So, what should we test here?

Well, as I told you before, you should always test the outcome of a method. So, just to refresh your memory, we have two types of functions, query functions that returns some kind of value, and command functions that carry out a task. In this case, what kind of function do you think

SendStatementEmails is? Is it a query function or is it a command function? Well it's a command function, that is also returning some value, a boolean, right? But its main task is to perform some action, and that is saving the statement and emailing it to the housekeeper.

So, to test the outcome of this method, we should have 1 or. more unit tests for the value it returns, maybe in some conditions it returns true, in other conditions it returns false. Also, because it's a command function we should make sure that this method carries out the tasks it's supposed to do. So in this case, we should have a bunch of interaction tests to make sure that this object, this housekeeper helper object, talks to other objects like statementGenerator, and emailSender

properly.

So interaction tests. Now, it's easier to start with state-based tests. So we want to write unit tests to make sure that this unit test returns the right value. Now look at this implementation, the only way to

exit this method is here, return true.

So no matter what, this method is always returning true there is nowhere in this code, where we are returning false. So there is really no value for writing such unit tests. And this makes me think this method would be better to define as void.

So, you don't really need this return statement, here, and we can reduce this implementation by one line. Making this method shorter, so this is another benefit of writing unit tests. They help you come up with better design, for your methods and functions. Because for every method, for every public method you're going to test, you look at that method from the perspective of it's clients. How are these clients going to use this method? Do they really need this return value, this boolean? Probably not, because if this method is always returning true, the clients of this method are not going to have any decision making, okay? So, with a simple change we no longer need to write any state-based tests for this method,

In the next lecture, we're going to start writing our first interaction tests.

We want to make sure that this class talks to statementGenerator properly.