```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 org.jetbrains.annotations.NotNull;

import java.util.ArrayList;

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

    /**
     * The Min student for session.
     */
    final int minStudentForSession = 2;
    /**
     * The Max student for session.
     */
    final int maxStudentForSession = 7;
    /**
     * The course id of the session.
     */
    private String courseIdOfSession;
    /**
     * The session id.
     */
    private String sessionId;
    /**
     * The teacher for session.
     */
    private Faculty teacher;
    /**
     * The student list in the session.
     */
    private ArrayList<Student> studentList;
    /**
     * The number of students in session.
     */
    private int studentsInSession = 0;
    /**
     * The session's filled status.
     */
    private boolean isFilled;
    /**
     * The session's cancelled status.
     */
    private boolean isCancelled;

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

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

    }

    /**
     * Instantiates a new Session.
     *
     * @param courseIdOfSession the course id of session
```

```
71       * @param sessionId         the session id
72       * @throws Exception the exception
73       */
74      public Session(String courseIdOfSession, String sessionId) throws Exception
75      {
76          setCourseIdOfSession(courseIdOfSession);
77          setSessionId(sessionId);
78          studentList = new ArrayList<>();
79      }
80
81      // --------------------------------------------------------------------------
82      // ACCESSORS - GETTER METHODS
83      // --------------------------------------------------------------------------
84
85      /**
86       * Gets min student for session.
87       *
88       * @return the min student for session
89       */
90      public int getMinStudentForSession() {
91          return minStudentForSession;
92      }
93
94      /**
95       * Gets max student for session.
96       *
97       * @return the max student for session
98       */
99      public int getMaxStudentForSession() {
100         return maxStudentForSession;
101     }
102
103     /**
104      * Gets teacher.
105      *
106      * @return the teacher
107      */
108     public String getTeacher() {
109         return teacher.toString();
110     }
111
112     public String getCourseIdOfSession() {
113         return courseIdOfSession;
114     }
115
116     /**
117      * Gets session id.
118      *
119      * @return the session id
120      */
121     public String getSessionId() {
122         return sessionId;
123     }
124
125     /**
126      * Gets student list.
127      *
128      * @return the student list
129      */
130     public ArrayList<Student> getStudentList() {
131         return studentList;
132     }
133
134     /**
135      * Gets students in session.
136      *
137      * @return the students in session
138      */
139     public int getStudentsInSession() {
140         return studentsInSession;
```

```
141     }
142
143     /**
144      * Gets session filled status.
145      *
146      * @return true if session is filled.
147      */
148     public boolean getFilledStatus() {
149         updateFilledStatus();
150         return isFilled;
151     }
152
153     /**
154      * Gets session cancelled status.
155      *
156      * @return true if session is cancelled.
157      */
158     public boolean getCancelledStatus() {
159         updateCancelledStatus();
160         return isCancelled;
161     }
162
163     // ------------------------------------------------------------------------
164     // MUTATORS - SETTER METHODS
165     // ------------------------------------------------------------------------
166
167     /**
168      * Sets course id of session.
169      *
170      * @param courseIdOfSession the course id of session
171      * @throws Exception the exception
172      */
173     public void setCourseIdOfSession(String courseIdOfSession) throws Exception {
174         checkStringValue(courseIdOfSession, "CourseIdOfSession");
175         this.courseIdOfSession = courseIdOfSession;
176     }
177
178     /**
179      * Sets session id.
180      *
181      * @param sessionId the session id
182      * @throws Exception the exception
183      */
184     public void setSessionId(String sessionId) throws Exception {
185         checkStringValue(sessionId, "SessionId");
186         this.sessionId = sessionId;
187     }
188
189     /**
190      * Sets teacher.
191      *
192      * @param teacher the teacher
193      */
194     public void setTeacher(Faculty teacher) {
195         this.teacher = teacher;
196     }
197
198     public void unsetTeacher()
199     {
200         this.teacher = null;
201     }
202
203     /**
204      * Sets cancelled status.
205      *
206      * @param status cancelled
207      */
208     public void setCancelledStatus(boolean status) {
209         isCancelled = status;
210     }
```

```
211
212     // ------------------------------------------------------------------------
213     // FUNCTIONALITY METHODS
214     // ------------------------------------------------------------------------
215
216     /**
217      * Check string value.
218      *
219      * @param valueString   the value string
220      * @param attributeName the attribute name
221      * @throws Exception the exception
222      */
223     protected void checkStringValue(String valueString, String attributeName) throws Exception {
224         if (valueString == null || valueString.trim().equals("")) {
225             throw new Exception(attributeName + " must be non-null and non-empty");
226         }
227     }
228
229     /**
230      * Update filled status.
231      */
232     public void updateFilledStatus()
233     {
234         isFilled = studentList.size() == maxStudentForSession;
235     }
236
237     /**
238      * Update cancelled status.
239      */
240     public void updateCancelledStatus()
241     {
242         isCancelled = studentList.size() < minStudentForSession;
243     }
244
245     public void updateStudentsInSession()
246     {
247         studentsInSession = studentList.size();
248     }
249
250     /**
251      * Add course.
252      *
253      * @param studentToAdd the student to add
254      */
255     public void addStudent(Student studentToAdd) {
256
257         if (!studentList.contains(studentToAdd)) {
258             studentList.add(studentToAdd);
259         }
260
261         updateFilledStatus();
262         updateCancelledStatus();
263         updateStudentsInSession();
264     }
265
266     /**
267      * Remove course boolean.
268      *
269      * @param studentToRemove the student to remove
270      * @return the boolean
271      */
272     public boolean removeStudent(Student studentToRemove) {
273
274         studentList.remove(studentToRemove);
275         updateFilledStatus();
276         updateCancelledStatus();
277         updateStudentsInSession();
278
279         return studentList.remove(studentToRemove);
280     }
```

```
281
282     public String printSessionIdOnly()
283     {
284         StringBuilder build = new StringBuilder();
285         build.append("Session Id: ");
286         build.append(sessionId);
287         build.append("\tAvailable seats: ");
288         build.append(maxStudentForSession - studentsInSession);
289         build.append("/");
290         build.append(maxStudentForSession);
291
292         return build.toString();
293     }
294
295     public String printStudentList()
296     {
297         StringBuilder build = new StringBuilder();
298         for(int i = 0; i < studentList.size(); i++)
299         {
300             build.append("\n\tStudent #");
301             build.append(i+1);
302             build.append(": ");
303             build.append(studentList.get(i).getName());
304             build.append("\t");
305             build.append(studentList.get(i).getId());
306         }
307
308         build.append("\n");
309
310         return build.toString();
311     }
312
313     // ------------------------------------------------------------------------
314     // OVERRIDDEN METHODS
315     // ------------------------------------------------------------------------
316
317     /**
318      * Override toString method.
319      *
320      * @return a string description of the Session class.
321      */
322     @Override
323     public String toString()
324     {
325         StringBuilder build = new StringBuilder();
326         build.append("Session Id: ");
327         build.append(sessionId);
328         build.append("\tAvailable seats: ");
329         build.append(maxStudentForSession - studentsInSession);
330         build.append("/");
331         build.append(maxStudentForSession);
332         build.append("\n\tInstructor:     ");
333
334         if(teacher == null)
335         {
336             build.append("N/A");
337         }
338         else
339         {
340             build.append(teacher.getName());
341         }
342
343         build.append("\n");
344
345         build.append(printStudentList());
346
347         return build.toString();
348     }
349
350     /**
```

```
351      * Override compareTo method.
352      *
353      * @param otherSession other session
354      * @return comparision result (i.e. -1, 0, -1) between Sessions.
355      */
356     @Override
357     public int compareTo(@NotNull Session otherSession) {
358         return this.sessionId.compareTo(otherSession.sessionId);
359     }
360
361     /**
362      * Override equals method.
363      *
364      * @param obj object
365      * @return the result of thisSession equals otherSession as a boolean
366      */
367     @Override
368     public boolean equals(Object obj) {
369         if (obj == null || !(obj instanceof Session)) {
370             return false;
371         }
372
373         return compareTo((Session) obj) == 0;
374     }
375 }
376
```