Homework 5: Build a Student Directory App

Due date: September 28, 2015

For this homework, you will build a student directory application. The idea is pretty much the same as CMU's Online Directory application (https://directory.andrew.cmu.edu), but your application is simpler. Your application will have only four data fields, just first name, last name, andrew id, and phone number. Also, the search operations will be more explicit, and you will allow the creation and deletion of entries.

Specification

- Student.java - This is a class to model a single student including first name, last name, andrew id, and phone number. These fields are all required.
- Directory.java - A class to model a collection of students using Map(s).

Here are some additional information for you.

Student class - You must provide the following methods in your Student class:

```
public Student(String andrewId)
public String getAndrewId()
public String getFirstName()
public String getLastName()
public String getPhoneNumber()
public void setFirstName(String s)
public void setLastName(String s)
public void setPhoneNumber(String s)
public String toString()
```

The toString method should return the string representation of a student object and should be formatted as follows:

```
Terry Lee (Andrew ID: eunsunl, Phone Number: 412-268-1476)
```

You may add additional methods to the Student class, if needed by your directory.

Directory class - This class uses three Maps as follows to maintain the collection of students. One Map will have the student's Andrew ID as the key and will map from Andrew ID to Student object. The other two maps will take either the first name or the last name as the key and will map the name to a List of Student objects. (It's a list because more than one student can have the same first name or the same last name or even the same first and last name, but every student will have a different Andrew ID.)

As Student objects are added to and deleted from your directory, all three Maps must be updated accordingly so that the Students in the directory can be found when searching by Andrew ID or first name or last name.

Your Directory class must provide the following methods:

```
public Directory()
public void addStudent(Student s)
public void deleteStudent(String andrewId)
public Student searchByAndrewId(String andrewId)
public List<Student> searchByFirstName(String firstName)
public List<Student> searchByLastName(String lastName)
public int size()
```

The methods have the following actions

- `addStudent` - given the student object, add the new student into the three maps if the given student's Andrew ID is not present in the directory. If the student's Andrew ID is present, throw `IllegalArgumentException`. Make sure to have proper field validations in place.
- `deleteStudent` - given the andrew id string value, this method should remove the corresponding student object from the three maps if present. If no andrew id matches, throw `IllegalArgumentException`.
- `searchByFirstName` - given the first name string value, this method should return a list containing all students that match the given first name. If no students in the directory have the given first name, return a zero length list of students.
- `searchByLastName` - given the last name string value, this method should return a list containing all students that match the given last name. If no students in the directory have the given last name, return a zero length list of students.
- `searchByAndrewId` - given the andrew id string value, this method should return the student in the directory. If no student in the directory has the given Andrew ID, return `null`.
- `size` - returns the number of students in the directory.

As you implement the methods above, you must keep in mind when returning mutable objects that you should make sure to not allow the code outside the class to modify private data inside the class.

If any of the above methods are given illegal arguments, you must throw an `IllegalArgumentException`.

Additional Requirements

Make sure to include javadoc comments and additional comments if necessary with all of your code. Also, keep your code style to conform to the SUN (Oracle) standard convention (http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html).

Testing

We encourage you to test your program thoroughly. In fact, try to write some test program before you write the required methods (any any associated helper methods).

Turning-in Your Work

Place all your .java files into a ZIP file called homework5.zip. The following JAR command must be used to do this:

```
jar cvf homework5.zip *.java
```

Submit your homework5.zip file which includes the required two files (Student.java and Directory.java) using AutoLab (http://autolab.cs.cmu.edu).

Grading

AutoLab will grade your assignment as follows:

- Java files exist and compile: 5 points
- Output is correct: 80 points
  - Addition : 15 points
  - Search : 30 points
  - Deletion : 15 points
  - Correct Data : 20 points
- Follows coding conventions: 10 points
  - We'll deduct one point for each coding convention issue detected.
- Author JavaDoc comment at beginning of file: 5 points

AutoLab will show you the results of its grading within approximately one or two minutes of your submission. You may submit multiple times so as to correct any problems with your assignment. Autolab uses the last submission as your grade.