San Francisco State University
CSC 413: Software Development
Dawson Zhou (zhoud@sfsu.edu)

Assignment 1: Classroom Management

Overview

In this assignment, you'll implement a simple Java class designed to manage course enrollment using a variety of data structures. You'll use Git and GitHub to submit your code as you make progress.

Submission

Assignment 1 is due on **February 12, 2021** at **11:59 pm PT**.

Late submissions will be accepted with a penalty:

- February 13, 2021 12:00 am PT to February 13, 2021 11:59 pm PT will receive 75%
- February 14, 2021 12:00 am PT to February 14, 2021 11:59 pm PT will receive 50%

The timestamp of your last GitHub commit will be used to determine whether your assignment is on time or will have a late penalty applied.

Grading

100 pts total:

- 10 pts: Correct initialization of instance variables.
- 10 pts: Correct implementation of `registerStudent`.
- 15 pts: Correct implementation of `enrollStudent`.
- 15 pts: Correct implementation of `dropStudent`.
- 10 pts: Correct implementation of `getEnrolledStudents`.
- 10 pts: Correct implementation of `getWaitlistedStudents`.
- 10 pts: Correct modified version of `getWaitlistedStudents` that reverses order.
- 10 pts: No changes made to method signatures or instance variable types. All instance variables used, and no new ones added.
- 5 pts: Separate commits made to GitHub repository (see below for details).
- 5 pts: README.md completely filled out.

Assignment Setup

Click on the GitHub Classroom link provided on iLearn to automatically create your GitHub repository for the assignment, which will include the starter code for the project.

See the "Git, GitHub, and IntelliJ Setup Guide" document for instructions on setting up the rest of the project on your local machine.

Detailed Requirements

Summary of requirements:

1. Implement the correct Classroom behavior detailed below in Base Version.
2. Base Version implementation is committed and submitted to the GitHub repository.
3. Implement the modification detailed below in Modified Version.
4. Modified Version implementation is committed and submitted to the GitHub repository, separate from the commit in requirement #2.
5. README.md file in the project is filled out.

Base Version

You are provided with a partially implemented Java file, **Classroom.java**, and a completed Java file, **Student.java**. Your task is to implement all of the methods in **Classroom** class. You will need to provide implementation code in every section of the file where you see the following comment:

```
// TODO: Implement.
```

The **Classroom** manages a roster of registered students. Students can be enrolled, waitlisted, or neither.

- When a student is registered, the Classroom object should track that student's full information, so that it can be retrieved later with just the student ID. However, registering a student does not enroll them in the class or add them to the waitlist.
- When a student is enrolled, they are placed in a collection tracking the IDs of students enrolled in the class. The student can only be enrolled if there is spare capacity. If not, there should be an attempt to place them in the waitlist.
- When a student is placed in the waitlist, they go to the back of the waitlist, behind any students that are already there. The student can only be waitlisted if there is spare capacity. If not, the student's enrollment request is ignored.

- When a student is dropped from the course, their ID is removed from the collection of enrolled students. If that frees up space in the class, the first student in line in the waitlist should automatically be enrolled. If a student is in the waitlist and is dropped from the course, that request is ignored.
- The Classroom can provide lists of student names when requested for those that are enrolled and those that are waitlisted.
    - The set of enrolled student IDs and the waitlist queue should not be modified in any way when these names are requested. You can use a "for each loop" to process each element in a collection:

```
for (int id: waitlistIds) {
    // Use id here.
}
```

You **must** use the collections provided for you in the list of instance variables (`registeredStudents`, `enrolledIds`, and `waitlistIds`).

You **cannot** change any of the method signatures (i.e. method names, parameter lists, and return types).

If you've implemented the base version correctly, the names of enrolled students printed should include everything from "Alice" through "Queen Latifa," except for "Eli" and "Klay." However, the order they're printed will **not** be in alphabetical order. The waitlist should have "Rupert," "Serena," "Tobias," and "Eli."

(I have just learned that I spelled "Queen Latifa" incorrectly. My apologies!)

Modified Version

Make a modification to your implementation of the `getWaitlistedStudents` method so that it returns the list of student names in **reverse order**. To get full credit, you must use the **Stack** data structure.

Commit that change separate from the base version implementation and submit it to GitHub.

If you've implemented the modified version correctly, you should see the same printout as above, except with the waitlist printed in reverse order.