File - C:\Users\ctqdt\IdeaProjects\CourseScheduler\src\com\company\data\course\Course.java

```java
/**
 * @author Caroline Ta
 * @since 05.19.2020
 */
package com.company.data.course;

import com.company.data.person.Faculty;
import com.company.data.person.Student;

import java.io.File;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.Scanner;

/**
 * The type Course.
 */
public class Course implements Comparable<Course> {

    /**
     * The Course department.
     */
    private String department;
    /**
     * The Course code.
     */
    private String code;
    /**
     * The Course description.
     */
    private String description;
    /**
     * The Course id.
     */
    private String courseId;
    /**
     * The Course minimum number of student.
     */
    private int minStudentForCourse;
    /**
     * The Course maximum number of student.
     */
    private int maxStudentForCourse;
    /**
     * The status if the Course is cancelled.
     */
    private boolean cancelledStatus;
    /**
     * The Student list.
     */
    public ArrayList<Student> studentList;
    /**
     * The Session list.
     */
    public ArrayList<Session> sessionList;

    // ------------------------------------------------------------------------
    // CONSTRUCTORS
    // ------------------------------------------------------------------------

    /**
     * Instantiates a new Course.
     */
    public Course() {

    }

    /**
     * Instantiates a new Course.
```

```
71       *
72       * @param department          the department
73       * @param code                the code
74       * @param description         the description
75       * @param courseId            the course id
76       * @param minStudentForCourse the min student for course
77       * @param maxStudentForCourse the max student for course
78       * @throws Exception the exception
79       */
80      public Course(String department,
81                    String code,
82                    String description,
83                    String courseId,
84                    int minStudentForCourse,
85                    int maxStudentForCourse) throws Exception {
86          setDepartment(department);
87          setCode(code);
88          setDescription(description);
89          setCourseId(courseId);
90          setMinStudentForCourse(minStudentForCourse);
91          setMaxStudentForCourse(maxStudentForCourse);
92
93          studentList = new ArrayList<>();
94          sessionList = new ArrayList<>();
95      }
96
97      // --------------------------------------------------------------------------
98      // ACCESSORS - GETTER METHODS
99      // --------------------------------------------------------------------------
100
101     /**
102      * Gets department.
103      *
104      * @return the department
105      */
106     public String getDepartment() {
107         return department;
108     }
109
110     /**
111      * Gets code.
112      *
113      * @return the code
114      */
115     public String getCode() {
116         return code;
117     }
118
119     /**
120      * Gets description.
121      *
122      * @return the description
123      */
124     public String getDescription() {
125         return description;
126     }
127
128     /**
129      * Gets course id.
130      *
131      * @return the course id
132      */
133     public String getCourseId() {
134         generateId();
135         return courseId;
136     }
137
138     /**
139      * Gets min student for course.
140      *
```

```
141      * @return the min student for course
142      */
143     public int getMinStudentForCourse() {
144         return minStudentForCourse;
145     }
146 
147     /**
148      * Gets max student for course.
149      *
150      * @return the max student for course
151      */
152     public int getMaxStudentForCourse() {
153         return maxStudentForCourse;
154     }
155 
156     /**
157      * Gets cancelled status.
158      *
159      * @return the cancelled status
160      */
161     public boolean getCancelledStatus() {
162         updateCancelledStatus();
163         return cancelledStatus;
164     }
165 
166     // ------------------------------------------------------------------------
167     // MUTATORS - SETTER METHODS
168     // ------------------------------------------------------------------------
169 
170     /**
171      * Sets department.
172      *
173      * @param department the department
174      * @throws Exception the exception
175      */
176     public void setDepartment(String department) throws Exception {
177         checkStringValue(department, "Department");
178         this.department = department;
179     }
180 
181     /**
182      * Sets code.
183      *
184      * @param code the code
185      * @throws Exception the exception
186      */
187     public void setCode(String code) throws Exception {
188         checkStringValue(code, "Code");
189         this.code = code;
190     }
191 
192     /**
193      * Sets description.
194      *
195      * @param description the description
196      * @throws Exception the exception
197      */
198     public void setDescription(String description) throws Exception {
199         checkStringValue(description, "Description");
200         this.description = description;
201     }
202 
203     /**
204      * Sets course id.
205      *
206      * @param courseId the course id
207      * @throws Exception the exception
208      */
209     public void setCourseId(String courseId) throws Exception {
210         checkStringValue(courseId, "CourseId");
```

```
211         this.courseId = courseId;
212     }
213 
214     /**
215      * Sets min student for course.
216      *
217      * @param minStudentForCourse the min student for course
218      */
219     public void setMinStudentForCourse(int minStudentForCourse) {
220         this.minStudentForCourse = minStudentForCourse;
221     }
222 
223     /**
224      * Sets max student for course.
225      *
226      * @param maxStudentForCourse the max student for course
227      */
228     public void setMaxStudentForCourse(int maxStudentForCourse) {
229         this.maxStudentForCourse = maxStudentForCourse;
230     }
231 
232     /**
233      * Sets cancelled status.
234      *
235      * @param cancelledStatus the cancelled status
236      */
237     public void setCancelledStatus(boolean cancelledStatus) {
238         this.cancelledStatus = cancelledStatus;
239     }
240 
241     // ------------------------------------------------------------------------
242     // SESSION-RELATED METHODS
243     // ------------------------------------------------------------------------
244 
245     /**
246      * Find session session.
247      *
248      * @param sessionId the session id
249      * @return the session
250      */
251     public Session findSession(String sessionId) {
252         for (Session s : sessionList) {
253             if (s.getSessionId().equals(sessionId)) {
254                 return s;
255             }
256         }
257         return null;
258     }
259 
260     /**
261      * Add session.
262      *
263      * @param session the session
264      * @throws Exception the exception
265      */
266     public void addSession(Session session) throws Exception {
267         Session sInDirectory = findSession(session.getSessionId());
268 
269         if (sInDirectory != null) {
270             throw new Exception("Session with id "
271                     + sInDirectory.getSessionId() + " is already in the list.\n");
272         }
273         sessionList.add(session);
274     }
275 
276     /**
277      * Remove session.
278      *
279      * @param session the session
280      */
```

```
281     public void removeSession(Session session) {
282         sessionList.remove(session);
283     }
284
285     /**
286      * Sort.
287      *
288      * @param comparator the comparator
289      */
290     public void sortSession(Comparator<Session> comparator) {
291         sessionList.sort(comparator);
292     }
293
294     /**
295      * Generate session list.
296      *
297      * @param pathName the path name
298      */
299     public void generateSessionList(String pathName) {
300         String line;
301         Session sessionToAdd;
302
303         try {
304             File sessionFile = new File(pathName);
305             Scanner input = new Scanner(sessionFile);
306
307             while (input.hasNext()) {
308                 line = input.nextLine();
309                 String[] lineAr = line.split("_");
310
311                 sessionList.add(new Session(courseId, lineAr[0]));
312                 sessionList.add(new Session(courseId, lineAr[1]));
313                 sessionList.add(new Session(courseId, lineAr[2]));
314                 sessionList.add(new Session(courseId, lineAr[3]));
315                 sessionList.add(new Session(courseId, lineAr[4]));
316             }
317
318         } catch (Exception e) {
319             e.printStackTrace();
320         }
321     }
322
323     // ------------------------------------------------------------------------
324     // FUNCTIONALITY METHODS
325     // ------------------------------------------------------------------------
326
327     /**
328      * Check string value.
329      *
330      * @param valueString   the value string
331      * @param attributeName the attribute name
332      * @throws Exception the exception
333      */
334     protected void checkStringValue(String valueString, String attributeName) throws Exception {
335         if (valueString == null || valueString.trim().equals("")) {
336             throw new Exception(attributeName + " must be non-null and non-empty");
337         }
338     }
339
340     /**
341      * Generate id.
342      */
343     public void generateId() {
344         this.courseId = this.department + this.code;
345     }
346
347     /**
348      * Update cancelled status.
349      */
350     public void updateCancelledStatus() {
```

```
351         if (sessionList.size() == 0 || studentList.size() < minStudentForCourse) {
352             cancelledStatus = true;
353         } else {
354             cancelledStatus = false;
355         }
356     }
357 
358     /**
359      * Enroll student.
360      *
361      * @param student the student
362      * @param session the session
363      */
364     public void enrollStudent(Student student, Session session) {
365         if (studentList.size() < maxStudentForCourse
366                 && session.getStudentList().size() < session.maxStudentForSession) {
367             studentList.add(student);
368 //            sessionList.add(session);
369         }
370 
371         updateCancelledStatus();
372     }
373 
374     /**
375      * Unenroll student.
376      *
377      * @param student the student
378      * @param session the session
379      * @return true if student is unenrolled successfully.
380      */
381     public boolean unenrollStudent(Student student, Session session) {
382         boolean unenrolled = false;
383 
384         if (studentList.remove(student) && sessionList.remove(session)) {
385             studentList.remove(student);
386 //            sessionList.remove(session);
387             unenrolled = true;
388             updateCancelledStatus();
389         }
390 
391         return unenrolled;
392     }
393 
394     // ------------------------------------------------------------------------
395     // OVERRIDDEN METHODS
396     // ------------------------------------------------------------------------
397 
398     /**
399      * Override toString method.
400      *
401      * @return a string description for the Course class
402      */
403     @Override
404     public String toString() {
405         StringBuilder build = new StringBuilder();
406         updateCancelledStatus();
407 
408         build.append("\nCourse:               ");
409         build.append(courseId);
410 
411         build.append("\nDescription:          ");
412         build.append(description);
413 
414         build.append("\nClass size (min-max): ");
415         build.append(minStudentForCourse);
416         build.append("-");
417         build.append(maxStudentForCourse);
418 
419         build.append("\nOpen seats:           ");
420         build.append(maxStudentForCourse - studentList.size());
```

```
421         build.append("/");
422         build.append(maxStudentForCourse);
423 
424         build.append("\nStudents enrolled:    ");
425         build.append(studentList.size());
426 
427         build.append("\nCancelled status:     ");
428         build.append(getCancelledStatus());
429 
430         build.append("\n\n");
431 
432         build.append(sessionList.toString());
433 
434         return build.toString();
435     }
436 
437     /**
438      * Override compareTo method.
439      *
440      * @param otherCourse the other course
441      * @return comparision result (i.e. -1, 0, 1) between Courses
442      */
443     @Override
444     public int compareTo(Course otherCourse) {
445         return this.courseId.compareTo(otherCourse.courseId);
446     }
447 
448     /**
449      * Override equals method.
450      *
451      * @param obj the object
452      * @return the result of thisCourse equals otherCourse as a boolean
453      */
454     @Override
455     public boolean equals(Object obj) {
456         if (obj == null || !(obj instanceof Course)) {
457             return false;
458         }
459 
460         return compareTo((Course) obj) == 0;
461     }
462 }
463 
```