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

package com.company.data.person;

import org.jetbrains.annotations.NotNull;

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

/**
 * The type Person directory.
 */
public class PersonDirectory {

    /**
     * The list of Person.
     */
    private ArrayList<Person> personList;
    /**
     * The list of Faculty.
     */
    private ArrayList<Faculty> facultyList;
    /**
     * The list of Student.
     */
    private ArrayList<Student> studentList;

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

    /**
     * Instantiates a new Person directory.
     */
    public PersonDirectory() {
        personList = new ArrayList<>();
        facultyList = new ArrayList<>();
        studentList = new ArrayList<>();
    }

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

    /**
     * Add person clone.
     *
     * @param person       the person
     * @param newPersonTel the new person tel
     * @throws Exception the exception
     */
    public void addPersonClone(@NotNull Person person, String newPersonTel) throws Exception
    {
        Person newPerson = (Person)person.clone();
        newPerson.setTel(newPersonTel);
        addPerson(newPerson);
    }

    /**
     * Sort.
     *
     * @param comparator the comparator
     */
    public void sortPersonName(Comparator<Person> comparator)
    {
        personList.sort(comparator);
        facultyList.sort(comparator);
```

```
71          studentList.sort(comparator);
72      }
73
74      /**
75       * Find person person.
76       *
77       * @param personTel the person tel
78       * @return the person
79       */
80      public Person findPerson(String personTel) {
81          for (Person p : personList) {
82              if (p.getTel().equals(personTel))
83              {
84                  return p;
85              }
86          }
87
88          return null;
89      }
90
91      /**
92       * Add person.
93       *
94       * @param person the person
95       * @throws Exception the exception
96       */
97      public void addPerson(@NotNull Person person) throws Exception {
98          Person pInDirectory = findPerson(person.getTel());
99
100         if (pInDirectory != null) {
101             throw new Exception("Person with phone number "
102                     + pInDirectory.getTel() + " is already in the list.\n");
103         }
104
105         personList.add(person);
106
107         if (person instanceof Faculty) {
108             facultyList.add((Faculty) person);
109         } else if (person instanceof Student) {
110             studentList.add((Student) person);
111         }
112
113     }
114
115     /**
116      * Remove person.
117      *
118      * @param person the person
119      */
120     public void removePerson(Person person) {
121
122         personList.remove(person);
123
124         if (person instanceof Faculty) {
125             facultyList.remove((Faculty) person);
126         } else if (person instanceof Student) {
127             studentList.remove((Student) person);
128         }
129     }
130
131     /**
132      * Gets faculty as string.
133      *
134      * @return the faculty as string
135      */
136     public String getFacultyAsString() {
137         StringBuilder build = new StringBuilder();
138
139         build.append("\n*************************************************************");
140         build.append("\n                      FACULTY DIRECTORY");
```

```
141         build.append("\n-------------------------------------------------------------\n");
142         for (Faculty f : facultyList) {
143             build.append(f);
144             build.append("\n");
145         }
146         build.append("\n*************************************************************\n\n");
147 
148         return build.toString();
149     }
150 
151     /**
152      * Gets student as string.
153      *
154      * @return the student as string
155      */
156     public String getStudentAsString() {
157         StringBuilder build = new StringBuilder();
158         build.append("\n*************************************************************");
159         build.append("\n                      STUDENT DIRECTORY");
160         build.append("\n-------------------------------------------------------------\n");
161         for (Person s : studentList) {
162             build.append(s);
163             build.append("\n");
164         }
165         build.append("\n*************************************************************\n\n");
166 
167         return build.toString();
168     }
169 
170 
171     /**
172      * Gets faculty output data.
173      *
174      * @return the faculty output data
175      */
176     public String getFacultyOutputData()
177     {
178         StringBuilder build = new StringBuilder();
179         build.append("\n*************************************************************");
180         build.append("\n                      FACULTY OUTPUT");
181 
182         for(int i = 0; i < facultyList.size(); i++)
183         {
184             build.append("\n-------------------------------------------------------------\n");
185             build.append(facultyList.get(i).toString());
186             build.append("\n");
187             build.append(facultyList.get(i).getFacultyCourseList().toString());
188             build.append("\n");
189         }
190 
191         build.append("\n\n*************************************************************\n\n");
192         return build.toString();
193     }
194 
195     /**
196      * Gets scheduled students output data.
197      *
198      * @return the scheduled students output data
199      */
200     public String getScheduledStudentsOutputData()
201     {
202         StringBuilder build = new StringBuilder();
203         build.append("\n*************************************************************");
204         build.append("\n                  SCHEDULED STUDENTS OUTPUT");
205 
206         for(int i = 0; i < studentList.size(); i++)
207         {
208             if(studentList.get(i).getScheduledStatus())
209             {
210                 build.append("\n-------------------------------------------------------------\n");
```

```
211                 build.append(studentList.get(i).toString());
212                 build.append("\n");
213                 build.append(studentList.get(i).getCourseScheduledList().toString());
214                 build.append("\n");
215             }
216         }
217 
218         build.append("\n\n************************************************************\n\n");
219         return build.toString();
220     }
221 
222     /**
223      * Gets unscheduled students output data.
224      *
225      * @return the unscheduled students output data
226      */
227     public String getUnscheduledStudentsOutputData()
228     {
229         StringBuilder build = new StringBuilder();
230         build.append("\n************************************************************");
231         build.append("\n                UNSCHEDULED STUDENTS OUTPUT");
232 
233         for(int i = 0; i < studentList.size(); i++)
234         {
235             if(!studentList.get(i).getScheduledStatus())
236             {
237                 build.append("\n------------------------------------------------------------\n");
238                 build.append(studentList.get(i).toString());
239                 build.append("\n");
240             }
241         }
242 
243         build.append("\n\n************************************************************\n\n");
244         return build.toString();
245     }
246 
247     // ------------------------------------------------------------------------
248     // OVERRIDDEN METHODS
249     // ------------------------------------------------------------------------
250 
251     @Override
252     public String toString() {
253 
254         StringBuilder build = new StringBuilder();
255 
256         build.append("\n************************************************************");
257         build.append("\n                      PERSON DIRECTORY");
258         build.append("\n------------------------------------------------------------\n");
259         for (Person p : personList) {
260             build.append("\n");
261             build.append(p);
262         }
263         build.append("\n\n************************************************************\n\n");
264 
265         return build.toString();
266     }
267 }
268 
```