In this lecture, we're going to look at testing methods

that return an array or a collection. So, back to our math class, we have another method here, that takes the limit, and returns the odd numbers

starting from 0, up to that limit.

So it's returning an IEnumberable of this version. When testing methods that return an array or a collection, again you need to do your own assessment on how general, or how specific your tests should be. Let me show you a few examples. So, back to our math tests class. I'm going to define a new test method here, public void GetOddNumbers.

Now what scenarios do we have here? Again,we don't want to look at the implementation, we want to think of this method as a black box. So it has one argument, limit, this limit can be a negative number, it can be 0, it can be a positive number.

So these are the three test cases I would write for this method. Now, in this lecture, I don't want to implement all these test cases, because

our focus is on testing arrays and collections. I'm going to show you

one test case, and then you can implement the other test cases on your own.

So, the scenario is LimitIsGreaterThanZero. It should return OddNumbers

UpToLimit. So, you have a math object to simply call math.GetOddNumbers.

And as the limit, I'm going to pass 5. You will ask why I didn't use 1 here. Because if we use one, we get an array with only one item.

When dealing with arrays, I would like my array to have three items.

So it's more real life. So 5, we get the result, now.

Let's look at a few different ways to write our Assertions. Here's the most general way to write an Assertion. Assert That Result Is Not Empty. Very general, I can have any numbers in this array and it will pass.

Sometimes you don't hear about specific values in an array or a collection,

you just want to make sure that there is something in the array or collection that is returned from your methods. In that case, this Assertion is perfectly fine.

But in this case, it's a little bit too general, a more specific

way is to test for the number of items in this array. So, AssertThat result.Count Is EqualTo3 A little bit more specific, but we can have any numbers in this array, you can have 1, 1, 1. It doesn't give us much confidence.

Again, another general way to write an Assertion is like this. Assert That result

Does Contain 1.

I'm going to duplicate this, it also contains 3, and 5.

So I just want to clarify something, I'm showing you different Assertions here, but that doesn't mean when you're testing this method, you should write all these Assertions, you should either use an Assertion like this, to make sure the result is not empty, or you should check for the count, or if you want to be more specific you want to check for the

existence of certain objects in the array, but you don't care about the order. because if you care about the order, your test is very specific.

Now sometimes, maybe that's what you want because your array

is supposed to return items that are sorted ahead of time. But in this case, chances are we don't care about the order of these items in the array. So we can write three Assertions to just make sure that we have 1, 3, and 5. But there is a shorter, and cleaner way to write the same three Assertions. You can write Assert, That, result Is Equivalent to, and here we create an array with three numbers. 1, 3, 5. So this is equivalent to

make sure that we have all these items in the result array, it doesn't care about the order, it just checks for the existence of each of these items. So this line is equivalent to writing these three lines.

So this is my preferred way to test the GetOddNumbers method. Let's run this test and make sure it

passes. Okay. It's passing, beautiful. Now, before we finish this lecture, I'm going to show you a couple more useful Assertions.Assert,

That result IsOrdered.

So if your method is supposed to sort the items, you can make sure they're ordered. You have another useful Assertion, Assert That result Is Unique. So with this you make sure that there are no duplicate items in your array.

For this particular method, we don't really need these Assertions.

But I want you to be aware of them, maybe you will need these in your applications.

So this is what I want you to take away. When you're testing methods thatreturn an array or a collection, make sure you'retest methods are not too general, or too specific.Find a right balance.