```
 1
 2 package control;
 3
 4 import adt.*;
 5 import boundary.CourseMaintenanceUI;
 6 import boundary.CourseProgramMaintenanceUI;
 7 import dao.DAO;
 8 import dao.Initializer;
 9 import entity.Course;
10 import entity.Course.Sem;
11 import entity.CourseCodeComparator;
12 import entity.CourseProgram;
13 import entity.CreatedAtComparator;
14 import entity.CreditHoursComparator;
15 import entity.Program;
16 import entity.SemesterComparator;
17 import entity.TitleComparator;
18 import entity.UpdatedAtComparator;
19 import java.io.IOException;
20 import java.util.Comparator;
21 import utility.*;
22
23 /**
24  *
25  * @author Chew Lip Sin
26  */
27 public class CourseMaintenance {
28
29 //    private ListInterface<Course> courseList = new ArrList<>();
30     private final CourseMaintenanceUI courseUI = new CourseMaintenanceUI();
31     private static final DAO<Course> cDAO = new DAO<>();
32     private static final DAO<Program> pDAO = new DAO<>();
33     private final DAO<CourseProgram> cpDAO = new DAO<>();
34     private static final CourseProgramMaintenance cpm = new CourseProgramMaintenance();
35     private final Initializer in = new Initializer();
36     private final TitleComparator titleC = new TitleComparator();
37     private final CourseCodeComparator cCodeC = new CourseCodeComparator();
38     private final CreditHoursComparator cHoursC = new CreditHoursComparator();
39     private final SemesterComparator semC = new SemesterComparator();
40     private final CreatedAtComparator createdAtC = new CreatedAtComparator();
41     private final UpdatedAtComparator updatedAtC = new UpdatedAtComparator();
42
43     public CourseMaintenance() {
44     }
45 //
46 //    public static void main(String[] args) throws IOException, InterruptedException {
```

```
47 //        CourseMaintenance cMain = new CourseMaintenance();
48 //        cMain.runCourseMaintenance();
49 //    }
50
51     public void runCourseMaintenance() throws IOException, InterruptedException {
52         ListInterface<Course> courseList = cDAO.retrieveFromFile("course.dat");
53 //        ListInterface<Course> courseList = in.initializeCourse();
54 //        cDAO.saveToFile(courseList, "course.dat");
55         ListInterface<Program> programList = pDAO.retrieveFromFile("program.dat");
56 //        ListInterface<Program> programList = in.ProgramInitializer();
57 //        pDAO.saveToFile(programList, "program.dat");
58         int choice;
59         do {
60             Command.cls();
61             choice = courseUI.getMenuChoices();
62             switch (choice) {
63                 case 0:
64                     break;
65                 case 1:
66                     addNewCourse(courseList);
67                     break;
68                 case 2:
69                     removeCourse(courseList);
70                     break;
71                 case 3:
72                     searchCourse(courseList, "search");
73                     break;
74                 case 4:
75                     searchCourse(courseList, "ammend");
76                     break;
77                 case 5:
78                     listedCourse(courseList);
79                     break;
80                 case 6:
81                     CourseProgramMaintenance coursePM = new CourseProgramMaintenance();
82                     coursePM.runCourseProgramMaintenance(courseList, programList);
83                     break;
84                 case 7:
85                     CourseGenerateReportMaintenance courseGRM = new
CourseGenerateReportMaintenance();
86                     courseGRM.runCourseGenerateReportMaintenance();
87                     break;
88                 default:
89                     MessageUI.displayInvalidChoiceMessage();
90
91             }
```

```
 92
 93          } while (choice != 0);
 94          pDAO.saveToFile(programList, "program.dat");
 95          cDAO.saveToFile(courseList, "course.dat");
 96
 97      }
 98
 99      private void addNewCourse(ListInterface<Course> courseList) {
100          boolean loop = false;
101          do {
102              Command.cls();
103              boolean newCourseFull, confirm = true;
104              Course newCourse = courseUI.inputCourseDetails(courseList);
105              newCourseFull = (!newCourse.equals(new Course(null, null, 0, null)));
106              if (newCourseFull == true) {
107                  confirm = askConfirmation(newCourse, "New", "add");
108                  if (confirm == true) {
109                      courseList.add(newCourse);
110                      cDAO.saveToFile(courseList, "course.dat");
111                      MessageUI.displaySuccessConfirmationMessage("Added");
112                      loop = courseUI.getConfirmationChoice("add", 0);
113                  } else {
114                      loop = false;
115                  }
116              } else {
117                  loop = false;
118              }
119              newCourse = null;
120              newCourseFull = false;
121          } while (loop);
122      }
123
124      public boolean askConfirmation(Course newCourse, String val, String val2) {
125          courseUI.displayCourse(newCourse, val);
126          return courseUI.getConfirmationChoice(val2, 1);
127      }
128
129      private void removeCourse(ListInterface<Course> courseList) {
130          boolean loop = false;
131          do {
132              Command.cls();
133              boolean deleteCourse, confirm;
134              int courseSelected = courseUI.inputRemoveCode(courseList);
135              deleteCourse = (courseSelected > -1);
136              if (deleteCourse == true) {
137                  confirm = askConfirmation(courseList.getEntry(courseSelected), "Remove",
```

```
"remove");
                if (confirm == true) {
                    LinkedListInterface<CourseProgram> courseProgramList =
cpDAO.dLLRetrieveFromFile("courseProgram.dat");
                    cpm.deleteCourse(courseList.getEntry(courseSelected),
courseProgramList);
                    courseList.remove(courseSelected);
                    cDAO.saveToFile(courseList, "course.dat");
                    MessageUI.displaySuccessConfirmationMessage("Removed");
                    Command.pressEnterToContinue();
                    loop = courseUI.getConfirmationChoice("remove", 0);
                } else {
                    loop = false;
                }
            } else {
                loop = false;
            }
        } while (loop);
    }

    private void searchCourse(ListInterface<Course> courseList, String val) {
        boolean loop = true;
        do {
            Command.cls();
            int choice = courseUI.getSearchMenuChoices();
            switch (choice) {
                case 0:
                    loop = false;
                    break;
                case 1:
                    searchCourseCode(courseList, val);
                    break;
                case 2:
                    searchCourseTitle(courseList, val);
                    break;
                default:
                    MessageUI.displayInvalidChoiceMessage();

            }
        } while (loop);
    }

    public void searchCourseCode(ListInterface<Course> courseList, String val) {
        boolean loop;
        do {
            int found = courseUI.getSearchCourseCode(courseList);
```

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

```
227         String newCourseCode = tempCourse.getCourseCode();
228         String newTitle = tempCourse.getTitle();
229         Sem newSem = tempCourse.getSemester();
230         int newCreditHours = tempCourse.getCreditHours();
231 //        do {
232         do {
233             Command.cls();
234             choice = courseUI.getAmmendMenuChoices();
235 
236             switch (choice) {
237                 case 0:
238                     break;
239                 case 1:
240                     newCourseCode = courseUI.inputCourseCode(courseList);
241                     if (newCourseCode.equals("0")) {
242                         newCourseCode = tempCourse.getCourseCode();
243                     }
244                     break;
245                 case 2:
246                     newTitle = courseUI.inputTitle();
247                     if (newTitle.equals("0")) {
248                         newTitle = tempCourse.getTitle();
249                     }
250                     break;
251                 case 3:
252                     newCreditHours = courseUI.inputCreditHour();
253                     if (newCreditHours == 0) {
254                         newCreditHours = tempCourse.getCreditHours();
255                     }
256                     break;
257                 case 4:
258                     newSem = courseUI.inputSemester();
259                     if (newSem == null) {
260                         newSem = tempCourse.getSemester();
261                     }
262                     break;
263                 default:
264                     MessageUI.displayInvalidChoiceMessage();
265             }
266         } while (choice != 0);
267         tempCourse = new Course(newCourseCode, newTitle, newCreditHours, newSem);
268         tempCourse.setCreatedAt(tempCourse2.getCreatedAt());
269         boolean match = courseList.contains(tempCourse);
270         int match2 = tempCourse2.compareSem(tempCourse.getSemester());
271         if (!match || match2 != 0) {
272             boolean confirm = askConfirmation(tempCourse, "Ammend", "ammend");
```

```
273             if (confirm == true) {
274                 LinkedListInterface<CourseProgram> cp = new DoublyLinkedList<>();
275                 cp = cpDAO.dLLRetrieveFromFile("courseProgram.dat");
276 
277                 cpm.modifyCourse(tempCourse, tempCourse2, cp);
278                 tempCourse.update();
279                 courseList.replace(found + 1, tempCourse);
280                 cDAO.saveToFile(courseList, "course.dat");
281                 MessageUI.displaySuccessConfirmationMessage("Ammend");
282                 Command.pressEnterToContinue();
283 //                    loop = courseUI.getConfirmationChoice("ammend", 0);
284             }
285         } else {
286             MessageUI.printFormattedText("No changing\n", ConsoleColor.YELLOW);
287         }
288 //        } while (loop);
289 
290     }
291 
292     public void listedCourse(ListInterface<Course> courseList) {
293         int choice;
294         ArrList.insertionSort(courseList, cCodeC, "asc");
295         courseUI.listAllCourses(courseList);
296         do {
297             choice = courseUI.getSortMenu(courseList);
298             Command.cls();
299             switch (choice) {
300                 case 0:
301                     break;
302                 case 1:
303                     ascListedCourse(courseList);
304                     break;
305                 case 2:
306                     desListedCourse(courseList);
307                     break;
308                 default:
309                     MessageUI.displayInvalidChoiceMessage();
310             }
311         } while (choice != 0);
312     }
313 
314     private void ascListedCourse(ListInterface<Course> courseList) {
315 
316         int choice;
317 
318         do {
```

```
319             choice = courseUI.getSortMenuChoice(courseList, "Ascending Order");
320             switch (choice) {
321                 case 0:
322                     break;
323                 case 1:
324                     ArrList.insertionSort(courseList, cCodeC, "asc");
325                     break;
326                 case 2:
327                     ArrList.insertionSort(courseList, titleC, "asc");
328                     break;
329                 case 3:
330                     ArrList.insertionSort(courseList, cHoursC, "asc");
331                     break;
332                 case 4:
333                     ArrList.insertionSort(courseList, semC, "asc");
334                     break;
335                 case 5:
336                     ArrList.insertionSort(courseList, createdAtC, "asc");
337                     break;
338                 case 6:
339                     ArrList.insertionSort(courseList, updatedAtC, "asc");
340                     break;
341                 default:
342                     MessageUI.displayInvalidChoiceMessage();
343             }
344             courseUI.listAllCourses(courseList);
345         } while (choice
346                 != 0);
347     }
348 
349     private void desListedCourse(ListInterface<Course> courseList) {
350         int choice;
351         do {
352             choice = courseUI.getSortMenuChoice(courseList, "Descending Order");
353             switch (choice) {
354                 case 0:
355                     break;
356                 case 1:
357                     ArrList.insertionSort(courseList, cCodeC, "des");
358                     break;
359                 case 2:
360                     ArrList.insertionSort(courseList, titleC, "des");
361                     break;
362                 case 3:
363                     ArrList.insertionSort(courseList, cHoursC, "des");
364                     break;
```

```
365                 case 4:
366                     ArrList.insertionSort(courseList, semC, "des");
367                     break;
368                 case 5:
369                     ArrList.insertionSort(courseList, createdAtC, "des");
370                     break;
371                 case 6:
372                     ArrList.insertionSort(courseList, updatedAtC, "des");
373                     break;
374                 default:
375                     MessageUI.displayInvalidChoiceMessage();
376             }
377             courseUI.listAllCourses(courseList);
378         } while (choice != 0);
379 
380     }
381 
382 }
```