CS 151
Principles of Computer Science, Part Deux
Alexa Sharp
Office: King 223C
Office Hours: M 1:30-3:30pm, T 12:00-1:30pm, F 3:30-4:30pm.

course website: www.cs.oberlin.edu/~asharp/cs151

please read the syllabus carefully, as well as the extended honour code available on the course website.

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 after class, ever.
Course Overview

Three major components:

1. **Data Structures** - the way in which we organize data
   
   ex. tree structures file system in Unix and Windows
   
   - how data is organized has an important impact on system performance

2. **Algorithms** - a clearly specified set of instructions to solve a problem
   
   ex. how exactly does mapquest find the best route from A to B?
   
   - algorithms need to be provably correct and efficient

3. **Abstraction** - hiding irrelevant information in order to reduce complexity
   
   ex. abstract data types: separate specification and implementation, as with ArrayLists and arrays
In this course, we will repeatedly

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 complexity.

4. Show how to use the different data structures in different situations.

In doing so, hopefully you will:

1. become a better problem solver,

2. learn how to store and manipulate data efficiently in different situations, and

3. think more about design of programs and reusability.
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, this, instanceof, static fields and methods. I expect you to be experts at this as well.

Please read Sections 6.1 - 6.3 for a more in-depth coverage of what is to follow. Also skim the starts of each section from 6.5 - 6.10.

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 website, 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](http://www.cs.oberlin.edu/~asharp/cs151/labs/lab00/lab0.html)
Data structure: specific representation of data and its allowable operations.

Data structures usually allow arbitrary insertion, but differ in their allowable access operations to the data.

Once have data structure, can reuse it over and over. We love that.

Collections API: set of Java classes that implement various data structures.

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()
Some Collections We’ll See

java.util.List – interface that supports sequences of data
   AnyType get(int index)
   AnyType set(int index, AnyType newval)

ArrayList – a list that is implemented with an array

LinkedList – a list implemented with doubly-linked list, head and tail pointers
   void addLast(AnyType obj)
   void addFirst(AnyType obj)
   AnyType getFirst()
   AnyType element()
   AnyType getLast()
   AnyType removeFirst()
   AnyType remove()
   AnyType removeLast()
Some Collections We’ll See

Stack – only allows access to most recently added object
    push()
    pop()
    peek() + others

Queue – only allows access to least recently added object
    add() (aka enqueue())
    remove() (aka dequeue())
    getFront()

PriorityQueue – allows access to the minimum item
Some Collections We’ll See

java.util.Set – interface that supports sets of elements, without duplicates

Some specific implementations of this interface are HashSet, TreeSet, and LinkedHashSet.

java.util.Map – interface that supports collection of entries that are (key,value) pairs

Some specific implementations of this interface are HashMap and TreeMap.