

Mock Objects

What are Mock Objects

- Any dummy object that stands in for a real object that is not available, or is difficult to use in a test case
- (More rigid): A mock object must have the ability to set up expectations and provide a self-validation mechanism

Why use Mock Objects?

- Help keep design decoupled
- Check code's usage of another object
- Test code from the inside out
- Make tests run faster
- Make it easier to write code for problematic resources
- Defer having to write a class
- Test components in isolation from the rest of the system
- Promote interface-based design
- Encourage composition over inheritance
- Refine interfaces
- Test unusual, unlikely, and exceptional situations

When to use Mock Objects

- Non-deterministic or random behavior
 - Difficult setup
 - Behavior hard to cause
 - Slow real objects
 - UIs
- Test needs to make queries not supported by real object
 - Non-existent real objects

How to use Mock Objects

- Create them by hand from scratch
- Create them by hand, using the MockObjects library
- Use MockMaker to generate the mock object prior to execution
- Use EasyMock to generate the mock object at runtime

By hand from scratch

- Create a class that implements the same interface as the real one
- Hard code return values
- Lots of work to create and validate expectations

By hand with MockObjects library

- Create the mocks
- Set state/expectations
- Execute code
- Have mocks verify expectations
- Lots of duplication across Mocks

MockMaker

- Given an interface or class, create a mock object for it
- Mock can then be tweaked

EasyMock

- Create mock object at runtime
- Two modes: record and replay

EasyMock adjustments

- Make the following adjustments to EasyMock example code in the book:
 - `EasyMock.niceControlFor` should be `MockControl.createControl`
 - `control.activate` should be `control.replay`

EasyMock Matcher

- Easymock by default uses `Object.equals()` type comparison
- For arrays, this doesn't look at content
- Solution: the first time you call a method that takes an array as a parameter, set the matcher before setting return value