Verifying That Mocked Methods Behave Accordingly

Nicolae Caprarescu
FULL-STACK SOFTWARE DEVELOPMENT CONSULTANT

www.properjava.com

Overview

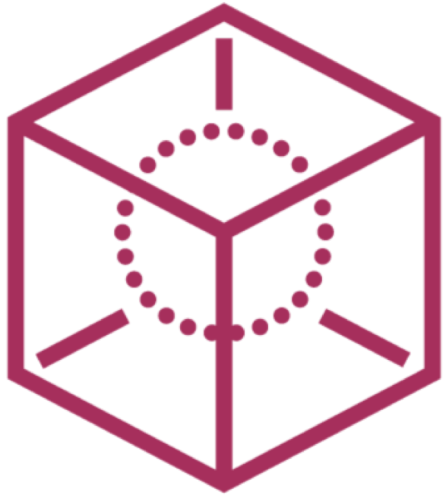
Testing state vs. testing behavior

Verifying mocks

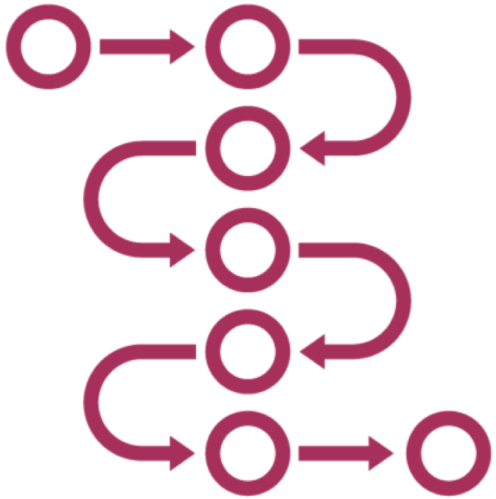
Using matchers for verification

Argument captors

Testing State vs. Testing Behavior



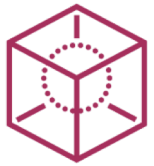
State-based testing



Behavior-based testing

Testing State

Verifying that the unit-under-test returns the correct result



You examine the state of the unit-under-test once the functionality has been exercised

Testing phases reminder: setup, exercise, verify, teardown

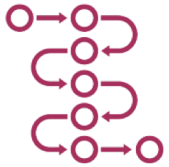
Classical, Detroit

```
Collaborator collab = new
Collaborator()

UnitUnderTest uut = new
UnitUnderTest(collab)


Result res = uut.exercise()

assertSomething(res)

teardown()
```

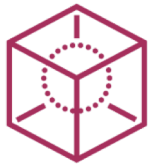
Typical State-testing Pseudocode

- Setup: non-mock collaborator object(s)
- Setup: unit-under-test
- Exercise
- Verify
- Teardown: doesn't have to be explicit

Testing Behavior



Verifying that the unit-under-test calls certain methods correctly



The checks are also carried out at verification stage. Often used in combination with state-based testing

Mocks always use this

Mockist, London

```
Collaborator mockCollab =
mock(Collaborator.class)

mockCollab.expectation()


UnitUnderTest uut = new
UnitUnderTest(mockCollab)



Result res = uut.exercise()


verify(mockCollab)

teardown()
```

Typical Behavior-testing Pseudocode

- Setup: mock collaborator object(s)
- Setup: expectations
- Setup: unit-under-test
- Exercise
- Verify
- Teardown: doesn't have to be explicit

Demo

Let’s carry out some verification

Using matchers

Argument captors

Summary

- Compared state testing and behavior testing
- Added behavior verification to tests
- Used matchers for verification
- Argument captors