CS 445: Data Structures
Spring 2017

Assignment 1

Assigned: Thursday, January 19 **Due:** Thursday, February 02 11:59 PM

---

1 Motivation

In CS 445, we often discuss the importance of data structure design and implementation to the wide variety of computing applications. Despite decades of study, organizations must still regularly develop custom data structures to fulfill their applications' specific needs, and as such the field remains hugely relevant to both computer scientists and software engineers.

As an example of the magnitude of impact that data structures can have on a large system, read the following news article:

`http://www.pcworld.com/article/2042979/the-tao-of-facebook-data-management.html`

In this assignment, you will implement two data structures to satisfy their specifications, which are provided in the form of interfaces.

> **Note:** Your goal in this assignment is to write classes that faithfully implement the ADTs described in the interfaces. Sample client code is provided as *one example* of how the classes may be used, and may also provide hints for how to solve certain problems in your implementations (since one of your classes will act as a client of the other) However, this sample code may not test all corner cases, and thus your goal is *not* simply to make this sample code work. You are *strongly encouraged* to write your own test client as well, to further test that your classes work in all the corner cases described in the interfaces.

2 Provided code

First, look over the provided code. You can find this code on Pitt Box in a folder named `cs445-a1-abc123`, where `abc123` is your Pitt username.

The `SetInterface<E>` interface describes a set, a data structure similar to a bag except that it does not allow duplicates. It is a generic interface that declares abstract methods for adding an item, removing an item (specified or unspecified), checking if the set is empty, checking the number of items in the set, and fetching the items from the set into an array. You should *not* modify this interface.

The SetFullException class is included to allow hypothetical implementations of SetInterface that have a fixed capacity. Your implementation should *not* be fixed capacity, and thus should not throw this exception.

The ProfileInterface interface describes a social network user's profile. It declares abstract methods for setting and getting the user's name and "about me" blurb, following other profiles, returning an array of the profile's followed profiles, and recommending a new user to follow based on this profile's "followed by those I follow" set. You should *not* modify this interface.

The SocialClient class is a sample client of both Set and Profile. It is a social networking simulator that allows the user to carry out following, unfollowing, etc. on a simple social network. This class maintains a set of profiles (stored as SetInterface<ProfileInterface>. It also stores its data in a file (SocialClientData.bin) when quitting so that it can restore this data when it is run again. As noted above, this code is provided only as an example, and may not test all functionality of the required classes.

3 Tasks

3.1 Implement Set, 50 points

Develop the generic class, Set<E>, a **dynamic-capacity array-based** implementation of the Set ADT described in SetInterface<E>. Read the interface carefully (including comments) to ensure you implement it properly; it will be graded using a client that assumes *all* of the functionality described in the SetInterface, not just the behavior demonstrated in SocialClient!

You must include a constructor public Set(int capacity) that initializes the array to the specified initial capacity, and a constructor public Set() that uses a reasonable default initial capacity. Whenever the capacity is reached, the array should resize, using the techniques discussed in lecture (i.e., you should never throw SetFullException).

Method	Points
Set()	4
Set(int)	4
int getCurrentSize()	2
boolean isEmpty()	2
boolean add(E)	7
boolean remove(E)	7
E remove()	6
void clear()	4
boolean contains(E)	6
E[] toArray()	8

3.2 Implement Profile, 50 points

Develop the Profile class, an implementation of the ADT described in ProfileInterface. Read the interface carefully (including comments) to ensure you implement it properly. As with Set, it will be graded using a client that expects the functionality described in its interface. The Profile class should be a client of the Set data structure. Use *composition* with your Set<E> class to store the "following" set as a data member of type Set<Profile> or Set<ProfileInterface>.

You must include a constructor `public Profile()` that initializes the name and "about me" blurb to be empty strings, and a constructor `public Profile(String name, String about)` that initializes these data members with the specified values. In the latter, you must check for `null` values for both, and replace any `null` value with an empty string.

Method	Points
`Profile()`	2
`Profile(String, String)`	4
`void setName(String)`	4
`String getName()`	2
`void setAbout(String)`	4
`String getAbout()`	2
`boolean follow(ProfileInterface)`	7
`boolean unfollow(ProfileInterface)`	7
`ProfileInterface[] following(int)`	8
`ProfileInterface recommend()`	10

3.3 Testing

`SocialClient` is provided as an example client of the `Profile` and `Set` classes. It *does not* exhaustively test the functionality of these classes. You are responsible for ensuring your implementations work properly in all cases, even those not tested by `SocialClient`, and follow the ADTs described in the provided interfaces. Thus, it is highly recommended that you write additional test client code to test all of the corner cases described in the interfaces. For help getting started, re-read the section of the textbook starting at Chapter 2.16.

> **Note:** For functionality that cannot be tested (e.g., methods that crash, cannot be compiled), up to 1/2 points will be awarded by inspection. At this level, turning in code that crashes or does not compile is not acceptable and will not yield success.

4 Submission

Upload your java files in the provided Box directory. If you overwrite the provided interfaces, remember to restore them to their original versions.

All programs will be tested on the command line, so if you use an IDE to develop your program, you must export the java files from the IDE and ensure that they compile and run on the command line. Do not submit the IDE's project files. Your TA should be able to download your `cs445-a1-abc123` directory from Box, and compile and run your code as discussed in Lab 1. For instance, `javac cs445/a1/SocialClient.java` and `java cs445.a1.SocialClient` must compile and run `SocialClient`.

In addition to your code, you may wish to include a `README.txt` file that describes features of your program that are not working as expected to assist the TA in grading the portions that do work as expected.

Your project is due at 11:59 PM on Thursday, February 02. You should upload your progress frequently, even far in advance of this deadline: **No late assignments will be accepted.**