Unit Testing

by Kip Irvine

COP 4814

Revision date: Jan 18, 2018

(c) 2008-2018 Kip Irvine, all rights reserved. You may modify and copy this slide show for your personal use as long as this copyright statement is included.
Overview

• What is a Unit Test?
• Why do Unit Testing?
• Goals
• Test-Driven Approach
• Common Excuses
• Example 1: Largest
• Running JUnit Tests
• Static methods & asserts
• Example 2: Stack
• Reference Sources
Goals of Software Testing

1. Does the application code accomplish the basic goals of the application?
2. Does the code work with a variety of inputs?
3. Will it stand up under extreme types of input?
4. Does the code document my intent?
What is a Unit Test?

• A method that tests results produced by another method
  • proves (or disproves) that individual methods work as expected

• Called a *structural test*, or *white box test* because it requires knowledge of the source code details
Two Classes

• Class being tested
  • called the *class under test*, or *target class*
  • contains methods that do the actual work

• Test class
  • also called a *Test Fixture*
  • contains methods that test the Application class methods
Here's What Unit Testing Can Do

• Reduces the time you spend debugging, particularly at the end of a project
• Improves reliability of application code
• Minimizes bug fixes that affect existing code
  (run regression tests after fixing a bug)
Cost of Fixing Defects

Stages

0.1
Requirement

0.5
Design

1
Coding

2
Unit

5
Acceptance

20
Maintenance

Relative cost of repair

source: www.verisoftindia.com
Test-Driven Approach

1. Write several unit tests before writing the target application code
2. The test should document the application code's intent
3. Run the tests and watch them fail
4. Write application code that permits the tests to pass
I'm already busy writing code! Why should I write even more?
Common Excuse for not testing:

Writing tests takes up too much time

Rebuttal:
• Unit Testing saves time—why?
  • It's easier to write tests while you still remember the code
• Waiting until the end means dealing with greater complexity
  • reworking defective code under pressure
  • tracking down bugs to their source in a large program
• you might create new bugs when fixing old ones
Task completion (x) vs Time (y)
The End