```
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;
 7 import adt.ArrList;
 8 import adt.DoublyLinkedList;
 9 import adt.LinkedListInterface;
10 import adt.ListInterface;
11 import adt.OrderClause;
12 import adt.StackInterface;
13 import entity.Course;
14 import entity.Course.Sem;
15 import entity.CourseCodeComparator;
16 import entity.CourseProgram;
17 import entity.CreditHoursComparator;
18 import entity. Semester Comparator;
19 import entity. Title Comparator;
20 import java.time.LocalDateTime;
21 import java.time.format.DateTimeFormatter;
22 import java.util.Iterator;
23 import utility.Command;
24 import utility.ConsoleColor;
25 import utility.InputValue;
26 import utility.MessageUI;
27
28 /**
29 *
30 * @author Chew Lip Sin
31 */
32 public class CourseGenerateReportMaintenanceUI {
33
34
       private LinkedListInterface<CourseProgram> cp = new DoublyLinkedList<>();
35
       private ListInterface<Course> courses = new ArrList<>();
36
       private final InputValue iv = new InputValue();
37
       /**
38
39
        * Constructs a CourseGenerateReportMaintenanceUI with the given course
40
         * programs and courses.
41
42
         * @param cp The linked list of course programs.
43
         * @param courses The list of courses.
44
         * /
```

```
public CourseGenerateReportMaintenanceUI(LinkedListInterface<CourseProgram> cp,
ListInterface<Course> courses) {
46
           this.cp = cp;
47
           this.courses = courses;
48
       }
49
50
       private final LocalDateTime myDateObj = LocalDateTime.now();
51
       DateTimeFormatter myFormatObj = DateTimeFormatter.ofPattern("dd-MM-yyyy");
52
       DateTimeFormatter myFormatObj2 = DateTimeFormatter.ofPattern("h:mm a");
53
       String formattedDate = myDateObj.format(myFormatObj);
54
       String formattedTime = myDateObj.format(myFormatObj2);
55
       private final CourseCodeComparator cCodeC = new CourseCodeComparator();
56
       private final CreditHoursComparator cHoursC = new CreditHoursComparator();
57
       private final TitleComparator titleC = new TitleComparator();
58
       private final SemesterComparator semC = new SemesterComparator();
59
60
61
        * Displays the header for the course report menu.
        * /
62
       public void displayHeader() {
63
64
           Command.cls();
65
           System.out.println("\t\t\t=========");
66
           System.out.println("\t\t\t||
                                            Course Report Menu
                                                                                 | | " ) ;
67
           System.out.println("\t\t\t=========");
68
       }
69
70
       /**
71
        * Displays the options available in the course report menu.
72
73
       public void displayReportMenu() {
74
           System.out.println("\t\t\t1. Generate Course and Program Report");
75
           System.out.println("\t\t\t\2. Generate Course Report");
76
           System.out.println("\t\t\t\0. Exit");
77
       }
78
79
       /**
80
        * Gets the user's choice from the course report menu.
81
82
        * @return The user's choice.
83
        * /
       public int getChoices() {
84
           int choice;
85
           do {
86
87
               System.out.print("\t\t\t\tEnter choice: ");
88
               choice = iv.readInteger();
               if (choice > 2 || choice < 0) {
89
```

```
MessageUI.displayInvalidChoiceMessage();
91
92
          } while (choice > 2 || choice < 0);</pre>
93
          return choice;
94
       }
95
       /**
96
97
        * Displays the header for the course program report.
98
        */
99
       public void displayCoursePReportHeader() {
100
          Command.cls();
101
          System.out.println("\t\t\t========");
102
          System.out.println("\t\t\t||
                                                                              | | " ) ;
                                              Course Program Report
103
          System.out.println("\t\t\t========");
104
       }
105
106
       /**
107
        * Displays the header for the course report.
108
       * /
109
       public void displayCourseReportHeader() {
110
          Command.cls();
111
          System.out.println("\t\t\t========");
112
          System.out.println("\t\t\t|)
                                                  Course Report
                                                                              | | " ) ;
113
          System.out.println("\t\t\t=========");
114
       }
115
116
       /**
117
        * Displays a progress animation, indicating the generation of a report.
118
119
       public void progress() {
120
          var anim = "|/-||;
121
          for (int i = 0; i \le 100; i = i + 50) {
122
              Command.progressPercentage(i, 100);
123
              try {
124
                 Thread.sleep(500);
125
              } catch (Exception e) {
126
              }
127
128
          MessageUI.displaySuccessConfirmationMessage("Generating Report");
129
          Command.pressEnterToContinue();
130
       }
131
132
       /**
133
        * Displays the course program report, showing course information grouped by
134
        * program.
135
        */
```

```
136
        public void displayCourseProgramReport() {
137
            String line = "";
138
            sortByProgramID();
139
            sortById();
140
            String oldCC = "";
141
            System.out.println("");
142
            System.out.println(String.format("\t\t\*-20s| %-20s|%-20s", "Course Code",
"Program Code", "Main/Elective"));
            for (int i = 0; i < 65; i++) {
143
                line += "-";
144
145
146
            Iterator<CourseProgram> it = cp.getIterator();
147
            while (it.hasNext()) {
148
                CourseProgram cp2 = it.next();
149
                if (oldCC.equals(cp2.getCourseCode())) {
150
                    System.out.print(String.format("\t\t\t\-20s|", ""));
151
                    String elective = strElective(cp2.isIsElective());
                    System.out.println(String.format(" %-20s| %-20s", cp2.getProgramCode(),
152
elective));
153
154
                } else {
155
                    System.out.println("\t\t\t" + line);
156
                    System.out.print(String.format("\t\t\-20s|", cp2.getCourseCode()));
157
                    String elective = strElective(cp2.isIsElective());
158
                    System.out.println(String.format(" %-20s| %-20s", cp2.getProgramCode(),
elective));
159
160
                oldCC = cp2.getCourseCode();
161
162
163
            }
164
165
            System.out.println("\t\t\t" + line);
166
            displayReportFooter();
167
            System.out.println("");
168
            System.out.print("\t\t\t");
169
            Command.pressEnterToContinue();
170
        }
171
        /**
172
173
         * Sorts the list of course programs by their course codes.
174
175
        public void sortById() {
176
            cp.orderBy((c1, c2)
177
                    -> c1.getCourseCode().compareTo(c2.getCourseCode()) < 0
178
                    ? OrderClause.MOVE FORWARD : OrderClause.MOVE BACKWARD);
```

```
179
180
181
        /**
182
         * Sorts the list of course programs by their program codes.
183
184
        public void sortByProgramID() {
185
            cp.orderBy((c1, c2)
186
                    -> c1.getProgramCode().compareTo(c2.getProgramCode()) < 0
187
                    ? OrderClause.MOVE FORWARD : OrderClause.MOVE BACKWARD);
188
        }
189
190
        /**
191
         * Converts a Boolean value indicating whether a course is elective or not
192
         * into a string representation.
193
194
         * @param isElective True if the course is elective, false otherwise.
195
         * @return The string representation "Elective" if true, or "Main" if false.
196
197
        private String strElective(boolean isElective) {
198
            if (isElective == true) {
199
                return "Elective";
200
            } else {
201
                return "Main";
202
203
204
        }
205
206
        /**
207
         * Displays the footer information for the generated report.
208
         * /
209
        public void displayReportFooter() {
210
            System.out.println("\n\t\tReport Generated by: University Education System");
211
            System.out.println("\t\tReport Generated at: " + formattedDate + " " +
formattedTime);
212
        }
213
214
        /**
215
         * Gets user choices for generating course reports and maintains a stack of
216
         * chosen reports.
217
218
         * @param choice A stack containing the user's report choices.
219
         * @return The updated stack of report choices.
220
         */
221
        public StackInterface<String> getCourseReportMenu(StackInterface<String> choice) {
222
            ListInterface<String> typeReport = new ArrList<>();
223
            boolean checkCode = false;
```

```
224
           boolean checkCredit = false;
225
           boolean checkSem = false;
226
           typeReport.add("Course Code Report");
227
           typeReport.add("Credit Hour Report");
228
           typeReport.add("Semester Report");
229
230
           int choice2;
231
           do {
232
               Command.cls();
                \verb|System.out.println("\t\t\t=========")| |
233
234
               System.out.println("\t\t\t|)
                                                           Course Report Menu
235
               System.out.println("\t\t\t========""),
236
               System.out.println("\t\t\t1. Course Code Report");
237
               System.out.println("\t\t\t\2. Credit Hour Report");
238
               System.out.println("\t\t\t\13. Semester Report");
239
               System.out.println("\t\t\t4. Undo");
240
               System.out.println("\t\t\t\0. Exit/Continue");
241
               System.out.print("\t\t\tEnter your choice(One report only choose one time):
");
242
               choice2 = iv.readInteger();
243
               if (choice2 < 0 \mid | choice2 > 4) {
244
                   MessageUI.displayInvalidChoiceMessage();
245
               } else if (choice2 == 1 && !checkCode) {
246
                   choice.push(typeReport.getEntry(1));
247
                   checkCode = true;
               } else if (choice2 == 2 && !checkCredit) {
248
249
                   choice.push(typeReport.getEntry(2));
250
                   checkCredit = true;
251
               } else if (choice2 == 3 && !checkSem) {
252
                   choice.push(typeReport.getEntry(3));
253
                   checkSem = true;
254
               } else if (choice2 == 4) {
255
                   if (!choice.isEmpty()) {
256
                       String popOut = choice.pop();
257
                       if (popOut.equals(typeReport.getEntry(1))) {
258
                           checkCode = false;
259
                       } else if (popOut.equals(typeReport.getEntry(2))) {
260
                           checkCredit = false;
261
                       } else if (popOut.equals(typeReport.getEntry(3))) {
262
                           checkSem = false;
263
                       }
264
                   } else {
265
                       MessageUI.printFormattedText("\t\t\tNothing can Undo\n",
ConsoleColor.YELLOW);
266
267
               } else if (choice2 == 0) {
```

```
System.out.println("");
268
269
                } else {
270
                    MessageUI.printFormattedText("\t\t\tYou have been assigned value into
the stack!\n", ConsoleColor.YELLOW);
271
272
                System.out.print("\t\t\t");
273
                Command.pressEnterToContinue();
274
            } while (choice2 != 0);
275
            return choice;
276
        }
277
278
        /**
279
         * Displays a report of courses grouped by their course codes.
280
         * /
281
        public void displayCourseCodeReport() {
282
            ArrList.insertionSort(courses, cCodeC, "asc");
283
            Iterator<Course> itA = courses.getIterator();
284
            Iterator<Course> itB = courses.getIterator();
285
            Iterator<Course> itF = courses.getIterator();
286
            Iterator<Course> itM = courses.getIterator();
287
288
            String line = "\t\t\t";
289
            for (int i = 0; i < 65; i++) {
290
                line += "-";
291
292
            int countA = 1, countB = 1, countF = 1, countM = 1;
293
            System.out.println(line);
294
            System.out.println("\t\tCourse Code Start with 'A'");
295
            while (itA.hasNext()) {
296
                Course course = itA.next();
297
                if (course.getCourseCode().charAt(0) == 'A') {
298
                    System.out.println(String.format("\t\t\2d. %8s %s", countA,
course.getCourseCode(), course.getTitle()));
299
                    countA++;
300
                }
301
            }
302
303
            System.out.println(String.format("\t\tTotal = %d", countA - 1));
304
            System.out.println(line);
305
            System.out.println("\t\tCourse Code Start with 'B'");
306
            while (itB.hasNext()) {
307
                Course course = itB.next();
308
                if (course.getCourseCode().charAt(0) == 'B') {
                    System.out.println(String.format("\t\t\t\2d. \88s \8s", countB,
309
course.getCourseCode(), course.getTitle()));
310
                    countB++;
```

```
311
312
                }
313
314
            System.out.println(String.format("\t\t\tTotal = %d", countB - 1));
315
            System.out.println(line);
316
            System.out.println("\t\tCourse Code Start with 'F'");
317
            while (itF.hasNext()) {
318
                Course course = itF.next();
319
                if (course.getCourseCode().charAt(0) == 'F') {
320
                    System.out.println(String.format("\t\t\2d. %8s %s", countF,
course.getCourseCode(), course.getTitle()));
321
                    countF++;
322
                }
323
324
            System.out.println(String.format("\t\tTotal = %d", countF - 1));
325
            System.out.println(line);
326
            System.out.println("\t\t\tCourse Code Start with 'M'");
327
            while (itM.hasNext()) {
328
                Course course = itM.next();
                if (course.getCourseCode().charAt(0) == 'M') {
329
330
                    System.out.println(String.format("\t\t\t\2d. %8s %s", countM,
course.getCourseCode(), course.getTitle()));
331
                    countM++;
332
                }
333
334
            System.out.println(String.format("\t\t\tTotal = %d", countM - 1));
335
336
        }
337
338
        /**
339
         * Displays a report of courses grouped by their credit hours.
340
341
        public void displayCreditHoursReport() {
342
            ArrList.insertionSort(courses, cHoursC, "asc");
343
            String line = "";
344
            int count1 = 1, count2 = 1, count3 = 1, count4 = 1, count5 = 1;
            for (int i = 0; i < 65; i++) {
345
346
                line += "-";
347
348
            System.out.println("\t\t\t" + line);
349
            System.out.println("\t\tCourse with 1 credit hour:");
350
            for (int i = 1; i <= courses.size(); i++) {</pre>
351
                if (courses.getEntry(i).getCreditHours() == 1) {
352
                    System.out.println(String.format("\t\t\t\2d. %s %s", count1,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
353
                    count1++;
```

```
354
355
356
            System.out.println(String.format("\t\t\tTotal = %d", count1 - 1));
357
            System.out.println("\t\t\t" + line);
358
            System.out.println("\t\tCourse with 2 credit hours:");
359
            for (int i = 1; i <= courses.size(); i++) {</pre>
360
                if (courses.getEntry(i).getCreditHours() == 2) {
361
                    System.out.println(String.format("\t\t\t\2d. %s %s", count2,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
362
                    count2++;
363
                }
364
365
            System.out.println(String.format("\t\t\tTotal = %d", count2 - 1));
366
            System.out.println("\t\t\t" + line);
367
            System.out.println("\t\tCourse with 3 credit hours:");
368
369
            for (int i = 1; i <= courses.size(); i++) {</pre>
370
                if (courses.getEntry(i).getCreditHours() == 3) {
371
                    System.out.println(String.format("\t\t\t\2d. %s %s", count3,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
372
                    count3++;
373
                }
374
375
            System.out.println(String.format("\t\t\tTotal = %d", count3 - 1));
376
            System.out.println("\t\t\t" + line);
377
            System.out.println("\t\tCourse with 4 credit hours:");
378
379
            for (int i = 1; i <= courses.size(); i++) {
380
                if (courses.getEntry(i).getCreditHours() == 4) {
381
                    System.out.println(String.format("\t\t\t\2d. %s %s", count4,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
382
                    count4++;
383
                }
384
385
            System.out.println(String.format("\t\t\tTotal = %d", count4 - 1));
386
            System.out.println("\t\t\" + line);
387
            System.out.println("\t\tCourse with 5 credit hours:");
388
389
            for (int i = 1; i <= courses.size(); i++) {</pre>
390
                if (courses.getEntry(i).getCreditHours() == 5) {
391
                    System.out.println(String.format("\t\t\2d. %s %s", count5,
courses.getEntry(i).getCourseCode(), courses.getEntry(i).getTitle()));
392
                    count5++;
393
                }
394
395
            System.out.println(String.format("\t\t\tTotal = %d", count5 - 1));
```

```
396
397
398
        /**
399
         * Displays a report of courses grouped by their intake semester (January or
400
         * July).
         */
401
402
        public void displayCourseSemesterReport() {
403
            Sem janu = Sem.JAN;
404
            Sem july = Sem.JUL;
405
406
            Sem all1 = Sem.ALL;
407
408
            ArrList.insertionSort(courses, titleC, "asc");
409
            ArrList.insertionSort(courses, semC, "asc");
410
            int countJan = 0, countJuly = 0;
411
            Iterator<Course> itJan = courses.getIterator();
412
            Iterator<Course> itJuly = courses.getIterator();
            String line = "";
413
            for (int i = 0; i < 65; i++) {
414
415
                line += "-";
416
417
            System.out.println("\t\t\t" + line);
418
            System.out.println("\t\tCourse intake on January:");
419
            while (itJan.hasNext()) {
420
                Course course = itJan.next();
421
                int all = course.compareSem(all1);
422
                int jan = course.compareSem(janu);
423
                if (all == 0 || jan == 0) {
424
                    countJan++;
425
                    System.out.println(String.format("\t\t\$2d. %8s %s", countJan,
course.getCourseCode(), course.getTitle()));
426
                }
427
428
            System.out.println("\t\t\tTotal = " + countJan);
429
            System.out.println("\t\t\" + line);
430
            System.out.println("\t\t\Course intake on July:");
431
            while (itJuly.hasNext()) {
432
                Course course = itJuly.next();
433
                int all = course.compareSem(all1);
434
                int jul = course.compareSem(july);
435
                if (all == 0 || jul == 0) {
436
                    countJuly++;
437
                    System.out.println(String.format("\t\t\t\2d. %8s %s", countJuly,
course.getCourseCode(), course.getTitle()));
438
                }
439
            }
```

