Mocks and stubs

A test double is an object that can stand in for a real object in a test, similar to how a stunt double stands in for an actor in a movie. The most common test doubles are mock and stubs.

Mocks and stubs are both dummy implementations of objects the code under test interacts with (they are types of 'test doubles')

In plain words it means that you replace a real thing (e.g. a class, module or function) with a fake version of that thing. The fake version looks and acts like the real thing (answers to the same method calls) but answers with canned responses that you define yourself at the beginning of your unit test.

The same object may be replaced with a stub in one test and a mock in another depending on the intent of the test.

Stubs can be thought of as inputs to the code under test, that hold predefined data. When called they behave a certain way – return a fixed value, throw an exception, calculate a return value based on parameters, pull from a sequence of values, etc. For example if your code queries a database using a DAO you might supply a stub DAO and have tests that verify the behaviour when your code finds one result, multiple results, zero results or when the query fails with an exception.

Mocks are objects that register calls they receive. In test assertion we can verify on Mocks that all expected actions were performed: A mock has expectations about the way it should be called, and a test should fail if it’s not called that way. We use mocks when we don’t want to invoke production code or when there is no easy way to verify that intended code was executed. Mocks are used to test interactions between objects, and are useful in cases where there are no other visible state changes or return results that you can verify e.g. an email service - We don’t want to send e-mails each time we run a test. One thing we can do though is mock the service and verify that it has been called. The mock records the interactions, and you can then examine these to ensure that the mock was correctly called and would of fired off the required functionality if it were the real-life Prod service.

Using test doubles is not specific to unit testing. More elaborate test doubles can be used to simulate entire parts of your system in a controlled way. However, in unit testing you're most likely to encounter a lot of mocks and stubs (depending of whether you're the sociable or solitary kind of developer), simply because lots of modern languages and libraries make it easy and comfortable to set up mocks and stubs.

Command Query Separation

Methods that return some result and do not change the state of the system, are called Query. When we have a method that performs some actions that changes the system state, but we don’t expect any return value from it, we call it Command. This creates a Separation of Concerns

Example

When you call a function getAmount(), you expect it to just return the amount and not change state of the system. It would be horrific if it did. Similarly, when you call setAmount(), it will have side-effects and you expect it to change the state of the system. But what do you expect setAmount() to return? Probably nothing.