Assignment 1

Due date

- 11.59 PM EST, September 19th

Git url

- https://classroom.github.com/a/UTR3Tr_a

Submit your code as per the provided instructions.

Updates

- Thu 05 Sep 2019 07:18:32 PM EDT: ant commands for compiling and running have been posted.

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.

Compiling and Running Commands

- Compilation: Your code should compile on remote.cs.binghamton.edu with the following command: ant -buildfile coursesRegistration/src/build.xml all
-
- Running the code: Your code should run on remote.cs.binghamton.edu with the following command: ant -buildfile coursesRegistration/src/build.xml run -Darg0="student_coursePrefs.txt" -Darg1="courseInfo.txt" -Darg2="registration_results.txt"

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 piazza if you have any questions about the requirements. Do NOT post your code to the piazza 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 9 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 positive integers, starting from 0.
    - •Course Name (A,B,C,D,E,F,G,H or I)
- •The following information is associated with each student.
    - •Student ID (a 3 digit integer in the range [100,999])
    - •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. A student can take a maximum of 3 courses. It is possible that some students will be assigned <3 courses as it depends on the input file.
3. All students are required to provide 9 preferences.
4. Total number of students can exceed the total capacity across all 9 courses.

5.A course is assigned to a first year student only if there is no 2nd or 3rd year student still waiting for it. Among 2nd and 3rd year students, priority has to be given to the 3rd year student.
6.If a course has been filled up, then any further registration requests for that course are rejected.
7.Each student has a satisfaction rating for each requested course. The order in which a student requests courses defines the satisfaction rating of that student for those courses given by *(9 - course.index)*.
For example, let **A,B,C,D,E,F,G,H,I** be the requested courses of a student. Then,
satisfaction rating of student for course **A** = 9 - A.index = 9 - 0 = 9
satisfaction rating of student for course **B** = 9 - B.index = 9 - 1 = 8
satisfaction rating of student for course **C** = 9 - C.index = 9 - 2 = 7
satisfaction rating of student for course **D** = 9 - D.index = 9 - 3 = 6
And so on...

Note: The average satisfaction rating for a student determines the quality of the course registration algorithm.

INPUT FORMAT

Your program should take two input files - *student_coursePrefs.txt* and *courseInfo.txt*. Note that the **input files will be well formatted**.

student_coursePrefs.txt will have the following format,

<student_id>
<PREF_1>,<PREF_2>,<PREF_3>,<PREF_4>,<PREF_5>,<PREF_6>,<PREF_7>,<PREF_8>,<PREF_9>::<student_level>

courseInfo.txt will have the following format,

<course_name> CAPACITY:<capacity>;CLASS_TIMING:<class_time>

INPUT EXAMPLES

student_coursePrefs.txt

```
111 D,C,A,B,G,I,H,F,E::FIRST_YEAR
222 F,E,D,C,B,A,H,I,G::SECOND_YEAR
333 D,A,F,E,I,C,H,B,G::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
G CAPACITY:45;CLASS_TIMING:4
H CAPACITY:25;CLASS_TIMING:3
I CAPACITY:10;CLASS_TIMING:6
```

OUTPUT

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

registration_results.txt will have the following format,

```
<student1_id>:<course_1>,<course_2>,<course_3>::SatisfactionR
ating=<value>
<student2_id>:<course_1>,<course_2>,<course_3>::SatisfactionR
ating=<value>
<student3_id>:<course_1>,<course_2>,<course_3>::SatisfactionR
ating=<value>
...
AverageSatisfactionRating=<value>
```

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

  - **Q**: *Can we assume that student_coursePrefs.txt and courseInfo.txt will be passed in as command line arguments in a specific order?*
  - **Ans**: Yes. Please make sure to include the assumed order of the input files in the README.
  - **Q**: *Can we assume that there will never be a class with multiple sections?*
  - **Ans**: Yes. Each course will have only one entry in courseInfo.txt file.
  - **Q**: *Can we assume capacity > 0? (for each class)*
  - **Ans**: Yes.
  - **Q**: *For output formatting, if a student does not get 3 classes should there be an extra comma?*
  - **Ans**: This is left to you. As nothing related to this was mentioned in the assignment, you are free to either remove the third comma or not. Please mention in the README, the formatting that you have chosen for students that were assigned < 3 courses.

Compiling and Running Java code

- Your submission must include a readme in markdown format with the name **README.md**.
- 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).
  - citations for external material utilized.
- 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 -zxvf firstName_lastName_assign1.tar.gz*.

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 columns 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** .