So, in the last section, you saw a glimpse of how you should name and structure your tests.

Let's quickly review the convention you should follow to organize your tests so you can easily locate them. For each project in your solution

you'll have a unit testing project. So, if you have a project called TestNinja, you should have a separate project called TestNinja.UnitTests. And as I told you before, you want to separate unit and integration tests, because unit test execute fast, integration tests take longer. So we want to run unit tests frequently as we're writing code, and run integration tests just before committing our code to the repository to make sure everything works. So, each project in the solution, you're going to have a separate unit testing project.

In this project, you often have a test class for each class in your production code. So if you have a class called reservation, you should have a class called reservation tests. Note the plural name here. This indicates that this class has multiple tests. Now, for each method in the reservation class, you should have one or more test methods.

How many tests do you need? It depends on what you're testing. Often, the number of tests is equal to or greater than the number of execution paths. Now the name of your test methods, should clearly specify the business rule you're testing, here are some core names. Test 1, SaveCustomerTest, GetMovie, look we can tell what business rule is being tested here. That's why we have a convention for naming our tests. The name of the method on our test is scenario testing and expected behavior. With this convention, we can look at the name of a test method, and tell what business rule is being tested.

Now, sometimes you're dealing with a large complex method with so many execution paths and edge cases. In that case, it may be better to dedicate a separate test class for that method. Because otherwise the test for this method may collude your test class. Let's say in our example, CanBeCancelled by method requires several tests. You can extract all the tests for this method, into a separate class and call it reservation, underline CanBeCancelledByTests. This way you can easily locate the tests for this method.