```
 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.ListInterface;
 8 import entity.Course;
 9 import entity.Course.Sem;
10 import java.util.Iterator;
11 import utility.Command;
12 import utility.ConsoleColor;
13 import utility.MessageUI;
14 import utility.InputValue;
15 import utility.Search;
16 import utility.Sort;
17
18 /**
19  *
20  * @author Chew Lip Sin
21  */
22 public class CourseMaintenanceUI {
23
24     InputValue iv = new InputValue();
25     Sort sort = new Sort();
26
27     public CourseMaintenanceUI() {
28     }
29
30     public int getMenuChoices() {
31         int choice = 0;
32         System.out.println("====================================");
33         System.out.println("||Course Management Subsystem Menu||");
34         System.out.println("====================================");
35         System.out.println("1. Add new course");
36         System.out.println("2. Remove course");
37         System.out.println("3. Search course");
38         System.out.println("4. Amend course details");
39         System.out.println("5. List all course");
40         System.out.println("6. Course and Program subsystem menu");
41         System.out.println("7. Generate Report");
42         System.out.println("0. Exit");
43         do {
44             System.out.print("Enter choice: ");
```

```
45              choice = iv.readInteger();
46              if (choice > 7 || choice < 0) {
47                  MessageUI.displayInvalidChoiceMessage();
48              }
49          } while (choice > 7 || choice < 0);
50
51          return choice;
52      }
53
54      public Course inputCourseDetails(ListInterface<Course> courseList) {
55          System.out.println("    ||==============||");
56          System.out.println("    ||Add New Course||");
57          System.out.println("    ||==============||");
58          System.out.println("");
59          String courseCode = inputCourseCode(courseList);
60          if ("0".equals(courseCode)) {
61              return new Course(null, null, 0, null);
62          }
63          String title = inputTitle();
64          if ("0".equals(title)) {
65              return new Course(null, null, 0, null);
66          }
67          int creditHour = inputCreditHour();
68          if (creditHour == 0) {
69              return new Course(null, null, 0, null);
70          }
71          Sem semester = inputSemester();
72          if (semester == null) {
73              return new Course(null, null, 0, null);
74          }
75          Course newCourse = new Course(courseCode, title, creditHour, semester);
76          System.out.println();
77          courseCode = "0";
78          title = "0";
79          creditHour = 0;
80          semester = null;
81          return newCourse;
82      }
83
84      public String inputCourseCode(ListInterface<Course> courseList) {
85          Iterator it = courseList.getIterator();
86          int i = 1;
87          String courseCode = "";
88          boolean match;
89          do {
90              match = false;
```

```
 91             System.out.print("Enter Course Code(Enter '0' to exit): ");
 92             courseCode = iv.readCourseCode();
 93             if ("0".equals(courseCode)) {
 94                 return "0";
 95             }
 96             courseCode = courseCode.toUpperCase();
 97             while (it.hasNext()) {
 98                 it.next();
 99                 String oldCourseCode = courseList.getEntry(i).getCourseCode();
100                 oldCourseCode = oldCourseCode.toUpperCase();
101                 match = oldCourseCode.equals(courseCode);
102                 i++;
103                 if (match) {
104                     MessageUI.printFormattedText("This " + courseCode + " course code
already exist\n", ConsoleColor.YELLOW);
105                     break;
106                 }
107             }
108         } while (match);
109         courseCode = courseCode.toUpperCase();
110         return courseCode;
111     }
112 
113     public String inputTitle() {
114         System.out.print("Enter the title(Enter '0' to exit): ");
115         String title = iv.readString();
116         if ("0".equals("0")) {
117             return title;
118         }
119         title = title.substring(0, 51);
120         return title;
121 
122     }
123 
124     public int inputCreditHour() {
125         int creditHour = 0;
126         do {
127             System.out.print("Enter credit hour(Enter '0' to exit): ");
128             creditHour = iv.readInteger();
129             MessageUI.displayInvalidCreditHourMessage(creditHour);
130         } while (creditHour < 0 || creditHour > 20);
131         return creditHour;
132     }
133 
134     public Sem inputSemester() {
135         Sem semester = null;
```

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

```
==========================================");
176
177     }
178
179     public String getAllCourses(ListInterface<Course> courseList) {
180 //        int currentIndex = 0;
181 //        for (int i = 1; i <= courseList.size(); i++) {
182 //            outputStr += courseList.getEntry(i) + "\n";
183 //        }
184         String outputStr = "";
185         Iterator it = courseList.getIterator();
186         int i = 1;
187
188         while (it.hasNext()) {
189 //            System.out.println(it.next());
190             outputStr += String.format("%-3d", i) + it.next() + "\n";
191             i++;
192         }
193         return outputStr;
194     }
195
196     public void displayCourse(Course course, String val) {
197         String word = " " + val + " Course Details";
198         MessageUI.printFormattedText(word + "\n", ConsoleColor.CYAN);
199         for (int i = 0; i < word.length() + 2; i++) {
200             MessageUI.printFormattedText("-", ConsoleColor.CYAN);
201         }
202         System.out.println("");
203         MessageUI.printFormattedText("Course Code : " + course.getCourseCode() + "\n",
ConsoleColor.CYAN);
204         MessageUI.printFormattedText("Course Title: " + course.getTitle() + "\n",
ConsoleColor.CYAN);
205         MessageUI.printFormattedText("Credit Hours: " + course.getCreditHours() + "\n",
ConsoleColor.CYAN);
206         MessageUI.printFormattedText("Semester    : " +
course.semToString(course.getSemester()) + "\n", ConsoleColor.CYAN);
207
208         Command.pressEnterToContinue();
209     }
210
211     //Ask confirmation message is 1 and 0 is ask again message.
212     public boolean getConfirmationChoice(String val, int type) {
213         int choice = 0;
214         do {
215             if (type == 1) {
216                 MessageUI.askConfirmationMessage(val);
```

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

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

```
304         key = key.toLowerCase();
305         while (it.hasNext()) {
306             it.next();
307             find = courseList.getEntry(i).getTitle().toLowerCase().contains(key);
308 
309             if (find) {
310                 courseList2.add(courseList.getEntry(i));
311                 find2 = true;
312             }
313 
314             i++;
315         }
316         if (find2) {
317             return 1;
318         }
319         return -1;
320     }
321 
322     public void displayCourseFounded(ListInterface<Course> courseList2) {
323         boolean loop = true;
324         int choice;
325         sort.insertionSort(courseList2, "courseCode");
326         sort.insertionSort(courseList2, "title");
327         do {
328             displayCourseFoundedList(courseList2);
329             System.out.print("Enter the choice you want to search for(Enter '0' to exit):
");
330             choice = iv.readInteger();
331             if (choice == 0) {
332                 loop = false;
333             } else if (choice > 0 && choice <= courseList2.size()) {
334                 displayCourse(courseList2.getEntry(choice), "Search");
335 
336             } else {
337                 MessageUI.displayInvalidChoiceMessage();
338             }
339         } while (loop);
340     }
341 
342     public void displayCourseFoundedList(ListInterface<Course> courseList2) {
343         int i;
344         System.out.println("Course");
345 
System.out.println("=============================================================================
====================");
346         System.out.println("No|Course Code |Course Title
```

```
|Credit Hours  |Semester");
System.out.println("================================================================================
====================");
        for (i = 1; i <= courseList2.size(); i++) {
            System.out.printf("%-2d|%-12s|%-52s|  %-2d          |  %-3s\n",
                    i,
                    courseList2.getEntry(i).getCourseCode(),
                    courseList2.getEntry(i).getTitle(),
                    courseList2.getEntry(i).getCreditHours(),
courseList2.getEntry(i).getSemester().getString(courseList2.getEntry(i).getSemester()));
        }
System.out.println("================================================================================
====================");
        MessageUI.printFormattedText(i - 1 + " result(s) founded!\n",
ConsoleColor.GREEN);
    }

    public int getAmmendMenuChoices() {
        int choice = 0;
        System.out.println("=================");
        System.out.println("||Ammend Course||");
        System.out.println("=================");
        System.out.println("1. Ammend Course Code");
        System.out.println("2. Ammend Course Title");
        System.out.println("3. Ammend Credit Hours");
        System.out.println("4. Ammend Semester");
        System.out.println("0. Exit/Continue");
        System.out.print("Enter your choice: ");
        do {
            choice = iv.readInteger();
            if (choice > 4 || choice < 0) {
                MessageUI.displayInvalidChoiceMessage();
            }
        } while (choice > 4 || choice < 0);
        return choice;
    }

    public int getCourseAmmend(ListInterface<Course> courseList, ListInterface<Course>
courseList2) {
        Search search = new Search();
        boolean loop = true;
        int choice;
        sort.insertionSort(courseList2, "courseCode");
```

```
385         sort.insertionSort(courseList2, "title");
386         do {
387             displayCourseFoundedList(courseList2);
388             System.out.print("Enter the choice you want to ammend for(Enter '0' to exit):
");
389             choice = iv.readInteger();
390             if (choice == 0) {
391                 loop = false;
392             } else if (choice > 0 && choice <= courseList2.size()) {
393                 displayCourse(courseList2.getEntry(choice), "Ammend");
394                 int index = search.binarySearch(courseList,
courseList2.getEntry(choice).getCourseCode());
395                 return index;
396             } else {
397                 MessageUI.displayInvalidChoiceMessage();
398             }
399         } while (loop);
400         return -1;
401     }
402 
403     public int getSortMenu(ListInterface<Course> courseList) {
404         int choice = 0;
405         System.out.println("Sort");
406         System.out.println("----");
407         System.out.println("1. Ascending Order");
408         System.out.println("2. Descending Order");
409         System.out.println("0. Exit");
410         do {
411             System.out.print("Enter choice: ");
412             choice = iv.readInteger();
413             if (choice > 2 || choice < 0) {
414                 MessageUI.displayInvalidChoiceMessage();
415             }
416         } while (choice > 2 || choice < 0);
417 
418         return choice;
419     }
420 
421     public int getSortMenuChoice(ListInterface<Course> courseList, String val) {
422         int choice = 0;
423         System.out.println(val);
424         System.out.println("----");
425         System.out.println("1. Course Code");
426         System.out.println("2. Course Title");
427         System.out.println("3. Credit Hours");
428         System.out.println("4. Semester");
```

```
429         System.out.println("5. Created At");
430         System.out.println("6. Updated At");
431         System.out.println("0. Exit");
432         do {
433             System.out.print("Enter choice: ");
434             choice = iv.readInteger();
435             if (choice > 6 || choice < 0) {
436                 MessageUI.displayInvalidChoiceMessage();
437             }
438         } while (choice > 6 || choice < 0);
439 
440         return choice;
441     }
442 
443 }
```