So in this lecture, I'm going to show you how to create fake or mock objects using the Moq library. So back in our solution, right click the unit testing project. And go to Manage NuGet Packages.

Here, you should search for Moq so you can see the current version, it's version is 4. 7. So we need to add this to our unit testing project.

Okay, alright, now to show you that we no longer need this FakeFileReader class, I'm going to delete it from our project, so here it is, delete.

So, back in our video service tests, instead of newing up a Fake FileReader, we're going to use the Moq library to create a dynamic mock.

It's very very easy, so, I'm going to create a fileReader and set this to a new Mock look, this is defined in the mock name space and this is a generic class. So, we specify the generic argument in this case IFileReader. So we're telling the Moq library we want an object that implements, this IFileReader interface, okay?

So, this fileReader object here is not the actual object, it's a mock object, because we set this to a new, mock of IFileReader, to make this more clear, you may prefer to rename this to mock FileReader. But I personally find this a little bit noisy. So I would rather use file Reader, okay? Now we need to program this mock, because by default, it doesn't have any behavior it's like an object that implements the iFileReader interface but doesn't do anything that doesn't have any code, now let's go back to our video service class, look, on this line, we're calling the read method a fileReader, and we're passing it, video.txt as an argument. So, they (?) need to program our mock, so when we call the read method and give it video.txt as a string, it's going to return a string.

Okay? So back in our unit test, the way we do this is via the setup method. So we call fileReader.Setup.

And here we're going to pass a lambda expression like this. fr as in short for FileReader, goes to fr.Read and we specify the argument, video.txt, so with this we are telling this mock fileReader that when we call the read method with this argument it should Return some string. In this unit test, because the scenario we're testing is for an empty file, I'm going to return an empty string. Now here we have other options,

we can Throw an exception, and you can specify the type of the exception right here.

If you want to find out the complete list of features available in these mock objects. Search for mock documentation. So here in the first link Quickstart moq4 wiki on Github.

On this page you have various examples of using this mock object that are perfectly documented so you don't need a course to learn how to use Moq, okay? So back here, our mock object is going to return an empty string when we call the read method with this argument.

Now as I told you this fileReader is not that object that implements

IFileReader, it's a mock object, but here when initializing this video service, we need to get that object. That's very easy. We simply pass

fileReader.Object. So this is the actual object that implements IFileReader.

Okay? Now when you use mocks our test methods get a little bit noisy, as you can see here. That's why I said you should reserve mocks only for dealing with external dependencies. If you want to mock everything, if you want to mock every dependency of this video service, you're going to end up with the fact test method with a lot of set up code like this. Okay?

So use mocks only for external dependencies. Now, let's run this test. Okay, it just passed, beautiful. Now one last thing before we finish this lecture. When I use mocks, I prefer to move some of the arrange part in the SetUp method. Let me show you what I mean. So, let's create a new method here, public void SetUp, and decorate this with a SetUp attribute. Now, we're going to use this service in every test method

in this class, right? So we should promote this to a private field in this class. So alt and enter, Refactor Introduce Field. If you are not using

ReSharper or Rider, you should have this in Visual Studio as well.

So here we're going to create a private field called \_videoService. Okay?

Also, we're going to use this mock object in other test methods in this class. Because in this test method we are testing this scenario empty file, but in another test method we're going to test a different scenario. So we don't want to create this mock object in every test method, right? So once again, I'm going to promote this variable, to a private field in this class. So alt and enter, Refactor, Introduce field.

Okay? Now, I'm going to keep the set up here, because this is very specific to this test, but I'm going to move the initialization of fileReader and videoService into our setup method. So, let's put these together so first we create our mock object, and then we create our videoService based on that mock object.

Now we can move these two lines inside the set up method here. And this makes our test very clean and focused. So this is the scenario we're

specifically testing for.