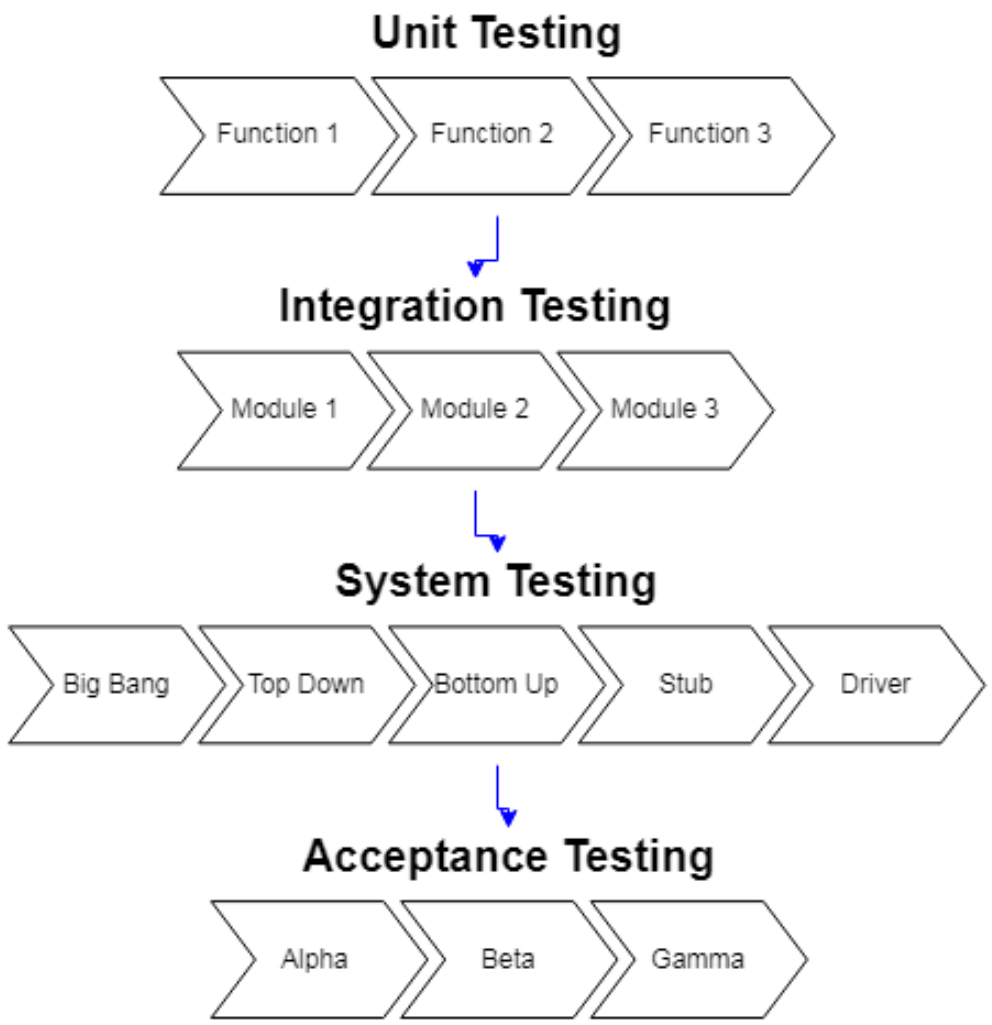
CS1632, Lecture 9: Unit Testing, part 1

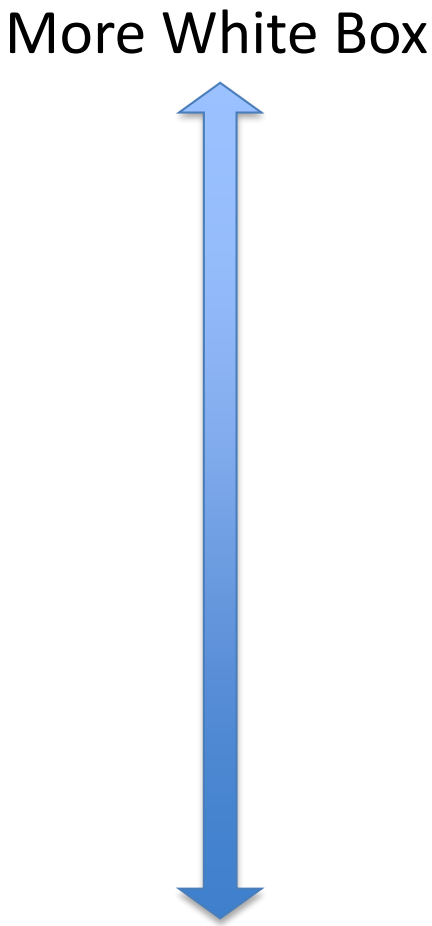
Wonsun Ahn

What is unit testing?

- Unit testing involves testing the smallest coherent "units" of code, such as functions, methods, or classes.
- It is white-box; you are looking at and testing the code directly.
- Ensures that the smallest pieces of the code work correctly (NOT that they work correctly with rest of the system – very localized)

The Four Levels of Software Testing





Examples

- Testing that sort() method actually sorts elements
- Testing that formatNumber() method formats number properly
- Testing that passing in a string to a function which expects an integer does not crash the program
- Testing that passing in a null reference throws an exception
- Testing that a send() and receive() methods exist on a class

Who does unit testing?

- Usually done by the developer writing the code
- Another developer (esp. in pair programming)
- (Very occasionally), a white-box tester.

What’s the point?

1. Problems found earlier
2. Faster turnaround time
3. Developer understands issues with his/her code
4. "Living documentation“
5. Unit tests in sum total form a test suite
   - Running full test suite (e.g. as part of regression test) allows quick detection of defects due to code changes with non-local impact
   - Easy to zero in on the failed unit if a unit test fails

What do unit tests consist of?

- A unit test is essentially a test case at the unit testing level
  - Same components: preconditions, execution steps, postconditions, …
- Anatomy of a unit test when implemented (e.g. using JUnit):
  - Preconditions: set up code (inits variables / data structures, …)
  - Execution Steps: one or more calls to unit tested method
  - Postconditions: assertions (checks postconditions are satisfied)
  - (Optional) tear down code (return to clean slate for next unit test)

Example of a Unit Test Case

- Preconditions: Two linked lists with one node each, where nodes contain the integer value 1
- Execution Steps: Compare the two lists with equality operator
- Postconditions: The result SHOULD be true

Unit test example

```
// Check that two LLs with the same Node value with a single
node are equal
@Test
public void testEqualsOneNodeSameVals() {
   LinkedList<Integer> ll11 = new LinkedList<Integer>();
   LinkedList<Integer> ll12 = new LinkedList<Integer>();
   ll11.addToFront(new Node<Integer>(new Integer(1)));
   ll12.addToFront(new Node<Integer>(new Integer(1)));
   assertEquals(ll11, ll12);
}
```

- `assertEquals`: Invokes equals method on arguments and asserts it returns true

More linked list test examples

sample_code/ junit_example/LinkedListTest.java

Postconditions = assertions

- When you think "should" or "must", that is the assertion. It's what you're testing for.
- It's the EXPECTED BEHAVIOR of the unit test.
- When you execute the test, that's when you'll find out the OBSERVED BEHAVIOR.
- If the expected behavior matches the observed behavior, the test passes; otherwise it fails.

JUnit assertions

- Some possible assertions using JUnit:
  - assertEquals, assertArrayEquals, assertSame, assertNotSame, assertTrue, assertFalse, assertNull, assertNotNull, assertThat(*something*),...
- fail() : assertion that always fails
  - Why would you want an assertion that always results in test failure?
  - Maybe you shouldn't have even gotten to that part of code

fail() example

```
//  Check that passing null to addToFront() results in an IllegalArgumentException
@Test
public void testAddNullToNoItemLL() {
   LinkedList<Integer> ll = new LinkedList<Integer>();
   try {
      ll.addToFront(null);
      fail("Adding a null node should throw an exception");
   } catch (IllegalArgumentException e) {
   }
}
```

- Code execution never reaches fail() due to exception, as designed

JUnit is not the only unit test framework out there!

- Not even for Java!
- But xUnit frameworks are common and easy to understand
  - C++: CPPunit
  - JavaScript: JSUnit
  - PHP: PHPUnit
  - Python: PyUnit
- Ideas should apply to other testing frameworks easily

Now Please Read Textbook Chapter 13

- In addition, look carefully into: sample_code/junit_example/LinkedListTest.java
- You can run all JUnit tests by executing runTests.sh
  - You will have to give execute permissions first (chmod +x runTests.sh)
  - Or invoke the shell explicitly (bash runTests.sh)