CS 445: Data Structures
Fall 2016

Assignment 1

Assigned: Monday, September 12 **Due:** Monday, September 26 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 and utilize simple data structures to build a simple social network prototype.

2 Provided code

First, carefully read the code provided at `https://cs.pitt.edu/~bill/445/a/a1code.zip`.

The `SetInterface<T>` 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 for implementations of `SetInterface` that have a maximum capacity.

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 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. The client allows the user to perform each of the following operations:

1. *List profiles* List the profiles that currently exist.
2. *Create profile* Create a new profile, and set its name and "about me" blurb.
3. *Show profile* Choose a profile from the list, and show its name, "about me" blurb, and a short preview of the profiles it is following (up to 4 profiles).
4. *Edit profile* Choose a profile from the list, and edit its "about me" blurb.
5. *Follow* Choose two profiles from the list, and make the first follow the second.
6. *Unfollow* Choose two profiles from the list, and make the first unfollow the second.
7. *Suggest a follow* Choose a profile from the list, and suggest a new profile to follow (using the `recommend` method from `Profile`).

3 Tasks

3.1 Implement Set, 50 points

Develop the generic class, `Set<T>`, that implements `SetInterface<T>` using an array. Read the interface carefully (including comments) to ensure you implement it properly; it will be tested using a client that assumes all of the functionality described in the `SetInterface`.

You **must** include a constructor `public Set(int capacity)` that initializes the array to a given capacity, **and** a constructor `public Set()` that initializes the array to a reasonable starting capacity. In either case, the array should resize when it is full, following the resize techniques we discussed in lecture.

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

Note: Your implementation of `Set` resizes when it is at capacity, so it should never throw `SetFullException` (this remains in `SetInterface` to accommodate fixed-capacity implementations).

3.2 Implement Profile, 50 points

Develop the `Profile` class that implements `ProfileInterface`. Read the interface carefully (including comments) to ensure you implement it properly. As with `Set`, it will be tested using a client that expects the functionality described in its interface.

The `Profile` class is a client of the `Set` data structure. You should use *composition* with your `Set<T>` class to store the "following" set. 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 member variables with values provided by the caller. 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, and follow the description of the ADTs provided 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 do not work, classes that cannot be compiled), up to 1/2 points will be awarded by manual inspection.

4 Submission

Create a zip file containing *only* java files (no class files!). You should include **unmodified** versions of the provided files `SetInterface.java`, `ProfileInterface.java`, `SetFullException.java`, and `SocialClient.java`. This way, the TA can unzip your submission, compile your code, and run `SocialClient` without any additional changes. 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.

Before uploading your final zip file, test that it works! Unzip your submission, compile your code, and run `SocialClient` from the command line to ensure the TA will be able to complete these steps with the file you are turning in.

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.

Submit your zip file according to the instructions at `https://cs.pitt.edu/~bill/445/#submission`

Your project is due at 11:59 PM on Monday, September 26. Be sure to test the submission procedure in advance of this deadline: **No late assignments will be accepted.**