CSC207.01 2014S, Class 37: Designing a Dictionary API

Overview

- Preliminaries.
  - Upcoming work.
  - Admin.
  - Extra credit.
  - Lecture on academic honesty.
  - Notes on exam 2.
  - Questions.
- Review: Designing ADTs and Data Structures.
- A Dictionary ADT.
- Simple implementations of dictionaries.

Preliminaries

Upcoming work

- No writeup today (particularly since there's no lab today)!
- No reading for tomorrow.
- HW 6 remains due Wednesday night.
- Exam 2 makeup due Thursday night.

Admin

- Today is probably not the best day to have prospectives. Oh well.
  We'll still make them introduce themselves.
- While I've finished grading your examinations, I need to revisit some issues. I hope to return them tomorrow. We will talk about some issues today.

Extra Credit

- "Pioneer Weekend is a three day event, sponsored by the Wilson Program and Grinnell AppDev, that is being organized to take place this month from April 18th - 20th. The objective of this event is to bring together student innovators from different backgrounds, to work together in teams of 3-6 people and complete a prototype of an idea that they come up with at the event." http://www.strikingly.com/pioneerweekend
- Math talk today at noon. CN on cool stuff that bridges math and CS.
- CS extra Thursday: Software to enhance wellness: The DavisJan team.
- CS table Friday: TBD.

- CS extra next Monday: Walker and Liberto on bluetooth.

Academic Honesty

- Most of you have had Walker. This stuff should have been drilled into you.
- I realize that many of my exam questions have similar code on the Interweb. It's clear that I ask interview-like questions (or vice versa). And it's okay with me that you use code you find, provided you cite it.
- Both the College and the CS department believe in upholding high standards of academic integrity.
- There are also practical reasons for you to develop good habits: In industry, if you borrow code and don't cite, no one knows that they have to license the code. If someone finds out, you can be putting your company at severe risk.
- When you appear to violate standards, there's a painful process (painful for almost everyone involved)

Notes on Exam 2

- See the document. I'm not repeating myself in the eboard.
- Four levels of protection
  - private - can't be accessed by other things
    - Evil HS teachers encourage its use, as do strange people on StackOverflow
  - protected
    - Subclasses
  - [nothing] "package"
    - Anything in the same package
    - Subclasses
  - public
    - Anything
- Makeup

Questions on the homework

How will the user interact with the calculator?

Similarly to `dc`. Read the inputs from stdin, print the outputs (from p and s) to stdout.

The bad version of queues has an iterator (which may be well designed). What do we have to do?

All collections have iterators, mostly so you can skim through the collection without mutating it.

Review: Designing ADTs and Data Structures

ADT

- What's the philosophy?
- What can we use it for?
- What methods do we need?

Data Structure

A Dictionary ADT

Simple Dictionary, indexed by strings

- What's the philosophy?
  - Something like a vector, but indexed by strings instead of by integers
- What can we use it for?
  - A traditional dictionary (or Wikipedia): Look things up by name
  - General storage of records; humans tend to think in terms of names rather than numbers
  - The fields of an object are like the keywords of a dictionary
- What other design decisions do we have?
  - Does it grow automatically? (Ye, we assumes.)
- What methods do we need?

The interface

```
public interface SimpleDictionary<T>
{
  /**
   * Change the entry for a particular index.
   */
  void set(String index, T value);

  /**
   * Remove a value based on its index.
   */
  void remove(String index);

  /**
   * Remove everything.
   */
  void clear();

  /**
   * Remove all the entries with a given value.
   */
  void removeAll(T val);

  /**
   * Get all of the indices in the dictionary.
   */
  Iterator<String> indices();

  /**
   * Get all of the values in the dictionary.
```

```
   */
  Iterator<T> iterator();

} // interface SimpleDictionary<T>
```

Generalizing: Replace "String" with a type variable

```
public interface Dictionary<I,T>
{
} // interface Dictionary<I,T>
```

Other notes about dictionaries

Lots of related names:

- Map
- Associative Array
- Hash
- Table

Simple implementations of dictionaries

Forthcoming.