Introduction to Eclipse, Unit Testing and JUnit

David Rabinowitz

Eclipse

- Eclipse is a kind of universal tool platform - an open extensible IDE for anything and nothing in particular.
- plugins
  - JDT
  - CDT
  - UML
  - Many many more

Working with Eclipse – Live Demonstration

- Project
- Views
- Code Assist
- Errors
- Debugging

Unit Tests

- First level of testing
- Done by the programmer
- Part of the coding process
- Delivered with the code
- Part of the build process

What's a Unit?

- Unit tests should be written in test classes, that replace 'drivers'
- For example, class Stack
  - Class Stack Has push, pop, count, …
  - Class TestStack Has testPush, testPop
- Tests are per-functionality
  - Not per method (testPushNull)
  - Not per class (testIterators)
- Unit testing can have several levels

What's a Unit Test?

- Call methods, and assert conditions

```
void testPush() {
    s = new Stack();
    s.push(new Integer(10));
    assertEquals(s.count() == 1); }
```

- Tests check themselves
  - Only output if a test fails
- Write tests after writing interface
- Run all tests after each change

Running Unit Tests

- A main() is required to run tests
- There are better options
  - JUnit for Java
  - CppUnit for C++
- Unit Testing Frameworks
  - Graphical user interface
  - Easily choose which tests to run
  - Elegantly support for test suites

Coding with Unit Tests

- Part of the design process
  - Design for testability --> Modularity
- Part of the coding process
  - Test-first coding
  - Run tests after each build
- Part of the build process
  - Build = compile + link + pass unit tests
  - A.k.a. "smoke tests"

Benefits of Unit Tests

- Regression Testing
  - For your own code
  - Daily build: for others' work
- Part of the usual work
  - Replaces work done anyway
  - Causes tests to be written
  - Validates the design before impl

Top 12 Reasons to Write Unit Tests Part I

- Tests Reduce Bugs in New Features
- Tests Reduce Bugs in Existing Features
- Tests Are Good Documentation
- Tests Reduce the Cost of Change

Top 12 Reasons to Write Unit Tests Part II

- Tests Improve Design
- Tests Allow Refactoring
- Tests Constrain Features
- Tests Defend Against Other Programmers

Top 12 Reasons to Write Unit Tests Part III

- Testing Is Fun
- Testing Forces You to Slow Down and Think
- Testing Makes Development Faster
- Tests Reduce Fear

JUnit

- Unit testing framework for Java
- Open Source
- Integrated into Eclipse
- TestCase
  - setUp(), tearDown()
  - Test<TestName>()
- TestSuite
- TestRunner

Resources

- Eclipse
  - http://www.eclipse.org/
  - http://www.eclipse-plugins.info/
- Top 12 Reasons to Write Unit Tests
  - http://www.onjava.com/pub/a/onjava/2003/04/02/javaxpckbk.html
- JUnit
  - http://www.junit.org/