

Test-Driven Data Structures

A way to learn about unit-testing

Prof. Joel Adams
Department of Computer Science

CALVIN
College

Introduction

Some things I learned about *testing* in my (1973) high school *Economics* class:

- “Mr. N.” wasn’t much of a teacher
- Prior knowledge of a test’s contents increases the pass rate for that test
 - Even pot-heads can get a perfect score if they know the test’s contents beforehand!

Test-Driven Development (TDD)

TDD is a software development technique.

To build a method named *m()*:

1. Build a “unit test” method *testM()* that:
 - a) Invokes *m()*, and
 - b) Tests the effect of *m()* by asserting what should be true after *m()* terminates
2. Run *testM()*; observe it *fail*
3. Build *m()*
4. Run *testM()*; observe it *pass* (hopefully)

Benefits of TDD

TDD offers many benefits, including:

1. Can detect errors sooner, producing higher quality code
2. Tests can be expanded incrementally (leave special cases, errors for last)
3. Automatic regression testing
4. Claim: Faster total development time
5. Building test methods first forces us to imagine how clients will use *m()*

Limitations of TDD

TDD has some limitations, including:

- 1. Code is only as good as its tests**
 - Poor tests → poor code
 - Writing good tests is an acquired skill
- 2. Does not help discover new algorithms (but it can help test them)**
- 3. Limited usefulness in some situations (e.g., GUIs, interactive I/O, ...)**

How to teach/learn/practice TDD?

The elementary data structures (*lists, stacks, queues, etc.*) have operations whose behaviors are well-defined.

- For professionals: The behaviors are *familiar*, so a professional can focus on learning how to write tests.
- For students: Test methods can help students understand what the behaviors that underlie the operations.

Demo

**Let's use TDD to build an array-based
Stack class...**

**To make it more challenging/interesting,
let's build it in C++...**

Conclusions

In TDD, testing drives the design.

- **Raises the profile/priority of testing.**
- **By writing the test first, you know what a method must do to pass it.**

The elementary data structures provide a good domain for learning TDD:

- **Data structures have well-defined APIs; defining post-operation state is easy**

Resources

- *Test-Driven Development by Example*, Kent Beck, Addison-Wesley, 2003.
- testdriven.com (online community)
- eclipse.org (eclipse plugin framework)
- junit.org (plugin for Java unit testing)
- web-cat.cs.vt.edu/eclipse (customized CxxTest plugin for C++ unit testing)

Thank you!