```
 1 /*
 2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
this license
 3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 4  */
 5 package boundary;
 6
 7 import adt.ArrList;
 8 import adt.ListInterface;
 9 import entity.Course;
10 import entity.Course.Sem;
11 import entity.CourseCodeComparator;
12 import entity.CreditHoursComparator;
13 import entity.SemesterComparator;
14 import entity.TitleComparator;
15 import java.util.Iterator;
16 import utility.Command;
17 import utility.ConsoleColor;
18 import utility.MessageUI;
19 import utility.InputValue;
20 import utility.Search;
21
22 /**
23  *
24  * @author Chew Lip Sin
25  */
26 public class CourseMaintenanceUI {
27
28     InputValue iv = new InputValue();
29     private final TitleComparator titleC = new TitleComparator();
30     private final CourseCodeComparator cCodeC = new CourseCodeComparator();
31     private final CreditHoursComparator cHoursC = new CreditHoursComparator();
32     private final SemesterComparator semC = new SemesterComparator();
33
34     public CourseMaintenanceUI() {
35     }
36
37     public int getMenuChoices() {
38         int choice = 0;
39         System.out.println("====================================");
40         System.out.println("||Course Management Subsystem Menu||");
41         System.out.println("====================================");
42         System.out.println("1. Add new course");
43         System.out.println("2. Remove course");
44         System.out.println("3. Search course");
```

```
45          System.out.println("4. Amend course details");
46          System.out.println("5. List all course");
47          System.out.println("6. Course and Program subsystem menu");
48          System.out.println("7. Generate Report");
49          System.out.println("0. Exit");
50          do {
51              System.out.print("Enter choice: ");
52              choice = iv.readInteger();
53              if (choice > 7 || choice < 0) {
54                  MessageUI.displayInvalidChoiceMessage();
55              }
56          } while (choice > 7 || choice < 0);
57
58          return choice;
59      }
60
61      public Course inputCourseDetails(ListInterface<Course> courseList) {
62          System.out.println("    ||==============||");
63          System.out.println("    ||Add New Course||");
64          System.out.println("    ||==============||");
65          System.out.println("");
66          String courseCode = inputCourseCode(courseList);
67          if ("0".equals(courseCode)) {
68              return new Course(null, null, 0, null);
69          }
70          String title = inputTitle();
71          if ("0".equals(title)) {
72              return new Course(null, null, 0, null);
73          }
74          int creditHour = inputCreditHour();
75          if (creditHour == 0) {
76              return new Course(null, null, 0, null);
77          }
78          Sem semester = inputSemester();
79          if (semester == null) {
80              return new Course(null, null, 0, null);
81          }
82          Course newCourse = new Course(courseCode, title, creditHour, semester);
83          System.out.println();
84          courseCode = "0";
85          title = "0";
86          creditHour = 0;
87          semester = null;
88          return newCourse;
89      }
90
```

```
 91     public String inputCourseCode(ListInterface<Course> courseList) {
 92         Iterator it = courseList.getIterator();
 93         int i = 1;
 94         String courseCode = "";
 95         boolean match;
 96         do {
 97             match = false;
 98             System.out.print("Enter Course Code(Enter '0' to exit): ");
 99             courseCode = iv.readCourseCode();
100             if ("0".equals(courseCode)) {
101                 return "0";
102             }
103             courseCode = courseCode.toUpperCase();
104             while (it.hasNext()) {
105                 it.next();
106                 String oldCourseCode = courseList.getEntry(i).getCourseCode();
107                 oldCourseCode = oldCourseCode.toUpperCase();
108                 match = oldCourseCode.equals(courseCode);
109                 i++;
110                 if (match) {
111                     MessageUI.printFormattedText("This " + courseCode + " course code
already exist\n", ConsoleColor.YELLOW);
112                     break;
113                 }
114             }
115         } while (match);
116         courseCode = courseCode.toUpperCase();
117         return courseCode;
118     }
119 
120     public String inputTitle() {
121         System.out.print("Enter the title(Enter '0' to exit): ");
122         String title = iv.readString();
123         if ("0".equals("0")) {
124             return title;
125         }
126         title = title.substring(0, 51);
127         return title;
128 
129     }
130 
131     public int inputCreditHour() {
132         int creditHour = 0;
133         do {
134             System.out.print("Enter credit hour(Enter '0' to exit): ");
135             creditHour = iv.readInteger();
```

```
136             MessageUI.displayInvalidCreditHourMessage(creditHour);
137         } while (creditHour < 0 || creditHour > 20);
138         return creditHour;
139     }
140 
141     public Sem inputSemester() {
142         Sem semester = null;
143         int choice;
144 
145         do {
146             System.out.println("SEMESTER");
147             System.out.println("1. JAN");
148             System.out.println("2. JUL");
149             System.out.println("3. ALL");
150             System.out.println("0. Exit");
151             System.out.print("Select course semester: ");
152             choice = iv.readInteger();
153             switch (choice) {
154                 case 1:
155                     semester = Sem.JAN;
156                     break;
157                 case 2:
158                     semester = Sem.JUL;
159                     break;
160                 case 3:
161                     semester = Sem.ALL;
162                     break;
163                 case 0:
164                     semester = null;
165                 default:
166                     break;
167             }
168             if (choice < 0 || choice > 3) {
169                 MessageUI.displayInvalidChoiceMessage();
170             }
171         } while (choice < 0 || choice > 3);
172         return semester;
173     }
174 
175     public void listAllCourses(ListInterface<Course> courseList) {
176         Command.cls();
177         System.out.println("\nList of Courses:");
178 
System.out.println("================================================================================
==============================================");
179         System.out.println("No |Course Code |Course Title
```

```
|Credit Hours  |Semester  |Created At  |Updated At");
System.out.println("================================================================================
==============================================");
        System.out.print(getAllCourses(courseList));
System.out.println("================================================================================
==============================================");

    }

    public String getAllCourses(ListInterface<Course> courseList) {
//        int currentIndex = 0;
//        for (int i = 1; i <= courseList.size(); i++) {
//            outputStr += courseList.getEntry(i) + "\n";
//        }
        String outputStr = "";
        Iterator it = courseList.getIterator();
        int i = 1;

        while (it.hasNext()) {
//            System.out.println(it.next());
            outputStr += String.format("%-3d", i) + it.next() + "\n";
            i++;
        }
        return outputStr;
    }

    public void displayCourse(Course course, String val) {
        String word = " " + val + " Course Details";
        MessageUI.printFormattedText(word + "\n", ConsoleColor.CYAN);
        for (int i = 0; i < word.length() + 2; i++) {
            MessageUI.printFormattedText("-", ConsoleColor.CYAN);
        }
        System.out.println("");
        MessageUI.printFormattedText("Course Code : " + course.getCourseCode() + "\n",
ConsoleColor.CYAN);
        MessageUI.printFormattedText("Course Title: " + course.getTitle() + "\n",
ConsoleColor.CYAN);
        MessageUI.printFormattedText("Credit Hours: " + course.getCreditHours() + "\n",
ConsoleColor.CYAN);
        MessageUI.printFormattedText("Semester    : " +
course.semToString(course.getSemester()) + "\n", ConsoleColor.CYAN);

        Command.pressEnterToContinue();
    }
```

```
217
218     //Ask confirmation message is 1 and 0 is ask again message.
219     public boolean getConfirmationChoice(String val, int type) {
220         int choice = 0;
221         do {
222             if (type == 1) {
223                 MessageUI.askConfirmationMessage(val);
224             } else {
225                 MessageUI.displayAskAgainMessage(val);
226             }
227             choice = iv.readInteger();
228             if (choice < 0 || choice > 1) {
229                 MessageUI.displayInvalidChoiceMessage();
230             }
231         } while (choice < 0 || choice > 1);
232         if (choice == 0) {
233             return false;
234         } else {
235             return true;
236         }
237     }
238
239     public int inputRemoveCode(ListInterface<Course> courseList) {
240         Iterator it = courseList.getIterator();
241         int i = 1;
242         boolean match = false;
243         System.out.print("Enter the course code you want to delete(Enter '0' to exit):
");
244         String courseCode = iv.readCourseCode();
245         if ("0".equals(courseCode)) {
246             return -1;
247         }
248         courseCode = courseCode.toUpperCase();
249         while (it.hasNext()) {
250             it.next();
251             String oldCourseCode = courseList.getEntry(i).getCourseCode();
252             oldCourseCode = oldCourseCode.toUpperCase();
253             match = oldCourseCode.equals(courseCode);
254
255             if (match) {
256                 return i;
257             }
258             i++;
259         }
260         MessageUI.printFormattedText("This " + courseCode + " is not in the list.\n",
ConsoleColor.YELLOW);
```

```
261         Command.pressEnterToContinue();
262         return -1;
263     }
264 
265     public int getSearchMenuChoices() {
266         int choice = 0;
267 
268         System.out.println("=================");
269         System.out.println("||Search Course||");
270         System.out.println("=================");
271         System.out.println("1. Search Course Code");
272         System.out.println("2. Search Course Title");
273         System.out.println("0. Exit");
274         System.out.print("Enter your choice: ");
275         do {
276             choice = iv.readInteger();
277             if (choice > 2 || choice < 0) {
278                 MessageUI.displayInvalidChoiceMessage();
279             }
280         } while (choice > 2 || choice < 0);
281         return choice;
282     }
283 
284     public int getSearchCourseCode(ListInterface<Course> courseList) {
285         Search search = new Search();
286         String key;
287         System.out.print("Enter the course code you want to search: ");
288         key = iv.readString();
289         key = key.toUpperCase();
290         int found = search.binarySearch(courseList, key);
291         return found;
292     }
293 
294     public void displayCourseDetailsFounded(ListInterface<Course> courseList, int found)
{
295         MessageUI.displayFoundMessage(courseList.getEntry(found + 1).getCourseCode());
296         Course courseSearch = courseList.getEntry(found + 1);
297         displayCourse(courseSearch, "Search");
298     }
299 
300     public int getSearchCourseTitle(ListInterface<Course> courseList,
ListInterface<Course> courseList2) {
301         Iterator it = courseList.getIterator();
302         String key;
303         boolean find, find2 = false;
304         int i = 1;
```

```
305
306         System.out.print("Enter the course title you want to search(Enter '0' to exit):
");
307         key = iv.readString();
308         if ("0".equals(key)) {
309             return 0;
310         }
311         key = key.toLowerCase();
312         while (it.hasNext()) {
313             it.next();
314             find = courseList.getEntry(i).getTitle().toLowerCase().contains(key);
315 
316             if (find) {
317                 courseList2.add(courseList.getEntry(i));
318                 find2 = true;
319             }
320 
321             i++;
322         }
323         if (find2) {
324             return 1;
325         }
326         return -1;
327     }
328 
329     public void displayCourseFounded(ListInterface<Course> courseList2) {
330         boolean loop = true;
331         int choice;
332         ArrList.insertionSort(courseList2, cCodeC,"asc");
333         ArrList.insertionSort(courseList2, titleC,"asc");
334         do {
335             displayCourseFoundedList(courseList2);
336             System.out.print("Enter the choice you want to search for(Enter '0' to exit):
");
337             choice = iv.readInteger();
338             if (choice == 0) {
339                 loop = false;
340             } else if (choice > 0 && choice <= courseList2.size()) {
341                 displayCourse(courseList2.getEntry(choice), "Search");
342 
343             } else {
344                 MessageUI.displayInvalidChoiceMessage();
345             }
346         } while (loop);
347     }
348 
```

```
349     public void displayCourseFoundedList(ListInterface<Course> courseList2) {
350         int i;
351         System.out.println("Course");
352 
System.out.println("=================================================================================
====================");
353         System.out.println("No|Course Code |Course Title
|Credit Hours  |Semester");
354 
System.out.println("=================================================================================
====================");
355         for (i = 1; i <= courseList2.size(); i++) {
356             System.out.printf("%-2d|%-12s|%-52s|  %-2d          |  %-3s\n",
357                     i,
358                     courseList2.getEntry(i).getCourseCode(),
359                     courseList2.getEntry(i).getTitle(),
360                     courseList2.getEntry(i).getCreditHours(),
361 
courseList2.getEntry(i).getSemester().getString(courseList2.getEntry(i).getSemester()));
362         }
363 
System.out.println("=================================================================================
====================");
364         MessageUI.printFormattedText(i - 1 + " result(s) founded!\n",
ConsoleColor.GREEN);
365     }
366 
367     public int getAmmendMenuChoices() {
368         int choice = 0;
369         System.out.println("=================");
370         System.out.println("||Ammend Course||");
371         System.out.println("=================");
372         System.out.println("1. Ammend Course Code");
373         System.out.println("2. Ammend Course Title");
374         System.out.println("3. Ammend Credit Hours");
375         System.out.println("4. Ammend Semester");
376         System.out.println("0. Exit/Continue");
377         System.out.print("Enter your choice: ");
378         do {
379             choice = iv.readInteger();
380             if (choice > 4 || choice < 0) {
381                 MessageUI.displayInvalidChoiceMessage();
382             }
383         } while (choice > 4 || choice < 0);
384         return choice;
385     }
```

```
386
387     public int getCourseAmmend(ListInterface<Course> courseList, ListInterface<Course>
courseList2) {
388         Search search = new Search();
389         boolean loop = true;
390         int choice;
391         ArrList.insertionSort(courseList2, cCodeC,"asc");
392         ArrList.insertionSort(courseList2, titleC,"asc");
393         do {
394             displayCourseFoundedList(courseList2);
395             System.out.print("Enter the choice you want to ammend for(Enter '0' to exit):
");
396             choice = iv.readInteger();
397             if (choice == 0) {
398                 loop = false;
399             } else if (choice > 0 && choice <= courseList2.size()) {
400                 displayCourse(courseList2.getEntry(choice), "Ammend");
401                 int index = search.binarySearch(courseList,
courseList2.getEntry(choice).getCourseCode());
402                 return index;
403             } else {
404                 MessageUI.displayInvalidChoiceMessage();
405             }
406         } while (loop);
407         return -1;
408     }
409
410     public int getSortMenu(ListInterface<Course> courseList) {
411         int choice = 0;
412         System.out.println("Sort");
413         System.out.println("----");
414         System.out.println("1. Ascending Order");
415         System.out.println("2. Descending Order");
416         System.out.println("0. Exit");
417         do {
418             System.out.print("Enter choice: ");
419             choice = iv.readInteger();
420             if (choice > 2 || choice < 0) {
421                 MessageUI.displayInvalidChoiceMessage();
422             }
423         } while (choice > 2 || choice < 0);
424
425         return choice;
426     }
427
428     public int getSortMenuChoice(ListInterface<Course> courseList, String val) {
```

```
429         int choice = 0;
430         System.out.println(val);
431         System.out.println("----");
432         System.out.println("1. Course Code");
433         System.out.println("2. Course Title");
434         System.out.println("3. Credit Hours");
435         System.out.println("4. Semester");
436         System.out.println("5. Created At");
437         System.out.println("6. Updated At");
438         System.out.println("0. Exit");
439         do {
440             System.out.print("Enter choice: ");
441             choice = iv.readInteger();
442             if (choice > 6 || choice < 0) {
443                 MessageUI.displayInvalidChoiceMessage();
444             }
445         } while (choice > 6 || choice < 0);
446 
447         return choice;
448     }
449 
450 }
```