Assignment 1

Due date

11.59 PM EST on September 22nd.

Git url

https://classroom.github.com/a/bwtm5nm3

Submit your code as per the provided instructions.

Updates

- Tue Sep 11 21:49:06 EDT 2018: ANT tarball has been updated, along with the directory structure required for this assignment.
- Thu Sep 13 10:39:12 EDT 2018: git url posted
- Sun Sep 16 22:44:12 EDT 2018: input format and example have been updated

Assignment Goal

A simple Java program.

Team Work

- No team work is allowed. Work individually. You cannot discuss the assignment with ANYONE other than the instructor and TA.

Programming Language

You are required to program using Java.

Compilation Method

- Compilation: Your code should compile on bingsuns or remote.cs.binghamton.edu with the following command:

  ```
  TBA
  ```

- Running the code: Your code should run on bingsuns or remote.cs.binghamton.edu with the following command: TBA

Policy on sharing of code

- EVERY line of code that you submit in this assignment should be written by you. Do NOT show your code to any other student. Do not copy any code from any online source. Code for File I/O or String operations, if found online, should be clearly cited, and you cannot use more than 5 lines of such online code.
- Code downloaded in its entirety from an online repository of code (GitHub, BitBucket, etc.) and submitted as student's own work, even if citied, is considered plagiarism.

- Code snippets, for File I/O, if used from an online source should be cited by mentioning it in the README.txt and also in the documentation of every source file in which that code appears.
- Post to the listserv if you have any questions about the requirements. Do NOT post your code to the listserv asking for help with debugging.

Project Description

Assignment Goal: Develop a program, using Java, to assign courses to students based on their preferences.

- There are 6 courses being offered. Each course has the following information.
  - Capacity - The total number of students that be registered for this course.
  - Class Timings - For simplicity, timings are just represented as integers, starting from 0.
  - Course Name (A,B,C,D,E or F)
- The following information is associated with each student.
  - Student ID (a 3 digit integer)
  - Levels:
    - FIRST_YEAR
    - SECOND_YEAR
    - THIRD_YEAR

  Hint: Enums can be used to specify the levels of a student.

The following rules MUST be followed when registering students to courses.

1. A student cannot be registered to multiple courses that have the same class timing.
2. Two courses (E and F) need to be assigned to a first year student only if there is no 2nd and 3rd year student still waiting for it. Among 2nd and 3rd year students, the priority has to be given to the 3rd year student.
3. If a course has been filled up, then any further registration requests for that course are rejected.

INPUT FORMAT
Your program should take two input files - *student_coursePrefs.txt* and *courseInfo.txt*.

student_coursePrefs.txt will have the following format,
<student_name> <PREF_1>,<PREF_2>,<PREF_3>,<PREF_4>,<PREF_5>,
<PREF_6>;STUDENT_LEVEL:<[first_yr,second_yr,third_yr]>

courseInfo.txt will have the following format,
<course_name>-CAPACITY:<capacity>;CLASS_TIMING:< class_time>
INPUT EXAMPLES
student_coursePrefs.txt

```
001 A, B, C, D, E, F; STUDENT_LEVEL: FIRST_YEAR
222 F, E, D, C, B, A; STUDENT_LEVEL: SECOND_YEAR
333 F, D, E, A, B, C; STUDENT_LEVEL: THIRD_YEAR
```

courseInfo.txt

```
A-CAPACITY:30; CLASS_TIMING: 7
B-CAPACITY:20; CLASS_TIMING: 8
C-CAPACITY:40; CLASS_TIMING: 7
D-CAPACITY:60; CLASS_TIMING: 9
E-CAPACITY:40; CLASS_TIMING: 2
F-CAPACITY:50; CLASS_TIMING: 8
```

OUTPUT
Your program should write the registration results to an output file called *registration_results.txt*.

registration_results.txt will have the following format,
<student_name>:<course_1>,<course_2>,<course_3>

NOTES ON GRADING

- Class participation points will be given to the first 10 students who post interesting sample input files.

Sample Input Files sent by students in this course

Please check piazza.

Clarifications based on student questions

- Is there a minimum or maximum number of courses a student can take? A student can take a max of 3 courses. It is possible that some students will be assigned less courses as it depends on the input file.
- Will all students give 6 preference options or less than 6 are allowed? All students are required to give 6 preferences.
- Can the number of students be much larger than the total capacity of 6 courses? Such that some students might not get assigned to any course at all? Yes.

Compiling and Running Java code

- Your README.md file should have the following information:
  - instructions on how to compile the code
  - instructions on how to run the code
  - justification for the choice of data structures (in terms of time and/or space complexity).
- You should have the following directory structure (replace firstName_lastName with your name).

```
./firstName_lastName_assign1
./firstName_lastName_assign1/coursesRegistration
./firstName_lastName_assign1/coursesRegistration/src
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/util
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/util/FileDisplayInterface.java
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/util/FileProcessor.java
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/util/Results.java
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/util/StdoutDisplayInterface.java
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/driver
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/driver/Driver.java
./firstName_lastName_assign1/coursesRegistration/src/coursesRegistration/scheduler
./firstName_lastName_assign1/coursesRegistration/src/build.xml
./firstName_lastName_assign1/README.md
```

```
[Other Java files you may need]
```

Code Organization

- Your directory structure should be EXACTLY as given in the code template
  - Download the ANT based tarball here. Use the command on linux/unix: *tar -xvf firstName_lastName_assign1.tar.gz*.

Submission

- Read this file for general guidelines on how to prepare a README for your submission.
- Make sure all class files, object files (.o files), executables, and backup files are deleted before creating a zip or tarball. To create a tarball, you need to "tar" and then "gzip" your top level directory. Create a tarball of the directory firstName_lastName_assign1. We should be able to compile and execute your code using the commands listed above.
- Instructions to create a tarball
  - Make sure you are one level above the directory firstName_LastName_assign1.
  - tar -cvf firstName_LastName_assign1.tar firstName_LastName_assign1/
  - gzip firstName_LastName_assign1.tar
- Upload your assignment to Blackboard, assignment-1.

General Requirements

- Start early and avoid panic during the last couple of days.
- Separate out code appropriately into methods, one for each purpose.
- You should document your code. The comments should not exceed 72 coloums in width. Use javadoc style comments if you are coding in Java. Include javadoc style documentation. It is acceptable for this assignment to just have the return type described for each method's documentation.
- Do not use "import XYZ.*" in your code. Instead, import each required type individually.
- All objects, in Java, that may be needed for debugging purposes should have the "toString()" method defined. By default, just place a toString() in every class.
- Every class that has data members, should have corresponding accessors and mutators (unless the data member(s) is/are for use just within the method.).

Design Requirements

Late Submissions

- The policy for late submissions is that you will lose 10% of the grade for each day that your submission is delayed. **There is NO difference in penalty for assignments that are submitted 1 second late or 23 hours late** .

Grading Guidelines

Grading guidelines have been posted here.

mgovinda at binghamton dot edu
Back to Programming Design Patterns