As I told you before, when I want to test a method, first I brainstorm all the different possibilities and test cases. So, I'm going to collapse this method so we don't get lost in the implementation detail, and look at it as a black box. What are various ntest cases we need to write for this method? Well, imagine we have an existing booking. It is possible that our new booking starts and finishes before this existing booking.

In this case, we don't have an overlap. The other scenario is our new booking finishes in the middle of an existing booking. In this case, we have an overlap, so our methods should return the reference of an existing booking. It is also possible that this new booking finishes after an existing booking.

Again, we have overlap, or, our new booking can start in the middle of an existing booking and finish after, or it can finish after the middle of an existing booking. And finally, the last possibility is where this new booking starts and finishes after an existing booking. In this case, we don't have an overlap, and our methods should return an empty string. Now, assuming that we have an overlap, we have two more scenarios, one is that the existing booking is cancelled, in this case this method will return an empty string because w don't have an overlap. Or, it is possible that this new booking is cancelled. Again, this method is going to assume that there is no overlap. So these are all test cases we need to write for this method. So back in our solution in our unit testing project, inside the Mocking folder, let's add a new class called BookingHelperTests.

And here we apply the test fixture attribute. And write our first unit test. Test so the method we are testing is Overlapping BookingsExist. First scenario is where our new booking starts and finishes before an existing booking, so, BookingStartsAndFinishesBefore AnExistingBooking.

And we expect this method to return an empty string. Because there is no overlap. So, ReturnEmptyString. Now, we can see that the name of this method is a little too long, and this is not necessarily a bad thing because our method name is very descriptive, it's telling us exactly what we are testing, however, even though I have reduced the font size the method name still doesn't completely fit on the screen. So here, to make it easier for you to see all these test methods I'm going to take the name of the method here, and put it in the class. So booking helper, underline OverlappingBookingsExistTests.

And this way we can make these methods a little bit shorter. That's better. So now, back in our BookingHelper class, let's expand this. So, we have this concrete dependency, and we need to get rid of it. So, in the next lecture, I'm going to extract this code into a separate class, and isolate it from our BookingHelper class.