Test Doubles

Test Doubles are lightweight versions of the components that are necessary to test some class or system, usually designated the System Under Test (SUT)

Kind of like stunt doubles in movies.

→ Why create Test Doubles ?

- The real components are not finished yet.
- It is computationally expensive to construct the real components.
- The real components are owned by someone else
- You want to avoid side-effects on real components as a result of testing.
- Test doubles help prevent flakey Tests: tests that work sometimes but not all time.
- Test doubles allow finer-grained

observation of the SUT.

→ Test Doubles provide the ecosystem

Dummy Objects 'fill in' objects required by the system under test (SUT) that are otherwise irrevalant for the test.

Test Stubs provide 'dummy' input data sources used by the SUT.

Fake Objects (→ could be an internal memory) are lightweight implementations of 'heavyweight' processes like DB.

Provide Test Inputs

Mock Objects check indirect results produced by the SUT.
(near-synonym) : Spy Object.

Monitor Test Outputs

→ Making It Concrete : mockito

- Mockito is a popular framework for creating test doubles in Java.
- It integrates well with jUnit for constructing unit test cases.
- It supports all the different kinds of

test doubles.

- Goal is to be able to mock away dependencies for unit test and inspect interactions between SUT and mocks.

Review :

> Working with software in context can be difficult to test.

> Test doubles provide a mean of 'faking' the ecosystem.

> Mockito is a framework that makes constructing test doubles simple.

Ref: https://site.mockito.org/

- When providing test inputs for test doubles, we want to use dummy objects and test stubs.

- During testing, you find that the unit tests require the use of a

computationally extensive database, you the use a fake object as a test double.

Test Doubles for Output (Mocks & Spys)

Mock Objects check indirect results produced by the SUT.
(near-synonym) : Spy Object.

} Monitor Test Outputs

- Why do we mock?

– Testing is more than checking function outputs.

What if function under test returns void?

– We want to test interactions :

- Methods called in proper order
- Methods called / not called
- Proper parameters to method calls

— Mock objects allow us to observe interactions of fake objects.

— Spy objects allow us to observe interactions of real objects.

- Test doubles provide the ecosystem.
- Mocks / spies allows fine grain monitoring of the SUT.
