**UIC CS FACULTY SCHEDULER**

BRYAN SPAHR

**Introduction**

Why did I choose undergraduate research? Well, I always relish the opportunity to set my own schedule (sometimes to my own detriment). I also liked the idea of a semester-long side project with Pat Troy. We’ve talked about various aspects of CS and coding in the past, especially feedback on the new CS curriculum and the curriculum as a whole.

When I came to Troy and told him we were doing research, we once again talked about the CS curriculum; this time, about possible gaps in subjects covered that could be filled in the future. One of these gaps is the topic of Web Development. Most CS students end up teaching themselves some web development at some point in their UIC career, but there isn’t any academia related to the topic. With the possibility in mind of a future tech elective, I was tasked with researching what materials such a class would need and, most importantly, whether or not the known UIC-hosted web service, <people.uic.edu>, would suffice for an entire class of students.

It didn’t take me long to realize that the best resources for such a course were all over the web, in fact, there was no need to create new materials for the course given the wide range of stuff available. I also quickly concluded after a little research and questioning that [people.uic.edu](file:///E:\workspace\uic_cs_faculty_scheduler\paper\people.uic.edu) would easily support a full class of students.

**What is this?**

Well, I had to do something to fulfil my research requirements for credit.

As it turns out, Professor Troy is in charge of choosing which CS professors teach each class every semester for the following semester. As such, he suggested this project as a potentially viable solution to this manual task by choosing professors using a GUI application.

This application does just that. Also included is a GUI editor for updating which classes each CS instructor teaches. Despite a relatively simple GUI, this project features an attention to detail and professional object oriented design throughout code. My main goal for this project was to make an application that could add value to what Professor Troy does, as opposed to just making something that will never see the light of day.

**How it works**

If you haven’t already, download the zip file release here: <http://bit.ly/1C18fQN>

Inside you should have an executable JAR file (note: requires Java Runtime to run) and a folder called data that contains several plaintext files. The JAR file and data folder need to be in the same directory in order to use the pre-loaded data provided in the zip file. Alternatively, you can put the JAR file in any directory you want and the application will create new data files for you to input your own data from scratch.

After launching the JAR file, you should see a small window with two buttons, Faculty Editor and Generate Instructors.

**Faculty Editor**

Clicking on the left button in the main window opens the editor. You’ll see a dropdown to select an instructor’s name. Every time you switch to a different instructor, the application will pre-load the data for that instructor into his/her course slots. Each instructor has up to 8 slots for commonly taught courses, which often aren’t all used, so the blank slots can just be left as “n/a”. When finished editing/adding courses for a particular instructor, make sure to click the save button or your changes will be lost. You should see console output as the application saves the data to both the application and the external data files.

**Generating Instructors**

Clicking on the right button in the main window calculates all instructors (from the available ones listed for each course) for the following semester. The results are displayed in the Results Window, and include all CS courses in the data, even those without any listed instructors. Also in this window is a button for saving out the results as a plaintext file for long-term purposes.

**Downsides/Future Improvements**

* I was originally going to have the main window include check boxes for each course so you could check only the courses being offered the following semester, making for a much cleaner result window. This functionality has been shelved for the time being.
* I was playing with the concept of putting a “reject” button next to each instructor pick in the results window. This would mean the user could reject multiple picks out of the original calculation and recalculate leaving the other picks untouched. This is less important, however.
* The application doesn’t take into account instructor teaching loads. This means 1 instructor could be picked for multiple classes, while another instructor could end up with none. This implies a greater need for the above point.
* Probably other things. Let me know if you think of other flaws.

**What I learned/gained experience**

* A few things about Java Swing and threading
* More experience building complex object-oriented applications in Java
* Javadoc commenting, the thorough kind
* How to use github.io
* How to use a compareTo to sort on multiple fields

Thanks for reading!

A complete API for this application follows this page.