7 Circular Data; State Change

Portfolio Problems

1. Design the classes that represent the transit system that consists of several train lines, with stations on each line.

   Every line has a name (usually a color name) and a list of stations it serves. Every station has a name and a list of lines that go through the station.

   (a) Make an example of a transit system that looks like the MBTA *Green Line, Red Line Blue Line, and Orange Line* and include two terminal stops and at least two transit stations for each line.

   (b) Design the method `isTransfer` that determines whether a station is a transfer station between one or more lines.

   (c) Design the method `sameLine` in the class `Station` that determines whether this station is on the same line as the given station.

   (d) Design the method `oneChange` that determines whether we can travel from this station to the given station making exactly one change at a transfer station.

Pair Programming Assignment

7.1 Problem

Complete problems 7.1 and 7.2 from Lab 7 and turn in the complete solution.

7.2 Problem

Complete problem 7.3 from Lab 7 and turn in the complete solution.

7.3 Problem

In this problem we will emulate a college registrar system.

A. Start a Java project and define the classes and interfaces that implement the class diagram shown on the next page.

   Notice, that we will need to break the circularity of this class diagram.

```
+-----------------------------------------------+
|             +-------------------+             |
|             |                   |             |
|             v                   |             v
|         +------+                |         +------+
|         | ALoS |<-------------+ |         | ALoC |<------------+
|         +------+              | |         +------+             |
|         +------+              | |         +------+             |
|            / \                | |            / \               |
|            ---                | |            ---               |
|             |                 | |             |                |
|     ----------------          | |     ----------------          |
|     |              |          | |     |              |          |
| +-------+   +---------------+ | | +-------+   +--------------+ |
| | MTLoS |   | ConsLoS       | | | | MTLoC |   | ConsLoC      | |
| +-------+   +---------------+ | | +-------+   +--------------+ |
| +-------+ +-| Student first | | | +-------+ +-| Course first | |
|           | | ALoS    rest  |-+ |           | | ALoC rest    |-+
|           | +---------------+   |           | +--------------+
|           |                     |           |
|           v                     |           v
|    +--------------+             |     +-------------+
|    | Student      |             |     | Course      |
|    +--------------+             |     +-------------+
|    | String name  |             |     | String name |
|    | int id       |             |     | int credits |
+----| ALoC courses |             +-----| ALoS roster |
     +--------------+                   +-------------+
```

B. Define examples of at least three students and eight courses, with every student enrolled in at least two courses.

C. Design the method `addCourse` in the class `Student` that will enroll the student in the given course. Throw an exception if the student is already enrolled in the course, or if the total number of credits would be over 20, after the enrollment is completed.

D. Design the method `dropCourse` that will drop the student from the given course. Throw an exception if the student is not enrolled in the given course, or if it would result in student being registered for fewer than two courses, after the drop is processed.

E. A student meets an attractive fellow student and wonders whether they may meet during one of the classes our student is enrolled in. Design the method `canMeet` that will tell our student whether there is a course our student is taking that the desired person is also enrolled in.

F. The registrar keeps a list of all students and a list of all courses. These are set up at the beginning of each quarter. Design the class `Registrar` and include your original examples in the registrar's database.

G. Design the method `register` that consumes the name of a student, the student's id, and the course name and enrolls the student in the course. The same restrictions as before apply. Additionally, the method should throw an exception if any of the information given is not correct (no student with the given name and id, no course with the given name).

H. Design the method `withdraw` that consumes the name of a student, the student's id, and the course name and drops the student from the course. The same restrictions as before apply. Additionally, the method should throw an exception if any of the information given is not correct (no student with the given name and id, no course with the given name).

Note: Most of these methods require several helper methods. At least half of the grade for this homework will assess the design of these methods - whether you truly follow the rule *one task — one method*.

Note: All of these methods, except the `canMeet` method are defined only to *effect* a change in the registrar system. Design your tests carefully. At least one half of the grade for this homework will assess the design and implementation of the tests for these methods.

Note: Did you notice that the above two criteria cover two halves of the grade for the homework? Well, there is some *wiggle room* there, but we really want you to understand how important these two issues are.