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

```java
/**
 * @author Caroline Ta
 * @since 05.19.2020
 */

package com.company.data.course;

import java.util.ArrayList;
import java.util.Comparator;

/**
 * The type Course directory.
 */
public class CourseDirectory {

    /**
     * The courseList list.
     */
    private ArrayList<Course> courseList;

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

    /**
     * Instantiates a new Course directory.
     */
    public CourseDirectory()
    {
        courseList = new ArrayList<>();
    }

    // ------------------------------------------------------------------------
    // ACCESSOR - GETTER METHODS
    // ------------------------------------------------------------------------

    /**
     * Gets course list.
     *
     * @return the course list
     */
    public ArrayList<Course> getCourseList() {
        return courseList;
    }

    // ------------------------------------------------------------------------
    // FUNCTIONALITY METHODS
    // ------------------------------------------------------------------------

    /**
     * Sort.
     *
     * @param comparator the comparator
     */
    public void sortCourse(Comparator<Course> comparator)
    {
        courseList.sort(comparator);
    }

    /**
     * Find course course.
     *
     * @param courseId the course id
     * @return the course
     */
    public Course findCourse(String courseId) {
        for (Course c : courseList) {
            if (c.getCourseId().equals(courseId))
            {
                return c;
```

```
71              }
72          }
73
74          return null;
75      }
76
77      /**
78       * Add course.
79       *
80       * @param course the course
81       * @throws Exception the exception
82       */
83      public void addCourse(Course course) throws Exception {
84          Course cInDirectory = findCourse(course.getCourseId());
85
86          if (cInDirectory != null) {
87              throw new Exception("Course "
88                      + cInDirectory.getCourseId() + " is already in the list.\n");
89          }
90
91          courseList.add(course);
92      }
93
94      /**
95       * Remove course.
96       *
97       * @param course the course
98       */
99      public void removeCourse(Course course) {
100         courseList.remove(course);
101     }
102
103     /**
104      * Generate session.
105      *
106      * @param courseId the course id
107      * @param path     the path
108      */
109     public void generateSession(String courseId, String path)
110     {
111         findCourse(courseId).generateSessionList(path);
112     }
113
114     // ------------------------------------------------------------------------
115     // OUTPUT METHODS
116     // ------------------------------------------------------------------------
117
118     /**
119      * Print course w session string.
120      *
121      * @return the string
122      */
123     public String printCourseWSession()
124     {
125         StringBuilder build = new StringBuilder();
126
127         build.append("\n************************************************************");
128         build.append("\n               COURSE SCHEDULE WITH SESSIONS");
129         for (Course p : courseList) {
130             build.append("\n-------------------------------------------------------------");
131             build.append("\n");
132             build.append(p);
133             build.append("\n\n");
134 //            for(int i = 0; i < p.sessionList.size(); i++)
135 //            {
136 //                build.append("\t#");
137 //                build.append(i+1);
138 //                build.append(" ");
139 //                build.append(p.sessionList.get(i).toString());
140 //            }
```

```
141         }
142         build.append("\n*************************************************************\n\n");
143 
144         return build.toString();
145     }
146 
147     /**
148      * Gets scheduled course sessions output data.
149      *
150      * @return the scheduled course sessions output data
151      */
152     public String getScheduledCourseSessionsOutputData()
153     {
154         StringBuilder build = new StringBuilder();
155         build.append("\n*************************************************************");
156         build.append("\n              SCHEDULED COURSE SESSIONS OUTPUT");
157 
158         for(int i = 0; i < courseList.size(); i++)
159         {
160             if(!courseList.get(i).getCancelledStatus())
161             {
162                 build.append("\n-------------------------------------------------------------\n");
163                 build.append(courseList.get(i).toString());
164                 build.append("\n");
165             }
166         }
167 
168         build.append("\n\n*************************************************************\n\n");
169         return build.toString();
170     }
171 
172     /**
173      * Gets unscheduled course sessions output data.
174      *
175      * @return the unscheduled course sessions output data
176      */
177     public String getUnscheduledCourseSessionsOutputData()
178     {
179         StringBuilder build = new StringBuilder();
180         build.append("\n*************************************************************");
181         build.append("\n            UNSCHEDULED COURSE SESSIONS OUTPUT");
182 
183         for(int i = 0; i < courseList.size(); i++)
184         {
185             if(courseList.get(i).getCancelledStatus())
186             {
187                 build.append("\n-------------------------------------------------------------\n");
188                 build.append(courseList.get(i).toString());
189                 build.append("\n");
190                 build.append(courseList.get(i).studentList.toString());
191                 build.append("\n");
192             }
193         }
194 
195         build.append("\n\n*************************************************************\n\n");
196         return build.toString();
197     }
198 
199     // -------------------------------------------------------------------------
200     // OVERRIDDEN METHODS
201     // -------------------------------------------------------------------------
202 
203     /**
204      * Override toString method.
205      *
206      *  @return a string description of the CourseDirectory class.
207      */
208     @Override
209     public String toString() {
210         StringBuilder build = new StringBuilder();
```

```
211         build.append("\n*************************************************************");
212         build.append("\n                      COURSE DIRECTORY");
213         build.append("\n-------------------------------------------------------------");
214         for (Course p : courseList) {
215             build.append("\n");
216             build.append(p);
217         }
218         build.append("\n\n*************************************************************\n\n");
219
220         return build.toString();
221     }
222 }
223
```