

```
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 control;
6
7  import adt.*;
8  import boundary.CourseMaintenanceUI;
9  import boundary.CourseProgramMaintenanceUI;
10 import dao.DAO;
11 import dao.Initializer;
12 import entity.Course;
13 import entity.Course.Sem;
14 import entity.CourseProgram;
15 import entity.Program;
16 import java.io.IOException;
17 import utility.*;
18
19 /**
20  *
21  * @author Chew Lip Sin
22  */
23 public class CourseMaintenance {
24
25     // private ListInterface<Course> courseList = new ArrList<>();
26     private final CourseMaintenanceUI courseUI = new CourseMaintenanceUI();
27     private final Sort s = new Sort();
28     private static final DAO<Course> cDAO = new DAO<>();
29     private static final DAO<Program> pDAO = new DAO<>();
30     private final DAO<CourseProgram> cpDAO = new DAO<>();
31     private static final CourseProgramMaintenance cpm = new CourseProgramMaintenance();
32     private final Initializer in = new Initializer();
33
34     public CourseMaintenance() {
35     }
36
37     public void runCourseMaintenance() throws IOException, InterruptedException {
38         ListInterface<Course> courseList = cDAO.retrieveFromFile("course.dat");
39 //         ListInterface<Course> courseList = in.initializeCourse();
40 //         cDAO.saveToFile(courseList, "course.dat");
41         ListInterface<Program> programList = pDAO.retrieveFromFile("program.dat");
42 //         pDAO.saveToFile(programList, "program.dat");
43 //         ListInterface<Program> programList = in.ProgramInitializer();
44 //         pDAO.saveToFile(programList, "program.dat");
```

```
45     int choice;
46     do {
47         Command.cls();
48         choice = courseUI.getMenuChoices();
49         switch (choice) {
50             case 0:
51                 break;
52             case 1:
53                 addNewCourse(courseList);
54                 break;
55             case 2:
56                 removeCourse(courseList);
57                 break;
58             case 3:
59                 searchCourse(courseList, "search");
60                 break;
61             case 4:
62                 searchCourse(courseList, "ammend");
63                 break;
64             case 5:
65                 listedCourse(courseList);
66                 break;
67             case 6:
68                 CourseProgramMaintenance coursePM = new CourseProgramMaintenance();
69                 coursePM.runCourseProgramMaintenance(courseList, programList);
70                 break;
71             case 7:
72                 CourseGenerateReportMaintenance courseGRM = new
CourseGenerateReportMaintenance();
73                 courseGRM.runCourseGenerateReportMaintenance();
74                 break;
75             default:
76                 MessageUI.displayInvalidChoiceMessage();
77         }
78     } while (choice != 0);
79
80
81 }
82
83 private void addNewCourse(ListInterface<Course> courseList) {
84     boolean loop = false;
85     do {
86         Command.cls();
87         boolean newCourseFull, confirm = true;
88         Course newCourse = courseUI.inputCourseDetails(courseList);
89         newCourseFull = (!newCourse.equals(new Course(null, null, 0, null)));
```

```

90         if (newCourseFull == true) {
91             confirm = askConfirmation(newCourse, "New", "add");
92             if (confirm == true) {
93                 courseList.add(newCourse);
94                 cDAO.saveToFile(courseList, "course.dat");
95                 MessageUI.displaySuccessConfirmationMessage("Added");
96                 loop = courseUI.getConfirmationChoice("add", 0);
97             } else {
98                 loop = false;
99             }
100         } else {
101             loop = false;
102         }
103         newCourse = null;
104         newCourseFull = false;
105     } while (loop);
106 }
107
108 public boolean askConfirmation(Course newCourse, String val, String val2) {
109     courseUI.displayCourse(newCourse, val);
110     return courseUI.getConfirmationChoice(val2, 1);
111 }
112
113 private void removeCourse(ListInterface<Course> courseList) {
114     boolean loop = false;
115     do {
116         Command.cls();
117         boolean deleteCourse, confirm;
118         int courseSelected = courseUI.inputRemoveCode(courseList);
119         deleteCourse = (courseSelected > -1);
120         if (deleteCourse == true) {
121             confirm = askConfirmation(courseList.getEntry(courseSelected), "Remove",
"remove");
122             if (confirm == true) {
123                 LinkedListInterface<CourseProgram> courseProgramList =
cpDAO.dLLRetrieveFromFile("courseProgram.dat");
124                 cpm.deleteCourse(courseList.getEntry(courseSelected),
courseProgramList);
125                 courseList.remove(courseSelected);
126                 cDAO.saveToFile(courseList, "course.dat");
127                 MessageUI.displaySuccessConfirmationMessage("Removed");
128                 Command.pressEnterToContinue();
129                 loop = courseUI.getConfirmationChoice("remove", 0);
130             } else {
131                 loop = false;
132             }

```

```
133         } else {
134             loop = false;
135         }
136     } while (loop);
137 }
138
139 private void searchCourse(ListInterface<Course> courseList, String val) {
140     boolean loop = true;
141     do {
142         Command.cls();
143         int choice = courseUI.getSearchMenuChoices();
144         switch (choice) {
145             case 0:
146                 loop = false;
147                 break;
148             case 1:
149                 searchCourseCode(courseList, val);
150                 break;
151             case 2:
152                 searchCourseTitle(courseList, val);
153                 break;
154             default:
155                 MessageUI.displayInvalidChoiceMessage();
156         }
157     } while (loop);
158 }
159
160
161 public void searchCourseCode(ListInterface<Course> courseList, String val) {
162     boolean loop;
163     do {
164         int found = courseUI.getSearchCourseCode(courseList);
165         if (found == -1) {
166             MessageUI.displayNotFoundMessage();
167         } else {
168             courseUI.displayCourseDetailsFounded(courseList, found);
169             if (val == "ammend") {
170                 ammendCourse(courseList, found);
171             }
172         }
173         loop = courseUI.getConfirmationChoice(val, 0);
174     } while (loop == true);
175 }
176
177
178 private void searchCourseTitle(ListInterface<Course> courseList, String val) {
```

```
179         boolean loop = true;
180         ListInterface<Course> courseList2 = new ArrList();
181         do {
182             int found = courseUI.getSearchCourseTitle(courseList, courseList2);
183             if (found != 0) {
184                 if (found == -1) {
185                     MessageUI.displayNotFoundMessage();
186                 } else if (found == 1) {
187                     if (val == "ammend") {
188
189                         int index = courseUI.getCourseAmmend(courseList, courseList2);
190                         if (index != -1) {
191                             ammendCourse(courseList, index);
192                         } else {
193                             loop = false;
194                         }
195                     } else {
196                         courseUI.displayCourseFounded(courseList2);
197                     }
198                 }
199                 courseList2.clear();
200                 loop = courseUI.getConfirmationChoice(val, 0);
201             } else if (found == 0) {
202                 loop = false;
203             }
204         } while (loop);
205     }
206
207     private void ammendCourse(ListInterface<Course> courseList, int found) {
208         int choice = 0;
209         Course tempCourse = courseList.getEntry(found + 1);
210         Course tempCourse2 = tempCourse;
211         String newCourseCode = tempCourse.getCourseCode();
212         String newTitle = tempCourse.getTitle();
213         Sem newSem = tempCourse.getSemester();
214         int newCreditHours = tempCourse.getCreditHours();
215         // do {
216         do {
217             Command.cls();
218             choice = courseUI.getAmmendMenuChoices();
219
220             switch (choice) {
221                 case 0:
222                     break;
223                 case 1:
224                     newCourseCode = courseUI.inputCourseCode(courseList);
```

```
225         if (newCourseCode.equals("0")) {
226             newCourseCode = tempCourse.getCourseCode();
227         }
228         break;
229     case 2:
230         newTitle = courseUI.inputTitle();
231         if (newTitle.equals("0")) {
232             newTitle = tempCourse.getTitle();
233         }
234         break;
235     case 3:
236         newCreditHours = courseUI.inputCreditHour();
237         if (newCreditHours == 0) {
238             newCreditHours = tempCourse.getCreditHours();
239         }
240         break;
241     case 4:
242         newSem = courseUI.inputSemester();
243         if (newSem == null) {
244             newSem = tempCourse.getSemester();
245         }
246         break;
247     default:
248         MessageUI.displayInvalidChoiceMessage();
249     }
250 } while (choice != 0);
251 tempCourse = new Course(newCourseCode, newTitle, newCreditHours, newSem);
252 tempCourse.setCreatedAt(tempCourse2.getCreatedAt());
253 boolean match = courseList.contains(tempCourse);
254 int match2 = tempCourse2.compareSem(tempCourse.getSemester());
255 if (!match || match2 != 0) {
256     boolean confirm = askConfirmation(tempCourse, "Ammend", "ammend");
257     if (confirm == true) {
258         LinkedListInterface<CourseProgram> cp = new DoublyLinkedList<>();
259         cp = cpDAO.dLLRetrieveFromFile("courseProgram.dat");
260
261         cpm.modifyCourse(tempCourse, tempCourse2, cp);
262         tempCourse.update();
263         courseList.replace(found + 1, tempCourse);
264         cDAO.saveToFile(courseList, "course.dat");
265         MessageUI.displaySuccessConfirmationMessage("Ammend");
266         Command.pressEnterToContinue();
267 //         loop = courseUI.getConfirmationChoice("ammend", 0);
268     }
269 } else {
270     MessageUI.printFormattedText("No changing\n", ConsoleColor.YELLOW);
```

```
271     }
272 //         } while (loop);
273
274     }
275
276     public void listedCourse(ListInterface<Course> courseList) {
277         int choice;
278         s.insertionSort(courseList, "courseCode");
279         courseUI.listAllCourses(courseList);
280         do {
281             choice = courseUI.getSortMenu(courseList);
282             Command.cls();
283             switch (choice) {
284                 case 0:
285                     break;
286                 case 1:
287                     ascListedCourse(courseList);
288                     break;
289                 case 2:
290                     desListedCourse(courseList);
291                     break;
292                 default:
293                     MessageUI.displayInvalidChoiceMessage();
294             }
295         } while (choice != 0);
296     }
297
298     private void ascListedCourse(ListInterface<Course> courseList) {
299         int choice;
300         do {
301             choice = courseUI.getSortMenuChoice(courseList, "Ascending Order");
302             switch (choice) {
303                 case 0:
304                     break;
305                 case 1:
306                     s.insertionSort(courseList, "courseCode");
307                     break;
308                 case 2:
309                     s.insertionSort(courseList, "title");
310                     break;
311                 case 3:
312                     s.insertionSort(courseList, "creditHours");
313                     break;
314                 case 4:
315                     s.insertionSort(courseList, "semester");
316                     break;
```

```
317         case 5:
318             s.insertionSort(courseList, "createdAt");
319             break;
320         case 6:
321             s.insertionSort(courseList, "updatedAt");
322             break;
323         default:
324             MessageUI.displayInvalidChoiceMessage();
325     }
326     courseUI.listAllCourses(courseList);
327 } while (choice != 0);
328 }
329
330 private void desListedCourse(ListInterface<Course> courseList) {
331     int choice;
332     do {
333         choice = courseUI.getSortMenuChoice(courseList, "Descending Order");
334         switch (choice) {
335             case 0:
336                 break;
337             case 1:
338                 s.insertionSortDes(courseList, "courseCode");
339                 break;
340             case 2:
341                 s.insertionSortDes(courseList, "title");
342                 break;
343             case 3:
344                 s.insertionSortDes(courseList, "creditHours");
345                 break;
346             case 4:
347                 s.insertionSortDes(courseList, "semester");
348                 break;
349             case 5:
350                 s.insertionSortDes(courseList, "createdAt");
351                 break;
352             case 6:
353                 s.insertionSortDes(courseList, "updatedAt");
354                 break;
355             default:
356                 MessageUI.displayInvalidChoiceMessage();
357         }
358         courseUI.listAllCourses(courseList);
359     } while (choice != 0);
360
361 }
362
```


363 }