Intro to Testing Patterns & Signal-to-Noise Ratio

Sean Olszewski
@__chefski__

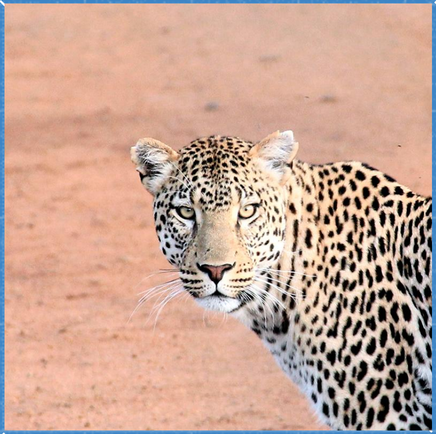
August 2018

Hello thar!

I AM SEAN OLSZEWSKI

- Care about designing effective test suites
- Practice test-driven development daily @ work (Pivotal Labs)
- Really TDD all the code I write

@__chefski__
github.com/seanrolszewski



0

Defining Signal-to-Noise Ratio

What's this thing all about?

Signal-to-noise ratio (SNR) is a measure that compares the level of a desired signal to the level of undesired noise.

SNR is sometimes used metaphorically to refer to the ratio of useful information to false or irrelevant data in a conversation or exchange.

SNR is a measure of how clearly a test failure indicates a specific fault or issue within your code base.

SNR is a measure that is applicable to an <u>entire test suite</u> and an <u>individual test</u>.

Various kinds of tests influence it differently.

Noise is anything that obscures the origin of a test failure.

Noise is subjective.

EXAMPLES OF NOISE

- Overly verbose/cluttered test suite logs
- Poorly chosen matchers/assertions
- Cascading test failures
- An unclear test subject

1

Benefits of Measuring Signal-to-Noise Ratio

Why should I care about this?

4

MAIN REASONS

WHY CARE ABOUT SNR?

Ease debugging by reducing search space

WHY CARE ABOUT SNR?

Gauge the balance of your test suite

WHY CARE ABOUT SNR?

Catch & prevent over-testing

WHY CARE ABOUT SNR?

Write tests that withstand & guide refactoring

WHY CARE ABOUT SNR?

- Ease debugging by reducing search space
- Gauge the balance of your test suite
- Catch & prevent over-testing
- Write tests that withstand & guide refactoring

2

Qualifying Signal-to-Noise Ratio

How do we measure SNR?

HOW DO WE QUALIFY SNR?

Low-signal	**Moderate-signal**	**High-signal**

HOW DO WE QUALIFY SNR?

Low-signal

More than one area of the code is the likely source of error.

HOW DO WE QUALIFY SNR?

Low-signal

More than one area of the code is the likely source of error.

The network and presentation layer need to be investigated.

High noise.

HOW DO WE QUALIFY SNR?

Moderate-signal

One area of the code is the likely source of error.

HOW DO WE QUALIFY SNR?

Moderate-signal

One area of the code is the likely source of error.

The network layer is the only thing that needs to be investigated.

Moderate noise.

HOW DO WE QUALIFY SNR?

High-signal

A very specific area of the code is the likely source of error.

HOW DO WE QUALIFY SNR?

High-signal

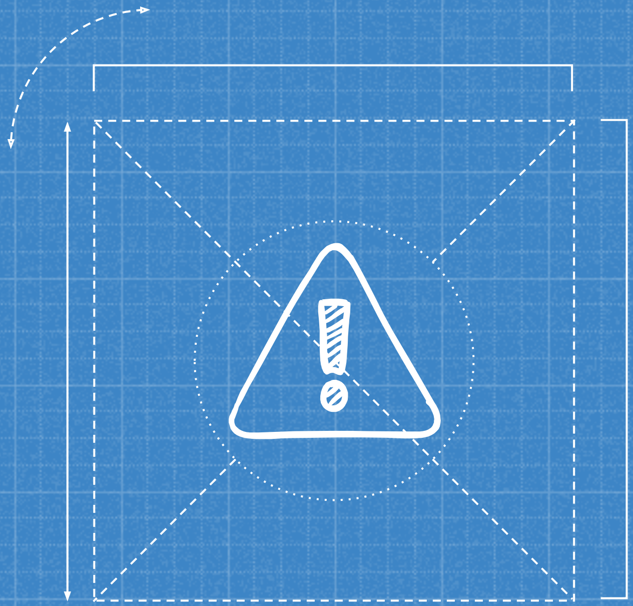
A very specific area of the code is the likely source of error.

One function call inside the *UrlBuilder* class needs investigation.

Low noise.

HOW DO WE QUALIFY SNR?

Low-signal	**Moderate-signal**	**High-signal**
More than one area of the code is the likely source of error.	One area of the code is the likely source of error.	A very specific area of the code is the likely source of error.
The network and presentation layer need to be investigated.	The network layer is the only thing that needs to be investigated.	One function call inside the *UrlBuilder* class needs investigation.
High noise.	Moderate noise.	Low noise.



BALANCE SIGNAL & NOISE

3

Techniques for Balancing Signal-to-Noise Ratio

How do we intentionally influence SNR?

2

TECHNIQUES

Test double usage

Testing pattern selection

TEST DOUBLE USAGE

A **test double** is a stand-in for something that would otherwise be real in the execution of your program.

Matt Parker, Pivotal Labs

There are 5 kinds of test doubles: dummies, spies, mocks, fakes, and stubs.

2

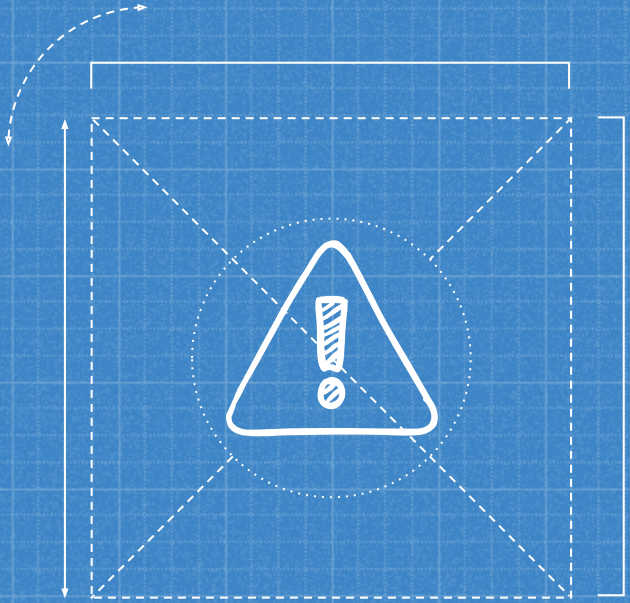
WAYS A TEST DOUBLE IMPROVES SNR

HOW TEST DOUBLES IMPROVE SNR

Isolate a test subject from its dependencies

HOW TEST DOUBLES IMPROVE SNR

Create **interaction points** for controlling and observing a test subject



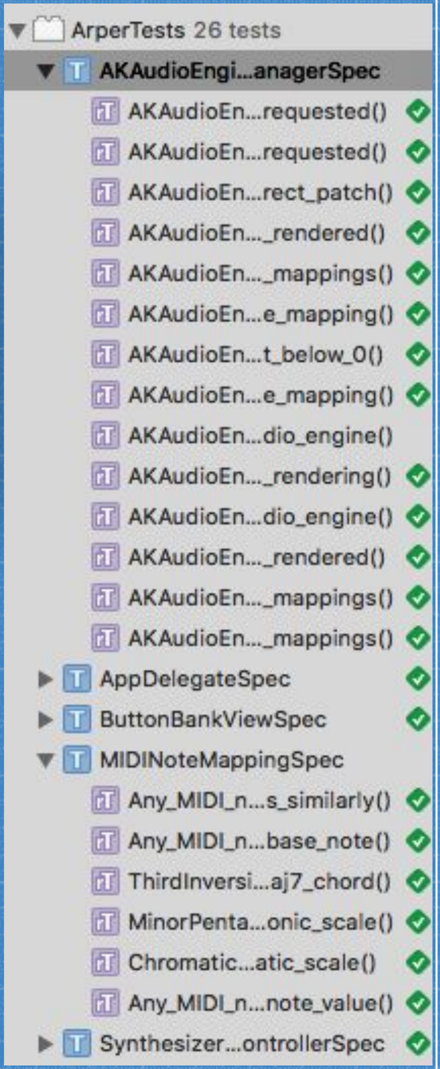
MINIMIZE TEST DOUBLE USAGE

EXAMPLE OF IMPROVING SNR WITH A TEST DOUBLE

BALANCING SNR

```swift
describe("AKAudioEngineManager") {
    beforeEach {
        audioEngineSpy = AudioEngineSpy()
        subject = AKAudioEngineManager(audioEngine: audioEngineSpy,
                                       noteMappings: [ChromaticNoteMapping(baseNote: 0)])
    }
    // 13 more tests are omitted
}
```

```swift
struct ChromaticNoteMapping: MIDINoteMapping {
    let intervals: [UInt8] = [1]
    //
}
```



```
▼ ArperTests 26 tests
  ▼ AKAudioEngi...anagerSpec
      AKAudioEn...requested() ✓
      AKAudioEn...requested() ✓
      AKAudioEn...rect_patch() ✓
      AKAudioEn..._rendered() ✓
      AKAudioEn..._mappings() ✓
      AKAudioEn...e_mapping() ✓
      AKAudioEn...t_below_0() ✓
      AKAudioEn...e_mapping() ✓
      AKAudioEn...dio_engine()
      AKAudioEn..._rendering() ✓
      AKAudioEn...dio_engine() ✓
      AKAudioEn..._rendered() ✓
      AKAudioEn..._mappings() ✓
      AKAudioEn..._mappings() ✓
  ▶ AppDelegateSpec ✓
  ▶ ButtonBankViewSpec ✓
  ▼ MIDINoteMappingSpec
      Any_MIDI_n...s_similarly() ✓
      Any_MIDI_n...base_note() ✓
      ThirdInversi...aj7_chord() ✓
      MinorPenta...onic_scale() ✓
      Chromatic...atic_scale() ✓
      Any_MIDI_n...note_value() ✓
  ▶ Synthesizer...ontrollerSpec ✓
```

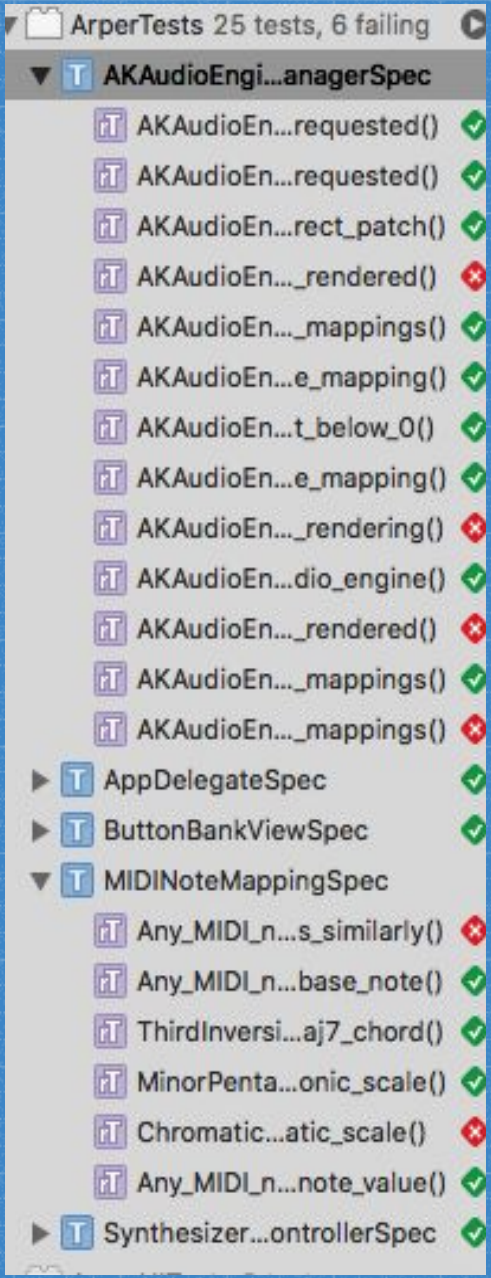
BALANCING SNR

BEFORE

```swift
struct ChromaticNoteMapping: MIDINoteMapping {
    let intervals: [UInt8] = [1]
    //
}
```

AFTER

```swift
struct ChromaticNoteMapping: MIDINoteMapping {
    let intervals: [UInt8] = [0]
    //
}
```



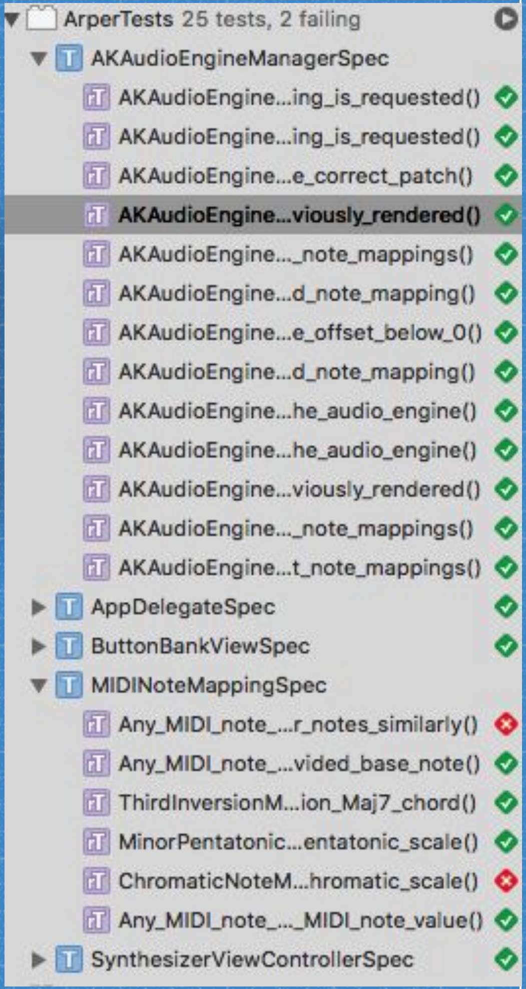
```
▼ ArperTests 25 tests, 6 failing
  ▼ AKAudioEngi…anagerSpec
      AKAudioEn…requested()        ✓
      AKAudioEn…requested()        ✓
      AKAudioEn…rect_patch()       ✓
      AKAudioEn…_rendered()        ✗
      AKAudioEn…_mappings()        ✓
      AKAudioEn…e_mapping()        ✓
      AKAudioEn…t_below_0()        ✓
      AKAudioEn…e_mapping()        ✓
      AKAudioEn…_rendering()       ✗
      AKAudioEn…dio_engine()       ✓
      AKAudioEn…_rendered()        ✗
      AKAudioEn…_mappings()        ✓
      AKAudioEn…_mappings()        ✗
  ▶ AppDelegateSpec                ✓
  ▶ ButtonBankViewSpec             ✓
  ▼ MIDINoteMappingSpec
      Any_MIDI_n…s_similarly()     ✗
      Any_MIDI_n…base_note()       ✓
      ThirdInversi…aj7_chord()     ✓
      MinorPenta…onic_scale()      ✓
      Chromatic…atic_scale()       ✗
      Any_MIDI_n…note_value()      ✓
  ▶ Synthesizer…ontrollerSpec      ✓
```

BALANCING SNR

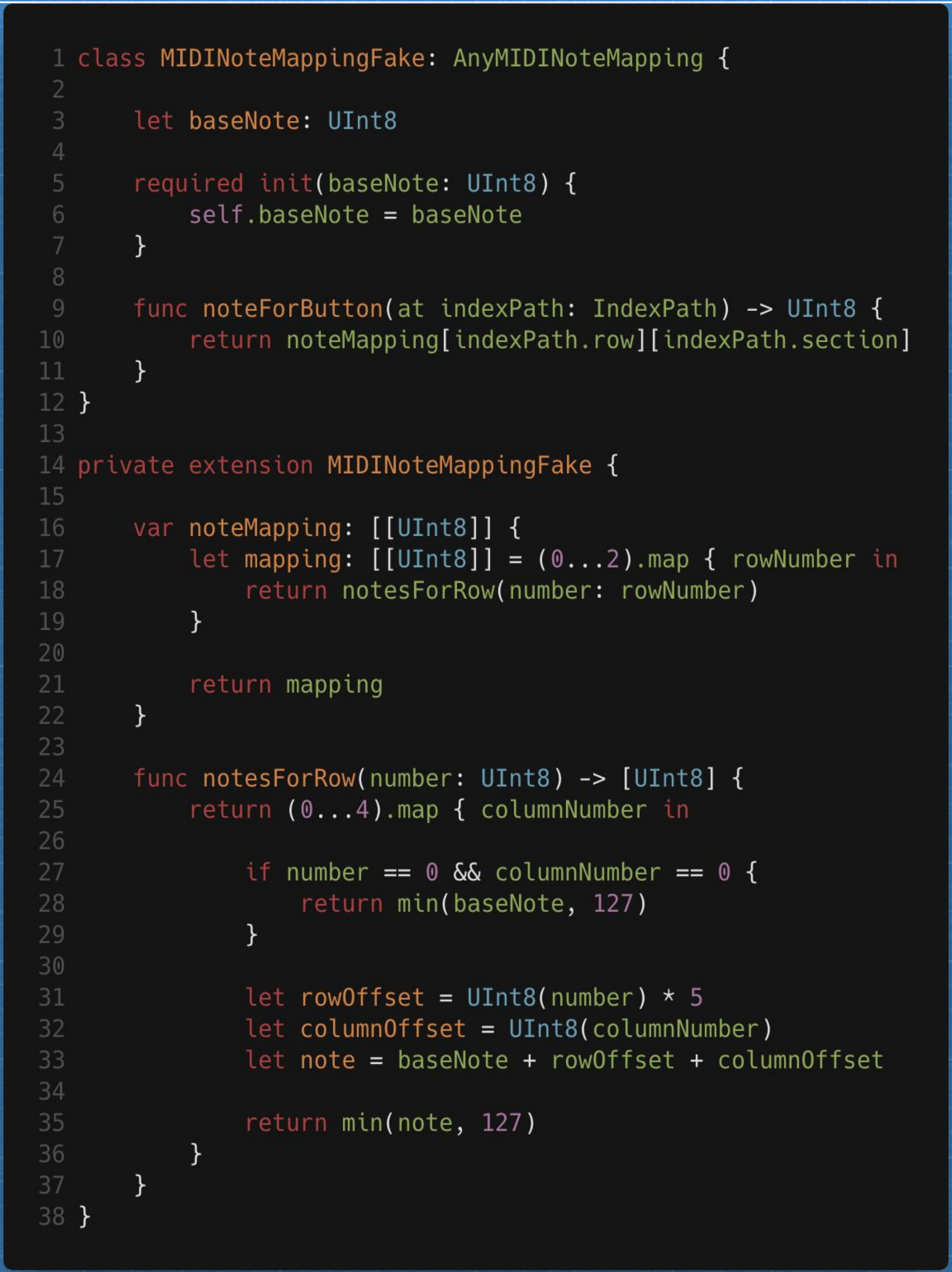
```swift
describe("AKAudioEngineManager") {
    beforeEach {
        audioEngineSpy = AudioEngineSpy()
        subject = AKAudioEngineManager(audioEngine: audioEngineSpy,
                                       noteMappings: [MIDINoteMappingFake(baseNote: 0)])
    }
}
```

SAME BREAKING CHANGE

```swift
struct ChromaticNoteMapping: MIDINoteMapping {
    let intervals: [UInt8] = [0]
    //
}
```



ArperTests 25 tests, 2 failing
- AKAudioEngineManagerSpec
  - AKAudioEngine...ing_is_requested() ✓
  - AKAudioEngine...ing_is_requested() ✓
  - AKAudioEngine...e_correct_patch() ✓
  - AKAudioEngine...viously_rendered() ✓
  - AKAudioEngine..._note_mappings() ✓
  - AKAudioEngine...d_note_mapping() ✓
  - AKAudioEngine...e_offset_below_0() ✓
  - AKAudioEngine...d_note_mapping() ✓
  - AKAudioEngine...he_audio_engine() ✓
  - AKAudioEngine...he_audio_engine() ✓
  - AKAudioEngine...viously_rendered() ✓
  - AKAudioEngine..._note_mappings() ✓
  - AKAudioEngine...t_note_mappings() ✓
- AppDelegateSpec ✓
- ButtonBankViewSpec ✓
- MIDINoteMappingSpec
  - Any_MIDI_note_...r_notes_similarly() ✗
  - Any_MIDI_note_...vided_base_note() ✓
  - ThirdInversionM...ion_Maj7_chord() ✓
  - MinorPentatonic...entatonic_scale() ✓
  - ChromaticNoteM...hromatic_scale() ✗
  - Any_MIDI_note_..._MIDI_note_value() ✓
- SynthesizerViewControllerSpec ✓



```swift
class MIDINoteMappingFake: AnyMIDINoteMapping {

    let baseNote: UInt8

    required init(baseNote: UInt8) {
        self.baseNote = baseNote
    }

    func noteForButton(at indexPath: IndexPath) -> UInt8 {
        return noteMapping[indexPath.row][indexPath.section]
    }
}

private extension MIDINoteMappingFake {

    var noteMapping: [[UInt8]] {
        let mapping: [[UInt8]] = (0...2).map { rowNumber in
            return notesForRow(number: rowNumber)
        }

        return mapping
    }

    func notesForRow(number: UInt8) -> [UInt8] {
        return (0...4).map { columnNumber in

            if number == 0 && columnNumber == 0 {
                return min(baseNote, 127)
            }

            let rowOffset = UInt8(number) * 5
            let columnOffset = UInt8(columnNumber)
            let note = baseNote + rowOffset + columnOffset

            return min(note, 127)
        }
    }
}
```

TESTING PATTERN SELECTION

CATEGORIES OF TESTING PATTERNS

Unit Tests

Integration Tests

CATEGORIES OF TESTING PATTERNS

Unit Tests

Usually very isolated

Integration Tests

Not isolated

CATEGORIES OF TESTING PATTERNS

Unit Tests	**Integration Tests**
Usually very isolated	Not isolated
Usually unaffected by cascading failures	Affected by cascading failures

CATEGORIES OF TESTING PATTERNS

Unit Tests	**Integration Tests**
Usually very isolated	Not isolated
Usually unaffected by cascading failures	Affected by cascading failures
White, gray, or black box test	Black box test

CATEGORIES OF TESTING PATTERNS

Unit Tests	**Integration Tests**
Usually very isolated	Not isolated
Usually unaffected by cascading failures	Affected by cascading failures
White, gray, or black box test	Black box test
Moderate to high signal	Low to moderate signal

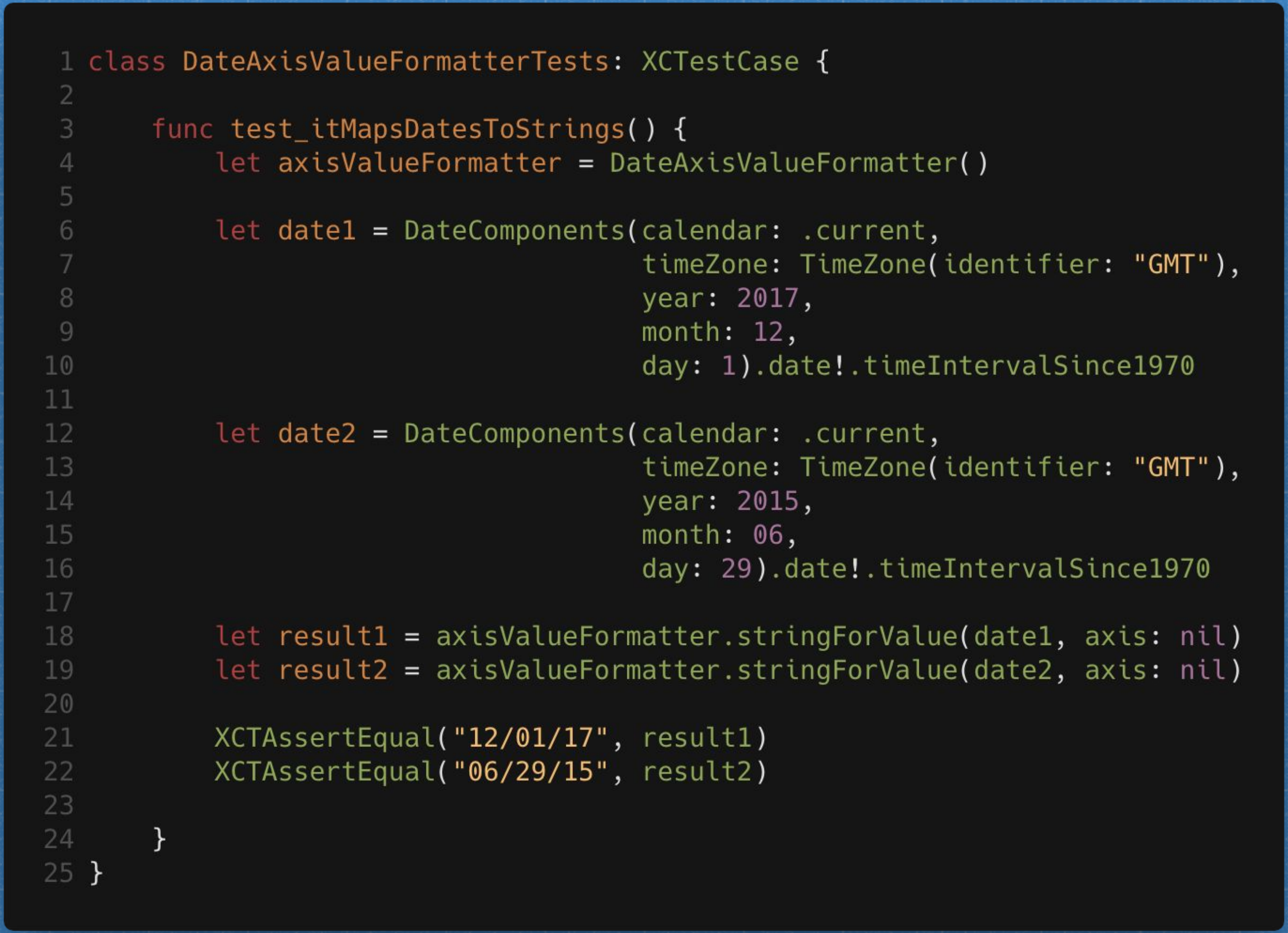
CATEGORIES OF TESTING PATTERNS

Unit Tests	**Integration Tests**
Usually very isolated	Not isolated
Usually unaffected by cascading failures	Affected by cascading failures
White, gray, or black box test	Black box test
Moderate to high signal	Low to moderate signal
Low to moderate noise	Moderate to high noise

EXAMPLES OF TESTING PATTERNS

Unit Tests	**Integration Tests**
Blackbox tests	Contract tests
Collaboration tests	UI tests

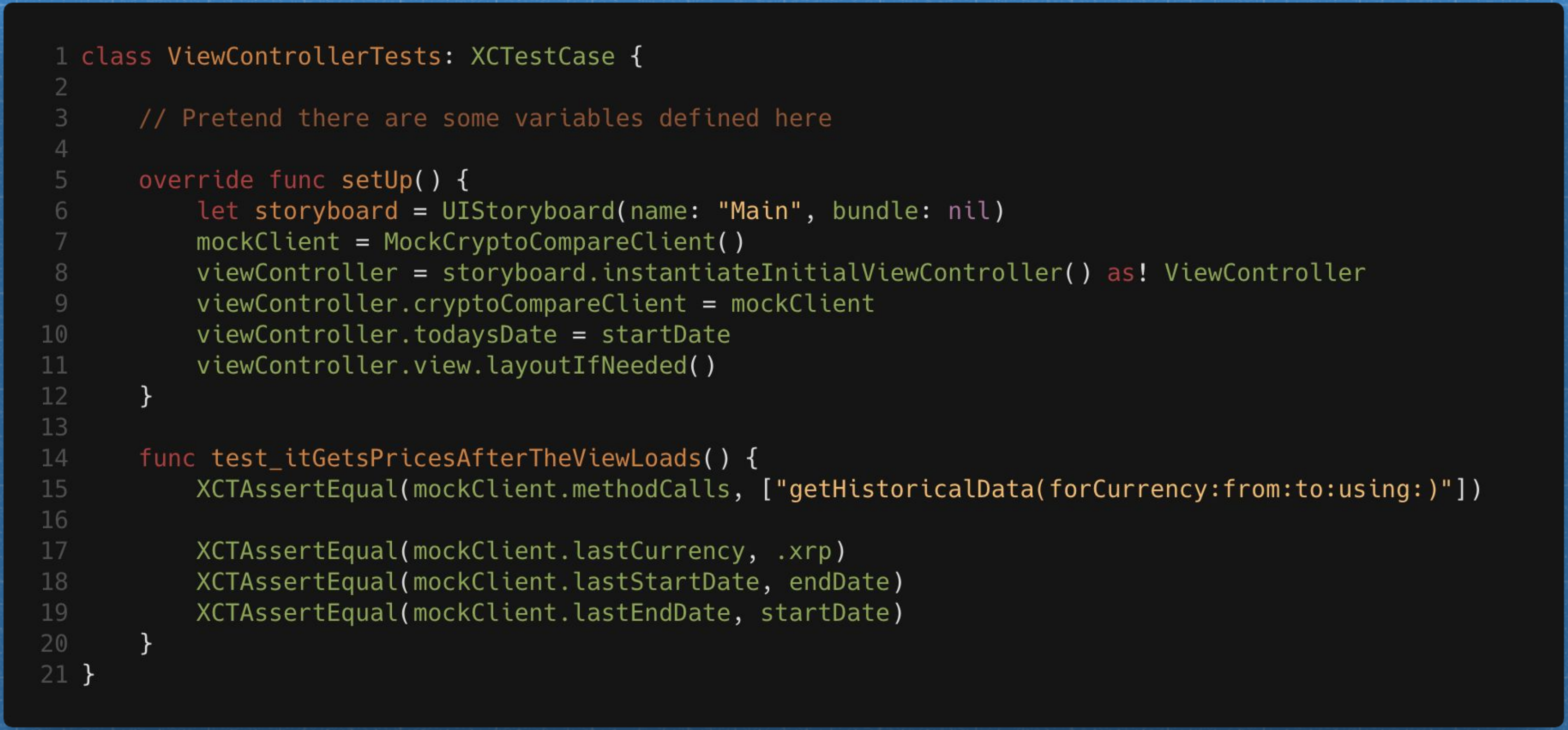
EXAMPLES OF TESTING PATTERNS - BLACKBOX TESTS



```swift
class DateAxisValueFormatterTests: XCTestCase {

    func test_itMapsDatesToStrings() {
        let axisValueFormatter = DateAxisValueFormatter()

        let date1 = DateComponents(calendar: .current,
                                   timeZone: TimeZone(identifier: "GMT"),
                                   year: 2017,
                                   month: 12,
                                   day: 1).date!.timeIntervalSince1970

        let date2 = DateComponents(calendar: .current,
                                   timeZone: TimeZone(identifier: "GMT"),
                                   year: 2015,
                                   month: 06,
                                   day: 29).date!.timeIntervalSince1970

        let result1 = axisValueFormatter.stringForValue(date1, axis: nil)
        let result2 = axisValueFormatter.stringForValue(date2, axis: nil)

        XCTAssertEqual("12/01/17", result1)
        XCTAssertEqual("06/29/15", result2)

    }
}
```

HIGH-SIGNAL, LOW-NOISE

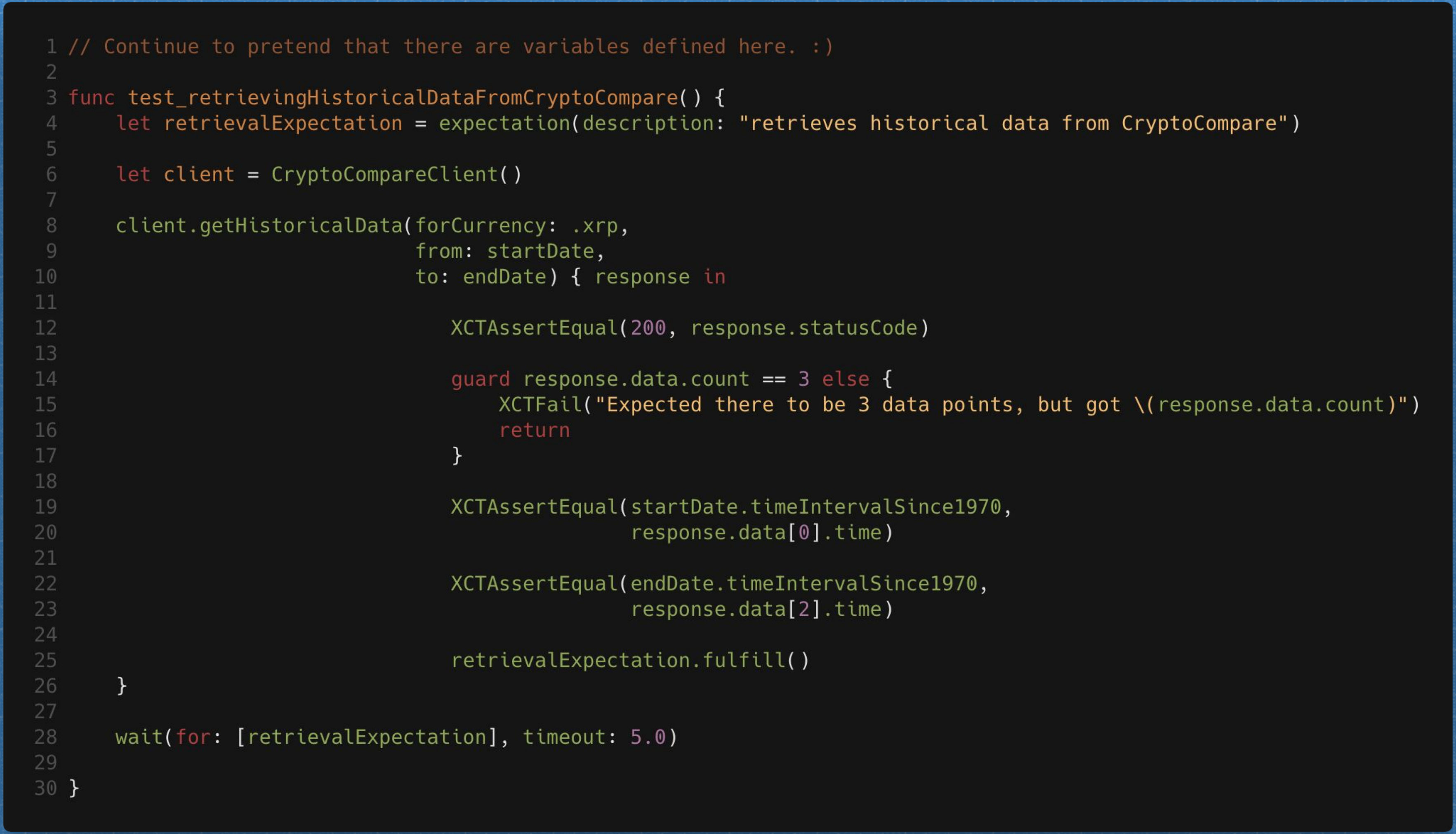
EXAMPLES CONTINUED - COLLABORATION TESTS



```swift
class ViewControllerTests: XCTestCase {

    // Pretend there are some variables defined here

    override func setUp() {
        let storyboard = UIStoryboard(name: "Main", bundle: nil)
        mockClient = MockCryptoCompareClient()
        viewController = storyboard.instantiateInitialViewController() as! ViewController
        viewController.cryptoCompareClient = mockClient
        viewController.todaysDate = startDate
        viewController.view.layoutIfNeeded()
    }

    func test_itGetsPricesAfterTheViewLoads() {
        XCTAssertEqual(mockClient.methodCalls, ["getHistoricalData(forCurrency:from:to:using:)"])

        XCTAssertEqual(mockClient.lastCurrency, .xrp)
        XCTAssertEqual(mockClient.lastStartDate, endDate)
        XCTAssertEqual(mockClient.lastEndDate, startDate)
    }
}
```

HIGH-SIGNAL, LOW-NOISE, WHITEBOX TEST

EXAMPLES CONTINUED - CONTRACT TESTS



```swift
// Continue to pretend that there are variables defined here. :)

func test_retrievingHistoricalDataFromCryptoCompare() {
    let retrievalExpectation = expectation(description: "retrieves historical data from CryptoCompare")

    let client = CryptoCompareClient()

    client.getHistoricalData(forCurrency: .xrp,
                             from: startDate,
                             to: endDate) { response in

                                XCTAssertEqual(200, response.statusCode)

                                guard response.data.count == 3 else {
                                    XCTFail("Expected there to be 3 data points, but got \(response.data.count)")
                                    return
                                }

                                XCTAssertEqual(startDate.timeIntervalSince1970,
                                               response.data[0].time)

                                XCTAssertEqual(endDate.timeIntervalSince1970,
                                               response.data[2].time)

                                retrievalExpectation.fulfill()
    }

    wait(for: [retrievalExpectation], timeout: 5.0)
}
```

MODERATE-SIGNAL, MODERATE-NOISE, BLACKBOX TEST

4

Summarizing Signal-to-Noise Ratio

Well, that was a lot.

What did we learn again?

WHAT DID WE LEARN?

- Defined SNR
- Reviewed why it matters
- Defined test doubles, then saw how they influence SNR
- Went through a refactoring example where we improve the SNR

WHAT DID WE LEARN?

- Reviewed how unit & integration tests affect it differently
- Went through some test patterns to see how they influence SNR
- And we now find ourselves here...

TECHNICAL RESOURCES

Test Driven Development: By Example

https://www.amazon.com/Test-Driven-Development-Kent-Beck/dp/0321146530

The Test Double Rule of Thumb

https://engineering.pivotal.io/post/the-test-double-rule-of-thumb/

Xunit Test Patterns

https://www.amazon.com/xUnit-Test-Patterns-Refactoring-Code/dp/01314950
54

Joe Masilotti’s Blog

http://masilotti.com/

CAREER RESOURCES

Integral

www.integral.io

Integrate Detroit

https://www.integral.io/enablement

Thank you very much!

ANY QUESTIONS?

sean.r.olszewski@gmail.com

github.com/seanrolszewski

@__chefski__