NEW! STRONGER THAN DIRT

ENTER.
tech with a heartbeat

{clean code}

CAUTION: EVEN BAD CODE CAN FUNCTION
40 FL OZ (1.25 QT) 1.18L

INTRODUCTION

- Test driven development is relatively new, it became popular in the mid 90s.

- The agile and TDD movements have encouraged many programmers to write automated unit tests, but many have missed some of the more subtle and important points on doing it good.

THE THREE LAWS OF TDD

- TDD asks us to write test first, before writing production code
  - **First law**: You may not write production code until you have written failing unit test.
  - **Second law**: You may not write more of a unit test than is sufficient to fail, and not compiling is failing.
  - **Third law**: You may not write more production code than is sufficient to pass the current failing tests.

Working according to the laws, you will write dozens of tests every day and those tests will cover virtually all your production code.

KEEPING TEST CLEAN

- Tests must change as the production code evolves.
- The dirtier the test the harder they are to change.
- The more tangled the test code, the more likely it is that you will spend more time cramming new tests into the suite than it takes to write the new production code.
- Test code is just as important as production code. It requires thought, design, and care.

TESTS ENABLE THE -ILITIES

- Having unit tests makes your production code flexible, maintainable and reusable.
- If you have tests you don’t have the fear of making changes to the code.
- Having an automated suite of unit tests that cover the production test is the key to keep your design and architecture as clean as possible. These enable all the -ilities because they enable change.

CLEAN TEST

- Three things makes a clean test:
  - ***Readability***, ***readability*** and ***readability***.

- It is more important in unit test than in production code and what makes it readable are clarity, simplicity and density of expression.

A DUAL STANDARD

- Test code must be simple, succinct and expressive but it need not to be as efficient as production code. Prioritize readability than efficiency in your test code.

ONE ASSERT PER TEST

- Tests designed in this way come to a single conclusion that is quick and easy to understand. It is not a rule though it is a guide line. But it is important that the number of asserts in a test ought to be minimized.

SINGLE CONCEPT PER TEST

- Another rule is that we want to test a single concept in each test function. We don’t want long test functions that go testing one miscellaneous thing after the another.

- So minimize the number of asserts per concept and test just on concept per test function.

F.I.R.S.T

- Clean test follow the following 5 other rules.

  - **Fast**: Tests should be fast, they should run quickly. If they are too slow, you probably will not run them frequently.
  - **Independent**: The tests should not depend on each other. One test should not set up conditions to the next. When they depend on each other, the first one to fail cause a cascade of downstream failures.
  - **Repeatable**: Test should be repeatable in any environment. You should be able to run the tests in the production environment.
  - **Self-validating**: The test should have a boolean output. Either pass or fail.
  - **Timely**: Test should be written in a timely fashion. Unit test should be written just before the production code that makes them pass. If you write tests after, you may find the production code hard to test. You may not design the production code to be testable.