**What is a Unit Test?**

A Unit Test is a minimal program to test that a particular area of functionality in another program works as designed.

For example, consider I wrote a function like this: int add (int x, int y)

How could I test this?

if (add(1,2) == 3) printf (“success”) else printf(“fail”)

This program is minimal (of course, it would need more to compile). Good unit tests don’t involve a lot of other code – the goal is a small, obviously correct program that proves our functionality.

**Why do we write Unit Tests?**

How can we prove that our code works? One approach is to build the entire program, then run it the way that a user would run it. There are problems with that:

1. It is 100% manual
2. When it doesn’t work, we don’t know what component to look at
3. It is not reproducible
4. Any change to the code means we should test the whole program again

Unit tests are automated, single component tests that we can reproduce. They should prove that any component that has been changed still works the way that it was intended to work.

**What unit tests should we write?**

How would you prove that add() works? Adding 1 and 2 to get 3 is a basic case. What other cases can you think of that might not work even if 1+2=3 DOES work? Maybe involve 0? Maybe involve negative numbers? Minimum and maximum value for int? You need to write enough tests to prove that your function works. If you need **many** tests, that indicates a function that does too much.

An example:

For “The Bit”, we need to test “AND”. This is a case where we can test all of the possibilities, since there are only 4.

I would test it something like this (untested pseudo-code):

public void testAnd() {

if (new bit(0).and(new bit(0)).getValue() !=0) throw new Exception(“0 AND 0 failed”);

if (new bit(0).and(new bit(1)).getValue() !=0) throw new Exception(“0 AND 1 failed”);

if (new bit(1).and(new bit(0)).getValue() !=0) throw new Exception(“1 AND 0 failed”);

if (new bit(1).and(new bit(1)).getValue() !=1) throw new Exception(“1 AND 1 failed”);

}

This process **must be repeated** for every method. I know that sounds like a lot of work. Honestly, it is. The value in this process is that you **know** that your code works.

**Other work to make this function**

Of course, this needs to go in an object (Bit\_test) and the program as a whole needs a main(). It is reasonable to make a method that calls all of your individual test. So you could have:

public class Bit\_test {

public static void main(String[] args) {

runTests();

}

public static void runTests() {

testAnd();

testOr();

etc…

}

}

There are unit tests systems (like jUnit) that will run your tests. You **may** use these, but you are not required to.