CMPT 473
Software Quality Assurance

Unit Testing & Testability

Nick Sumner - Fall 2014

Levels of Testing

- Recall that we discussed different levels of testing for test planning:
  - Unit Tests
  - Integration Tests
  - System Tests
  - …

Levels of Testing

- Recall that we discussed different levels of testing for test planning:
  - Unit Tests
  - Integration Tests
  - System Tests
  - …
- The simplest of these is *Unit Testing*
  - Testing the smallest possible fragments of a program

Unit Testing

- Try to ensure that the *functionality* of each component works in isolation

Unit Testing

- Try to ensure that the *functionality* of each component works in isolation
  - Unit Test a car:

    Wheels work. Steering wheel works....

Unit Testing

- Try to ensure that the *functionality* of each component works in isolation
  - Unit Test a car:

    Wheels work. Steering wheel works....
  - Integration Test a car:

    Steering wheel turns the wheels....

Unit Testing

- Try to ensure that the *functionality* of each component works in isolation
  - Unit Test a car:
    Wheels work. Steering wheel works....
  - Integration Test a car:
    Steering wheel turns the wheels....
  - System Test a car:
    Driving down the highway with the air conditioning on works...

Unit Testing

- Try to ensure that the *functionality* of each component works in isolation
  - Unit Test a car:
    Wheels work. Steering wheel works....
  - Integration Test a car:
    Steering wheel turns the wheels....
  - System Test a car:
    Driving down the highway with the air conditioning on works....
- Not testing how well things are glued together.

Unit Testing

- Try to ensure that the *functionality* of each component works in isolation
  - Unit Test a car:
    Wheels work. Steering wheel works....
  - Integration Test a car:
    Steering wheel turns the wheels....
  - System Test a car:
    Driving down the highway with the air conditioning on works....
- Not testing how well things are glued together.

Why? How is this beneficial?

Unit Tests

- A dual view:
  - They specify the expected behavior of individual components

Unit Tests

- A dual view:
  - They specify the expected behavior of individual components
  - An executable specification

Unit Tests

- A dual view:
  - They specify the expected behavior of individual components
  - An executable specification
- Can even be built first & used to guide development
  - Usually called Test Driven Development

Unit Tests

- A dual view:
  - They specify the expected behavior of individual components
  - An executable specification
- Can even be built first & used to guide development
  - Usually called Test Driven Development

Have any of you experienced this in your jobs?

Unit Tests

- Some guiding principles:
  - *Focus* on one component *in isolation*
  - Be *simple* to set up & run
  - Be easy to *understand*

Unit Tests

- Some guiding principles:
  - *Focus* on one component *in isolation*
  - Be *simple* to set up & run
  - Be easy to *understand*
- Usually managed by some automating framework

Unit Tests

- Some guiding principles:
  - *Focus* on one component *in isolation*
  - Be *simple* to set up & run
  - Be easy to *understand*
- Usually managed by some automating framework

What frameworks have you used to write unit tests?

JUnit

- Widely used unit testing framework for Java
  - And I *know* that you should have seen it before

JUnit

- Widely used unit testing framework for Java
  - And I *know* that you should have seen it before
- We'll use it to drive our discussion, but its features exist in other frameworks, too.

What features do your unit testing frameworks provide?

JUnit

- Basic Features

JUnit

- Basic Features
  - Per class set up & tear down
    @BeforeClass & @AfterClass

JUnit

- Basic Features
  - Per class set up & tear down
    @BeforeClass & @AfterClass
  - Per test set up & tear down
    @Before & @After

JUnit

- Basic Features
  - Per class set up & tear down
    @BeforeClass & @AfterClass
  - Per test set up & tear down
    @Before & @After
  - Test identification & properties
    @Test (expected=Exception.class) (timeout=100)

JUnit

- Basic Features
  - Per class set up & tear down
    @BeforeClass & @AfterClass
  - Per test set up & tear down
    @Before & @After
  - Test identification & properties
    @Test (expected=Exception.class) (timeout=100)
  - Test hiding
    @Ignore

JUnit

- Basic Features
  - Per class set up & tear down
    @BeforeClass & @AfterClass
  - Per test set up & tear down
    @Before & @After
  - Test identification & properties
    @Test (expected=Exception.class) (timeout=100)
  - Test hiding
    @Ignore
  - Expectations
    assertEquals, assertTrue, assertFalse, assertNull, ...

JUnit

- Let's work through a simple example

(There shouldn't be anything new here, but it should be a refresher if you haven't seen it in awhile)

Testability

- What makes testing hard?
  - Not just difficult to get adequacy
  - What makes it difficult to *write* tests?

Testability

- What makes testing hard?
  - Not just difficult to get adequacy
  - What makes it difficult to *write* tests?
- Dependencies
  - Connections between classes

Testability

- What makes testing hard?
  - Not just difficult to get adequacy
  - What makes it difficult to *write* tests?
- Dependencies
  - Connections between classes
  - Singletons

Testability

- What makes testing hard?
  - Not just difficult to get adequacy
  - What makes it difficult to *write* tests?
- Dependencies
  - Connections between classes
  - Singletons
  - Nondeterminism

Testability

- What makes testing hard?
  - Not just difficult to get adequacy
  - What makes it difficult to *write* tests?
- Dependencies
  - Connections between classes
  - Singletons
  - Nondeterminism
  - Static binding

Testability

- What makes testing hard?
  - Not just difficult to get adequacy
  - What makes it difficult to *write* tests?
- Dependencies
  - Connections between classes
  - Singletons
  - Nondeterminism
  - Static binding
  - Mixing construction & application logic
  - ...

Testability

- What makes testing hard?
  - Not just difficult to get adequacy
  - What makes it difficult to *write* tests?
- Dependencies
  - Connections between classes
  - Singletons
  - Nondeterminism
  - Static binding
  - Mixing construction & application logic
  - ...

But solutions exist!
You can *design* code to be testable!

Testability (by example)

- Let's work together to improve some difficult to test code....

Testability

- Keys things to notice:
  - *Mocks* & *stubs* allow us to isolate components under test

Testability

- Keys things to notice:
  - *Mocks* & *stubs* allow us to isolate components under test
  - *Dependency Injection* allows us to use mocks and stubs as necessary

Testability

- Keys things to notice:
  - *Mocks* & *stubs* allow us to isolate components under test
  - *Dependency Injection* allows us to use mocks and stubs as necessary
  - But doing this can lead to a lot more work and boilerplate code when written by hand

Testability

- Keys things to notice:
  - *Mocks* & *stubs* allow us to isolate components under test
  - *Dependency Injection* allows us to use mocks and stubs as necessary
  - But doing this can lead to a lot more work and boilerplate code when written by hand

Given dependency injection,
what happens to the way we create objects?

Testability

- Keys things to notice:
  - *Mocks* & *stubs* allow us to isolate components under test
  - *Dependency Injection* allows us to use mocks and stubs as necessary
  - But doing this can lead to a lot more work and boilerplate code when written by hand

Given dependency injection,
what happens to the way we create objects?

How might we mitigate boilerplate issues?

Mocking Framework Example

- Frameworks exist that can automate the boilerplate behind:

Mocking Framework Example

- Frameworks exist that can automate the boilerplate behind:
  - Mocking

    e.g. Mockito, Jmock, etc.

    [DEMO]

Mocking Framework Example

- Frameworks exist that can automate the boilerplate behind:
  - Mocking

    e.g. Mockito, Jmock, etc.
  - Dependency Injection

    e.g. Google Guice, Pico Container, etc.