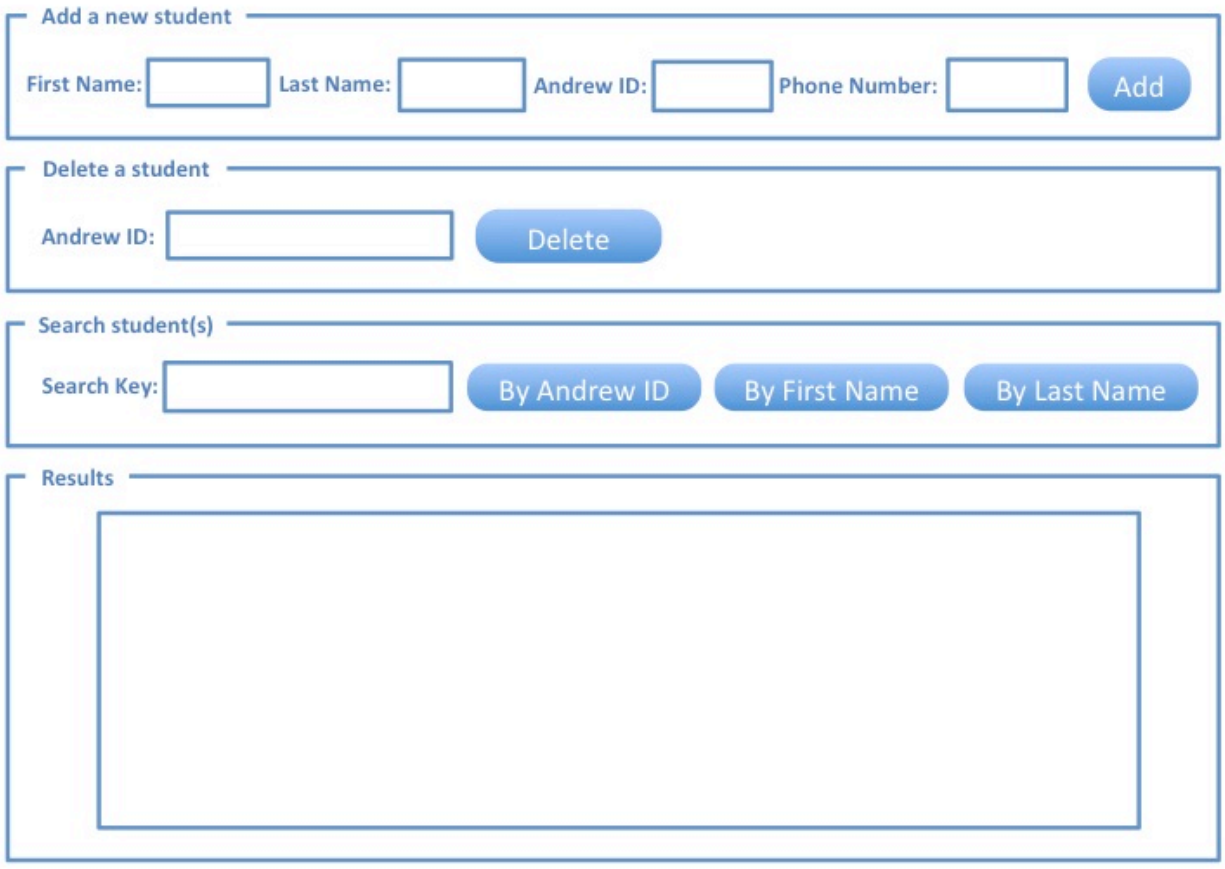
Homework 6: Swing for Student Directory

Due date: October 3, 2016 (Mon)

For this homework, you will build a swing interface for the data access object you created in previous assignment. Your swing interface will allow a user to create new student entries, search for entries, and delete entries.

You will have flexibility in how you layout the swing GUI, as long as all the components are present and it looks nice. Here's a sketch of the interface we would like you to build.



Specification

- **Swing GUI** -- Build a Swing GUI, based on the sketch above, in `DirectoryDriver.java` file that contains the main method.

- **Load Data** -- When running your program (`DirectoryDriver` class), it must accept an optional command line parameter with the name of a CSV file that will contain initial data you must load before displaying the Swing GUI. (If no parameter is provided, you should just start your Swing GUI with an empty data.) The CSV file will have the following header and format. The first line is the header and the following lines are student entries (one entry per line) that should be loaded. This does NOT have anything to do with any database management system such as MySQL. It simply means your program should be able to read and load some data from a CSV file into memory.

  "First Name","Last Name","Andrew ID","Phone Number"
  "Terry","Lee","eunsunl","412-268-1078"
  "Jeffrey","Eppinger","je0k","412-268-7688"

- **Stress Test** -- Your Student.java and Directory.java must pass the same stress tests run in the previous homework. Keep in mind that this application might evolve in the future. For example, we might add a new functionality to update existing students' records (first name, last name, and/or phone number). This means Student class is mutable. It should be possible to change the fields through setters multiple times.

- **Search by Andrew ID** -- Allow the user to provide a search key and click on the search "By Andrew ID" button. Display the results as follows:
  - **No Matches** -- If there are no entries in your data for this Andrew ID, display a message in the Results area that indicates this. (The message must include the search key.)
  - **Match** -- If there is an entry in your data for this Andrew ID, display (in the Results area) all the fields for the entry formatted as follows.

    ```
    Terry Lee (Andrew ID: eunsunl, Phone Number: 412-268-1078)
    ```

- **Search by Last Name or First Name** -- Allow the user to provide a search key and click either the search "By Last Name" or "By First Name" button. Display the results as follows:
  - **No Matches** -- If there are no entries in your data for this search key, display a message in the Results area that indicates this (be sure to indicate in the message whether the search was by last name or by first name and show the value of the search key in the message).
  - **One Match** -- If there is one entry in your data for this search key, display (in the Results area) all the fields for the entry formatted as follows.

    ```
    Terry Lee (Andrew ID: eunsunl, Phone Number: 412-268-1078)
    ```

  - **Multiple Matches** -- If there is more than one entry in your data for this search key, display a list of matches in the results area. Use the same format as above and display each entry in a new line.

- **New Entry** -- Allow the user to enter the data for a new entry in the data. Allow the user to enter

all the fields for a new student entry. Andrew ID, first name, and last name fields are required. The other fields are optional. When the user clicks on the "Add" button, check the data. If there are no errors, store the entry in your data and display in the Results area a message stating a new entry was added.

- **Delete by Andrew ID** -- Allow the user to provide an Andrew ID and click on the "Delete" button. Display the results as follows:
  - **No Matches** -- If there are no entries in your data for this Andrew ID, display a message in the Results area that indicates this. (The message must include the Andrew ID.)
  - **Match** -- If there is an entry in your data for this Andrew ID, display in the Results area a message stating that the entry was deleted and all the fields for the deleted entry formatted as follows.

```
Terry Lee (Andrew ID: eunsunl, Phone Number: 412-268-1078)
```

- **Error Handling** -- If there is error when adding, deleting, or searching for entries, display the error message in the Results area. The message must indicate the problem and specifically state which field has the error. If there are multiple errors, you must display only one error message -- you may decide which. Here is a list of the errors you must detect:
  - Andrew ID missing
  - Last Name missing
  - First Name missing
  - Data already contains an entry for this Andrew ID

- **UI Fine Points** -- To make this homework easier for the TAs to grade, provide the following features in your main window:
  - When the window opens, the focus must be in the search key text field.
  - When typing in the search key text field, pressing "enter" key executes the "search by Andrew ID" action (same as clicking on the "By Andrew ID" button.)
  - After a button is clicked, clear the text fields if there is no error. If there is an error, do not clear the text fields, so that the user can see what the input was.

Turning in your work

Place all your .java files into a ZIP file called homework6.zip. The following JAR command must be used to do this:

```
jar cvf homework6.zip *.java
```

Submit your homework6.zip file which includes the required three files (`Student.java`, `Directory.java`, and `DirectoryDriver.java`) using AutoLab (https://autolab.andrew.cmu.edu).

Grading

AutoLab will grade your assignment as follows (total of 25 points):

- Java files exist and compile: 5 points
- Data Stress Test: 5 points
- Follows coding conventions: 10 points
    - We'll deduct one point for each coding convention issue detected.
- Author JavaDoc comment at beginning of each 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 the AutoLab-graded portion of your assignment. Autolab uses the last submission as your grade.

For this assignment, TAs will run and grade the other requirements as follows (total of 75 points):

- Swing GUI: 30 points
- Load Data: 5 points
- Search by Andrew ID: 5 points
- Search by Last Name or First Name: 10 points
- New Entry: 10 points
- Delete by Andrew ID: 5 points
- Error Handling: 5 points
- UI Fine Points: 5 points

Note that the TAs will grade their portion of the homework after the due date. You will **NOT** get to resubmit this homework after the due date. The TAs will only grade their portion once.