Now let's move onto the next method.

So here we have this method called max, which returns the maximum of two numbers. How many execution paths do we have here? We have two execution paths. And that is dependent upon this condition. So if A is greater, we're going to return A, otherwise we're going to return B.

So there are two execution paths in this method. And the number of tests is often equal to or greater than the number of execution paths. So, back in our testclass, I'm going to create another test method, test, public, void, you're testing the max method.

Now when anyone(?) write tests for a method, I would rather create all the test methods first before implementing them. I just want to brainstorm all the different scenarios that need to be tested. This way I make sure that I won't miss a scenario when I get busy writing the

tests.

So, the first scenario I want to test here is when the first argument is greater. So, first argument is greater, it should return the first argument.

Now, I'm going to duplicate this, the second scenario is where?

The second argument is greater. So, second argument is greater and it should return the second argument.

Now here we have a third scenario. Can you tell what scenario I'm talking about here? That is the scenario where the two arguments are equal. Now, this is something that I want you to pay great attention to here. So far, in order to write tests, I started by looking at the implementation of method that I wrote tests based on this implementation.

This is a good approach to get started, but it's not enough, it's not something that you should rely on. Because with this approach, when you write your tests based on implementation, it is possible that implementation might have problems.

Maybe your implementation is missing something. So you don't want to rely on that implementation. The best way to write test for method, is to think of it as a black box. Let's imagine we don't know what is inside the max method. Max method is just one method that takes two

arguments. A ending. What are different possibilities here? One possibility is A is greater, another possibility is B is greater, and the last possibility is their equal. So, I'm going to go back to our test class, duplicate the last test, and change the name of the test method.

So, this scenario condition is arguments are equal it should return the same argument. Okay? Now, let's start implementation. So first we should create an instance of the Math class. Math.

Now enact, call the Math method, here the first argument should be greater. So, I'm going to pass 2 and 1. Get the result, and Assert. That result, is equal to 2. Let's run this test. It's passing, beautiful. Now, the second test. To save time, I'm just going to copy these few lines, but in the real world you shouldn't copy paste code, it's better to write it from scratch, because sometimes when you copy paste code, you make mistakes. So, here I'm going to change these arguments to 1 and 2. So the second argument is greater, and the Assertion should still be the same. And one last time,

Going to paste these few lines. And as the arguments I'm going to pass one and one and Assert that result is equal to one. Now, I'm going to run all the tests in this class. So I put my curser on the class name, right here, and press command and T, and command and R. So you can see, all the tests in this class are passing. Beautiful.