmail(newTutorEmail);
239             modifyTutor1.setTutorSpecialization(newTutorSpecialization);
240             modifyTutor1.setTutorExpYear(newTutorExpYear);
241             modifyTutor1.setTutorType(newTutorType);
242 
243             tutorDAO.saveToFile(tutorList);
```

```
244             MessageUI.displaySuccessfulModify();
245             choice1 = tutorUI.inputContinue();
246             if (!choice1.equals("Y")) {
247                 MessageUI.displayExitProgram();
248                 um.runMenu();
249             }
250         }
251     }
252 
253     public String getAllTutor(SortedListInterface<Tutor> tutorList1) {
254         String outputStr = "";
255 
256         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
257             outputStr += tutorList.getEntry(i) + "\n";
258         }
259         return outputStr;
260     }
261 
262     public void displayTutor() {
263         tutorUI.listAllTutor(getAllTutor(tutorList));
264 
265     }
266 
267     private int choice = 0;
268 
269     public void findTutor() {
270         do {
271             choice = tutorUI.printFind();
272 
273             if (choice == 0) {
274                 MessageUI.displayInvalid();
275             } else if (choice == -1) {
276                 MessageUI.displayInvalid();
277             } else {
278                 switch (choice) {
279                     case 1:
280                         findTutorById();
281                         break;
282                     case 2:
283                         findTutorByName();
284                         break;
285                     default:
286                         MessageUI.displayInvalid();
```

```
287                         break;
288                 }
289             }
290         } while (choice != 1 && choice != 2);
291     }
292
293     public void findTutorById() {
294         String findTutorId = tutorUI.inputTutorId();
295         Tutor foundTutor = null;
296         String choice = "";
297         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
298             if
(tutorList.getEntry(i).getTutorId().contentEquals(findTutorId)) {
299                 foundTutor = tutorList.getEntry(i);
300                 tutorUI.printTutorDetails(foundTutor);
301                 choice = tutorUI.inputContinue();
302                 if (!choice.equals("Y")) {
303                     MessageUI.displayExitProgram();
304                     um.runMenu();
305                 }
306             }
307         }
308         if (foundTutor == null) {
309             System.out.println("The tutor you found does not exist!");
310         }
311     }
312
313     public void findTutorByName() {
314         String findTutorName = tutorUI.inputTutorName().toUpperCase(); //
Convert input to uppercase
315         ListInterface<Tutor> matchingTutors = new ArrayList<>();
316
317         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
318             String tutorName =
tutorList.getEntry(i).getTutorName().toUpperCase(); // Convert stored name to
uppercase
319
320             if (tutorName.contains(findTutorName)) { // Check if the
stored name contains the input substring
321                 matchingTutors.add(tutorList.getEntry(i));
322             }
323         }
324         if (!matchingTutors.isEmpty()) {
```

```
325             for (int i = 1; i <= matchingTutors.getNumberOfEntries(); i++)
{
326                 Tutor foundTutor = matchingTutors.getEntry(i);
327                 tutorUI.printTutorDetails(foundTutor);
328                 String choice = tutorUI.inputContinue();
329                 if (!choice.equals("Y")) {
330                     MessageUI.displayExitProgram();
331                     um.runMenu();
332                 }
333             }
334         } else {
335             System.out.println("No matching tutor found!");
336         }
337     }
338 
339     public String filterTutor(SortedListInterface<Tutor> tutorList1) {
340         String outputStr = "";
341         int choice;
342 
343         do {
344             choice = tutorUI.printFilter();
345 
346             if (choice == 0 || choice > 3) {
347                 System.out.println("Invalid choice. Please enter a valid
option.");
348             } else {
349                 switch (choice) {
350                     case 1:
351                         tutorList1.selectionSortAscending();
352                         outputStr = tutorUI.printSortedList(tutorList1,
"ascending");
353                         System.out.println(outputStr);
354                         break;
355                     case 2:
356                         tutorList1.selectionSortDescending();
357                         outputStr = tutorUI.printSortedList(tutorList1,
"descending");
358                         System.out.println(outputStr);
359                         break;
360                     case 3:
361                         // Exit
362                         break;
363                 }
```

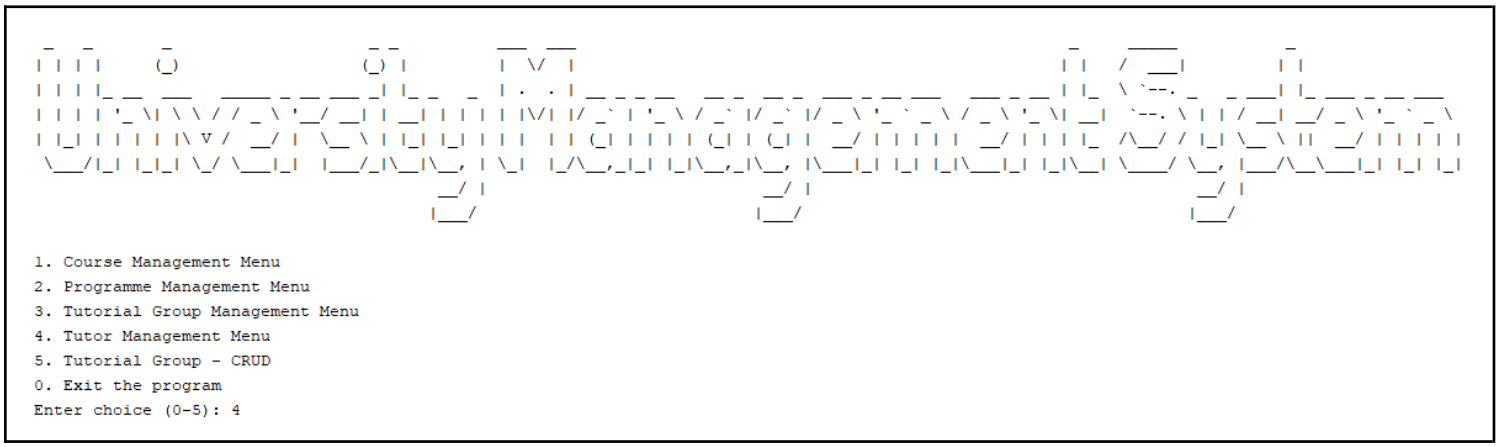
```
364             }
365         } while (choice != 3);
366 
367         return outputStr;
368     }
369 
370     public void tutorReport() {
371         int reportChoice;
372 
373         do {
374             reportChoice = tutorUI.printReport();
375 
376             switch (reportChoice) {
377                 case 1:
378                     fullTimeTutors();
379                     partTimeTutors();
380                     break;
381                 case 2:
382                     experiencedTutorReport();
383                     break;
384                 case 3:
385 
386                     break;
387                 default:
388                     MessageUI.displayInvalid();
389                     break;
390             }
391         } while (reportChoice != 1 && reportChoice != 2 && reportChoice !=
3);
392     }
393 
394     public void fullTimeTutors() {
395         ListInterface<Tutor> fullTimeTutors = new ArrayList<>();
396 
397         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
398             Tutor tutor = tutorList.getEntry(i);
399 
400             if (tutor.getTutorType().equalsIgnoreCase("FT")) {
401                 fullTimeTutors.add(tutor);
402             }
403         }
404 
405         if (!fullTimeTutors.isEmpty()) {
```

```
406             tutorUI.printFtTutors(fullTimeTutors);
407         } else {
408             System.out.println("No full-time tutors found!");
409         }
410     }
411 
412     public void partTimeTutors() {
413         ListInterface<Tutor> partTimeTutors = new ArrayList<>();
414 
415         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
416             Tutor tutor = tutorList.getEntry(i);
417 
418             if (tutor.getTutorType().equalsIgnoreCase("PT")) {
419                 partTimeTutors.add(tutor);
420             }
421         }
422 
423         if (!partTimeTutors.isEmpty()) {
424             tutorUI.printPtTutors(partTimeTutors);
425         } else {
426             System.out.println("No full-time tutors found!");
427         }
428     }
429 
430     public void experiencedTutorReport() {
431         ListInterface<Tutor> lessThan2Years = new ArrayList<>();
432         ListInterface<Tutor> between2And5Years = new ArrayList<>();
433         ListInterface<Tutor> moreThan5Years = new ArrayList<>();
434         String categoryHeader1 = "Tutor Experience Less Than 2 Years";
435         String categoryHeader2 = "Tutor Experience Between 2 And 5 Years";
436         String categoryHeader3 = "Tutor Experience More Than 5Years";
437 
438         for (int i = 0; i < tutorList.getNumberOfEntries(); i++) {
439             Tutor tutor = tutorList.getEntry(i);
440 
441             int experience = tutor.getTutorExpYear();
442             if (experience < 2) {
443                 lessThan2Years.add(tutor);
444             } else if (experience >= 2 && experience < 5) {
445                 between2And5Years.add(tutor);
446             } else if (experience > 5) {
447                 moreThan5Years.add(tutor);
448             }
```

```
449         }
450         tutorUI.displayTutorCategory(categoryHeader1, lessThan2Years);
451         tutorUI.displayTutorCategory(categoryHeader2, between2And5Years);
452         tutorUI.displayTutorCategory(categoryHeader3, moreThan5Years);
453     }
454
455 }
```

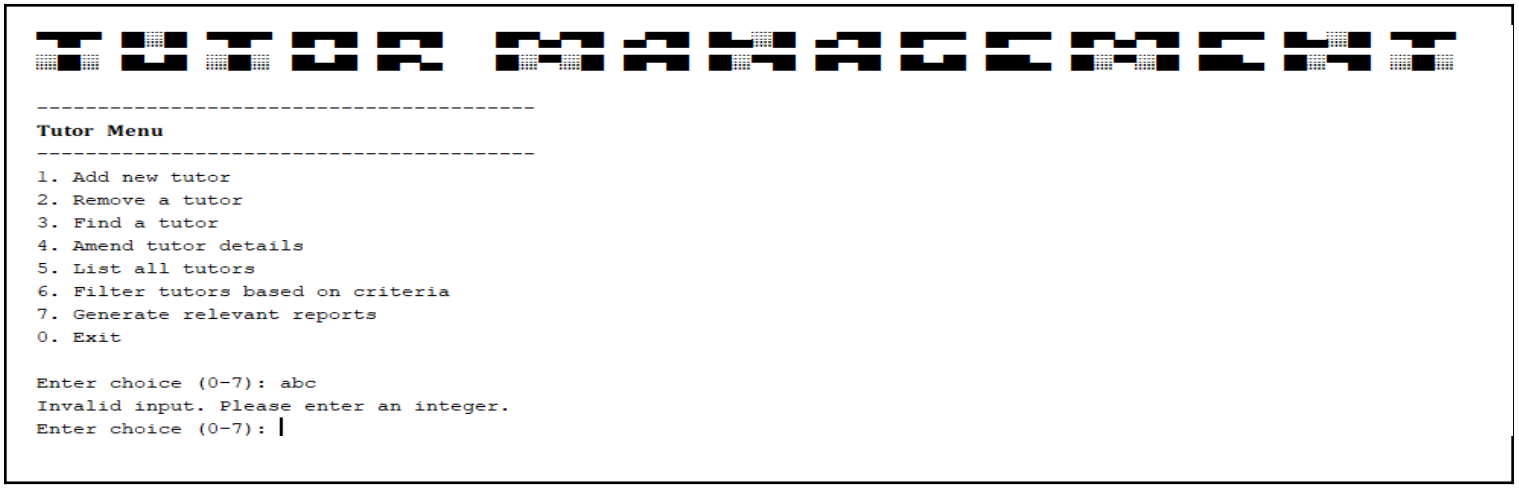
2.4.2 Screenshot of sample user interface

University Management System Main Menu



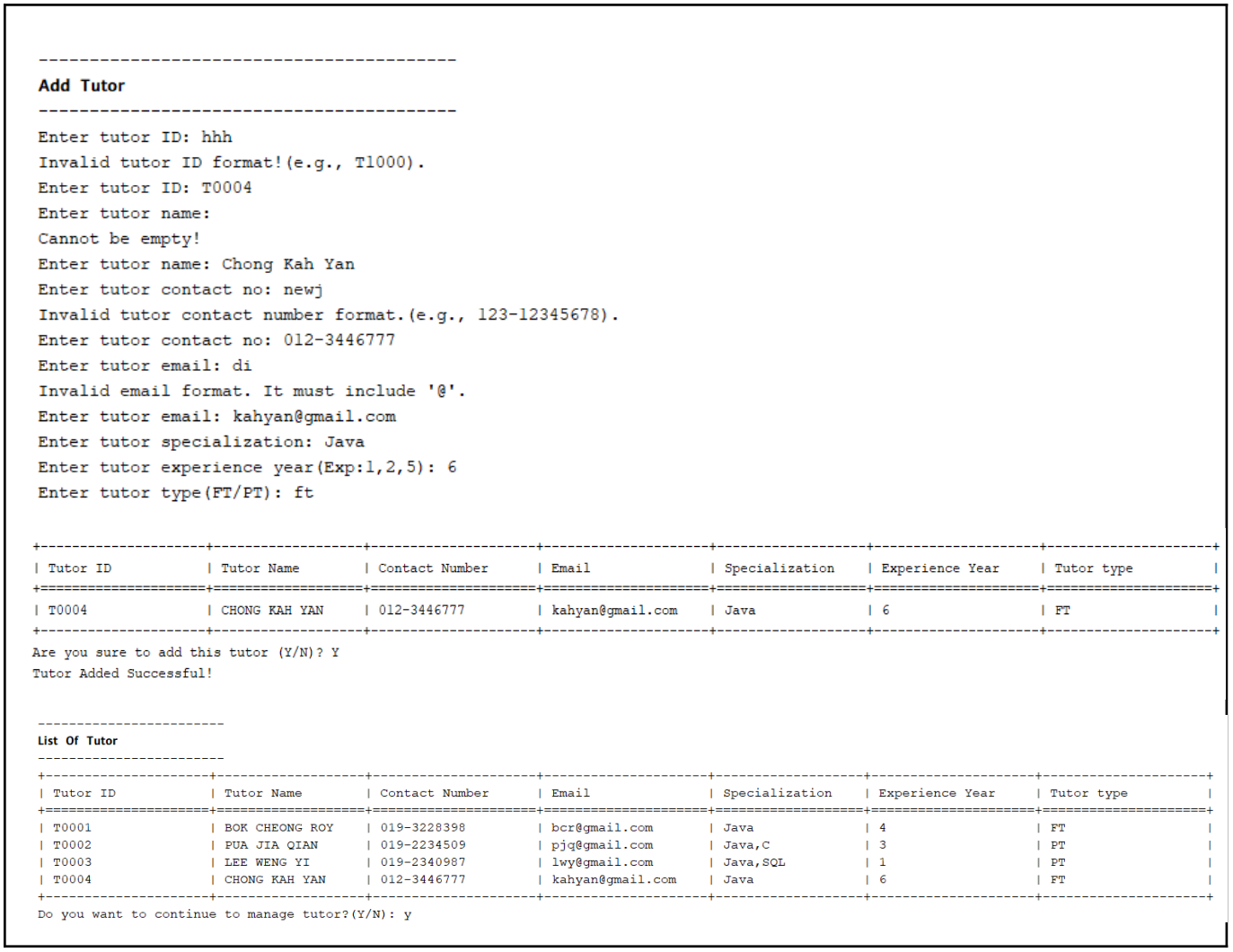
User entered 4 in the main menu will redirect the user to the Tutor management menu.

Tutor management menu



When going into the Tutor management,it will first show the Tutor management title and also display a tutor menu.The tutor menu will let user to choice the 7 option of function in the tutor management and enter 0 to exit.It will only accept integer from 0-7 and it will prompt an error and ask user to to reenter .

1.Add Tutor

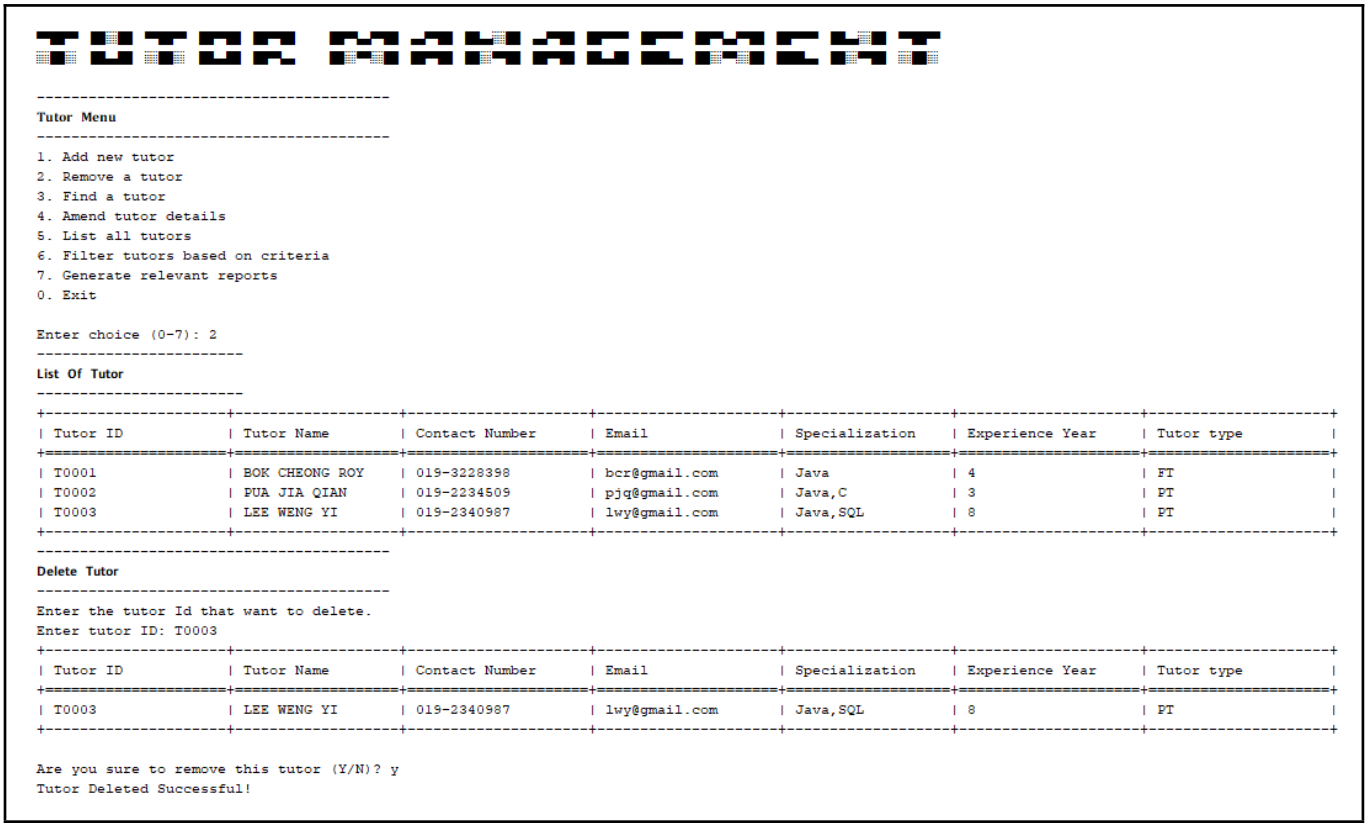


```
-----------------------------------------
Add Tutor
-----------------------------------------
Enter tutor ID: hhh
Invalid tutor ID format!(e.g., T1000).
Enter tutor ID: T0004
Enter tutor name:
Cannot be empty!
Enter tutor name: Chong Kah Yan
Enter tutor contact no: newj
Invalid tutor contact number format.(e.g., 123-12345678).
Enter tutor contact no: 012-3446777
Enter tutor email: di
Invalid email format. It must include '@'.
Enter tutor email: kahyan@gmail.com
Enter tutor specialization: Java
Enter tutor experience year(Exp:1,2,5): 6
Enter tutor type(FT/PT): ft

+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name       | Contact Number     | Email              | Specialization   | Experience Year    | Tutor type         |
+====================+==================+====================+====================+==================+====================+====================+
| T0004              | CHONG KAH YAN    | 012-3446777        | kahyan@gmail.com   | Java             | 6                  | FT                 |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
Are you sure to add this tutor (Y/N)? Y
Tutor Added Successful!


------------------------
List Of Tutor
------------------------
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name       | Contact Number     | Email              | Specialization   | Experience Year    | Tutor type         |
+====================+==================+====================+====================+==================+====================+====================+
| T0001              | BOK CHEONG ROY   | 019-3228398        | bcr@gmail.com      | Java             | 4                  | FT                 |
| T0002              | PUA JIA QIAN     | 019-2234509        | pjq@gmail.com      | Java,C           | 3                  | PT                 |
| T0003              | LEE WENG YI      | 019-2340987        | lwy@gmail.com      | Java,SQL         | 1                  | PT                 |
| T0004              | CHONG KAH YAN    | 012-3446777        | kahyan@gmail.com   | Java             | 6                  | FT                 |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
Do you want to continue to manage tutor?(Y/N): y
```

It will go into add tutor when user enters option 1 in the tutor menu.The add tutor will let user enter the information of a tutor which is tutor id, tutor name,tutor contact number,tutor email,tutor specialization ,tutor experience and tutor type.It will aslo display error message and let user enter again if there are any error or empty input.For example,there are format required for Id which is need to form with T and 4 integer.Then,the phone number must be only between 10-11 integer with symbol ‘-’.Then email aslo conataint format required which is need to include @.

Then it will ask for confirmation to add a user.If yes,it will display the latest list of users and ask users whether they want to continue to manage tutor.if no,it will exit from add user to tutor menu.

2.Delete User



When user entered option 2 in the tutor menu,it will display a list of existing tutor list to let user refer when doing delete tutor .The user need to enter the tutor id they would like to delete and the system will list the following tutor’s details and ask for confirmation.Id yes,it will show successful message andask user whether want to continue to manage tutor.if no,it will exit from add user to tutor menu.

3.Find Tutor

```
----------------------------------------
Find Tutor
----------------------------------------
Search tutor by id or name
1.ID
2.Name
Enter(1-2): 1
Enter tutor ID: t0003
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name         | Contact Number     | Email              | Specialization     | Experience Year    | Tutor type         |
+====================+====================+====================+====================+====================+====================+====================+
| T0003              | LEE WENG YI        | 019-2340987        | lwy@gmail.com      | Java,SQL           | 1                  | PT                 |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
Do you want to continue to manage tutor?(Y/N): y
```

The user can access the find tutor when enter option 3 in the tutor menu.The system will first display a find tutor menu to let user choose whether is find by ID or name.If the user enter option 1,it will ask user to enter the tutor id that want to find and then display the respective tutor details.If the tutor are not exists ,it will prompt error message and ask user to reenter again.

```
----------------------------------------
Find Tutor
----------------------------------------
Search tutor by id or name
1.ID
2.Name
Enter(1-2): 2
Enter tutor name: weng
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name         | Contact Number     | Email              | Specialization     | Experience Year    | Tutor type         |
+====================+====================+====================+====================+====================+====================+====================+
| T0003              | LEE WENG YI        | 019-2340987        | lwy@gmail.com      | Java,SQL           | 1                  | PT                 |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
Do you want to continue to manage tutor?(Y/N): y
```

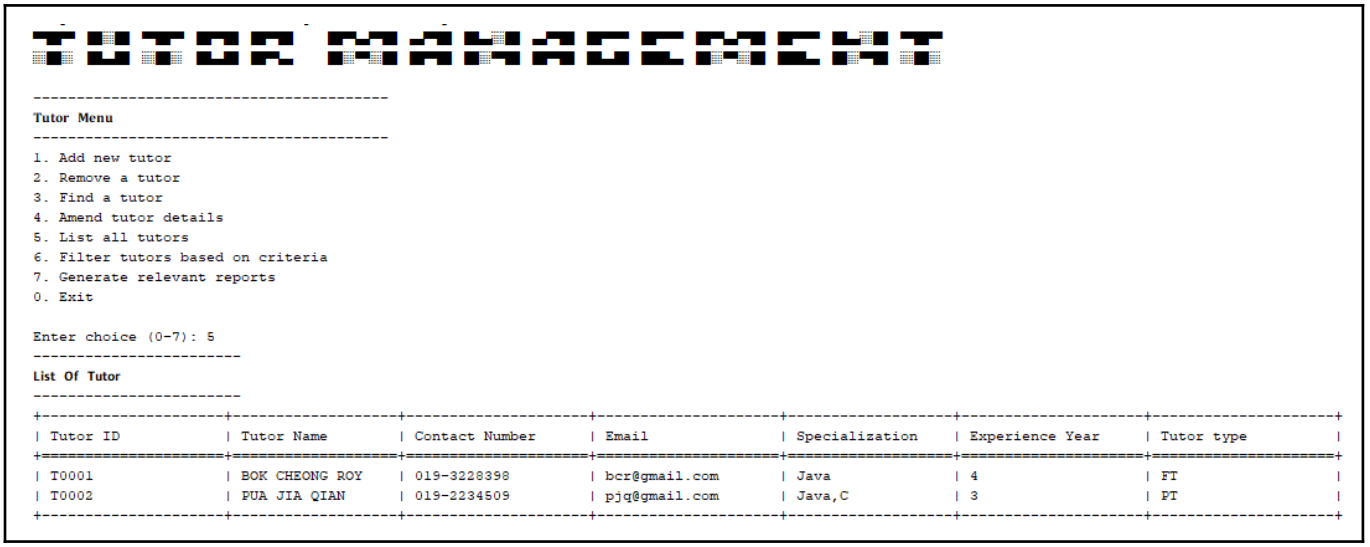
When the user enter option 2,it will ask user to enter the tutor name that want to find.thye user can enter a few word of the tutor name to find.Then,the system will display the tutor information.

4.Modify Tutor

```
------------------------
List Of Tutor
------------------------
+---------------------+-------------------+--------------------+--------------------+-------------------+---------------------+---------------------+
| Tutor ID            | Tutor Name        | Contact Number     | Email              | Specialization    | Experience Year     | Tutor type          |
+=====================+===================+====================+====================+===================+=====================+=====================+
| T0001               | BOK CHEONG ROY    | 019-3228398        | bcr@gmail.com      | Java              | 4                   | FT                  |
| T0002               | PUA JIA QIAN      | 019-2234509        | pjq@gmail.com      | Java,C            | 3                   | PT                  |
| T0003               | LEE WENG YI       | 019-2340987        | lwy@gmail.com      | Java,SQL          | 1                   | PT                  |
+---------------------+-------------------+--------------------+--------------------+-------------------+---------------------+---------------------+
----------------------------------------
Modify Tutor
----------------------------------------
Enter the tutor id that want to modify.
Enter tutor ID: T003
Invalid tutor ID format!(e.g., T1000).
Enter tutor ID: T0003
+---------------------+-------------------+--------------------+--------------------+-------------------+---------------------+---------------------+
| Tutor ID            | Tutor Name        | Contact Number     | Email              | Specialization    | Experience Year     | Tutor type          |
+=====================+===================+====================+====================+===================+=====================+=====================+
| T0003               | LEE WENG YI       | 019-2340987        | lwy@gmail.com      | Java,SQL          | 1                   | PT                  |
+---------------------+-------------------+--------------------+--------------------+-------------------+---------------------+---------------------+

 Select which information you want to modify:
 1. Tutor Name
 2. Tutor Conatct No
 3. Tutor Email
 4. Tutor Specialization
 5. Tutor Experience Year
 6. Tutor Type(FT/PT)
 7. Exit

 Enter(1-7): t
 Invalid input. Please enter an integer.
 Enter(1-7): 5
 Enter tutor experience year(Exp:1,2,5): 8
 Select which information you want to modify:
 1. Tutor Name
 2. Tutor Conatct No
 3. Tutor Email
 4. Tutor Specialization
 5. Tutor Experience Year
 6. Tutor Type(FT/PT)
 7. Exit

 Enter(1-7): 7
 ----Exiting Edit Tutor system-----

 Are you sure to modify this tutor (Y/N)? y
 Tutor Modified Successful!

 Do you want to continue to manage tutor?(Y/N):
```

When a user wants to modify a tutor,the system will display an existing tutor list to let users to refer and enter the tutor's tutor id that they want modify.If the entered tutor id is not exists,it will prompt a error message and ask user to reenter.After a exist and correct tutor id have been entered,it will display modify tutor menu to let user choose what information that want to modify.Then,the user can enter choice 7 to exit when finish modify.

5.List Tutor



```
TUTOR MANAGEMENT

-----------------------------------------
Tutor Menu
-----------------------------------------
1. Add new tutor
2. Remove a tutor
3. Find a tutor
4. Amend tutor details
5. List all tutors
6. Filter tutors based on criteria
7. Generate relevant reports
0. Exit

Enter choice (0-7): 5
------------------------
List Of Tutor
------------------------
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name       | Contact Number     | Email              | Specialization   | Experience Year    | Tutor type         |
+====================+==================+====================+====================+==================+====================+====================+
| T0001              | BOK CHEONG ROY   | 019-3228398        | bcr@gmail.com      | Java             | 4                  | FT                 |
| T0002              | PUA JIA QIAN     | 019-2234509        | pjq@gmail.com      | Java,C           | 3                  | PT                 |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
```

The list tutor will list all the tutur details that have been added.

6.Filter Tutor

```
----------------------------------------
Sorting Tutor
----------------------------------------
Select the meyhod of sorting tutor
1.Ascending sorting tutor name
2.Descending sorting tutor name
3.Exit to tutor menu
Enter(1-3): 1
Ascending Sorted List
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+----------------------+
| Tutor ID           | Tutor Name       | Contact Number     | Email              | Specialization   | Experience Year    | Tutor type           |
+====================+==================+====================+====================+==================+====================+======================+
| T0001              | BOK CHEONG ROY   | 019-3228398        | bcr@gmail.com      | Java             | 4                  | FT                   |
| T0002              | PUA JIA QIAN     | 019-2234509        | pjq@gmail.com      | Java,C           | 3                  | PT                   |
| T0003              | LEE WENG YI      | 019-2340987        | lwy@gmail.com      | Java,SQL         | 8                  | PT                   |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+----------------------+

----------------------------------------
Sorting Tutor
----------------------------------------
Select the meyhod of sorting tutor
1.Ascending sorting tutor name
2.Descending sorting tutor name
3.Exit to tutor menu
Enter(1-3): 2
Descending Sorted List
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+----------------------+
| Tutor ID           | Tutor Name       | Contact Number     | Email              | Specialization   | Experience Year    | Tutor type           |
+====================+==================+====================+====================+==================+====================+======================+
| T0003              | LEE WENG YI      | 019-2340987        | lwy@gmail.com      | Java,SQL         | 8                  | PT                   |
| T0002              | PUA JIA QIAN     | 019-2234509        | pjq@gmail.com      | Java,C           | 3                  | PT                   |
| T0001              | BOK CHEONG ROY   | 019-3228398        | bcr@gmail.com      | Java             | 4                  | FT                   |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+----------------------+
```

There will be a function of sorting the tutor list when the user enters choice 4 in the tutor menu.The users can choose to filter the tutor by selecting 1 for ascending sorting or 2 for descending sorting.Then it will display the entered tutor details based on the user selection.It will aslo display an error message and ask the user to reenter if user enter the choice that other than 1-3.

```
----------------------------------------
Sorting Tutor
----------------------------------------
Select the meyhod of sorting tutor
1.Ascending sorting tutor name
2.Descending sorting tutor name
3.Exit to tutor menu
Enter(1-3): 3
TUTOR MANAGEMENT

----------------------------------------
Tutor Menu
----------------------------------------
1. Add new tutor
2. Remove a tutor
3. Find a tutor
4. Amend tutor details
5. List all tutors
6. Filter tutors based on criteria
7. Generate relevant reports
0. Exit

Enter choice (0-7):
```

Then,the user can enter choice 3 to exit the tutor management menu when they finish filtering the tutor.

7.Tutor Report

```
----------------------------------------
Tutor Report
----------------------------------------
Select the type of report
1.Part-Time/Full-Time Tutor Report
2.Experienced Tutor Report
3.Exit to tutor menu
Enter(1-3): e
Invalid input. Please enter an integer.
Enter(1-3): 1

----------------------------------------
Full Time Tutor
----------------------------------------
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name       | Contact Number     | Email              | Specialization   | Experience Year    | Tutor type         |
+====================+==================+====================+====================+==================+====================+====================+
| T0001              | BOK CHEONG ROY   | 019-3228398        | bcr@gmail.com      | Java             | 4                  | FT                 |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
| Total Tutor: 1                                                                                                                               |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
----------------------------------------
Part-Time Tutor
----------------------------------------
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name       | Contact Number     | Email              | Specialization   | Experience Year    | Tutor type         |
+====================+==================+====================+====================+==================+====================+====================+
| T0003              | LEE WENG YI      | 019-2340987        | lwy@gmail.com      | Java,SQL         | 8                  | PT                 |
| T0002              | PUA JIA QIAN     | 019-2234509        | pjq@gmail.com      | Java,C           | 3                  | PT                 |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
| Total Tutor: 2                                                                                                                               |
+--------------------+------------------+--------------------+--------------------+------------------+--------------------+--------------------+
```

When going into the tutor report,the user needs to select the tutor report by entering choice 1 or 2 in the tutor report menu .If the user the the choice 1,it will display the Part-time /Full Time tutor report.It will show this two types of tutor by using two different tables.There will also show the total number of the type of tutor.

```
----------------------------------------
Tutor Report
----------------------------------------
Select the type of report
1.Part-Time/Full-Time Tutor Report
2.Experienced Tutor Report
3.Exit to tutor menu
Enter(1-3): 2
----------------------------------------
Tutor Experience Less Than 2 Years
----------------------------------------

+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name         | Contact Number     | Email              | Specialization     | Experience Year    | Tutor type         |
+====================+====================+====================+====================+====================+====================+====================+
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Total Count: 0                                                                                                                                      |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
----------------------------------------
Tutor Experience Between 2 And 5 Years
----------------------------------------
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name         | Contact Number     | Email              | Specialization     | Experience Year    | Tutor type         |
+====================+====================+====================+====================+====================+====================+====================+
| T0002              | PUA JIA QIAN       | 019-2234509        | pjq@gmail.com      | Java,C             | 3                  | PT                 |
| T0001              | BOK CHEONG ROY     | 019-3228398        | bcr@gmail.com      | Java               | 4                  | FT                 |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Total Count: 2                                                                                                                                      |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+

----------------------------------------
Tutor Experience More Than 5Years
----------------------------------------
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Tutor ID           | Tutor Name         | Contact Number     | Email              | Specialization     | Experience Year    | Tutor type         |
+====================+====================+====================+====================+====================+====================+====================+
| T0003              | LEE WENG YI        | 019-2340987        | lwy@gmail.com      | Java,SQL           | 8                  | PT                 |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
| Total Count: 1                                                                                                                                      |
+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+
```

When the user enters choice 2 in the tutor report menu,it will display the Experience tutor report.The report will be show by 3 different ranges of year in different tables.The range will be Tutor experience less than 2 years,tutor experience between 2 and 5 years ,and tutor experience more than 5 years.Then, it will display the tutor details base on the range with total amount of each range.