Contact Information

Alexa Sharp
Office: King 223C
course website: www.cs.oberlin.edu/~asharp/cs151
Office Hours: M 1:30-3:30pm, T 12:00-1:30pm, F 3:30-4:30pm.

Comments:
- Computers are not to be used in the classroom. Please put yours away.
- If you’re not yet registered, OK. Come to class and lab and you’ll get in.
- I have a class at 11 and thus will not be able to meet other classes, ever.
- Please read the syllabus carefully, as well as the extended honour code.

Course Overview

1. Define a new abstract data type (data and method specification)
2. Discuss various implementations of the abstract data type via various data structures and algorithms
3. Compare and contrast the implementations according to their algorithmic complexities
4. Show how to use the different data structures in different situations

In this course, we will repeatedly

1. Think more about design of programs and reusability
2. Learn how to store and manipulate data efficiently in different situations, and
3. Become a better problem solver.

In doing so, hopefully you will:

Three major components:

1. Data Structures - the way in which we organize data
2. Algorithms - a clearly specified set of instructions to solve a problem
3. Abstraction - hiding irrelevant information in order to reduce complexity

ex. An object data type: separate specification and implementation, as

ex. Objected data types: separate specification and implementation, as

ex. Abstraction - hiding irrelevant information in order to reduce complexity

ex. Algorithms need to be properly covered and efficient

ex. How exactly does MapQuest find the best route from A to B?

Course Overview: Principles of Computer Science, Part Deux

CS 151
Homework

Please skim Chapters 1 and 2 for a review of Java and reference types. I expect you to be experts at this stuff already.

Please skim Chapter 3 for a review of objects and classes. This includes sections on mutators and accessors, constructors, the this keyword, instanceof, static fields and methods. I expect you to be experts at this stuff already.

Please read Sections 4.1 - 4.2 for a review of inheritance (extends, super, final, polymorphism, etc.) and Sections 4.3 - 4.7 for Friday’s lecture. These readings are posted on the course webpage, in case you forget.

ALSO, lab 0 is out and is due on Sunday at 8pm. Find it on the website, or www.cs.oberlin.edu/~asharp/cs151/labs/lab00/lab0.html

Tuesday, September 4, 2012

Collections API

Collections API: set of Java classes that implement various data structures. Data structure: specific representation of data and its allowable operations.

Collections interface -- represents a group of objects / elements. Must support:

- boolean isEmpty()
- int size()
- boolean add(AnyType x)
- boolean contains(Object x)
- boolean remove(Object x)
- void clear()
- Object[] toArray()
- <OtherType> OtherType[] toArray(OtherType[] arr)
- java.util.Iterator<AnyType> iterator()

Data structures usually allow arbitrary insertion, but differ in their allowable access operations. Some Collections We’ll See

- LinkedList -- a list implemented with a doubly-linked list, head and tail pointers. Allows insertions at arbitrary locations.
- ArrayList -- a list that is implemented with an array. Allows insertions at arbitrary locations.
- Stack -- only allows access to most recently added object. Push(), Pop().
- Queue -- only allows access to least recently added object. Add() and remove() operations. Some Collections We’ll See

- PriorityQueue -- allows access to the minimum element. Min-heap implementation.
- java.util.List -- interface that supports sequences of data. Allows access to elements at any location. Get(int index), set(int index, AnyType newval).
- ArrayList -- a list that is implemented with an array that supports sequences of data.
- Stack -- only allows access to most recently added object. Push(), Pop().
- Queue -- only allows access to least recently added object. Add() and remove() operations.
Some Collections We'll See

Java.util.Set -- interface that supports sets of elements, without duplicates. Some specific implementations of this interface are HashSet and TreeSet.

Java.util.Map -- interface that supports collection of entries that are (key, value). Some specific implementations of this interface are HashMap and TreeMap.

Some Collections We'll See