

TDD with JUnit

Test-Driven Development (TDD)

- TDD Life Cycle
- TDD vs DLP (Debug Later Programming)
- Why TDD is matter
- Unit Testing with F.I.R.S.T
- Code and Test Coverage
- Structure of good unit testing (GUT)

Unit Testing with JUnit

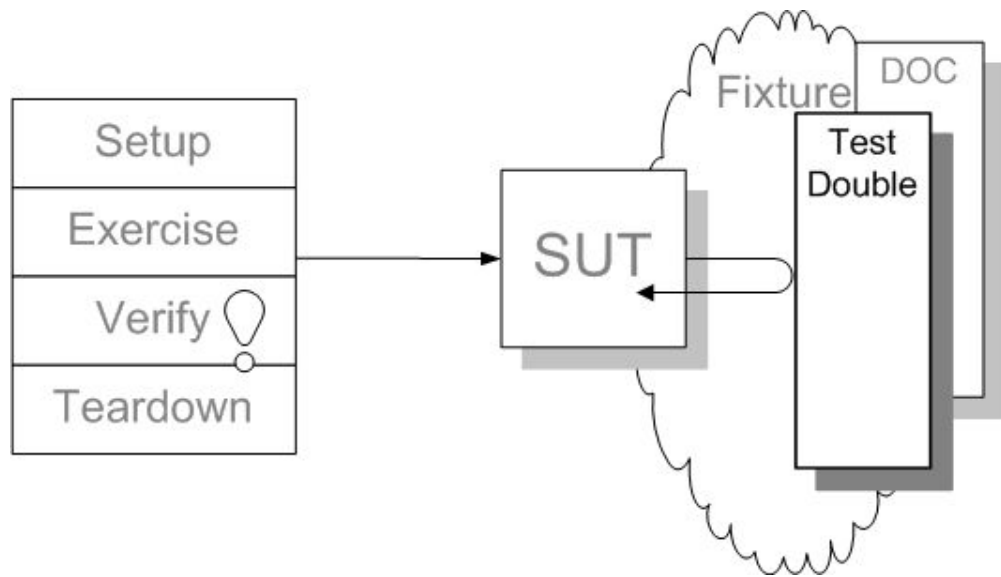
- JUnit lifeCycle
- Assertion
- Data-Driven Test with JUnit
- JUnit features
- Timeout
- Conditional
- **Category**
- **Suite**
- Running testing

Test Double

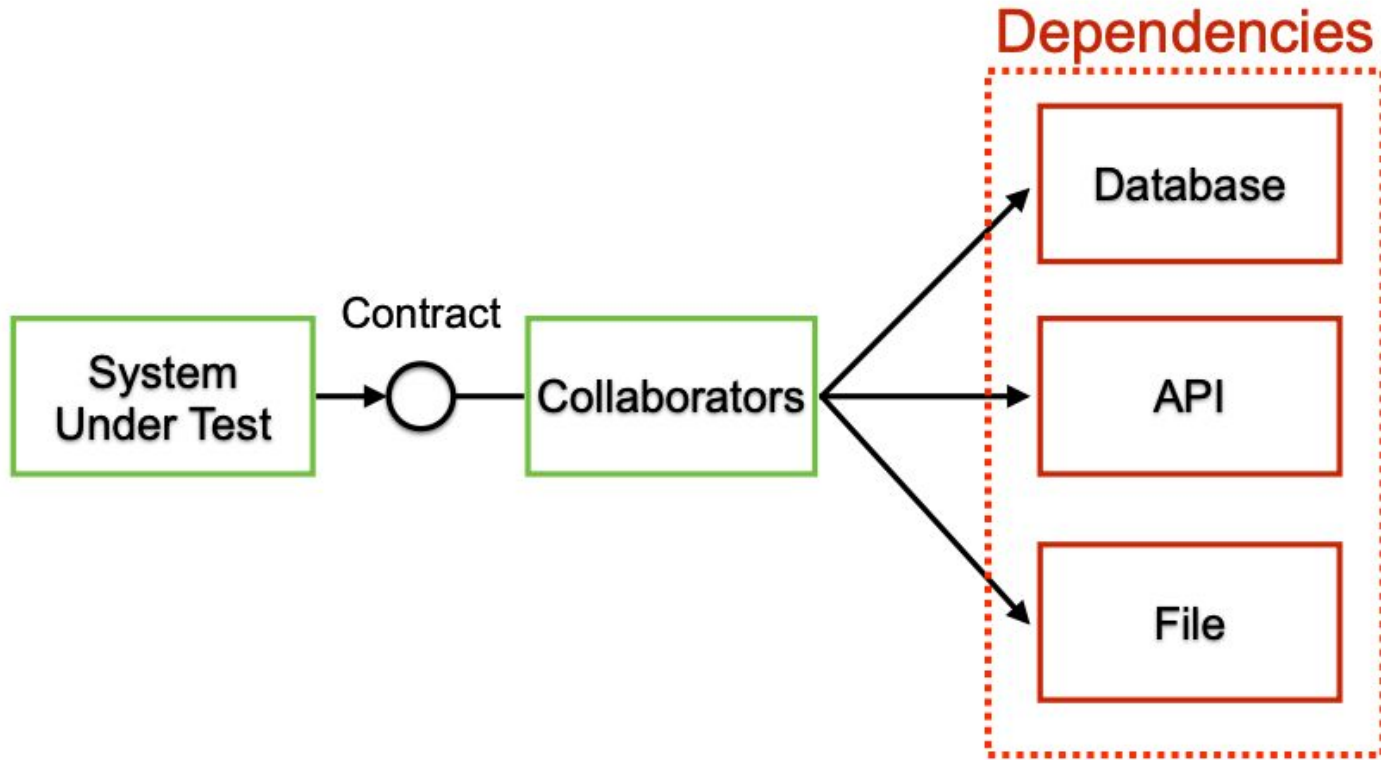
- **Dummy**
- **Stub**
- **Spy**
- **Mock**
- **Fake**

Test Double

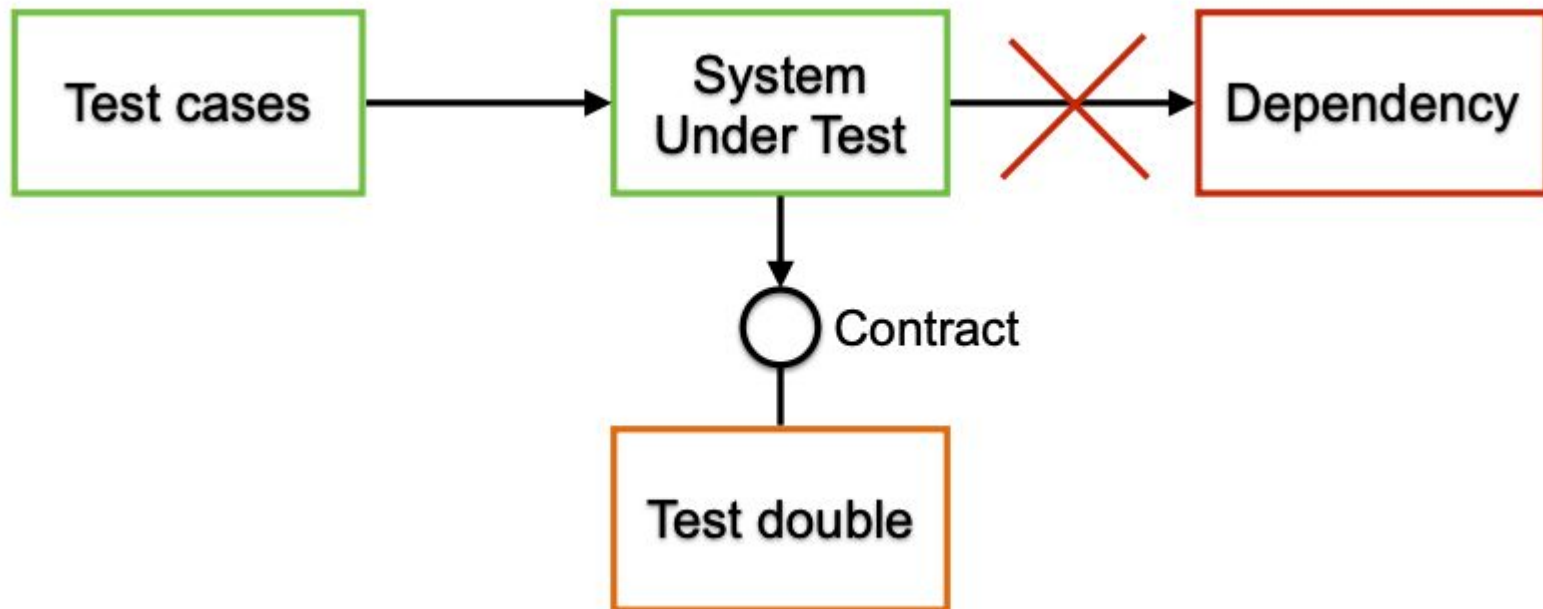
How can we verify logic independently ?
How can we avoid Slow tests ?



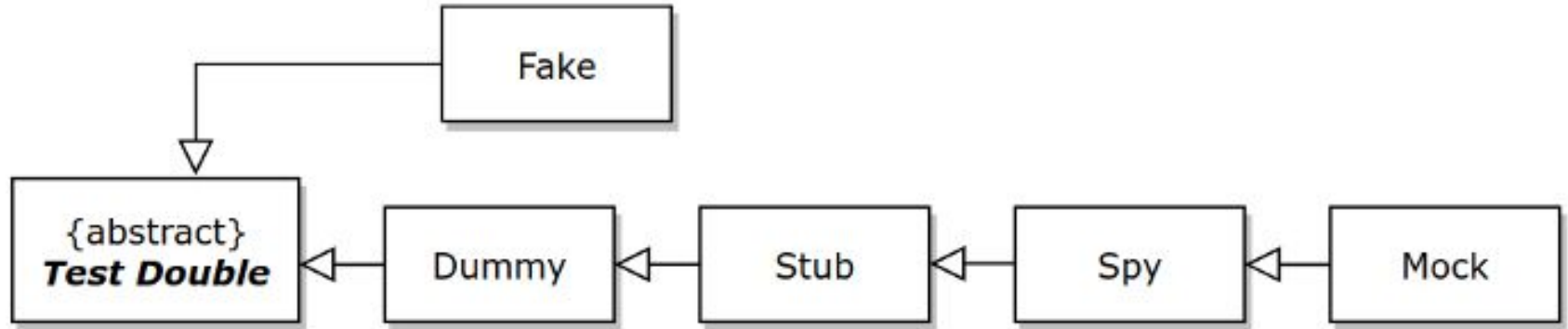
Production Code



With Test double



The five types of Test Doubles



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.

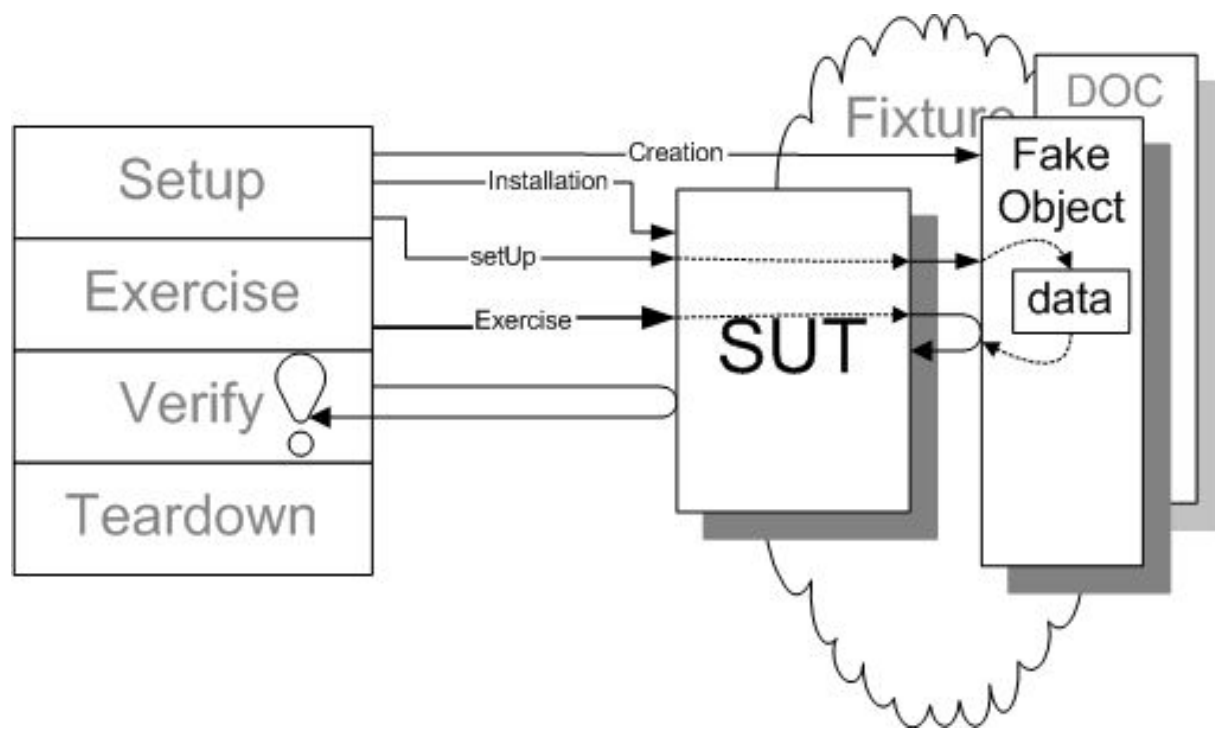
Fake

Working with implementation but take some shortcut

Not suitable for production

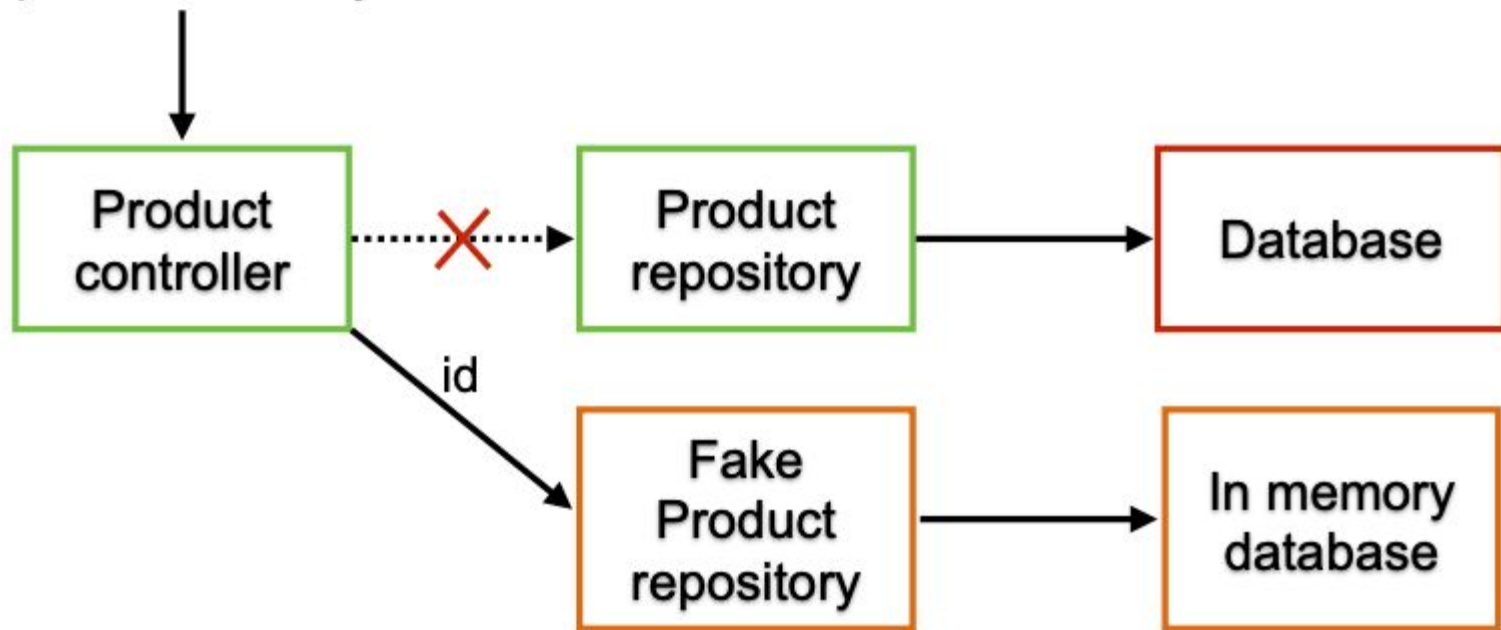
Use with read and write operations

E.g. In-memory database, Fake API server



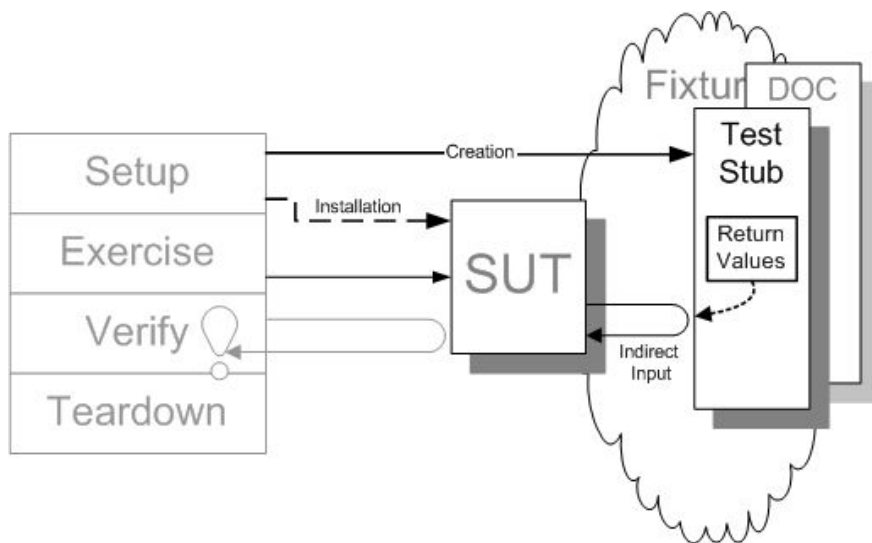
Fake

Get product detail by ID



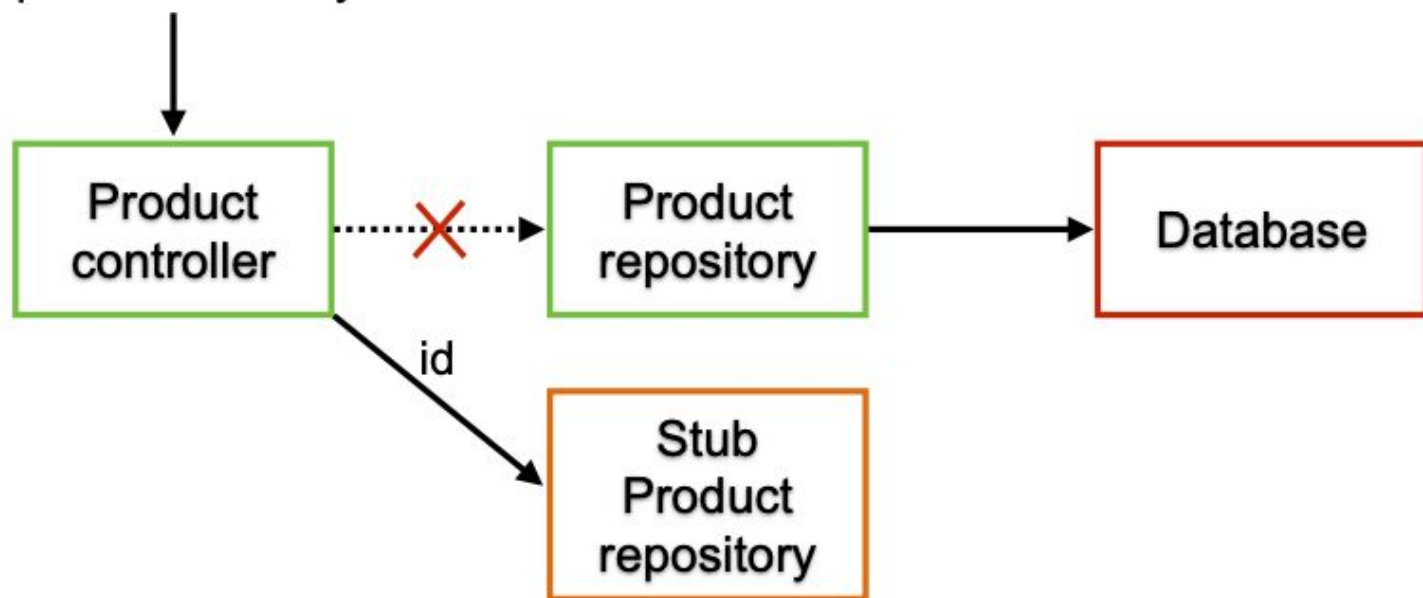
Stub

Provide answers to calls made during the test
A double with hardcode return values



Stub

Get product detail by ID

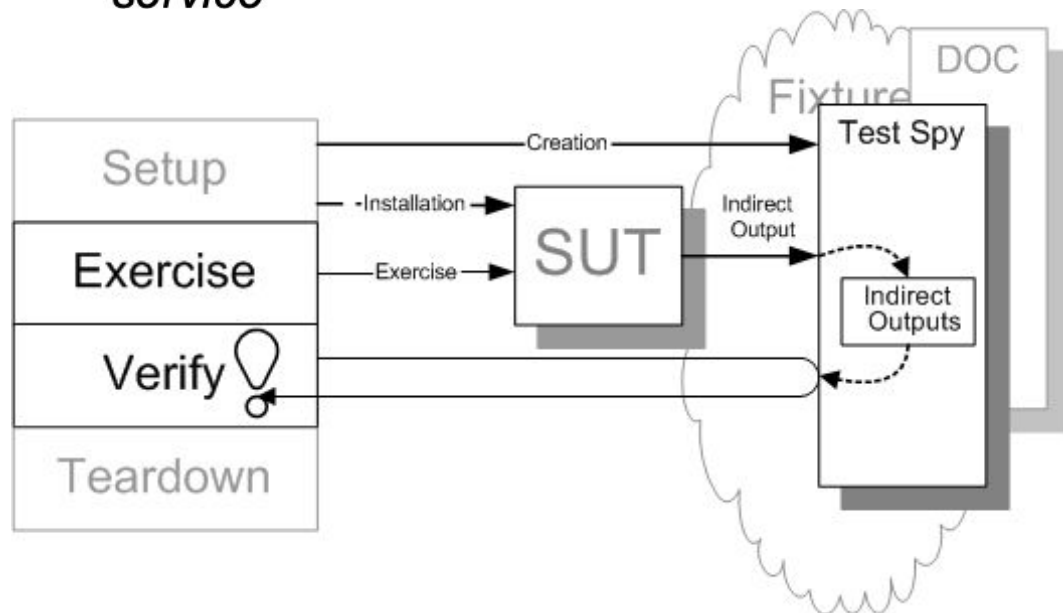


Spy

Like stub

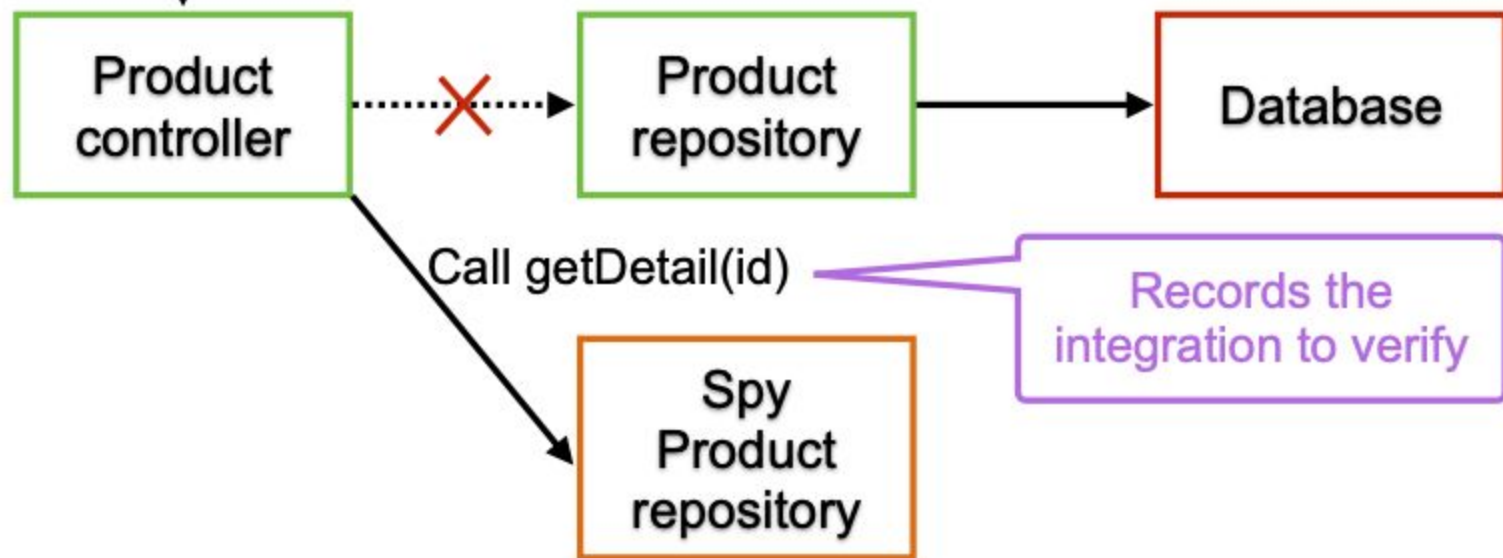
Record some information based on how its called

E.g. how many message it was sent via email service



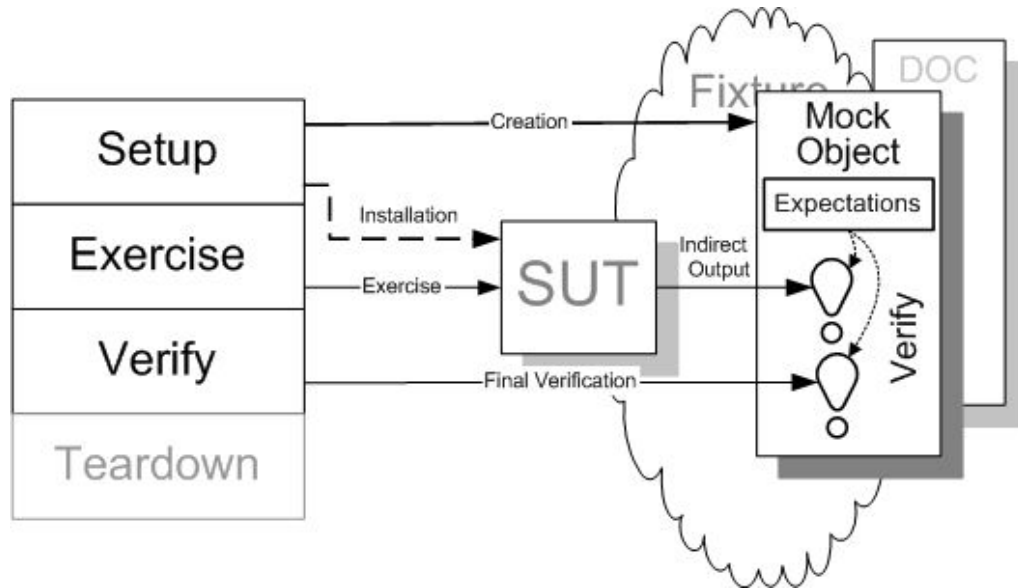
Spy

Get product detail by ID



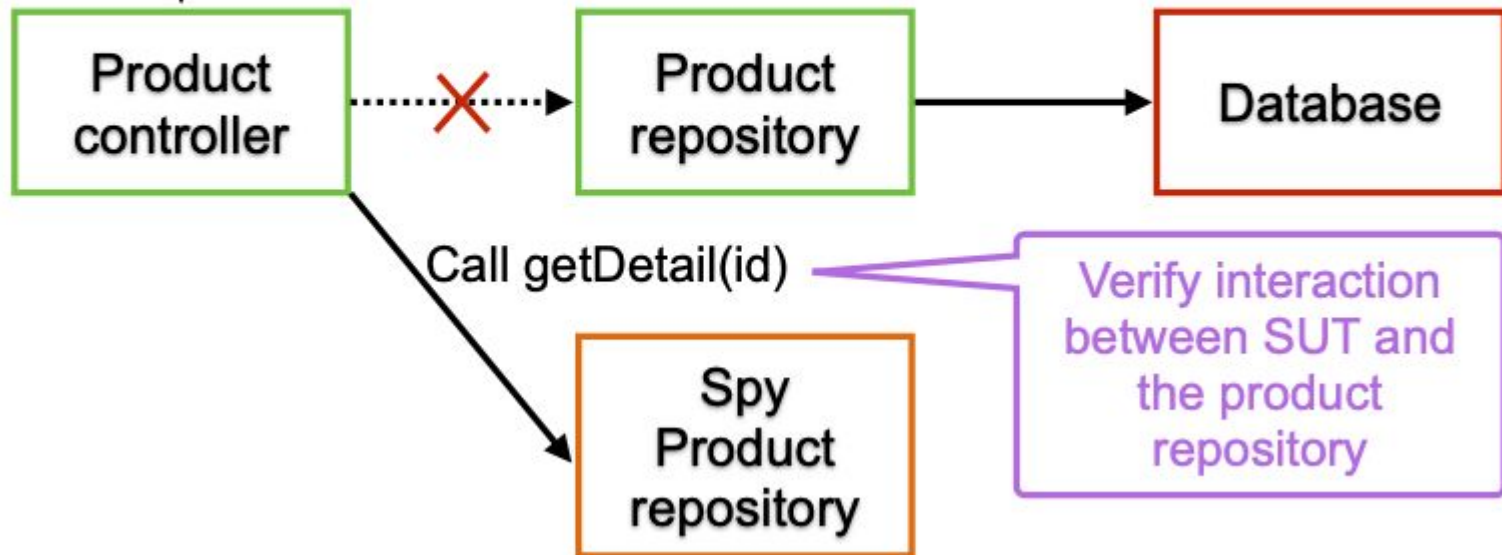
Mock object

Pre-programmed with expectations with spec
Mock object can throw an exception if receive a call
that don't expect



Mock object

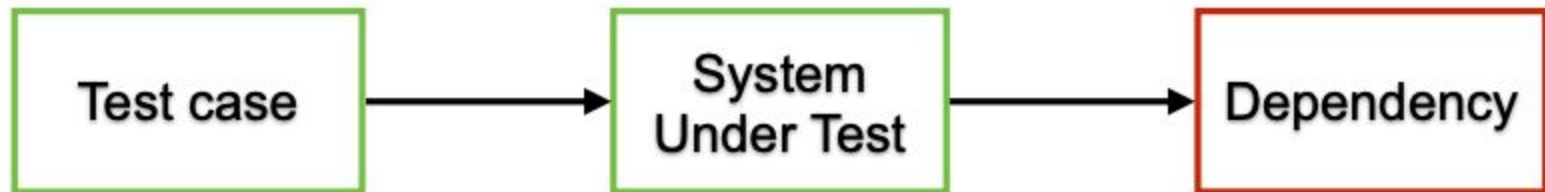
Get product detail by ID



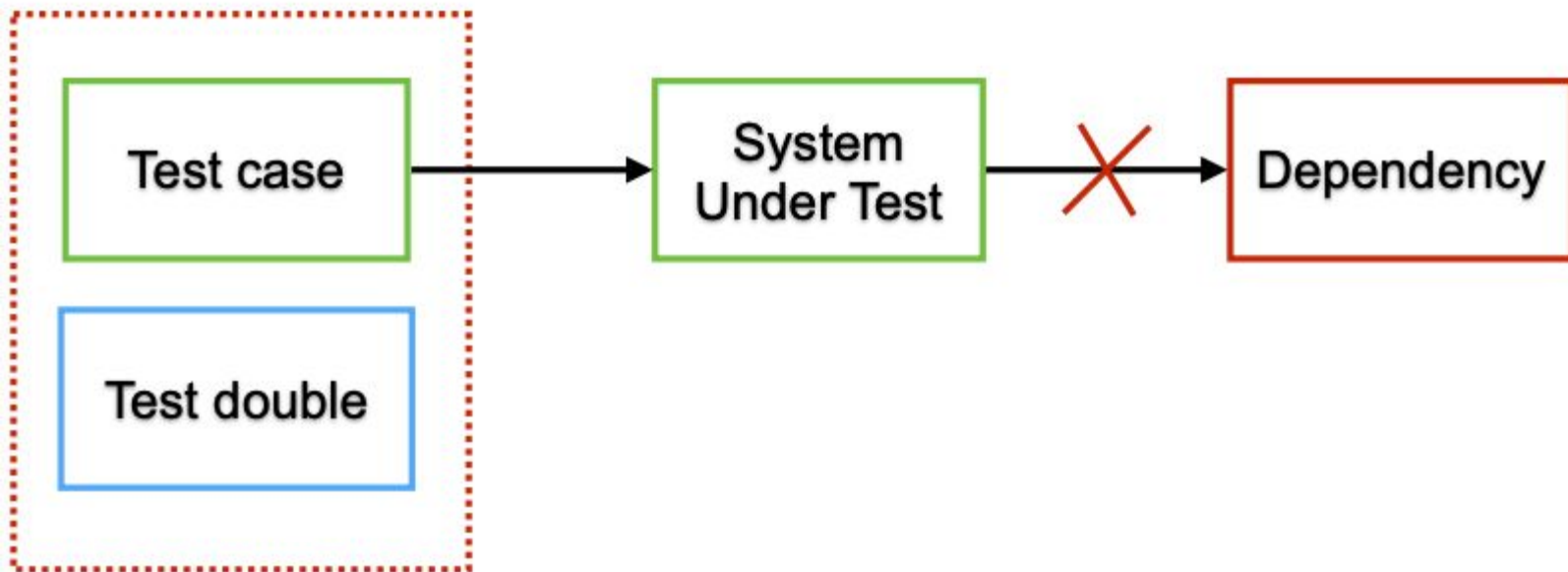
Dummy object

Passed around but never actually used
Used to fill parameter lists

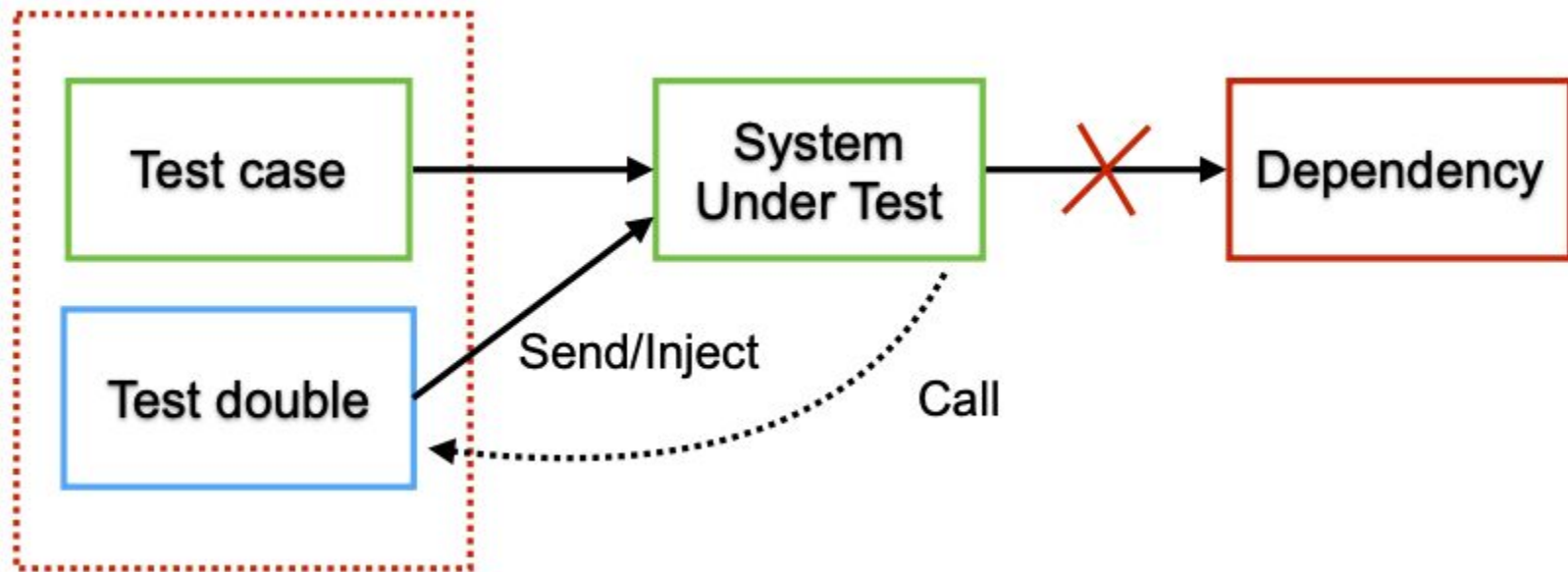
Test double ?



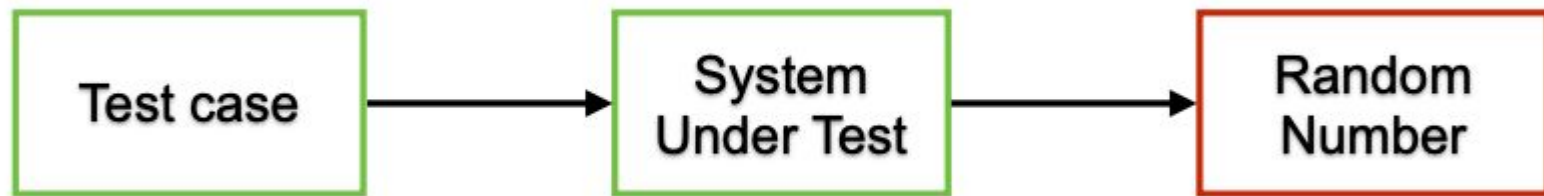
Create test double

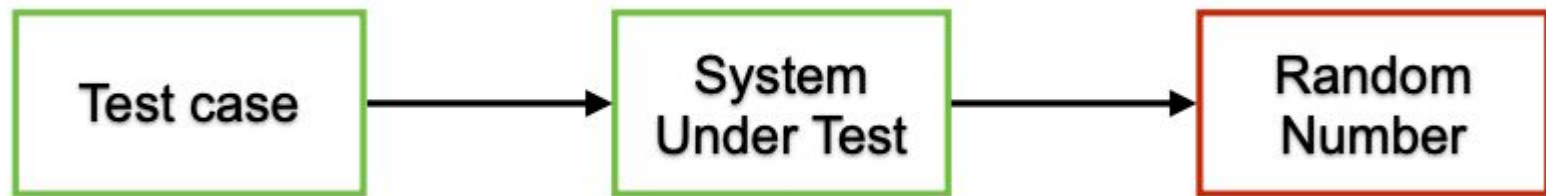


Send/inject test double to SUT



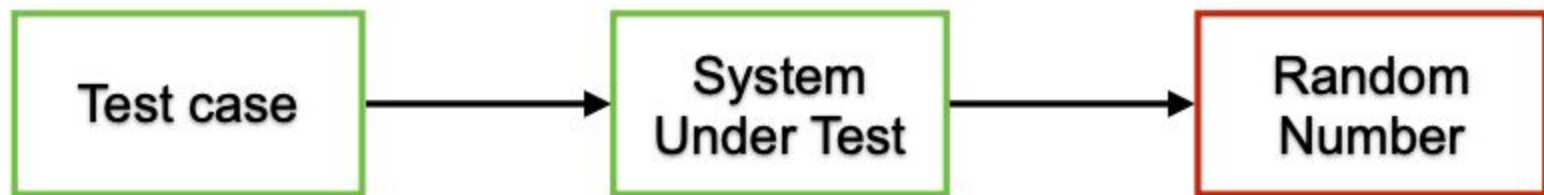
Workshop





Test random number = 5 ?

Workshop



Test random number must called 1 time ?

Mockito



Using Mockito with JUnit 5

```
@ExtendWith(MockitoExtension.class)
public class DemoWithMockito {

    @Mock
    Random random;

    @Test
    public void usingMockito() {
        // Create stub
        when(random.nextInt(10))
            .thenReturn(5);
    }
}
```


Test suite

Test suite is used to bundle a few unit test cases and run them together. In JUnit, both `@RunWith` and `@Suite` annotations are used to run the suite tests. This chapter takes an example having two test classes, `TestUnit1` & `TestUnit2`, that run together using `Test Suite`.

```
import org.junit.runner.RunWith;
import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses({
    TestUnit1.class,
    TestUnit2.class
})

public class JunitTestSuite {
}
```