The State Universtiy of New York, Korea
Computer Science

CSE 114 Handout 13: PS 9 May 15, 2019

This problem set is due **Thursday, May 23 at 11:59pm, KST**. Note that the due date that you see on Blackboard is not accurate since it shows the time in EST. You should go by the due date in this handout.

- Follow the specification exactly. It makes grading much easier.
- Each method that you write must be properly commented and attractively formatted.
- You may create and run your programs using Eclipse if you want, or continue using emacs and command line interface.
- Your programs should be formatted in a way thats readable. In other words, indent appropriately, use informative names for variables, etc. If you are uncertain about what a readable style is, see the examples from class and textbook as a starting point for a reasonable coding style.
- **Be sure to include** a comment at the top of each file submitted that gives **your name** and **email address**.

What to Submit

Submit the following files as a **single zip or tar file** on **Blackboard**. Multiple submissions are allowed before the due date. Please do **not** submit `.class` files, Eclipse-specific project files, or any that I did not ask for.

```
Application.java
ProcessApplications.java
(an input file that for sure works with your solution)
and any other Java file(s) I will need to run your program.
```

What Java Features to Use

For this assignment, you are not allowed to use more advanced features than what we have covered in Lecture 1 through Lecture 18.

Partial vs. Complete Solutions

Please read what I said in PS 1 on this issue.

Naming Conventions In Java And Programming Style In General

Refer to the ones given in PS 1.

Problem (100 points)

Suppose you are writing a program that will help our Admissions Office process the applications into the University. In this problem your program will open an input file containing applicant information and read it into an array of application objects. Once you have them read into an array, you can do some interesting operations with the data that you just read in.

See a sample input file named `a0.txt` on the course web. I added multiple input files of various sizes for your consumption if you need them. An entry (an application record) in the input file has six attributes and is formatted as follows:

- The very first line of the input file contains a number indicating the number of records contained in the entire input file.
- The first line of an application record is the name of an applicant consisting of first name, middle name (or middle initial with a period), and last name in that order in a single line, all separated by a blank space.
- Next line is the street address in one line, city name in the next line, state in another line, and finally zip code in a separate line.
- The next line is a phone number consisting of the area code, prefix, and last four digits all separated by a blank space.
- The next line is the id number of the application.
- The next line is the intended major, e.g., Computer Science, Economics, etc. A multi-word name should be allowed.
- The next line is the applicant's high school GPA, e.g., 3.96.
- The next line is an indication of whether the applicant is applying for a scholarship or not.
- If there is another application record in the file, it will follow next in the same format.

Your task is to read a file containing application records and do what is asked to be done in `ProcessApplications.java` that is provided in the [given] folder.

Your solution should consist of the following files:

- `Application.java`: This file contains a class definition that represents an application.
- `ProcessApplications.java`: This file contains a `main` that reads in an input file and processes the application data. (A skeleton is provided in the [given] folder.)
- It is likely that you will want to add another file, perhaps `State.java` to represent a state and use it in the `main`. More on this in the FAQ.

Hand in your Java files and at least one input file that works with your program. I assume that your input file would be of the same format as mine, but I would still like to have one of your input files that for sure works with your program.

When you test your program as you develop it, use a small input file, for example start with one line and make sure it works; then two lines and make sure it works, and so on. Incremental development, right? It is much easier to deal with a smaller input file. After you think you are done debugging your program, use a large input file to test it again before you hand it in.

Note: I included a program that you can use to generate an input file of any arbitrary size. The file is called `RandomApplications.java`. Take a look if you are interested.

Hints: Read the FAQ!