Test-Driven Development

Emerson Murphy-Hill
CRC Cards?
What’s Required

• Most any programming language
• An xUnit testing framework
  – SUnit, smalltalk
  – JUnit, Java
  – NUnit, .NET
  – etc…
Unit Testing

• A testing method that can be used test-first or test-last or test-while

• A unit test is a class

• In a program, a unit tests make sure everything works by covering almost every class

• In practice, it doesn’t replace functional/system, acceptance, or user/GUI tests
Unit Tests: Details

• A unit test is a subclass of TestCase
• In each unit test, you have a bunch of methods, each of which tests a specific piece of functionality
• Each method that is tested must start with “test”, such as testAddCustomer, testAsString, etc.
• No arguments to testX methods
Coverage

class String
- append:
- substring:to:
- at:putChar:
- charAt:

TestCase subclass TestString
- testAppendChar
- testAppendString
- testSubstring
- testSubstringBoundaries
- testPutChar
Unit Tests: The Internals

• Write like any other method, but assert what you want to happen

  ```plaintext
testAppendChar
  |string|
  string := 'go'.
  string append: ' banana!'.
  self assert: string = 'go banana!'.
  ```

• Tests are run with TestRunner GUI
• So let’s go write some tests for Strings!
Asserting more things

- **assert**: takes what you expect to be true
- **deny**: takes what you expect to be false
- **should:raise**: takes a block and what kind of error it should raise
- **shouldnt:raise**: 
Unit Tests: More Details

- The setUp method happens before each testX method (the framework assures this happens)
- The tearDown method happens after
- Let take a look at the official StringTest…
Best Practices

• Test everything that you want to work
• More test methods in your TestCase than in the class you are testing
• Tests should be as fine grained as possible
• Tests should be independent
• Should not take long to run (a few seconds)
• Easy to understand and read
So why Unit Test?

- Not much work to write or run
- Documents your class
- Gives you / others confidence that your code works
- No need to wait for “testing team”
- Tests are fine grained – can be run independently
- Tests can be aggregated easily
- Which tests fail give you a hint of where a bug was introduced
- Form a fairly complete regression test
What is Test-Driven Development?

- A new way to build software
- A strict development method:
  1. Add a test.
  2. Run the test.
  3. Make a small change.
  4. Run the tests again. (If they fail, go back to 3)
  5. Refactor (while testing)
Where did this come from?

• Test-First Development (+refactoring)
• A practice of Extreme Programming
  – Accept and love change
  – Release early, release often
• There are many supposed advantages, but we’ll discuss those after we try it
So let’s do it…

- We’ll build a little application that represents a network of friends.
- We’ll build incrementally
- Build acceptance/unit tests out of “user stories”
- I’ll be both the customer and lead developer
  - The customer is on site, so you can ask him questions, but he won’t interject
- You’ll be developers, too
So why Test-First?

• You always know what to do next: write a test or make a test pass
• You test code while you are writing it, instead of after you have forgotten about it
• Your tests are always up to date – no backlogs of testing to-do
• You take the customer’s point of view – what do I really want the code to do
• The code you have is exactly what is requested – no more, no less
Cute Stories

- Mean time to failure
- Test orb
- It scales
References

• Material from this presentation gathered from:
  – http://www.agiledata.org/essays/tdd.html
  – http://en.wikipedia.org/wiki/Test_driven_development