So, once we generate a statement, then we should email it to the housekeeper.

So, back to our test class, I'm going to select this method duplicate it, the scenario is WhenCalled, it should EmailTheStatement. And here we are assuming a happy path, so no errors, so, let's remove the first line. Now this time in our arrange part we need to program this statementGeneratorMock.

So, I'm going to select this, cut, and move it here, instead of verify, we're going to use SetUp. Okay? And SetUp doesn't take the second argument, Times.Never, so let's delete this, instead we should call .Returns, here we should return the statement file name.

So, like this. filename. But as you learned, it's better to extract this and store it in a private field to reduce the noise in the code. So, alt and enter. Refactor, introduce field, let's call this statementFileName. Okay? Now, let's delete this line here and initialize this field upon declaration. So, private readonly string, set this to fileName.

Okay? Now back in our test, so here's our arrange part but I don't like the way this code is formatted, it's very confusing. So, see how I format this. I put SetUp here, then, put the close (?) in parenthesis, on this line and a line returns a SetUp. We can also remove this line break that's better. Now finally our Assertion. So, here's our emailSender object, let's Verify es goes to, we want to verify that email method is called with these arguments, so the email address of the housekeeper is Housekeeper. Email. And for email body, we should use houseKeeper.statement EmailBody. And the third argument here, is the statement file name.

And we have that as a private field, so statement FileName. Now this line is getting a little too long so let's break it up into multiple lines. So each argument on a separate line. And finally, the last argument should be the subject of the email. So if you look at the service look this is where we construct the subject. We can put this string in our test, but I'm not quite a fan of that, because it is possible that some times in the future we're going to change the subject of the email.

So in our unit test, I don't want to check for this exact string. Otherwise, this test is going to be fragile. So, back in the test, we can use It.IsAnystring This is a little bit more flexible. Okay? Now, let's run this test. It passed, I want to make sure we're testing the right thing, so back in our service, this is where we send the email, if you comment out this line, that test should fail.

So, back here let's run this test one more time. It failed, beautiful. So we have a trustworthy test, now back in the service let's bring this back in. Beautiful. Now look at this conditional statement, here. If this statement generator returns a file name that is null or white space, we are not going to send the email. So here we need 3 more unit tests.

If the file name is null, white space, or empty string. And then we want to verify that this method is not called. Just like before. So, to save time, I'm going to duplicate this method.

And change the scenario to statement FileNameIsNullAnd ShouldNotEmailTheStatement. So here, in the arrange part, you want to change the return value, to null. Now here we have a compilation issue because this method return has two overloads. One that returns a string and the other that returns a funk. When we pass null here the compiler is not sure which overload we are interested in. So, to get around that we can simply pass a funk here, a lambda expression that returns null.

So this is the arrange part, we act, now when verifying we want to pass a second argument for the Verify method, and set the number of times never. Now here, Times.Never. Also, we can remove the noise in this test because we don't really care about these exact arguments because we are not going to call this method anyway. So I would rather to replace these three strings with It IsAny string.

So, quickly, It Is Any string, duplicate it twice, and delete these two arguments. Now the test is cleaner, let's run it, it passed, beautiful, now finally let's duplicate it a couple more times, and test for white space and empty string. So StatementFileIs EmptyString, so here in the arrange part we want to return an empty string and the Assertion is exactly the same, now one more time I'm going to duplicate this, and this time,

we should change the scenario to Statement FileNameIsWhiteSpace.

So, a simple change like this. Now, let's run all the tests in this class. So, I put the cursor on the class name, and run the tests, all tests are passing, beautiful.