ALEX ROZUMII, TOPTAL

TESTS ARE NOT YOUR ORDINARY CODE

WHO AM I?

WHY TESTING?

TALK STRUCTURE

- Lot of ~~boring~~ bullet points
    - I’ve found the ‘Keynote’ app in my laptop!
- More abstract things first
- More real-world stuff down the road
- …and yes, tell me please where I’m wrong

TESTING

WHAT IS NOT PRESENTED HERE

- How to get a job as a QA. Sorry, developers
- Stress testing vs load testing
- Test plan
- Bug workflow
- Test case
- Types of testing
- QA vs QC vs quality management

AT ITS CORE, TESTING IS THE PROCESS OF COMPARING "WHAT IS" WITH "WHAT OUGHT TO BE"

Lee Copeland

WHAT IS TESTING AUTOMATION?

- Test automation is the use of **special software** to **control the execution of tests** and the comparison of actual outcomes with predicted outcomes.

PATH CONCEPT

- Developers usually focus on the positive path (aka happy path)
- But… That’s not the only path
- What about bad data (Sad Path)
- Hacker attacks (Evil Path)
- Web services going down (Weird Path)?

TEST DESIGN

WTF IS TEST DESIGN?

90% of audience

EXAMPLE

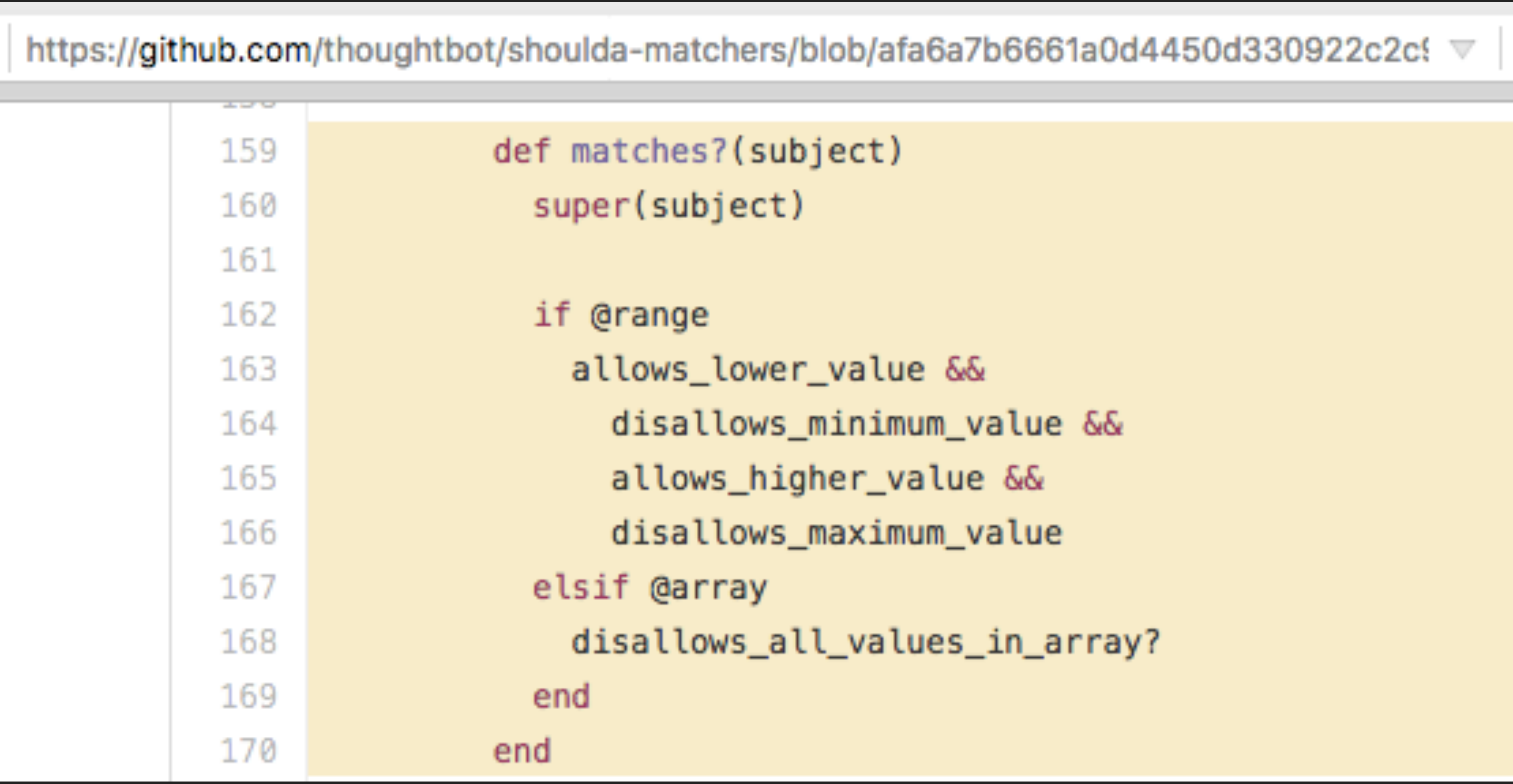
- Consider a printer that has an input option of the number of copies to be made, from 1 to 99.
- Three classes:
    - < 1 - invalid
    - 1-99 - valid
    - >99 - invalid

EQUIVALENCE CLASS PARTITIONING

- The idea behind this technique is to divide a set of test conditions into groups or sets that can be considered the same.

EXAMPLE

- Boundaries:
  - < 1
    - 0
  - 1-99 - valid
    - 1
    - 99
  - >99 - invalid
    - 100



https://github.com/thoughtbot/shoulda-matchers/blob/afa6a7b6661a0d4450d330922c2c

```
      def matches?(subject)
        super(subject)

        if @range
          allows_lower_value &&
            disallows_minimum_value &&
            allows_higher_value &&
            disallows_maximum_value
        elsif @array
          disallows_all_values_in_array?
        end
      end
```

LEVELS OF TESTING (CLASSIC)

- Unit testing
- Component testing
- Integration testing
- Component integration testing
- System integration testing
- Alpha testing
- Beta testing

LEVELS OF TESTING (PRACTICAL)

- It's relative. Your unit might be someone else's component.
- So, basically, it gets down to induction:

  A. Get a small block

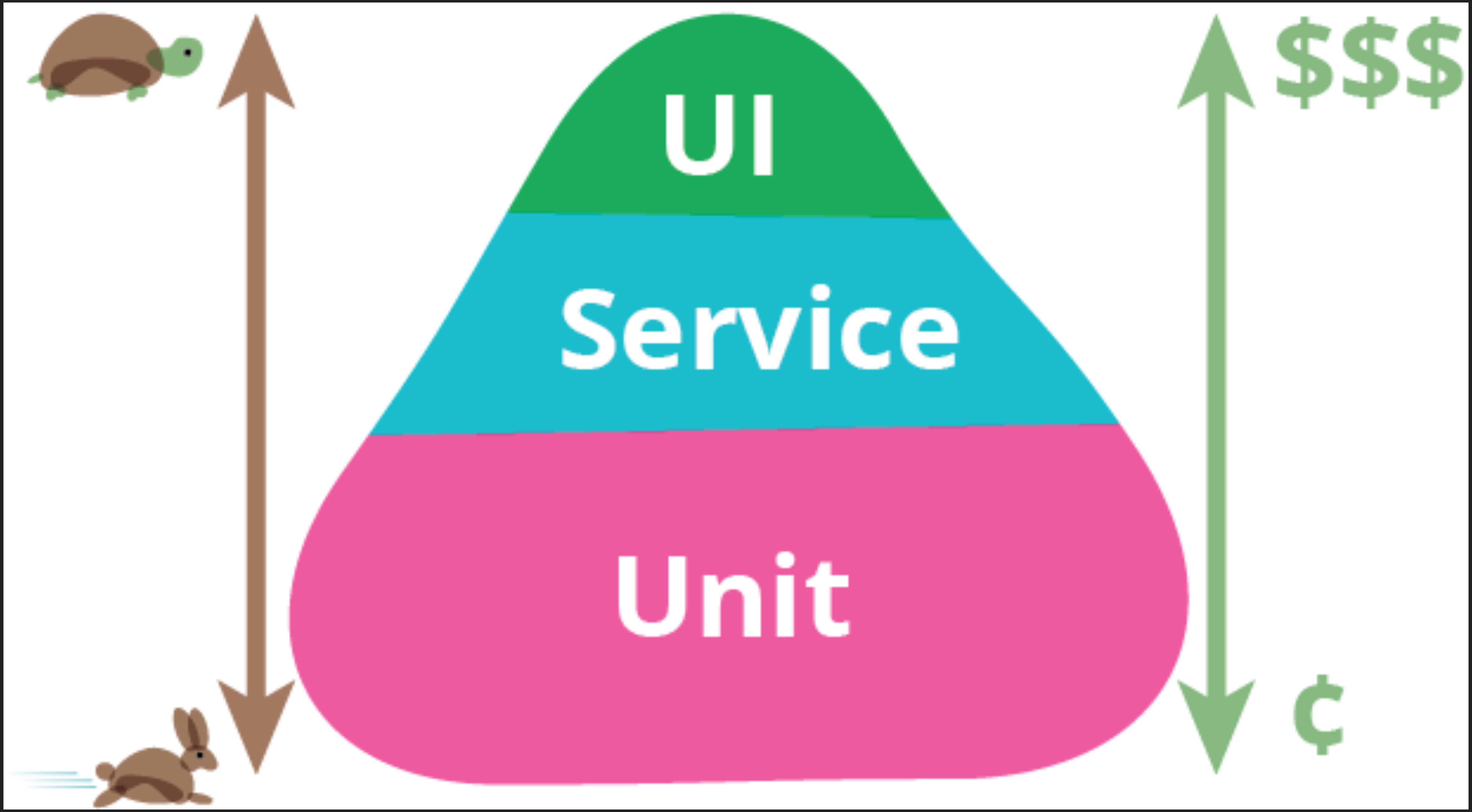
  B. Test it

  C. Combine few tested blocks

  D. Do integration testing on them

  E. Given the thing we've just tested is a small block, go to (A)

TESTING PYRAMID



NICE THING ABOUT LEVELS OF TESTING

- If you have a bug - than you have at least one gap at each level of your testing. ;)

TESTING AND DEVELOPMENT

FOCUS OF DEVELOPMENT

- Functionality ↑
- Correctness ↑
- Performance ↑

FOCUS OF TESTING

- Functionality ~
  - Depends on test design
- Correctness ↑ ↑ ↑
  - Explicitness
- Performance ~

STANDARD DEVELOPMENT APPROACHES

- DRY
- SRP
- More abstractions
- Use all the features of language

TESTING

- Simpler is better
    - Even capybara-like implicitness is considered harmful
- One case - one test
- Less abstractions
- Less language features used

TESTING PATTERNS?

- Okay, I needed to say about those
- Page object
- Composition
- Factory
- Decorator
- Hardcoding

IF YOU CAN'T BUILD A WELL-STRUCTURED MONOLITH, WHAT MAKES YOU THINK YOU CAN BUILD A WELL-STRUCTURED SET OF MICROSERVICES?

Simon Brown

MODULARITY

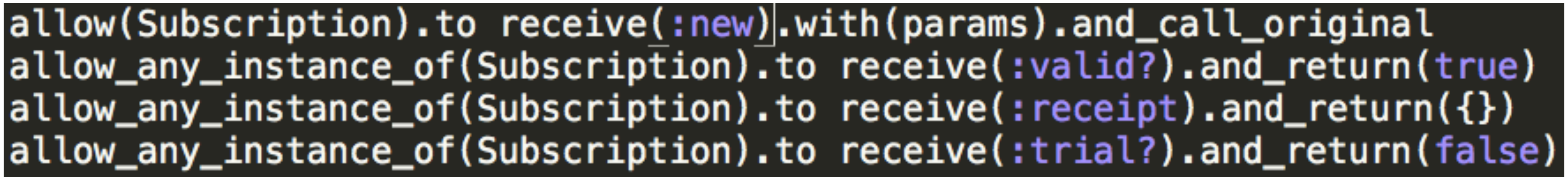
- Testability is one of the architecture attributes
- Number of system states: average_unit_permutation_number ^ unit_number
    - 5 units, 10 permutations each, 9,765,625 states of a system.
    - 10 units, 5 permutation each, 100,000 states of a system.

USING MODULARITY TO IMPROVE TESTABILITY

- Less states of each part -> less cases (**SRP**)
- Smaller interfaces -> easier to reproduce (**ISP**)
- Hiding implementation details -> simpler to write appropriate tests
- It’s hard to test something
  - Maybe it’s a god object?
  - Maybe something can be extracted?

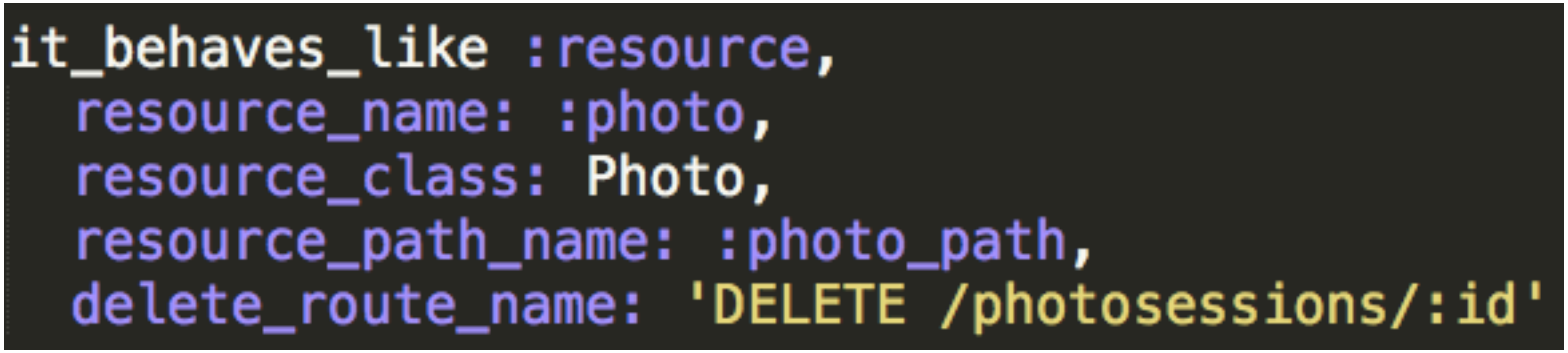
LET'S HAVE SOME REAL-LIFE CASES

IMPLEMENTATION DETAILS



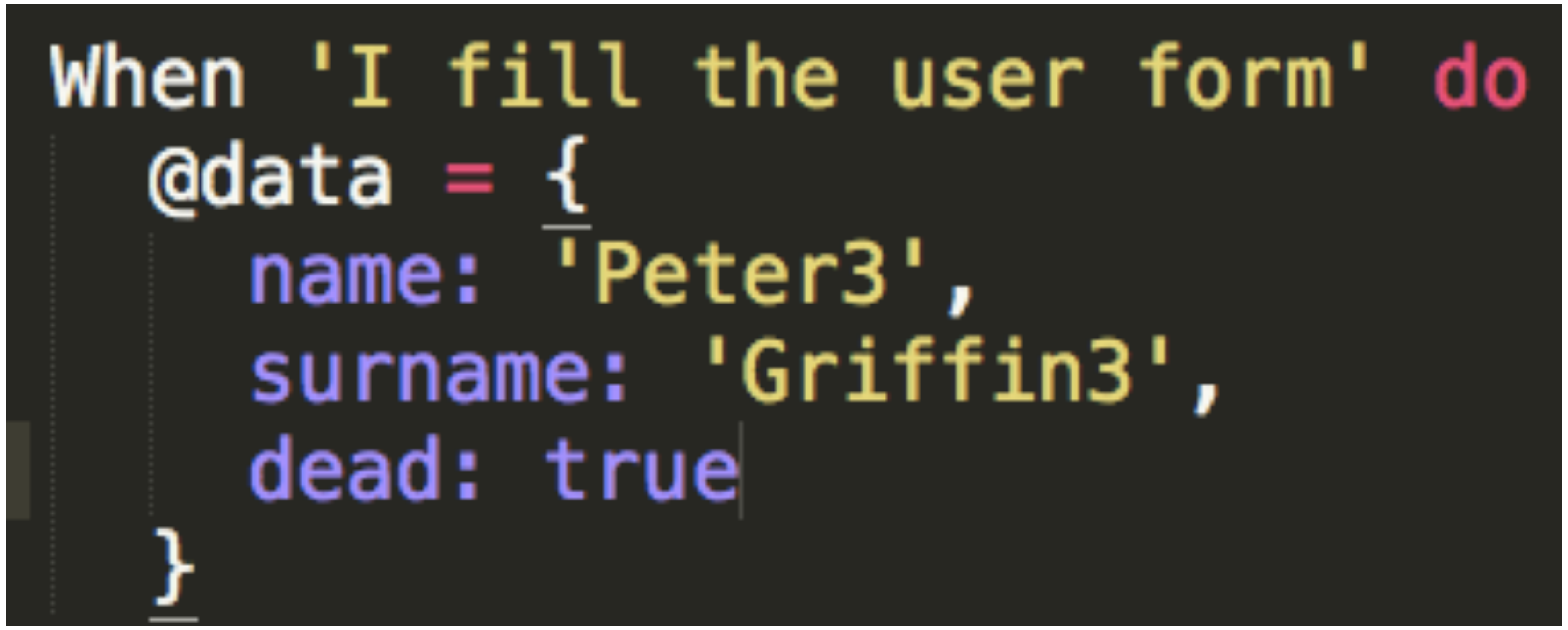
```
allow(Subscription).to receive(:new).with(params).and_call_original
allow_any_instance_of(Subscription).to receive(:valid?).and_return(true)
allow_any_instance_of(Subscription).to receive(:receipt).and_return({})
allow_any_instance_of(Subscription).to receive(:trial?).and_return(false)
```

LET'S (NOT) DO SOME META!



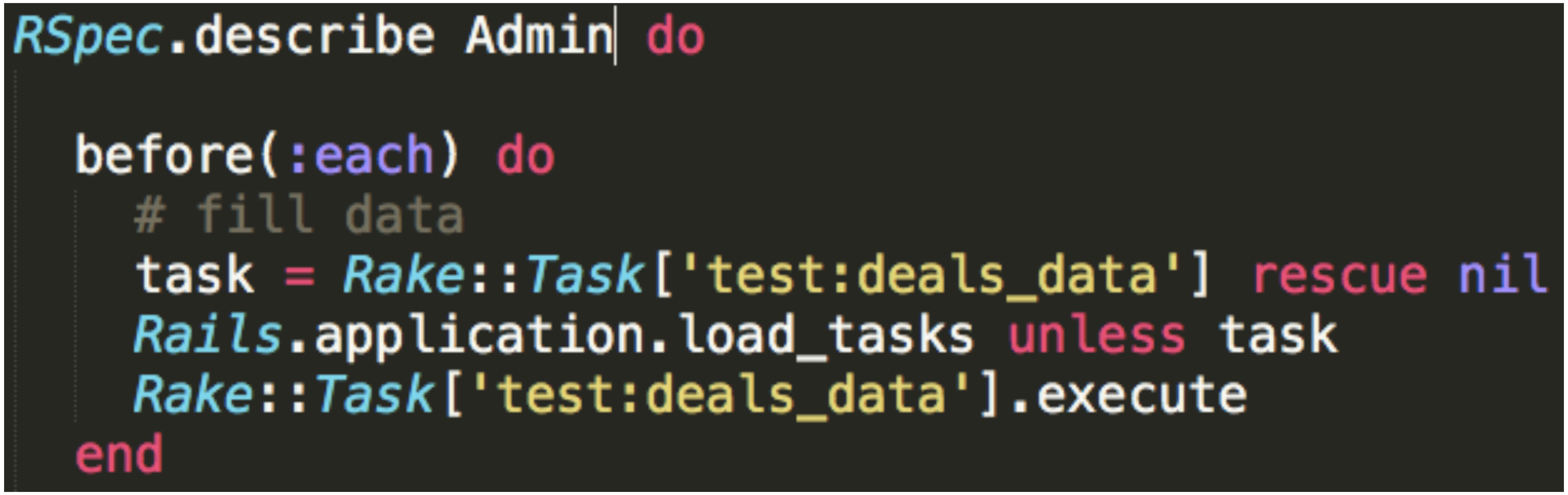
```
it_behaves_like :resource,
  resource_name: :photo,
  resource_class: Photo,
  resource_path_name: :photo_path,
  delete_route_name: 'DELETE /photosessions/:id'
```

HARDCODING



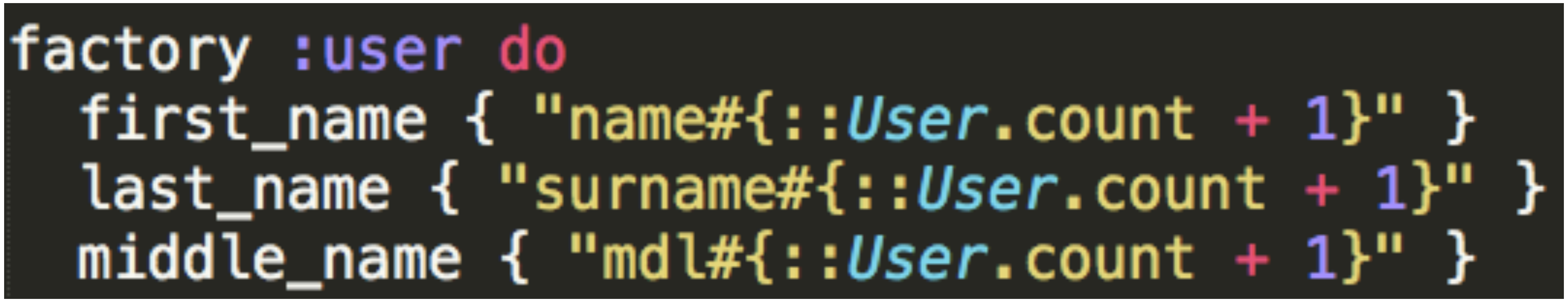
```
When 'I fill the user form' do
  @data = {
    name: 'Peter3',
    surname: 'Griffin3',
    dead: true
  }
```

WE ALREADY HAVE THIS CODE IN THE APP!



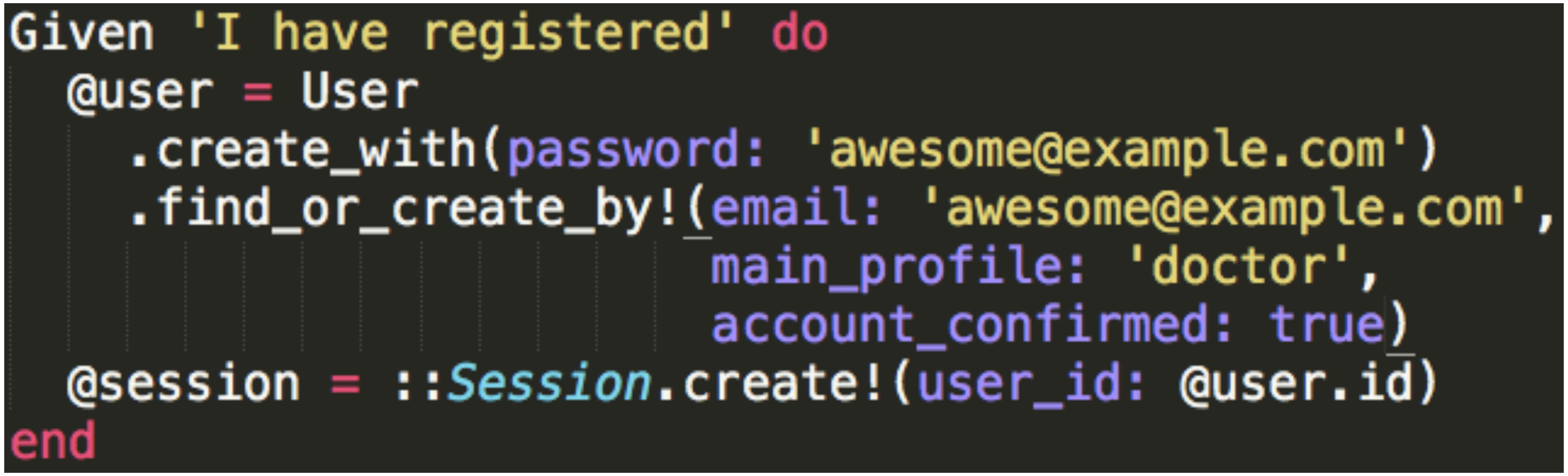
```
RSpec.describe Admin do

  before(:each) do
    # fill data
    task = Rake::Task['test:deals_data'] rescue nil
    Rails.application.load_tasks unless task
    Rake::Task['test:deals_data'].execute
  end
```

WHAT'S WRONG HERE?



```ruby
factory :user do
  first_name { "name#{::User.count + 1}" }
  last_name { "surname#{::User.count + 1}" }
  middle_name { "mdl#{::User.count + 1}" }
```

OBSOLETE DETAILS



```ruby
Given 'I have registered' do
  @user = User
    .create_with(password: 'awesome@example.com')
    .find_or_create_by!(email: 'awesome@example.com',
                        main_profile: 'doctor',
                        account_confirmed: true)
  @session = ::Session.create!(user_id: @user.id)
end
```

SUMMARY

SUMMARY

- Split and simplify everything in tests - less is more
- Limit the size of test suite (test design helps)
- Follow not only the ‘happy’ path
- Read the docs for the test tool ;)

THANKS!

QUESTIONS!