Alex Meredith

MUN Committee Scheduler

5/4/15

Functional Overview

My program will assign students to MUN committees based on their preference of committee, region or country preference, gender, history class, and grade in the class. It will run just like our earlier assignments in Java, and it will import data from text files for input. It will then print scheduled committees to an output file. It is eventually meant to be used by the Lakeside history department, but in its current implementation will likely be too complex in its IDE and input formatting requirements for their use. Eventually, I would like to reimplement this program with a GUI so that it will be easy for students and teachers to use.

Design Overview

My program will have a client that implements the students and committees and it will print committees to an output file. I plan to have expected inputs from text files containing different classes of students and different committees of countries. In the initial implementation, there will not be any user input, only file I/O. If I have time, I would like to implement settings that allow the user to change default factors that go into committee assignments and maybe user input of student information. Eventually, I would like to reimplement this program in a GUI so that students and teachers can easily submit their data and I can eliminate the need for file I/O.

Design Details

|  |
| --- |
| Client (MUN.java)   * Has ArrayLists of Student objects grouped by history class (which contain Committee and Region objects) and has Committee objects, which contain Region and Student objects * Assigns students to different committees * Has method getStudents() which scans input files of student data and converts the data into Student objects in the ArrayLists * Has method getCommittees() which scans input files of committees and converts them into Committee Objects * Has method assignCommittees() which assigns students to committees * Has method chooseCommittees() which chooses which committees to implement on MUN day (there are more committee options than actual committees) * Has method printFinalResults() which prints the final committees to an output file * Potentially will have method alterSettings() which allows the user to change the criteria for committee assignment * Potentially will have method enterStudentData() which will allow the user to enter student data |
| Student.java   * Has an ArrayList of Committee objects as an instance variable (this contains the student’s top three committee choices) * Has an ArrayList of Region objects as an instance variable (this contains the student’s preferred two regions) * Has a double grade as an instance variable * Has a character gender (“M” or “F” only, to keep things simple) as an instance variable * Has a String name as an instance variable * Implements the Comparable interface * Has a compareTo() method which ranks students alphabetically * Has accessor and mutator methods for all of its instance variables |
| Committee.java   * Has a Map as an instance variable with Region objects as the keys and Students as the values, but all Students are initially null * Contains a String issueDescription describing the issue that committee will address as an instance variable * Contains a String committeeName as an instance variable * Contains an integer preference initially set to 0 as an instance variable (this will be set to demonstrate whether the committee is the student’s first, second, or third choice) * Contains mutator and accessor methods for instance variables * Contains printCommittee() method, which prints out the students assigned to various countries |
| Region.java   * Has an ArrayList of Strings as an instance variable that contains country names * Has a String preferredCountry initially an empty String that contains the name of a preferred country if there is one as an instance variable * Has a String regionName as an instance variable * Contains mutator and accessor methods for instance variables * Contains equals() method which checks if it exactly matches another region * Contains equalsWithoutPreference() method which checks if it matches another region except for the preferredCountry variable |

Testing Design

I will initially test the program with all of the possible committees but only a few students and check to make sure the file I/O works, no exceptions are thrown and no runtime errors occur. I will then check the output to make sure it is reasonable. If everything seems to work, I will gradually add students for more tests. I will check what happens if all of the students have the same preferences or a few students have totally incompatible preferences (basically worst-case scenarios). I will not test with other users until I have a working GUI (probably not within this project timeline). If I can, I would love to get country and committee preference data for the past few years of MUN from the history department to help test this program. I will also test cases with incomplete data (some students did not put all three choices of committee, for example)

Grading Rubric

Program assigns all students to committees, and gives every student one of their top committee and region choices if at all possible; program is externally correct (19 points)

Design is time efficient and avoids redundancies (8 points)

Code is readable and well documented; overall the program and flow of control are easy to understand, and it is clear why the design was chosen (8 points)

Program has clearly been tested, and passes basic testing (5 points)

Proposed Implementation Schedule

Friday, May 8 - Design document is complete; coding has begun

Monday, May 11 – Student object, Region object, and Committee object are implemented; work on MUN.java has begun, especially on File I/O

Friday, May 15 – File I/O in MUN.java works, and all methods in it except chooseCommittees() and assignCommittees() are functional or nearly there

Monday, May 18 – All methods in MUN.java have been implemented successfully; basic testing has begun

Friday, May 22 – MUN.java passes all tests; code is being refactored, checked, and commented well

Monday, May 25 – Program is ready for release; if there is time, implement alterSettings() and research GUI possibilities

Potential Obstacles and Open Questions

Deciding on an algorithm to optimize committee assignment is likely to be tricky, and I think that assignCommittees() and chooseCommittees() are likely to be the most difficult to implement. I recognize that alterSettings() may also be difficult, and I will only implement it if I have time. I can research scheduling algorithms and discuss possible algorithms with Dr. Butler, Dr. Bricker, and possibly even Mr. Rona or Mr. Ballard (because they have had input on the Lakeside scheduling program).

Resources

I do not yet know what resources, if any, I will utilize to complete this project. I may discuss scheduling algorithms with Mr. Rona or Mr. Ballard, and I will almost certainly do research on them using the internet or a programming textbook.