Tuesday, January 30

Wrapping up **Phases of software development**

- Specification - what software needs to do
- Design - how to do it with pieces
- Implementation - write the code
- Analysis - how well does code do it
- Test - find the bugs before the user does
- Maintenance and evolution - enhance the software

**Design - how to do it with pieces**

- often think of data representation at this point
- each function/method should do one thing well - cohesion

**Implementation - write the code**

- consider style
- reviewed examples from assignment due last night

**Analysis - how well does code do it**

- skim example in chapter 1
- consider steps/comparisons program uses
  - visiting all n elements is O(n)
- ex. Find smallest value in array of n elements
  - O(n) normally
  - O(1) if array is sorted

**Test - find the bugs before the user does**

- a _failure_ is when a program is observed doing something wrong
- _faults_ in the code cause failures
- we make _errors_ when we allow faults into the code

- Test cases should have input and expected output
- Execute program and compare output to expected output
- JUnit
- Faults lurk in corners and edges (boundaries and interfaces)
- be aware of kinds of bugs you make/create and double-check those

**Maintenance and evolution - enhance the software**

- Change happens. Plan for it.
Abstract data types

**data type**: a set of values and the operations on those values

**abstract data type (ADT)**: an encapsulated set of values and a set of operations that act on those values so that the implementation and the representation are hidden

one abstract type can have two or more data representations and

we don't need to know what the representation is to use it

**Information hiding**

loose coupling between parts

Java classes

text will discuss a throttle class
our examples - timer and CoinPurse (money)

Javadoc comments can generate fancy documentation