8 Breaking the Circle.

Introduction

In this assignment we focus on circularly referential data. We also leverage again the power of the Abstract Data Type we used in the last assignment (*IMapping*) and introduce a new ADT that allows us to traverse over a structured data *from outside*.

Start by defining the following classes and interfaces- but without examples yet. The schedule represents a list of courses students is taking during this semester. Roster contains a list of students enrolled in the course offered this semester. We assume student's name is unique and the course key value is unique as well.

```
+--------------------+   +-------------------+
| Student            |   | Course            |
+--------------------+   +-------------------+
| String name        |   | String key        |
| String major       |   | int no            |
| IMappping schedule |   | String instructor |
+--------------------+   | IMapping roster   |
                         +-------------------+

+------------------------------------+
| interface IMapping                 |
+------------------------------------+
| IMapping add(String key, Object obj) |
| Object find(String key)            |
| boolean contains(String key)       |
+------------------------------------+

+-----------------------------------------------------+
| interface Traversal                                 |
+-----------------------------------------------------+
| // produce the currently first object in the data set |
| Object current()                                    |
|                                                     |
| // advance to the next element of the data set      |
| Traversal advance()                                 |
|                                                     |
| // are there more items to produce/advance over?    |
| boolean hasMore()                                   |
+-----------------------------------------------------+
```

8.1 Problem: Representing Circularly Referential Data

Define the classes that represent students and courses.

A. Design first the classes that implement a list of objects (either students or courses) and implement the *IMapping* interface.

B. Extend the definition so it also implements the *Traversal* interface. Throw an exception if the client attempts to perform an illegal operation, such as advancing in an empty data set.

C. Use these classes to now define lists of students and lists of courses and make examples of data.

D. Define the *ISame* method for the classes that represent students and courses. Compare only the student's name and major when comparing two students, and only the course key, course number, and the instructor when comparing two courses.

E. Define the *toString* method for the student class that prints only the student's name and the major. It does not print the schedule.

F. Define the *toString* method for the course class that prints the course information: the key, the number, and the instructor. It does not print the roster.

∎

8.2 Problem: Designing Methods for Circularly Referential Data

Define the class *Registrar* that contains a list of all students at the university and a list of all courses taught this semester. Use the list of objects you defined earlier to handle both lists. In the *Registrar* class define the following methods (use helper methods in this class or in other classes as needed):

A. Design the method *showStudent* in the class *Student* that produces a String where the first line is the information about the student and each of the following lines contains the information about one course in the student's schedule.

B. Design the method *showCourse* in the class *Course* that that produces a String where the first line is the information about the course and each of the following lines contains the information about one student enrolled in the course.

C. Design the method in the *Registrar* class that produces a list of all courses on a schedule for a student with the given name.

D. Design the method in the *Registrar* class that determines whether two students are enrolled in the same course.

E. Design the method in the *Registrar* class that produces a list of all students enrolled in the given course.

∎

8.3 Problem: Mutating Circularly Referential Data

Of course, the registrar has only one list of students and one list of courses. Actions such as a new student enrolling in the university, student adding a course to her schedule, or dropping a course should only change the contents of the instance of the registrar object.

A. Design the *addStudent* method to the *Registrar* class that adds a student with the given name and major to the registrar database. Initially this student is not enrolled in any courses.

B. Design the method *add* that enrolls a given student in a specified course. Analyze the problem first and describe in a couple of lines of comments what is the information the registrar instance needs to perform this task, and what will be the effects.

C. Design the method that allows the registrar to change the instructor assigned to teach a course.

∎

8.4 Problem: Mutating a Binary Search Tree

Finish the problem from **Lab 8** that asked you to design a mutable version of a binary search tree. ∎

8.5 Writing Problem

Writing assignments are separate from the rest of the assignment for the week. You should work on this assignment alone, and submit your work individually on the due date for the rest of the homework. The answer should be about two paragraphs long – not to exceed half a page or 300 words.

Read the following article on outsourcing. Write a short essay that describes your thoughts on this issue.
http://www.acm.org/globalizationreport/ ∎