Testing and debugging

"Tests can only establish the *presence* of bugs in a program; they cannot establish their *absence*" --Dijkstra

Unit tests test each component of a program in isolation.

A *test harness* is a program that runs tests by feeding parameters to methods.

Tests should be *repeatable*.

What kinds of test cases should we make?

Kinds of tests

- Positive cases
  - You know the answer, and are verifying that you get the expected result
- Boundary cases
  - Cases on the edges of acceptable input
- Negative cases
  - Cases where method should fail without crashing

Wait a minute! If we **already know the answer**, why do we need the method we're testing?

Verifying results

- Use an *oracle*
  - slower/undesirable means of producing same answer
    - "calculate results by hand" is a lo-fi oracle
- Verify that properties of the output hold
  - e.g. sqrt(x) * sqrt(x) == x

We want maximal *code coverage* for our tests -- we should exercise every path through each method.

What happens when we find a bug that isn't covered by our tests?

When we find a bug, make a test that exposes that bug!

(oh, and we **never** get rid of test cases)

We can then *regression test*, to make sure future changes don’t bring back old bugs.

How do we find what's causing bugs?

One solution: using a Logger.

`java.util.logging.Logger`

The logger is a class that acts sort of like PrintStream, but you can turn it on and off.

Programmers may insert trace messages into their code to verify that it is working as expected.

Sometimes, you have the source code for the class you're testing. Sometimes, you don't. **How can we deal with the latter case?**

Exercise

SimpleDate
- month, day, year, hour, minute
+ boolean isValid() + String timeToString() + String dateToString() + String toString()

develop a tester class for `SimpleDate`.

The *debugger* assists in code tracing.

Debugger capabilities

- Breakpoints
  - unconditional vs. conditional
- Printing locals
- Printing memory
- Stepping through code

It beats tracing by hand!

For more info

- CS 302 debugging lab
  - going to lab: always a good plan!
- Eclipse debugger tutorial:
  - http://tinyurl.com/zl77o