CS1632, LECTURE 9: UNIT TESTING, PART 1

BILL LABOON

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 the rest of the system – very localized)

EXAMPLES

1. Testing that a .sort method actually sorts elements
2. Testing that a nil/null throws an exception
3. Testing that a formatNumber method formats a number properly
4. Checking that passing in a string to a function which expects an integer does not crash
5. Testing that a .send and .receive method exist on a class

UNIT TESTING

This is usually done by the developer writing the code, another developer (esp. in pair programming), or (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. Able to tell if your changes caused issues elsewhere by running full test suite

WHAT DO UNIT TESTS CONSIST OF?

- (optional) Set up code
- Preconditions
- Execution Steps
- Postconditions - a/k/a Assertions (a/k/a asserts, shoulds, musts)
- (optional) Tear down code

EXAMPLE (IN NATURAL LANGUAGE, NOT CODE)

I create two linked lists with the same number of nodes and same data in each node.

If I compare them with the equality operator, they SHOULD be equal.

(or "they MUST be equal.")

(or "I ASSERT that they will be equal")

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, assertTrue, assertFalse, assertNull, assertNotNull, assertSame, assertThat(*something*), assertNotSame…
- fail()

fail() makes a test automatically fail. Usually because you know it's going to fail anyways and you don't want to waste time running it, or check that a certain part of code is not reached.

FAIL

UNIT TEST EXAMPLE

```
//  0. A LL should always equal itself
@Test
public void testEqualsSelf() {
    LinkedList<Integer> ll = new LinkedList<Integer>();
    assertEquals(ll, ll);
}
```

MORE LINKED LIST TEST EXAMPLES

sample_code/LinkedListTest.java

JUNIT IS NOT THE ONLY UNIT TEST FRAMEWORK OUT THERE!

- Not even for Java!
- But the xUnit frameworks are common and easy to understand
- Ideas should apply to other testing frameworks easily