

```
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.DoublyLinkedList;
9  import adt.LinkedListInterface;
10 import adt.ListInterface;
11 import entity.Course;
12 import entity.CourseProgram;
13 import entity.Program;
14 import java.util.Iterator;
15 import utility.InputValue;
16 import utility.MessageUI;
17 import dao.DAO;
18 import utility.Command;
19 import utility.ConsoleColor;
20
21 /**
22  *
23  * @author Chew Lip Sin
24  */
25 public class CourseProgramMaintenanceUI {
26
27     private final InputValue iv = new InputValue();
28     private final DAO<CourseProgram> cpDAO = new DAO();
29
30     public CourseProgramMaintenanceUI() {
31     }
32
33
34     public int getMenuChoice() {
35         Command.cls();
36         int choice = 0;
37         System.out.println("=====");
38         System.out.println("||Course and Program Management Subsystem Menu||");
39         System.out.println("=====");
40         System.out.println("1. Add new program to the course");
41         System.out.println("2. Remove program from the course");
42         System.out.println("0. Exit");
43         do {
44             System.out.print("Enter choice: ");
```

```

45         choice = iv.readInteger();
46         if (choice > 2 || choice < 0) {
47             MessageUI.displayInvalidChoiceMessage();
48         }
49     } while (choice > 2 || choice < 0);
50
51     return choice;
52 }
53
54 public int getCourseChoices(ListInterface<Course> courseList) {
55     int choice = 0;
56     do {
57         System.out.print("Enter choice(Enter '0' to exit): ");
58         choice = iv.readInteger();
59         if (choice > courseList.size() || choice < 0) {
60             MessageUI.displayInvalidChoiceMessage();
61         }
62     } while (choice > courseList.size() || choice < 0);
63     return choice;
64 }
65
66 public void addCourseProgramUI(LinkedListInterface<CourseProgram> courseProgramList,
ListInterface<Course> courseList, ListInterface<Program> programList, int index) {
67     ListInterface<Program> tempProgram = new ArrList<>();
68     ListInterface<Program> tempProgram2 = new ArrList<>();
69     ListInterface<Boolean> isElective = new ArrList<>();
70     ListInterface<Program> tempProgram3 = new ArrList<>();
71     ListInterface<Program> tempProgram4 = new ArrList<>();
72
73     boolean loop = true;
74     for (int i = 1; i <= programList.size(); i++) {
75         tempProgram3.add(programList.getEntry(i));
76     }
77     int choice = 0, choice2 = 0;
78     validateProgram(courseProgramList, courseList, programList, tempProgram2, index);
79     for (int i = 1; i <= tempProgram2.size(); i++) {
80         tempProgram3.remove(tempProgram2.getEntry(i));
81     }
82     tempProgram4 = tempProgram2;
83     do {
84         int no = 1;
85         if (tempProgram3.size() > 0) {
86             no = displayProgramList(tempProgram3, no);
87             choice = getProgramChoice(choice, no);
88
89             if (choice != 0) {

```

```
90         choice2 = displayElective(choice2, isElective);
91         if (choice2 == 1 || choice2 == 2) {
92             tempProgram.add(tempProgram3.getEntry(choice));
93             tempProgram2.add(tempProgram3.getEntry(choice));
94             tempProgram3.remove(choice);
95         }
96
97     }
98
99     if (choice == 0 || choice2 == 0) {
100         loop = false;
101     }
102
103 }
104 if (tempProgram3.size() == 0) {
105     loop = false;
106 }
107 } while (loop);
108 if (tempProgram.size() > 0) {
109     askAddConfirmation(tempProgram, courseList, isElective, courseProgramList,
index);
110     for (int i = 0; i < courseProgramList.sizeOf(); i++) {
111         System.out.println(String.format("%s %s %s",
courseProgramList.get(i).getCourseCode(), courseProgramList.get(i).getProgramCode(),
courseProgramList.get(i).isIsElective()));
112     }
113
114 }
115 if (tempProgram.size() == 0 && tempProgram4.size() == programList.size()) {
116     MessageUI.printFormattedText("There is no available program to choose\n",
ConsoleColor.YELLOW);
117     Command.pressEnterToContinue();
118 }
119 tempProgram.clear();
120 tempProgram2.clear();
121 tempProgram3.clear();
122 isElective.clear();
123
124 }
125
126 public void validateProgram(LinkedListInterface<CourseProgram> courseProgramList,
ListInterface<Course> courseList,
127 ListInterface<Program> programList, ListInterface<Program> tempProgram2,
128 int index) {
129     for (int i = 0; i < courseProgramList.sizeOf(); i++) {
130         if
```

```

(courseProgramList.get(i).getCourseCode().equals(courseList.getEntry(index).getCourseCode()))
{
131         for (int j = 1; j <= programList.size(); j++) {
132             if
(courseProgramList.get(i).getProgramCode().equals(programList.getEntry(j).getCode())) {
133                 tempProgram2.add(programList.getEntry(j));
134             }
135         }
136     }
137 }
138
139 }
140
141 private void askAddConfirmation(ListInterface<Program> tempProgram,
142     ListInterface<Course> courseList, ListInterface<Boolean> isElective,
143     LinkedListInterface<CourseProgram> courseProgramList, int index) {
144     int choice;
145     do {
146         MessageUI.printFormattedText("Are you sure to add it?(1 is yes, 0 is no): ",
ConsoleColor.BRIGHTBLUE);
147         choice = iv.readInteger();
148         if (choice == 1) {
149             for (int i = 1; i <= tempProgram.size(); i++) {
150                 CourseProgram cp = new
CourseProgram(courseList.getEntry(index).getCourseCode(),
151                 tempProgram.getEntry(i).getCode(),
152                 isElective.getEntry(i).booleanValue());
153                 courseProgramList.add(cp);
154             }
155             MessageUI.displaySuccessConfirmationMessage("Added");
156             Command.pressEnterToContinue();
157             cpDAO.saveToFile(courseProgramList, "courseProgram.dat");
158
159             } else if (choice > 1 || choice < 0) {
160                 MessageUI.displayInvalidChoiceMessage();
161             }
162         } while (choice > 1 || choice < 0);
163     }
164
165     private int displayProgramList(ListInterface<Program> tempProgram3, int no) {
166         Iterator<Program> it = tempProgram3.getIterator();
167         System.out.println("Program List:");
168         System.out.println(String.format("%-5s|%-15s|%-80s", "No.", "Program Code",
"Program Name"));
169         String line = "";
170

```

```
171     for (int i = 0; i <= 100; i++) {
172         line += "-";
173     }
174     System.out.println(line);
175     while (it.hasNext()) {
176         Program program = it.next();
177         System.out.println(String.format("%2d    |%-15s|%-80s", no, program.getCode(),
program.getName()));
178         no++;
179     }
180
181     return no;
182
183 }
184
185 private int getProgramChoice(int choice, int no) {
186     do {
187         System.out.print("Enter choice(Enter '0' to exit/continue): ");
188         choice = iv.readInteger();
189         if (choice >= no || choice < 0) {
190             MessageUI.displayInvalidChoiceMessage();
191         }
192     } while (choice >= no || choice < 0);
193     return choice;
194 }
195
196 private int displayElective(int choice2, ListInterface<Boolean> isElective) {
197     do {
198         System.out.println("Elective");
199         System.out.println("=====");
200         System.out.println("1. Yes");
201         System.out.println("2. No");
202         System.out.println("0. Exit");
203         System.out.print("Is it an Elective?: ");
204         choice2 = iv.readInteger();
205         if (choice2 > 2 || choice2 < 0) {
206             MessageUI.displayInvalidChoiceMessage();
207         }
208     } while (choice2 > 2 || choice2 < 0);
209     if (choice2 == 1) {
210         isElective.add(Boolean.TRUE);
211     } else if (choice2 == 2) {
212         isElective.add(Boolean.FALSE);
213     }
214     return choice2;
215 }
```

```

216
217     public void removeCourseProgramUI(LinkedListInterface<CourseProgram>
courseProgramList, ListInterface<Course> courseList, ListInterface<Program> programList, int
index) {
218         ListInterface<Program> tempProgram = new ArrList<>();
219         ListInterface<Program> tempProgram2 = new ArrList<>();
220         ListInterface<Boolean> isElective = new ArrList<>();
221         ListInterface<Program> tempProgram3 = new ArrList<>();
222         ListInterface<Program> tempProgram4 = new ArrList<>();
223         LinkedListInterface<CourseProgram> cpl = new DoublyLinkedList<>();
224         boolean loop = true;
225         int choice = 0;
226         validateProgram(courseProgramList, courseList, programList, tempProgram2, index);
227         for (int i = 1; i <= tempProgram2.size(); i++) {
228             tempProgram3.add(tempProgram2.getEntry(i));
229         }
230         for (int i = 1; i <= tempProgram2.size(); i++) {
231             tempProgram4.add(tempProgram2.getEntry(i));
232         }
233         do {
234             int no = 1;
235             if (tempProgram2.size() > 0) {
236                 no = displayDelProgramList(tempProgram2, no);
237                 choice = getProgramChoice(choice, no);
238
239                 if (choice != 0) {
240                     tempProgram.add(tempProgram2.getEntry(choice));
241                     tempProgram2.remove(choice);
242
243                 }
244
245                 if (choice == 0) {
246                     loop = false;
247                 }
248
249             }
250             if (tempProgram2.size() == 0) {
251                 loop = false;
252             }
253         } while (loop);
254         if (tempProgram.size() > 0) {
255             validateCourseProgram(tempProgram, courseList, courseProgramList, index,
cpl);
256             askDelConfirmation(tempProgram, courseList, courseProgramList, index, cpl);
257             for (int i = 0; i < courseProgramList.sizeOf(); i++) {
258                 System.out.println(String.format("%s %s %s",

```

```

courseProgramList.get(i).getCourseCode(), courseProgramList.get(i).getProgramCode(),
courseProgramList.get(i).isIsElective()));
259         }
260
261     }
262     if (tempProgram4.size() == 0) {
263         MessageUI.printFormattedText("There is no available program to delete\n",
ConsoleColor.YELLOW);
264         Command.pressEnterToContinue();
265     }
266     tempProgram.clear();
267     tempProgram2.clear();
268     tempProgram3.clear();
269     isElective.clear();
270 }
271
272 private int displayDelProgramList(ListInterface<Program> tempProgram2, int no) {
273     Iterator<Program> it = tempProgram2.getIterator();
274     System.out.println("Program List:");
275     System.out.println(String.format("%-5s|%-15s|%-80s", "No.", "Program Code",
"Program Name"));
276     String line = "";
277
278     for (int i = 0; i <= 100; i++) {
279         line += "-";
280     }
281     System.out.println(line);
282     while (it.hasNext()) {
283         Program program = it.next();
284         System.out.println(String.format("%2d    |%-15s|%-80s", no, program.getCode(),
program.getName()));
285         no++;
286     }
287
288     return no;
289 }
290
291
292 private void askDelConfirmation(ListInterface<Program> tempProgram,
ListInterface<Course> courseList,
293     LinkedListInterface<CourseProgram> courseProgramList,
294     int index,
295     LinkedListInterface<CourseProgram> cp1) {
296     int choice;
297     do {
298         MessageUI.printFormattedText("Are you sure to delete it?(1 is yes, 0 is no):

```

```
", ConsoleColor.BRIGHTBLUE);
299         choice = iv.readInteger();
300         if (choice == 1) {
301             for (int i = 0; i < cp1.sizeOf(); i++) {
302                 courseProgramList.remove(cp1.get(i));
303             }
304             cpDAO.saveToFile(courseProgramList, "courseProgram.dat");
305             MessageUI.displaySuccessConfirmationMessage("Deleted");
306             Command.pressEnterToContinue();
307         } else if (choice > 1 || choice < 0) {
308             MessageUI.displayInvalidChoiceMessage();
309         }
310     } while (choice > 1 || choice < 0);
311 }
312
313 private void validateCourseProgram(ListInterface<Program> tempProgram,
314     ListInterface<Course> courseList,
315     LinkedListInterface<CourseProgram> courseProgramList,
316     int index, LinkedListInterface<CourseProgram> cp1) {
317     for (int i = 0; i < courseProgramList.sizeOf(); i++) {
318         if
319         (courseProgramList.get(i).getCourseCode().equals(courseList.getEntry(index).getCourseCode()))
320     {
321         for (int j = 1; j <= tempProgram.size(); j++) {
322             if
323             (courseProgramList.get(i).getProgramCode().equals(tempProgram.getEntry(j).getCode())) {
324                 cp1.add(courseProgramList.get(i));
325             }
326         }
327     }
```