Dealing with Team Dependencies in Tests

Diego Pacheco

About me...

- ❏ Cat's Father
- ❏ Head of Software Architecture
- ❏ Agile Coach
- ❏ SOA/Microservices Expert
- ❏ DevOps Practitioner
- ❏ Speaker
- ❏ Author

diegopacheco

@diego_pacheco

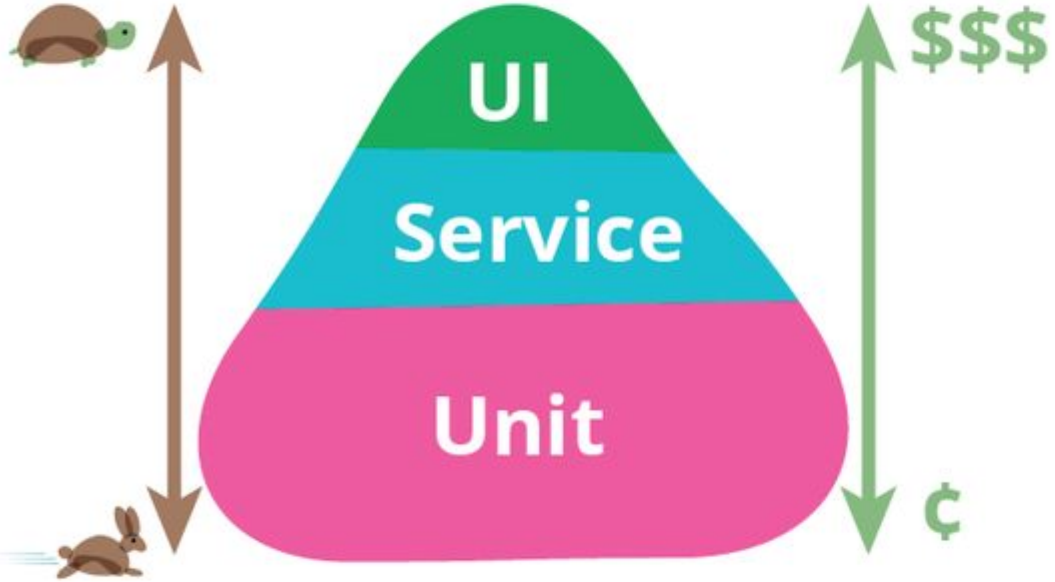
http://diego-pacheco.blogspot.com.br/

Diego Pacheco
Building Applications with Scala
Write modern, scalable, and reactive applications with the power of Scala
Packt>

Diego Pacheco
Building Effective Microservices
Explore microservices and their implementation hands-on
Packt>

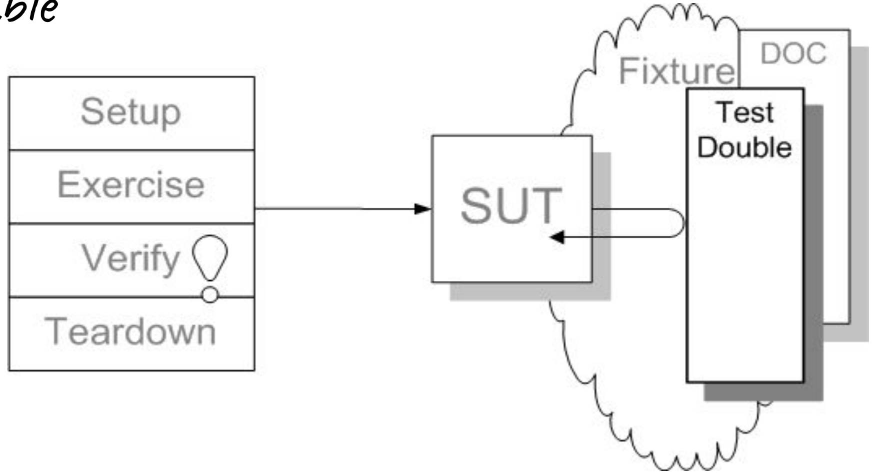
https://diegopacheco.github.io/

Going Down == Going Faster? Dependencies?

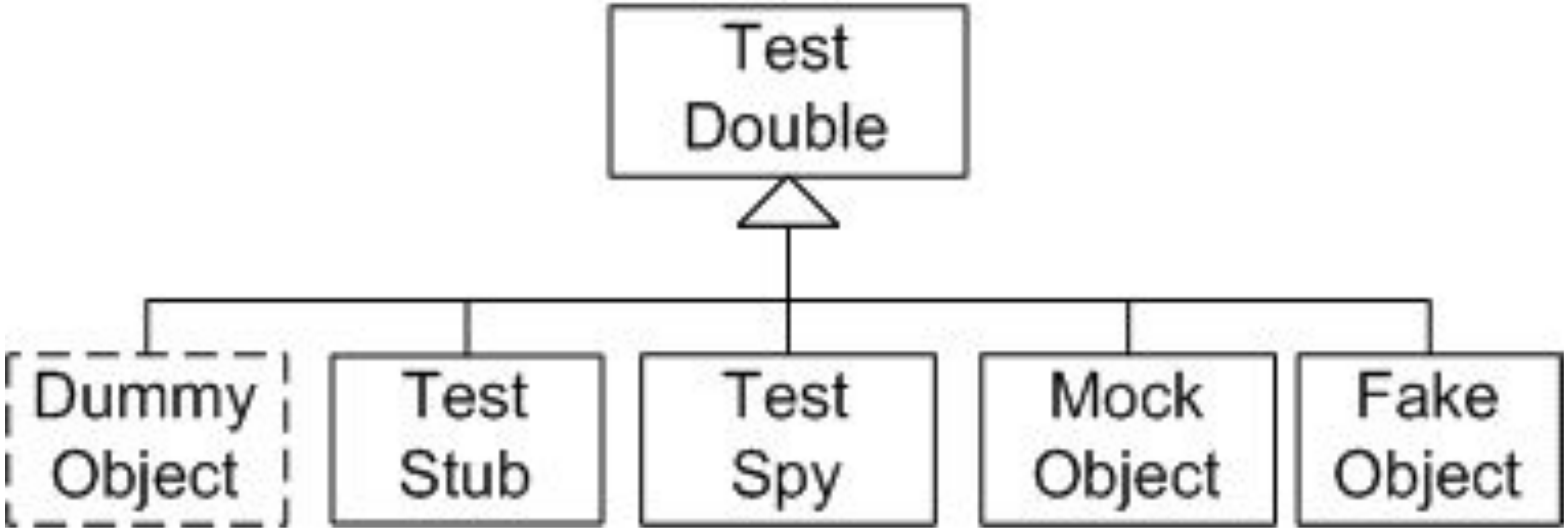


Test Doubles

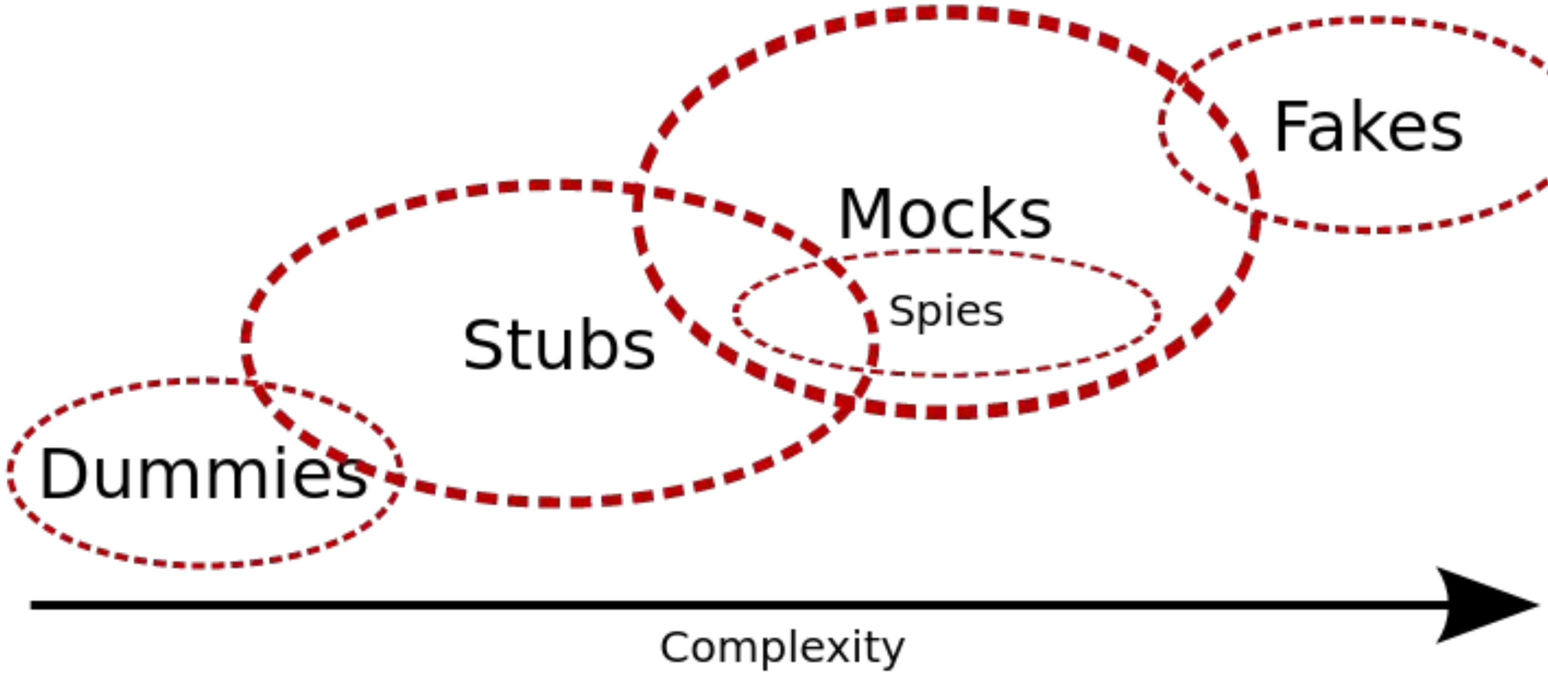
- Also known as "Impostors". Much before AmongUs!
- There are multiple Flavors: Fakes, Mocks, Spys, Stubs and Dummy.
- Motivations:
  - Verify code when dependency is unusable
  - Avoiding Slow Tests
- Wrongly: People often call all "Mocks".



Ontology

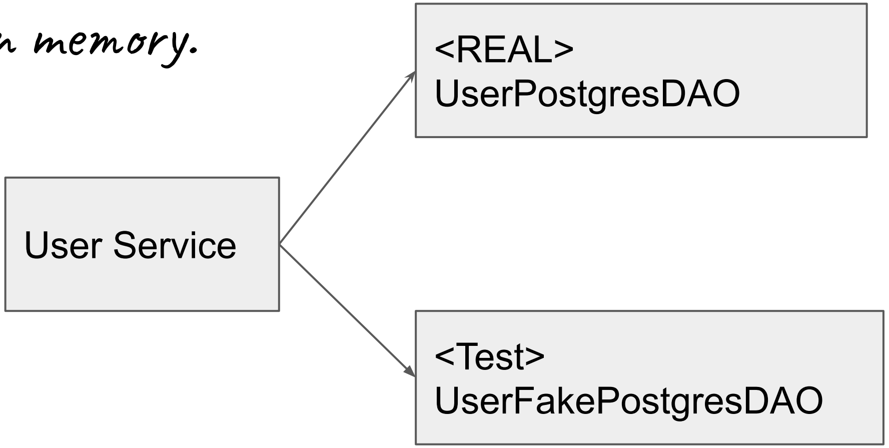

Test Double
Dummy Object
Test Stub
Test Spy
Mock Object
Fake Object

Dealing with dependencies is not a free lunch



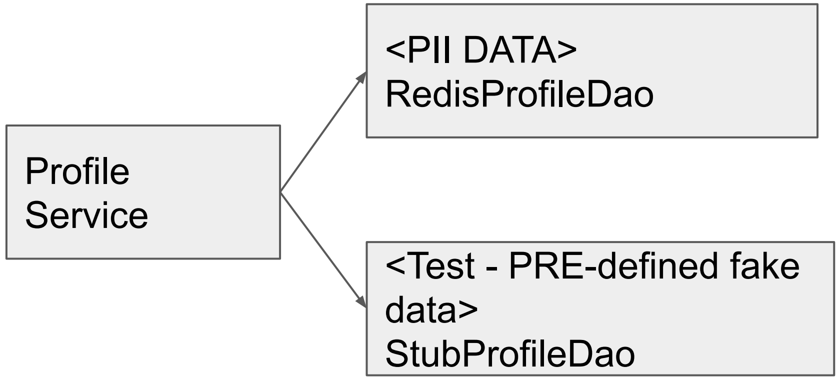
Fakes

- Objects with Working Implementations but not the same as the REAL production ones.
- Usually Dynamic. You can add data.
- Usually using gortcuts and simplified versions.
- I.e: Faking Redis with a HashMap in memory.
- I.e: Postgres DAO in memory.



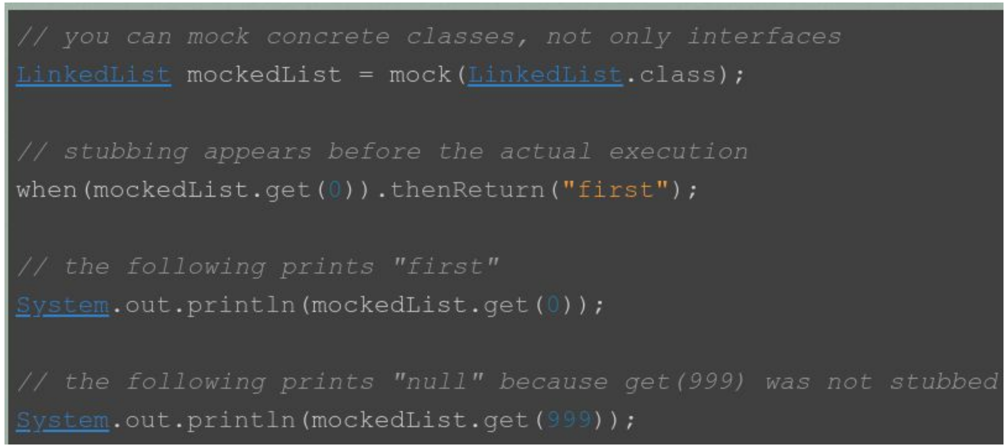
Stub

- ❏ Holds predefined Data and use it to answer questions on tests.
- ❏ Use it when we don't want we cannot use real data because side effects.
- ❏ Often used when the target code does not allow you to control input objects.
- ❏ Similar to FAKE but static. Fake is Dynamic.
- ❏ I.e: You have PII Data and don't want return real user data.
- ❏ I.e: You have to test a GET record but the api takes 7 days to activate it.


Profile Service
<PII DATA> RedisProfileDao
<Test - PRE-defined fake data> StubProfileDao

Mocks

- ❑ Register Calls they received.
- ❑ You can verify is specific actions happened. More complex than Stubs.
- ❑ Solutions: Mockitto, EasyMocks, PowerMocks.
- ❑ I.e: Return void and send emails. Want to verify that email service was called.



```
// you can mock concrete classes, not only interfaces
LinkedList mockedList = mock(LinkedList.class);

// stubbing appears before the actual execution
when(mockedList.get(0)).thenReturn("first");

// the following prints "first"
System.out.println(mockedList.get(0));

// the following prints "null" because get(999) was not stubbed
System.out.println(mockedList.get(999));
```

Spys

- ❏ It might be better change the design - think twice before doing it.
- ❏ Special kind of Mocks.
- ❏ Register some information on how they were called.
- ❏ Capture the target method to know is was called or not.
- ❏ i.e : A Mail service, how many messages were sent?
- ❏ I.e: Legacy code that might be impossible to test with Stubs

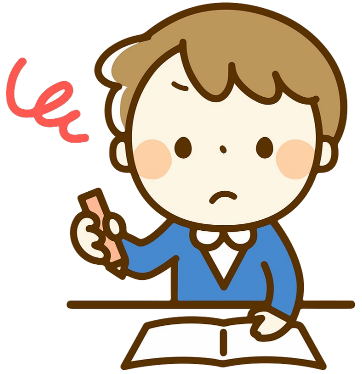
And mocks.

Dummy

- No Implementation. Simple to do it.
- Often used to satisfy an method argument need.
- Could be either Dummy values or Dummy Objects.
- I.e: You need pass a parameter that you don't care or is not useful for you test. Like a notification callback or customization /parametrization you don't care.

Isolation Matters!

- ❏ A good design - It should be easy to test right?
- ❏ Specially for shared libraries
- ❏ Hiding Implementation Details
- ❏ Less public classes
- ❏ Avoid Coupling with DI/IoC Libraries.
- ❏ Avoid coupling - Reduce surface, reduce customization.
- ❏ As Results - Harder to Test. Not necessary wrong.



Cross Cutting Concerns like i.e: AUTH.

- Functional and Nonfunctional requirements can be:
    - Cutting and Cross-cutting concerns.
- Even business rules can be cross-cutting concerns.
- AUTH often is implemented in a centralized fashion.
- But imagine is was implemented Component by component basis

Would you easily disable it in order to speed up tests? For sure you
Should be able to mock project by project but would that be
The right call?

Dealing with Team Dependencies in Tests

Diego Pacheco