Tuesday, November 15

Testing

Testing

- Waterfall model show testing as an activity or box
  - In practice, testing is performed constantly
- There has never been a project where there was too much testing.
  - Products always ship with some defects
- Test cases are a valuable resource
  - Should be managed like code

Review

- Name and describe four types of testing.
- What is the difference between black box and white box testing?

Quality Assurance Activities

- Verification
  - Check product against specification
  - Building the system right
- Validation
  - Check product against world (stakeholder expectations)
  - Building the right system
- van Vliet considers all quality assurance activities as testing

Testing Objectives

- Goal of testing is to make the software misbehave
  - Failures tell you a lot more than successes
- Your reward is finding a bug, even if it's your own code
  - No prizes for test cases that pass
- Testing can only tell you about the presence of defects
  - Need to use proofs and other checks to show correctness

The Tester's Role on Agile Projects

Testers in their traditional role	Tester role in an agile project
▪ A separate QA group ▪ Tests are derived from detailed requirements and specifications ▪ QA may or may not participate in planning sessions, but is not usually informed about design considerations until after they have been finalized	▪ Is part of the team and attends all team sessions ▪ Is an integral part of the planning game ▪ Practices pair testing, i.e. collaborates with the developers to get good tests

http://www.ucalgary.ca/~ageras/wshop/abstracts/2003/role-agile-tester.htm

Thursday, November 18

Automated Testing using JUnit

Automated Testing

- Idea: Testing is repetitive. Get a computer to do the work for you.
  - Computers are good at repetitive sequences and don't get bored.
  - More reliable and robust than testing by hand.
- Benefits
  - Can test frequently at little additional cost
  - Greater confidence in the code
- Costs
  - Tests need to maintained along with code
    - e.g. refactoring

JUnit

- Framework for performing unit testing on Java programs
  - Test cases are sub-classed from an interface
- Available as a stand-alone application and built into Eclipse
  - Cppunit available for C++ code, httpunit for web pages
- Framework executes the test cases and records the results
  - Displays results in a GUI
  - "Keep the bar green to keep the code clean."

Unit Testing

- A unit test typically tests one class in the system
  - A unit test suite contains many test cases
- Each test case typically tests one method in the system
- There can be many test cases for each method in the system
- Each test case either succeeds or fails, there is no gray area
- If a test case has an error, that is also a failure
- A test or test suite can be said to succeed to a certain percentage

How to use JUnit

```
class C {
   method m1();
   method m2();
}
```

```
class CTest  extends TestCase {
   method testM1();
   method testM2();
}
```

```
class D {
   method m3();
   method m4();
}
```

```
class DTest extends TestCase {
   method testM3();
   method testM4();
}
```

- Each test class exercises one class in the system. Each test method exercises one method in the system. You also write additional test methods to exercise combinations of system methods.

How to use JUnit

- Each test method consists of a sequence of steps, and some checks of the results.
- Once you have the unit tests written, you run them. You could run them directly from main(), but it is easier to use a test running utility
  - Options: JUnit TestRunners or the Ant junit task.

JUnit Methods

- `assertEquals(x, y)` – Test passes if `x` and `y` are equal
  - x and y can be primitives or any type with an appropriate equals method
  - Three argument versions exist for floating point numbers
- `assertFalse(b)` – Test passes if boolean value `b` is false
- `assertTrue(b)` – Test passes if boolean value `b` is true

- `assertNull(o)` – Test passes if object `o` is null
- `assertNotNull(o)` – Test passes if object `o` is not null

- `assertSame(ox, oy)` – Test passes if `ox` and `oy` refer to the same object
- `assertNotSame(ox, oy)` – Test passes if `ox` and `oy` do not refer to the same object

JUnit Test Runner Sequence

- Test runner is given a list of test classes
- For each test class

  Create an instance of the test class
  For each test*() method
  Run setUp() method
  Run test method steps and checks
  If a check fails, an assertion is thrown and the test method fails
  Run tearDown() method
- Test runner produces a report
- Some test runners work interactively

Example from JUnit Primer

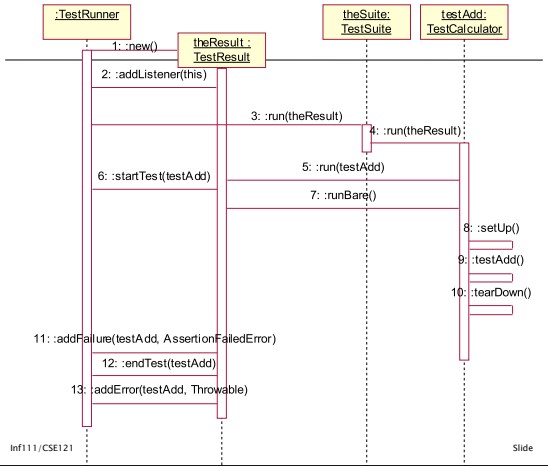
http://www.clarkware.com/articles/JUnitPrimer.html

Checking for Duplicate Objects

- Why can't you just use a different collection class?
  - The need to check for an attempt to add a duplicate object will arise with all collection classes.
  - This is a conceptual problem, not a logic problem.
- Isn't it expensive to have to iterate through the array every time?
  - It's computationally expensive, but it's a small price to pay to prevent/catch human errors.
  - Can be made cheaper with a different collection class.

Things to Notice

- setUp() method makes some variables that are used in the tests
  - Officially called “fixtures”
- tearDown() frees memory, prevents results of one test from affecting the next
- Only the first failure in a test method is reported
  - Don’t do too much in a single test
- Missing test cases: a new cart should be empty, add the same product twice, remove a product that was already removed, test isEmpty(), etc.



Other Details

- Ordering of Test Cases
  - Not guaranteed
    - Could be in order of presentation in file
    - Could be something else
  - Can control by manually loading into a test suite
    - More work and can be error prone, but more predictable
- Sequences of Tests
  - Same as above
- Exceptions
- Customization
  - Can write your own JUnit runners

Tests with Exceptions

```
public void testRemoveItem(){
    try {
       _testCart.removeItem(_secondItem);
       fail("Should raise a product not found
  exception");
    }
       catch (ProductNotFoundException pnfe){
       assertNotNull(pnfe);
    }
}
```

More Information

- Eclipse Help
  - Help -> Help Contents -> Java Development User Guide -> Getting Started -> Basic Tutorial -> Writing and running JUnit tests
- JUnit Home Page
  - http://www.junit.org
- JUnit Primer
  - http://www.clarkware.com/articles/JUnitPrimer.html