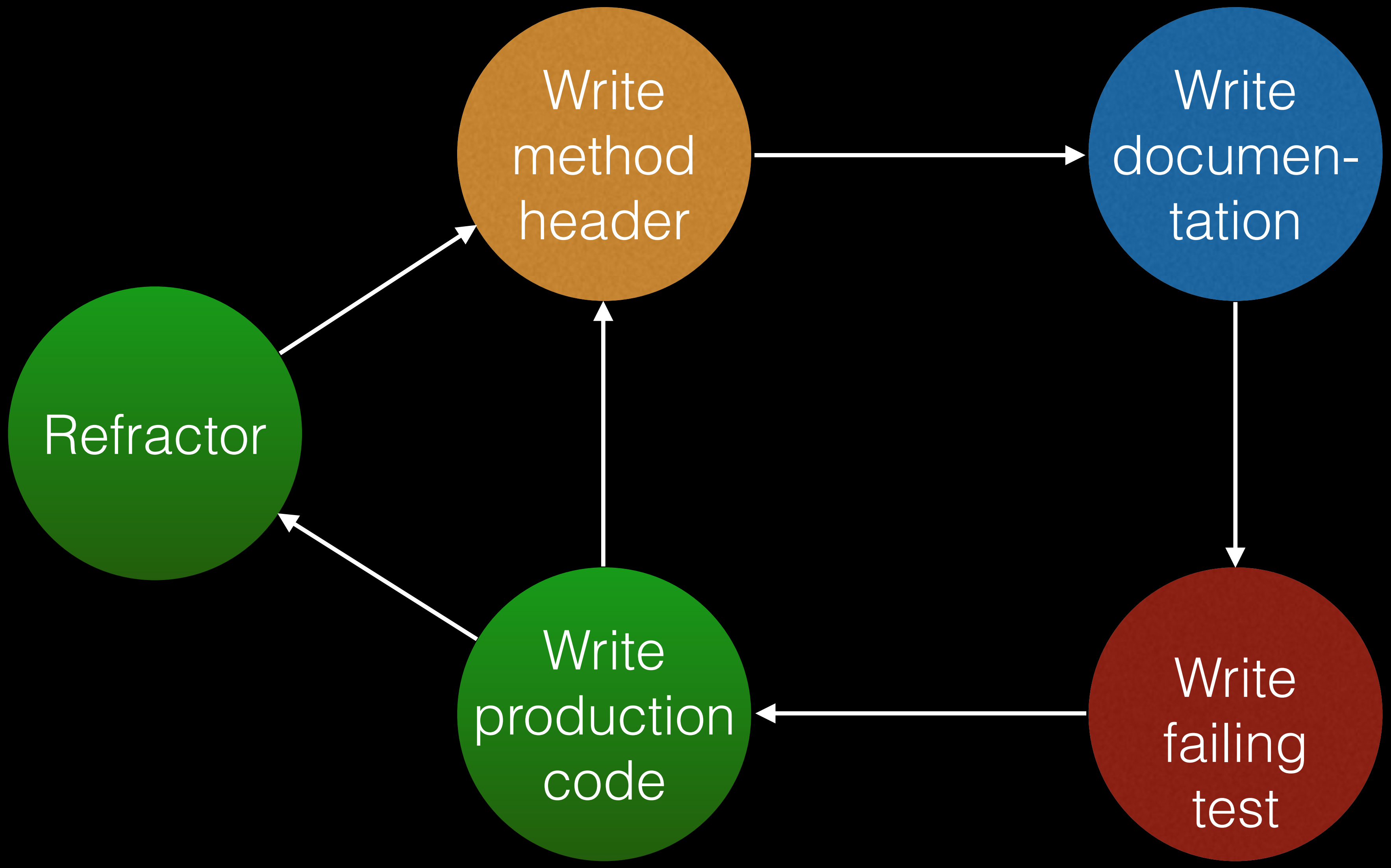
Test-driven development – Why and how to do it in Swift

Alex Hoppen – CocoaHeads Aachen – 26.2.2015

Why?

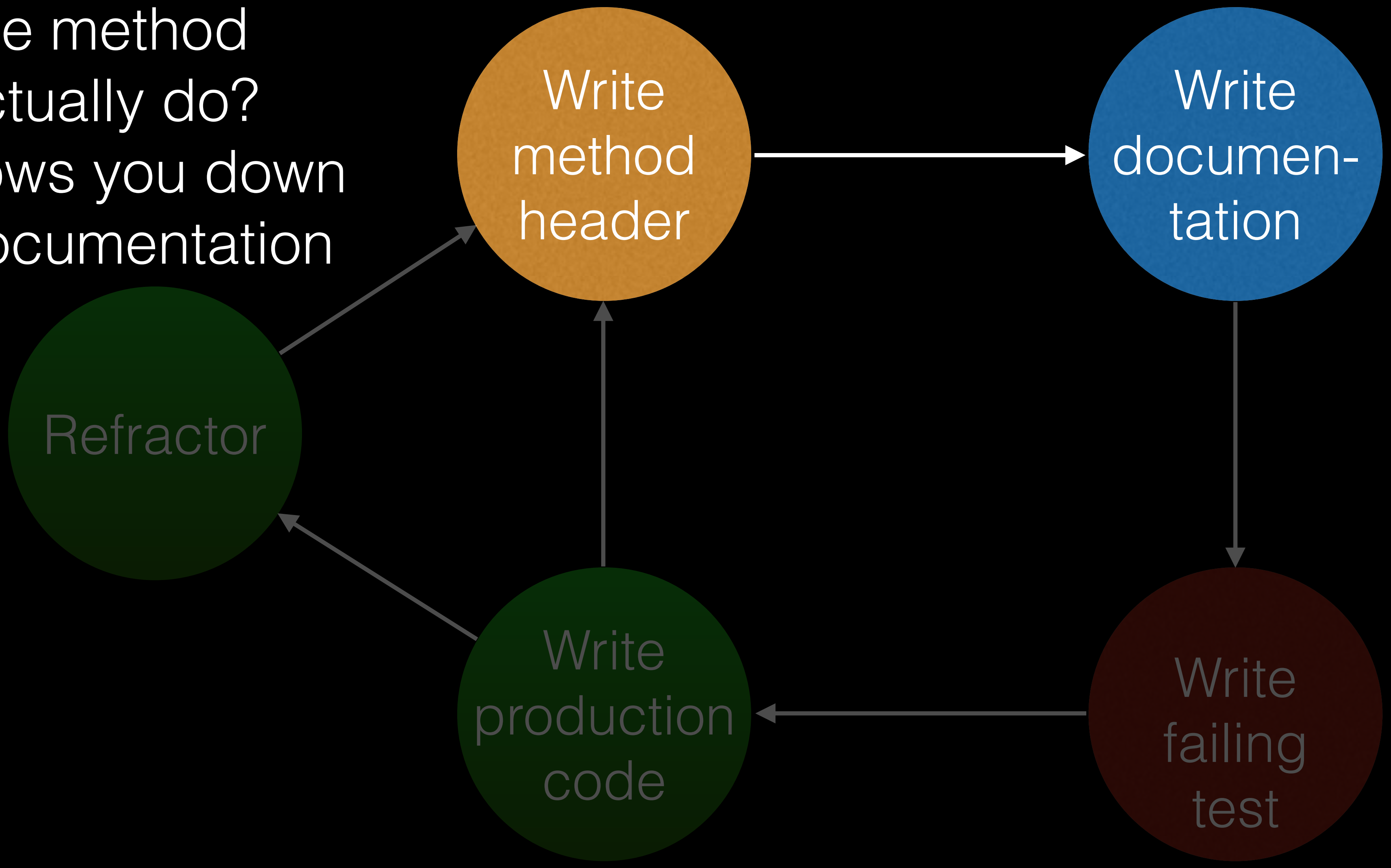
- Create App bottom-up instead of top down
- Better architecture
- Makes you think about what you are going to do
- Documentation
- Find bugs early
- Confidence when refactoring
- Good feeling ;-)

My TDD workflow



My TDD workflow

What should
the method
actually do?
→ slows you down
→ documentation



Write method header

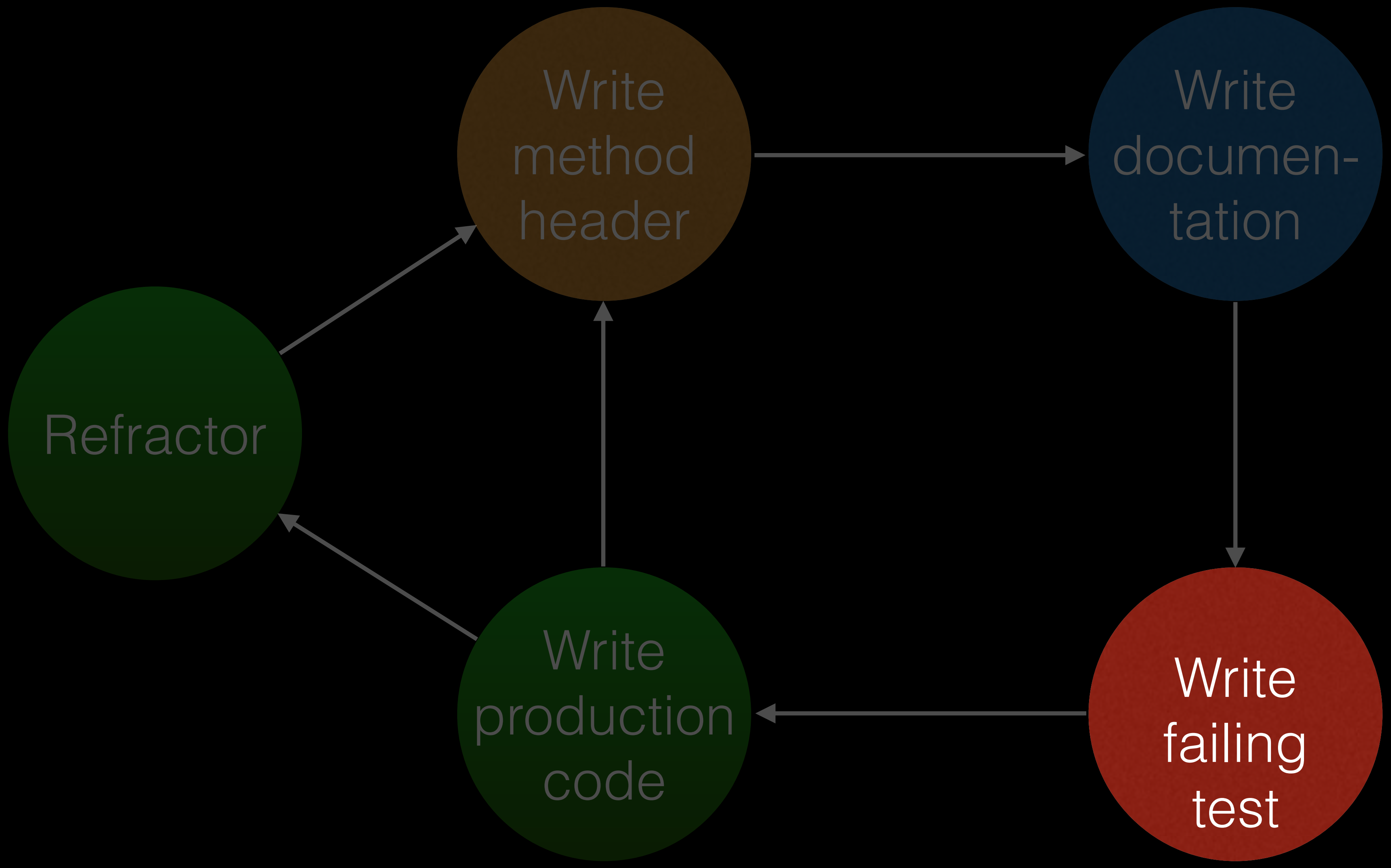
Write documentation

Write failing test

Write production code

Refractor

My TDD workflow



Write method header

Write documentation

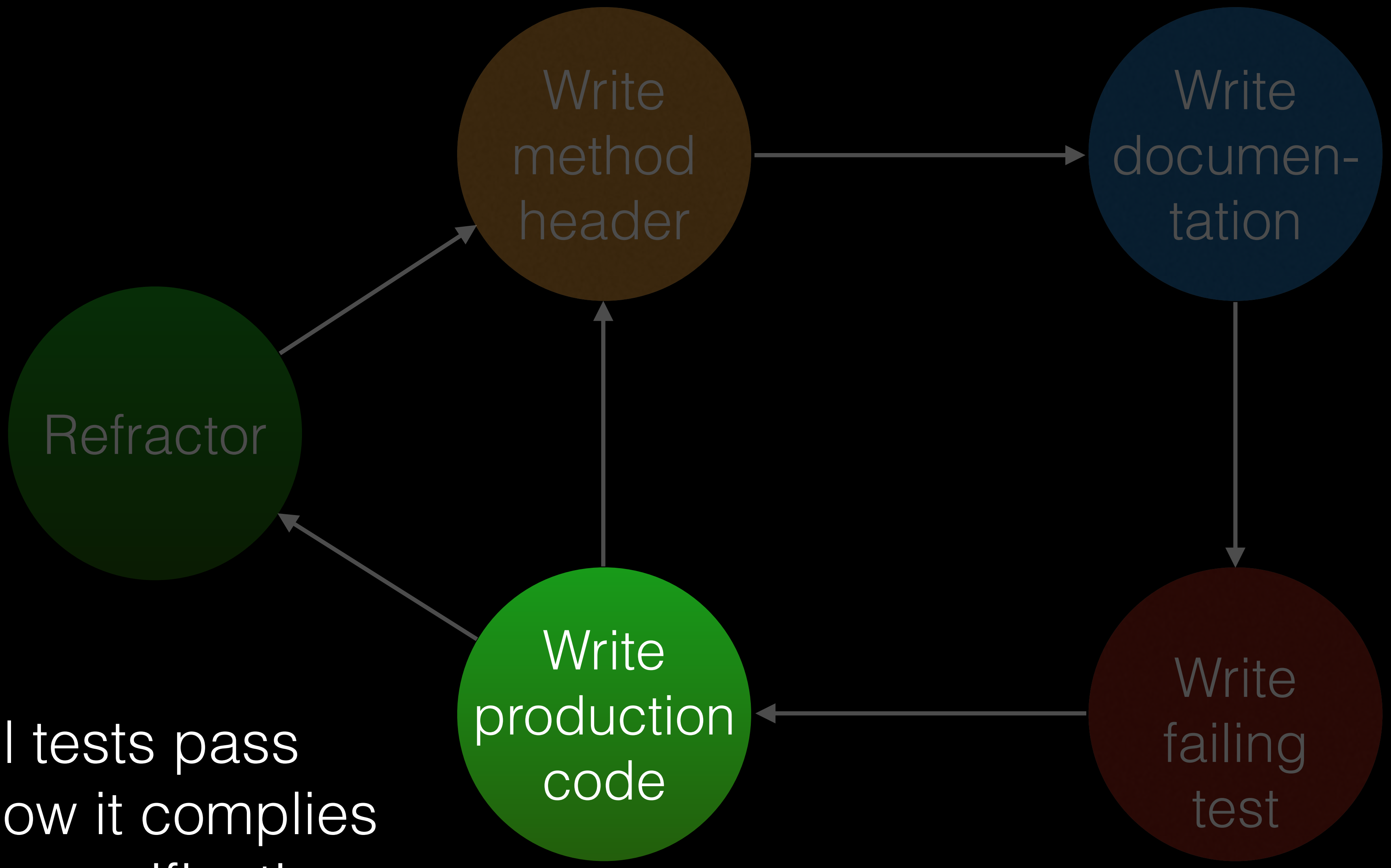
Write failing test

Write production code

Refractor

Does the API „feel good“?
→ better
architecture

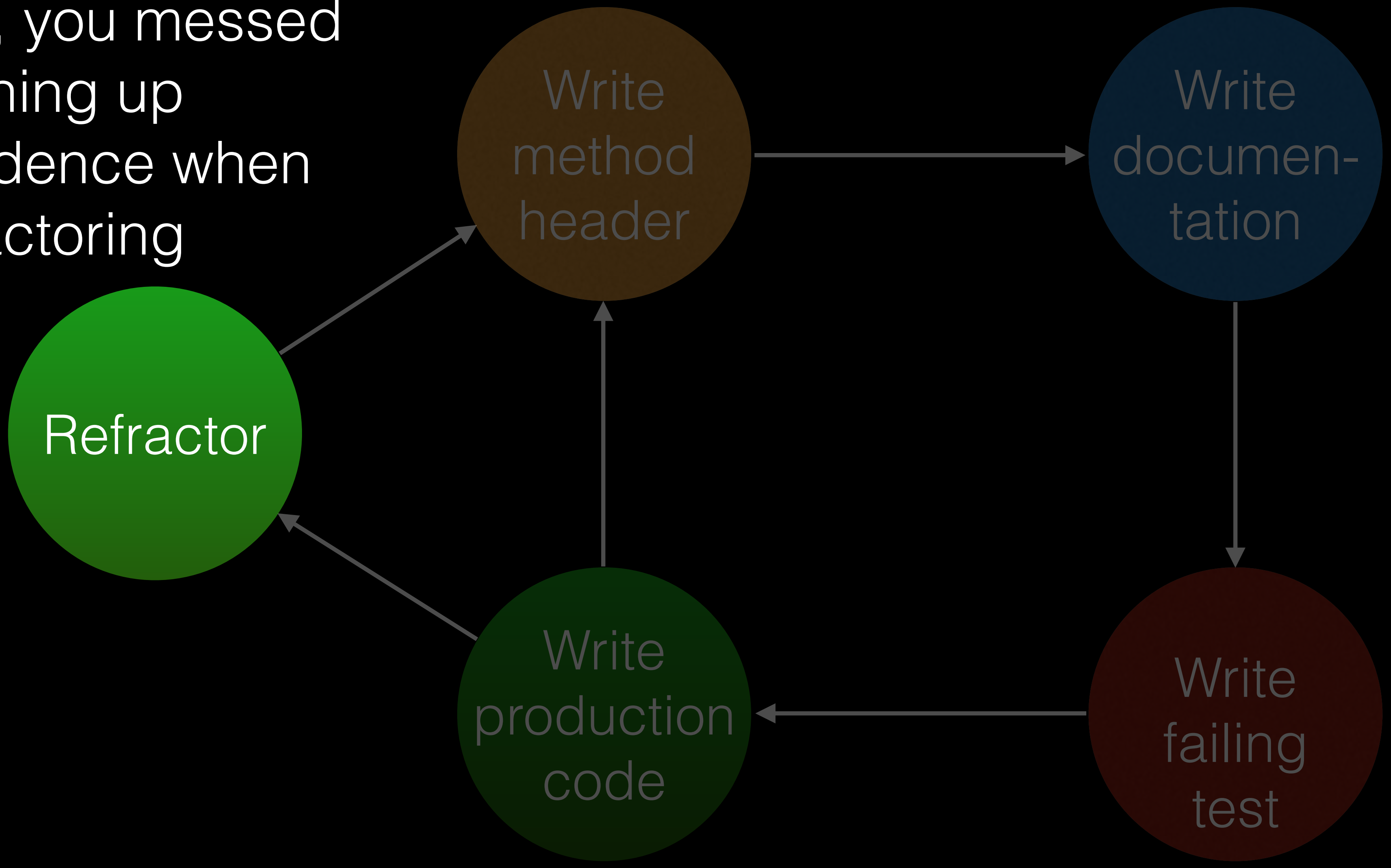
My TDD workflow



If all tests pass
you know it complies
to the specification
→ Find bugs early

My TDD workflow

If all tests stay green you know, you messed nothing up
→ Confidence when refactoring



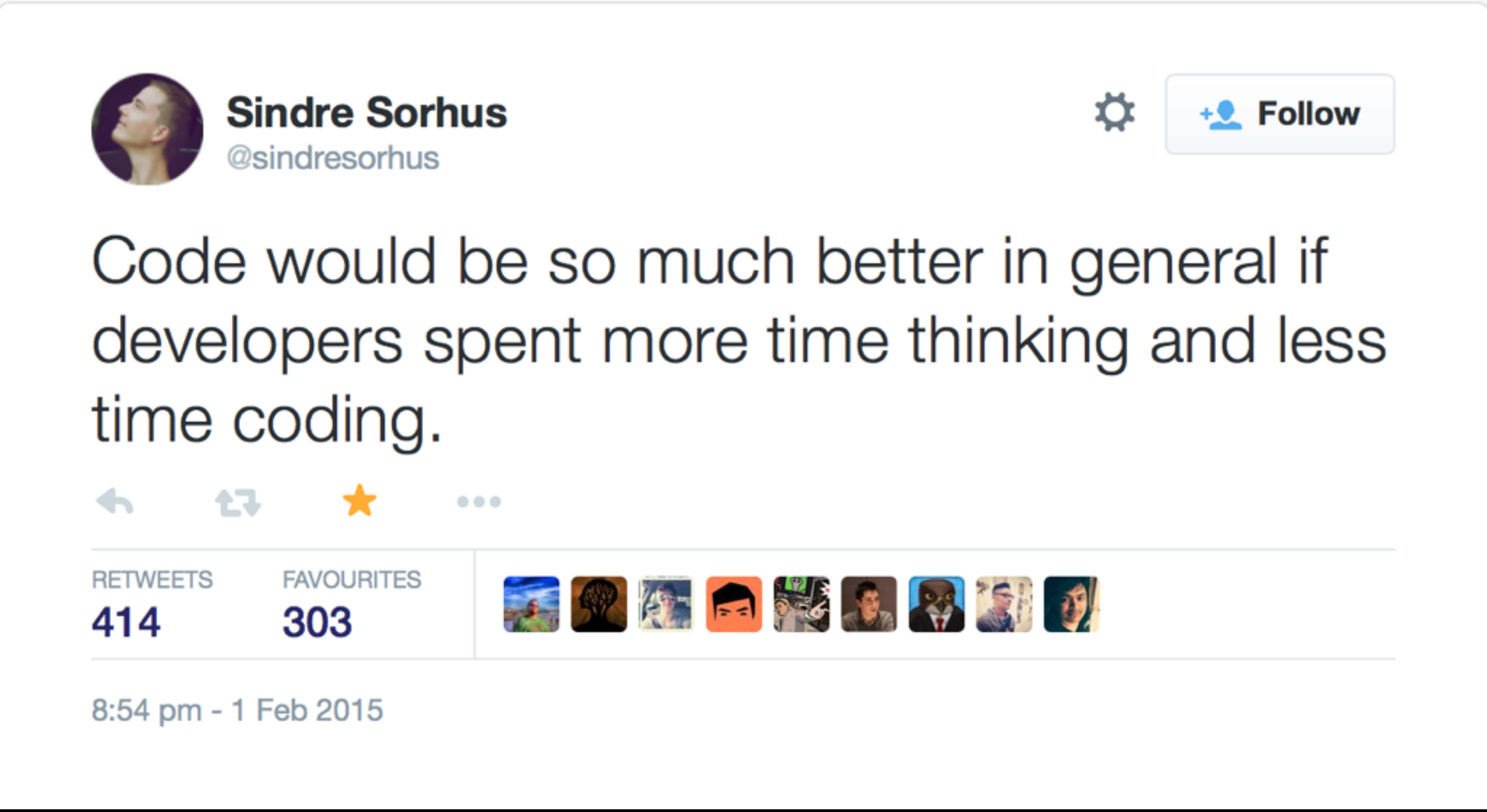
Are you slower?

Are you slower?

To be honest: Yes

but

Are you slower?



Sindre Sorhus
@sindresorhus

Code would be so much better in general if developers spent more time thinking and less time coding.

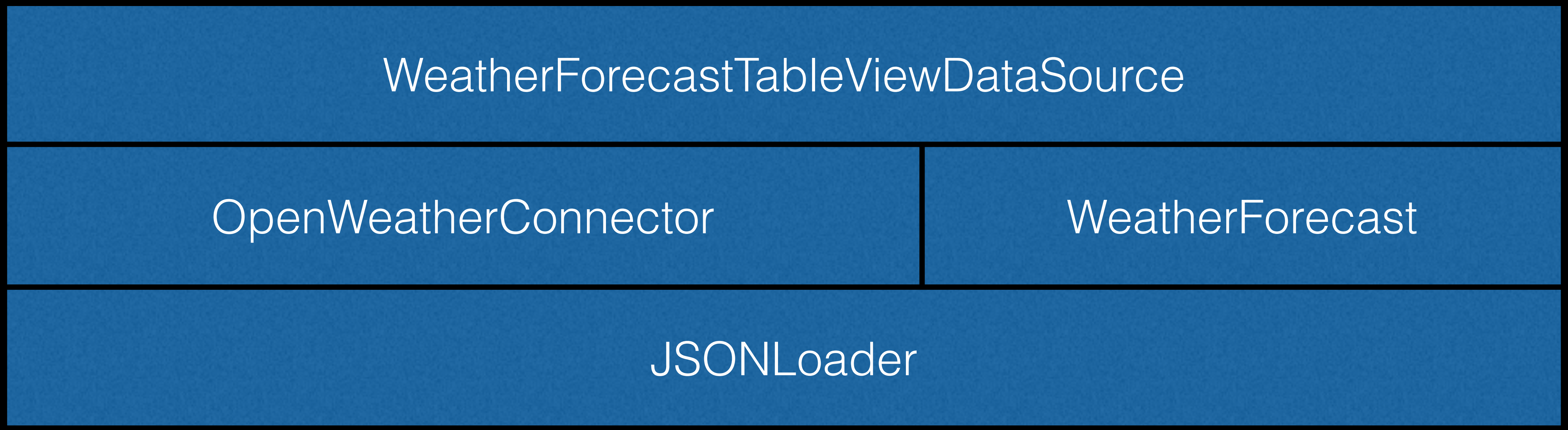
RETWEETS 414 FAVOURITES 303

8:54 pm - 1 Feb 2015

Demo

- Let’s make a crap app that fetches temperatures from openweathermap.org for Aachen and display them in a table view
- What makes this app non-trivial:
  - External dependencies
  - Asynchronous code
  - Swift 😜

Demo – overview



WeatherForecastTableViewDataSource

OpenWeatherConnector

WeatherForecast

JSONLoader

Demo

https://github.com/ahoppen/CocoaWeather

What we just saw

- Make all dependencies of an object explicit via dependency injection
- Mock external dependencies to really test a specific unit
- Use XCTAssert… to test values
- Use XCTestExpectation for asynchronous assertions

Final words – my opinion

- A test / production code ratio of 2 is usual. At the beginning it may even be higher.
- Don’t worry if it feels awkward at first. It get’s better with time and saves you time after the initial creation of the app
- Don’t write code without a test for it

Useful resources to get started

- Trivial examples: Just google
- Another (slightly more complex) example: http://qualitycoding.org/objective-c-tdd/
- Swift-specific unit testing-problems: http://www.andrewcbancroft.com/2014/12/19/swift-unit-testing-resources/
- objc.io Testing article: http://www.objc.io/issue-15/editorial.html