Producing Production Quality Software

Lecture 5: Testing
Prof. Arthur P. Goldberg
Fall, 2005

Testing Topics

• Testing phases
• ‘Glass box’ unit testing
• JUnit
• Holostic testing
• ‘Black Box’ testing
• Limitations
Testing Phases

- Unit
- Regression
- Integration (interfaces)
- System
- Acceptance

‘Glass box’ unit testing

- Written by programmer
- Packaged with code
- Ideally written concurrent with, or before, program
- Reusable for regression testing
Java Unit Test Package: JUnit

- Reusable
- Built on Asserts, e.g.
  
  ```java
  Assert.assertTrue(!m12CHF.equals(null));
  Assert.assertEquals(m12CHF, m12CHF);
  ```

- See
  - Gamma, Beck, *JUnitTest Infected: Programmers Love Writing Tests*
  - Nice tutorial about JUnit in Eclipse
    http://www.cs.umanitoba.ca/~eclipse/10-JUnit.pdf
  - Andy Hunt, Dave Thomas, *Pragmatic Unit Testing in Java with JUnit*
  - Johannes Link, *Unit Testing in Java: How Tests Drive the Code*

JUnit Test framework

- TestCases, e.g.
  ```java
  public class MoneyTest extends TestCase {
      // ...
      public void testSimpleAdd() {
          Money m12CHF= new Money(12, "CHF");  // (1)
          Money m14CHF= new Money(14, "CHF");
          Money expected= new Money(26, "CHF");
          Money result= m12CHF.add(m14CHF);     // (2)
          Assert.assertTrue(expected.equals(result));  // (3)
      }
  }
  ```

- A **TestCase** subclass implements multiple
  **public void test...() methods**
JUnit Test framework, cont.

- The TestCase calls setUp() and tearDown() before each test...() method
- setUp() creates a ‘fixture’
- A TestSuite runs all the tests
- TestSuite extracts the test methods from the TestCase subclass, e.g.
  ```java
  public static Test suite() {
      return new TestSuite(MoneyTest.class);
  }
  ```

JUnit Test framework, cont.

- The TestSuite tracks the results of all the Assert methods
- Reports
  - Number of successes
  - Individual failures
- GUI too
More Systematic Unit Testing

- Coverage testing: executing all statements, including error handling code
- Structured basis testing
  - Minimal coverage testing
  - Number of tests required given by $Test$
    
    ```
    Test = 1;
    For each branch top (if, while, case, and, or, for, ...)
    Test++;
    ```
- See structured basis example

‘Holistic’ Testing

- Often complex software can be tested by examining boundary conditions
- Mathematica does this
- For example
  - Exploit symmetries to generate tests
    - ‘Pool ball’ simulation example
  - Implement more than once
    - Subset example
Quality Assurance (QA)

- Independent testing
- In military software testing, must be a different company (IV&V)
- Would typically perform these tests:
  - Regression
  - Integration (interfaces)
  - System

‘Black Box’ Testing

- J. Whittaker, *How to Break Software*
- Philosophy
  - Attack weaknesses in the code
  - Assume no specification
- Types of ‘attacks’
  - UI
    - I/O
    - Data and computation
  - System interface
    - File system
    - OS interface
- Hostile Environment Application Tester (HEAT)
Limitations

• What doesn’t the testing we’ve discussed do?

Recommended Books

• Cem Kaner, Jack Falk, Hung Q. Nguyen, *Testing Computer Software*
• Cem Kaner, James Bach, Bret Pettichord, *Lessons Learned in Software Testing*