Executable Documentation
Exploring the Spec in RSpec

David Chelimsky
Articulated Man, Inc
Lead Developer/Maintainer of RSpec
Author: The RSpec Book

Traditional Documentation

- *Decoupled from the software it documents*
- *Expensive and error prone change management process*
- *Can lose value over time as momentum to maintain it decreases*

Executable Documentation

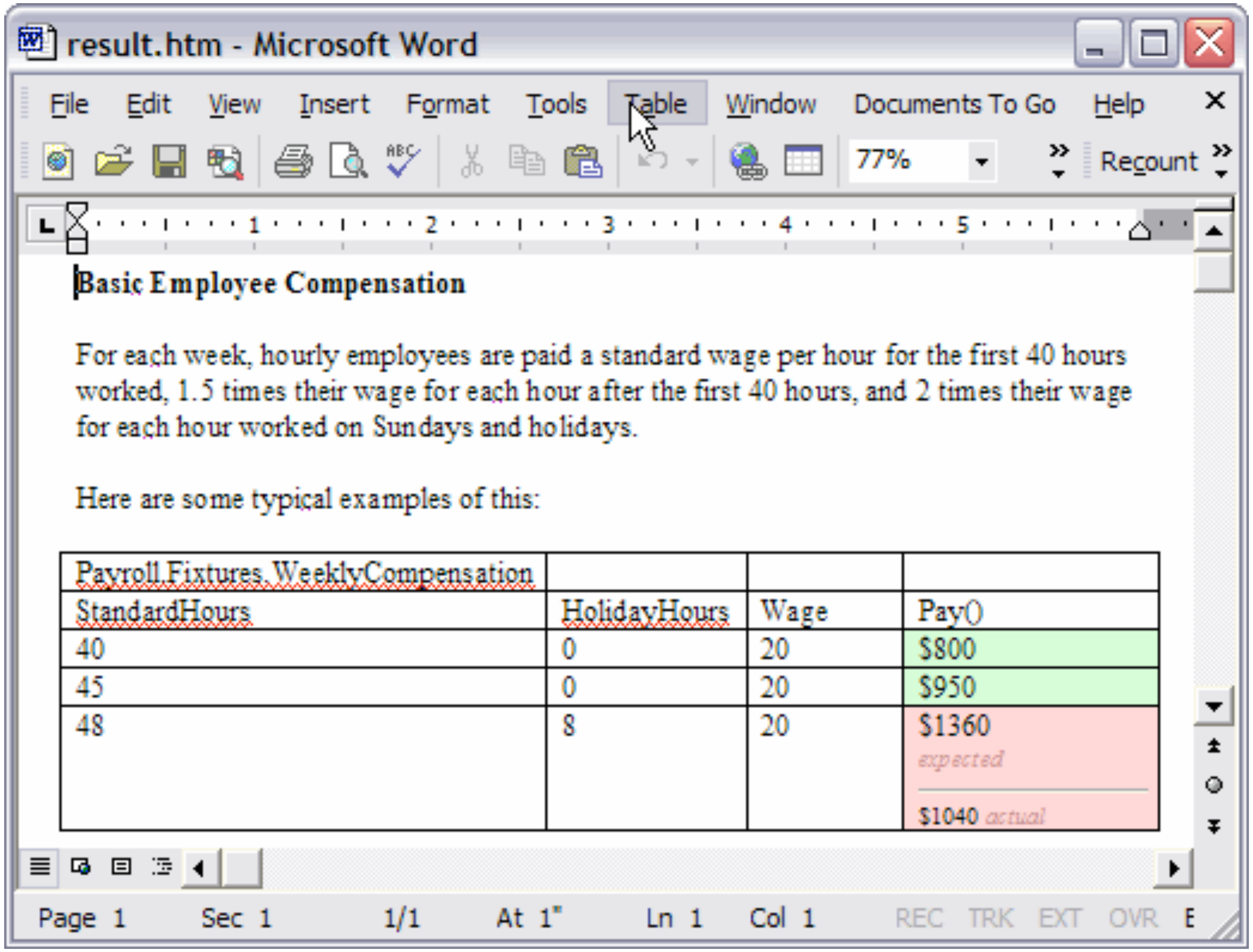
- *Documentation that generates tests*
- *Programatically bound to the software it documents*
- *Changes to documentation and software happen in parallel*
  - *or at least in close proximity*

Tools / Audience

Project Stakeholders

FIT

- *Framework for Integrated Test*
  - *Ward Cunningham*


result.htm - Microsoft Word
File Edit View Insert Format Tools Table Window Documents To Go Help
77%
Recount
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1 REC TRK EXT OVR

Basic Employee Compensation

For each week, hourly employees are paid a standard wage per hour for the first 40 hours worked, 1.5 times their wage for each hour after the first 40 hours, and 2 times their wage for each hour worked on Sundays and holidays.

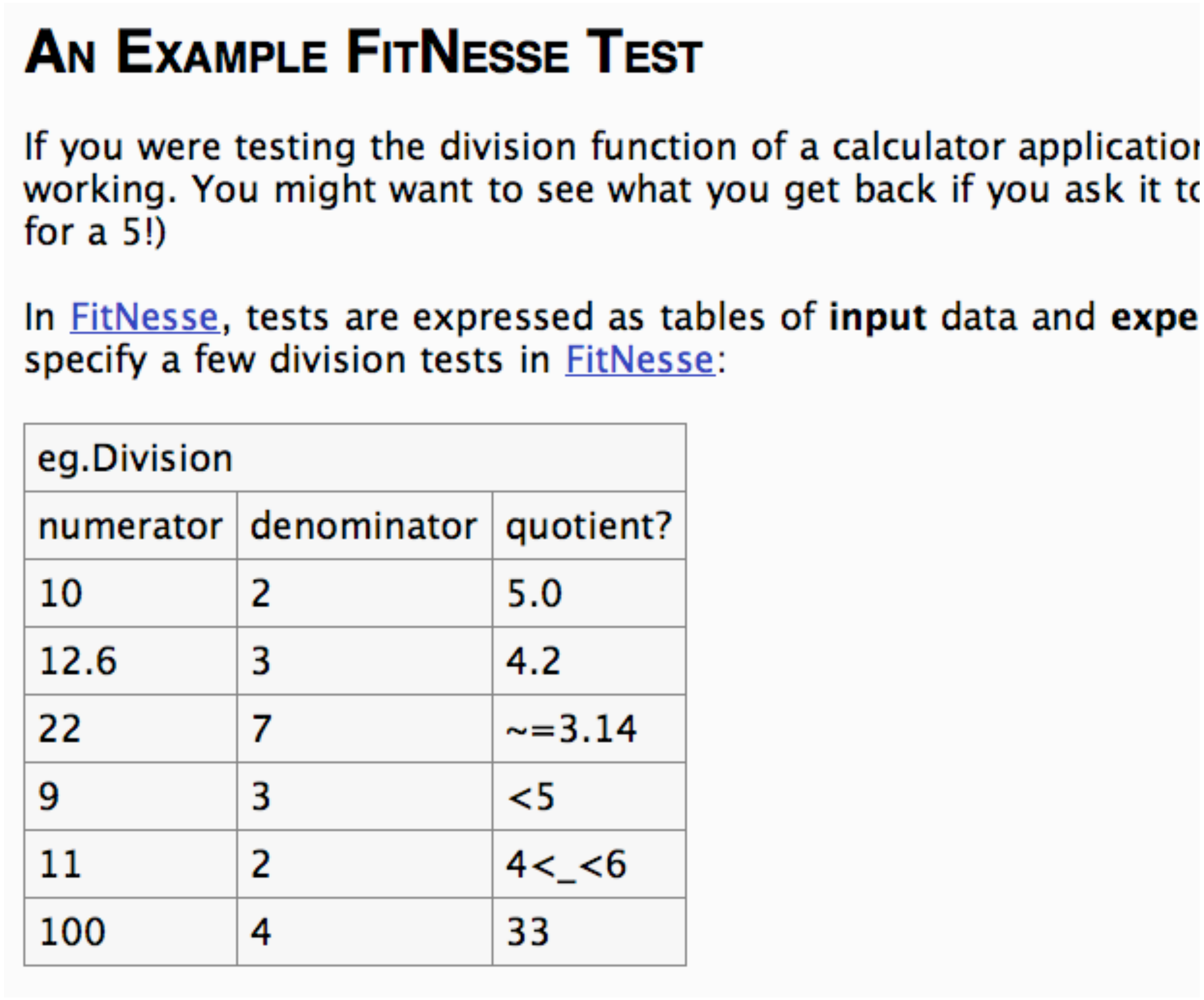
Here are some typical examples of this:

Payroll.Fixtures.WeeklyCompensation			
StandardHours	HolidayHours	Wage	Pay()
40	0	20	$800
45	0	20	$950
48	8	20	$1360 *expected* / $1040 *actual*

http://fit.c2.com

FitNesse

- *FIT in a browser*
  - *+ built-in web server*
  - *+ built-in wiki*
- *by Robert and Micah Martin*



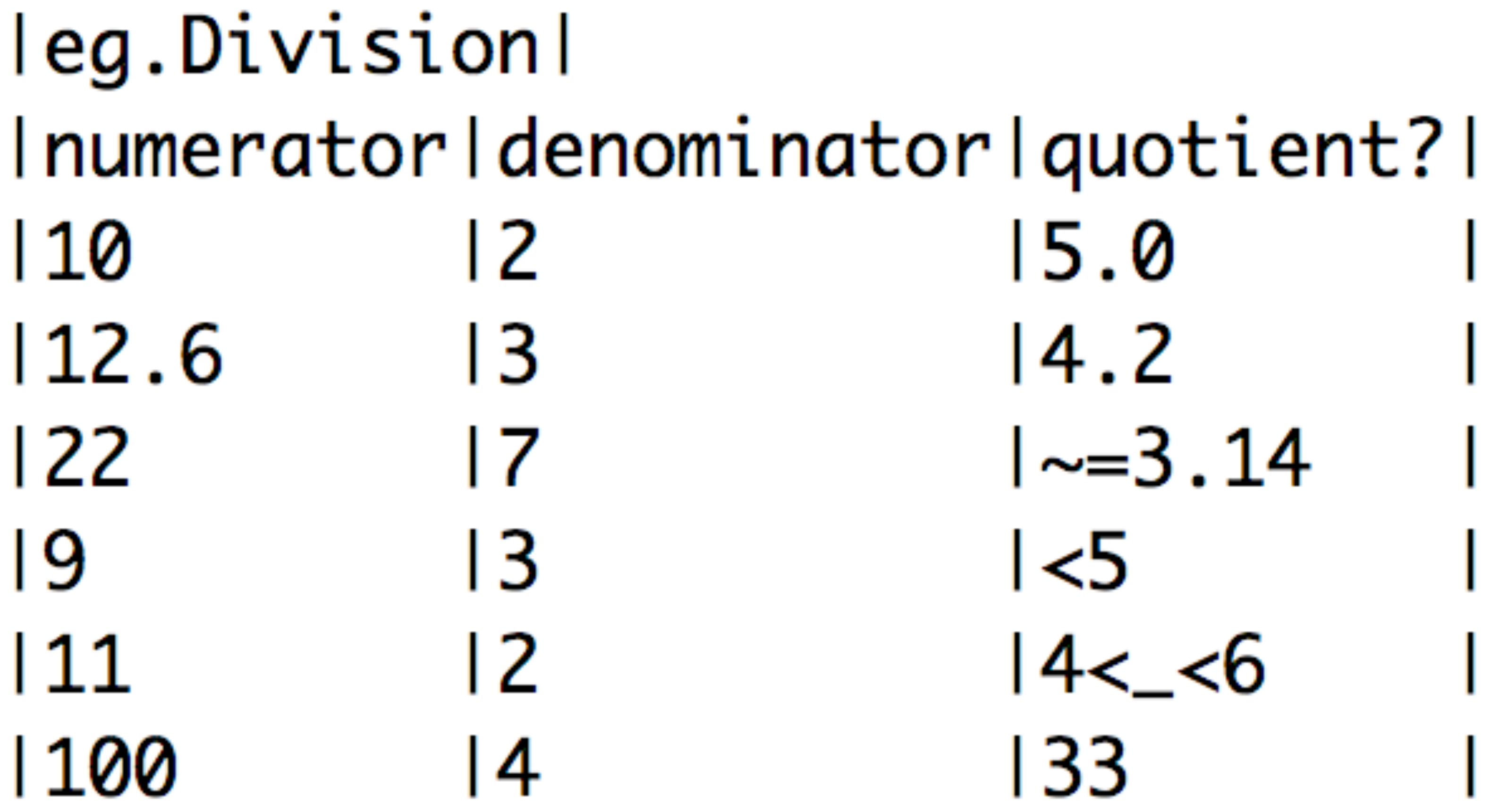
An Example FitNesse Test

If you were testing the division function of a calculator applicatio
working. You might want to see what you get back if you ask it t
for a 5!)

In FitNesse, tests are expressed as tables of **input** data and **expe**
specify a few division tests in FitNesse:

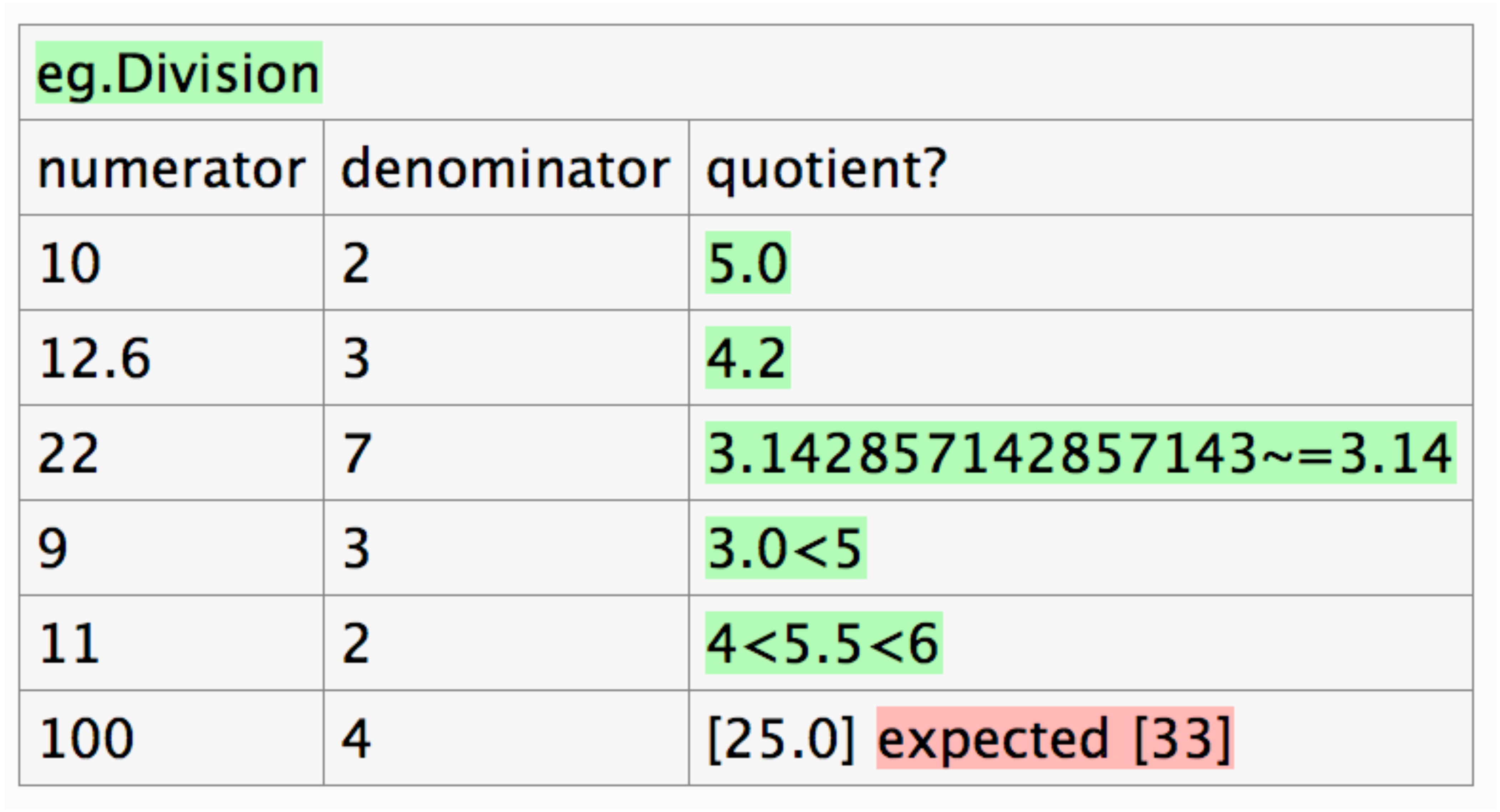
eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	~=3.14
9	3	<5
11	2	4<_<6
100	4	33

http://fitnesse.org



```
|eg.Division|
|numerator|denominator|quotient?|
|10       |2          |5.0      |
|12.6     |3          |4.2      |
|22       |7          |~=3.14   |
|9        |3          |<5       |
|11       |2          |4<_<6    |
|100      |4          |33       |
```

http://fitnesse.org



eg.Division		
numerator	denominator	quotient?
10	2	5.0
12.6	3	4.2
22	7	3.142857142857143~=3.14
9	3	3.0<5
11	2	4<5.5<6
100	4	[25.0] expected [33]

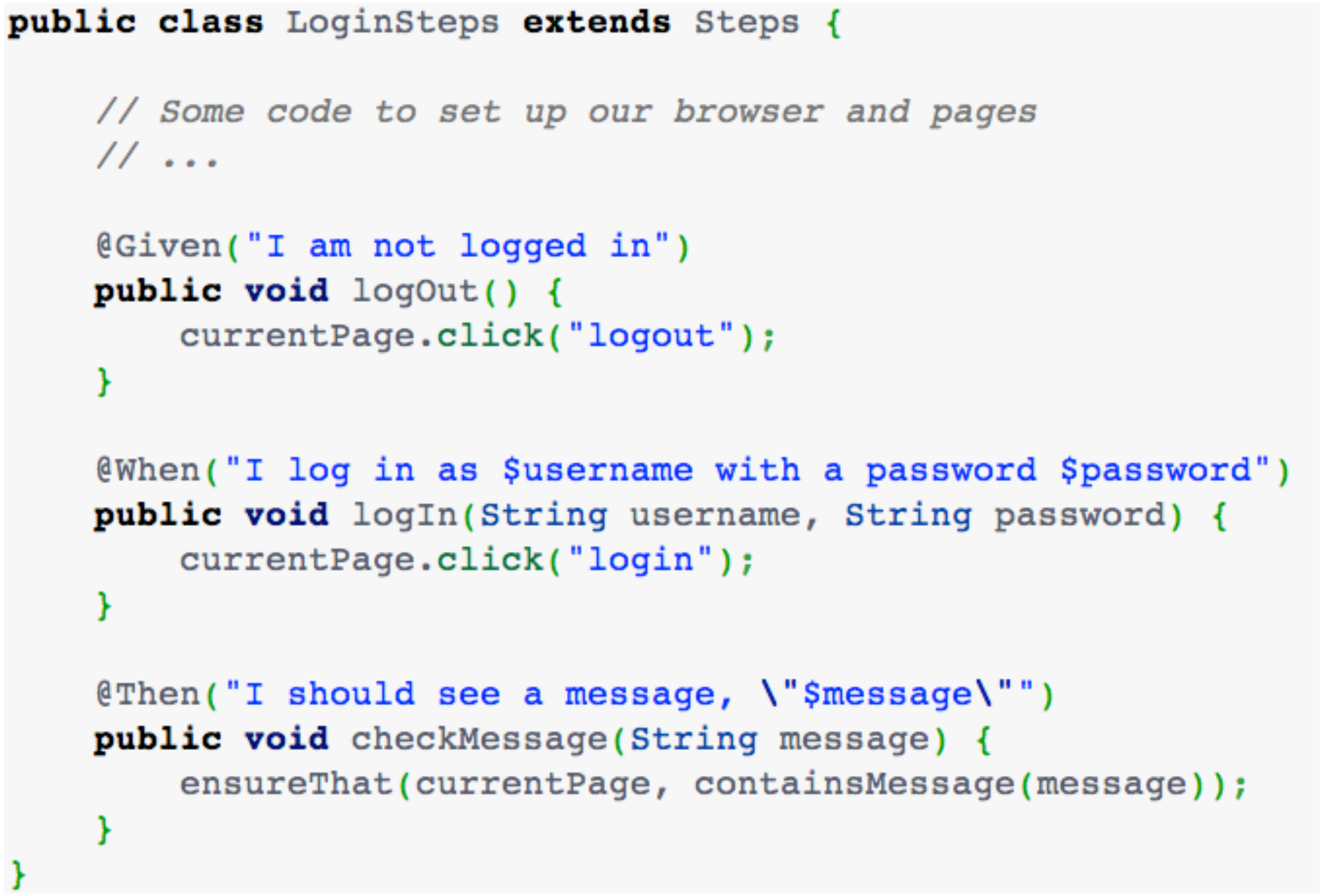
http://fitnesse.org

JBehave

- *The original Behaviour Driven Development Framework*
  - *by Dan North*

```
Given I am not logged in
When I log in as Liz with a password JBehaver
Then I should see a message, "Welcome, Liz!"
```

http://jbehave.org/



```
public class LoginSteps extends Steps {

    // Some code to set up our browser and pages
    // ...

    @Given("I am not logged in")
    public void logOut() {
        currentPage.click("logout");
    }

    @When("I log in as $username with a password $password")
    public void logIn(String username, String password) {
        currentPage.click("login");
    }

    @Then("I should see a message, \"$message\"")
    public void checkMessage(String message) {
        ensureThat(currentPage, containsMessage(message));
    }
}
```

http://jbehave.org/

RBehave

RSpec Story Runner

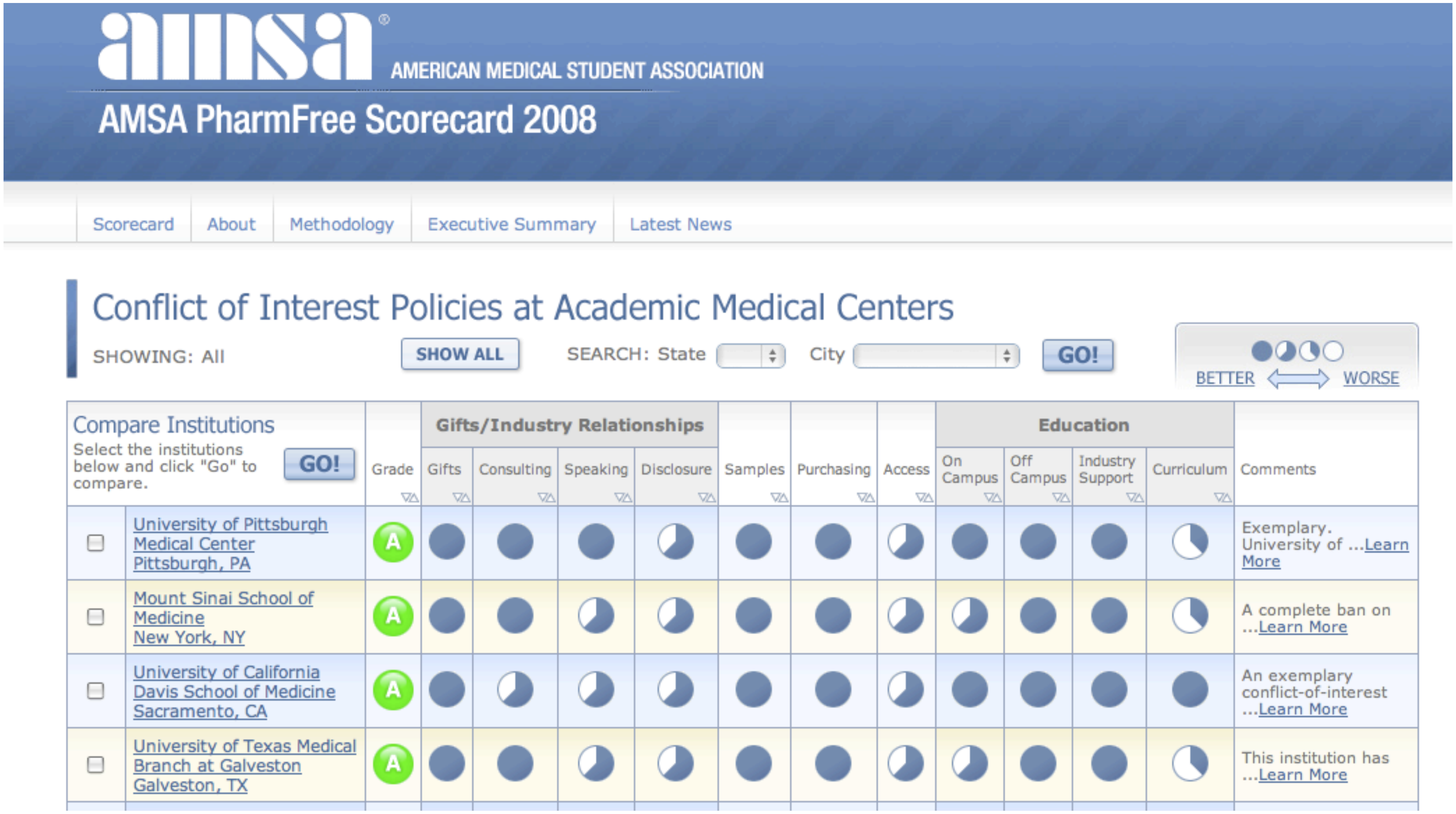
Cucumber

RBehave

- *Dan North’s Port of JBehave*
  - *used RSpec to develop*

RSpec Story Runner

- *Merged RBehave into RSpec*
- *Added Plain Text Story capability*



AMERICAN MEDICAL STUDENT ASSOCIATION

AMSA PharmFree Scorecard 2008

Scorecard | About | Methodology | Executive Summary | Latest News

Conflict of Interest Policies at Academic Medical Centers

SHOWING: All   SHOW ALL   SEARCH: State   City   GO!

● ◕ ◔ ○ BETTER ⟺ WORSE

Compare Institutions

Select the institutions below and click "Go" to compare. GO!

			Gifts/Industry Relationships							Education				
	Institution	Grade	Gifts	Consulting	Speaking	Disclosure	Samples	Purchasing	Access	On Campus	Off Campus	Industry Support	Curriculum	Comments
☐	University of Pittsburgh Medical Center Pittsburgh, PA	A	●	●	●	◕	●	●	◕	●	●	●	◔	Exemplary. University of ...Learn More
☐	Mount Sinai School of Medicine New York, NY	A	●	●	◕	◕	●	●	◕	◕	●	●	◔	A complete ban on ...Learn More
☐	University of California Davis School of Medicine Sacramento, CA	A	●	◕	◕	◕	●	●	◕	●	●	●	●	An exemplary conflict-of-interest ...Learn More
☐	University of Texas Medical Branch at Galveston Galveston, TX	A	●	●	◕	◕	●	●	◕	◕	●	●	◔	This institution has ...Learn More

Story: system generates letter grade

As an assessor
I want the system to generate a letter grade for me
So I don't have to get the math right every time

Scores for domains with subdomains are calculated as the average of the best three subdomain scores.

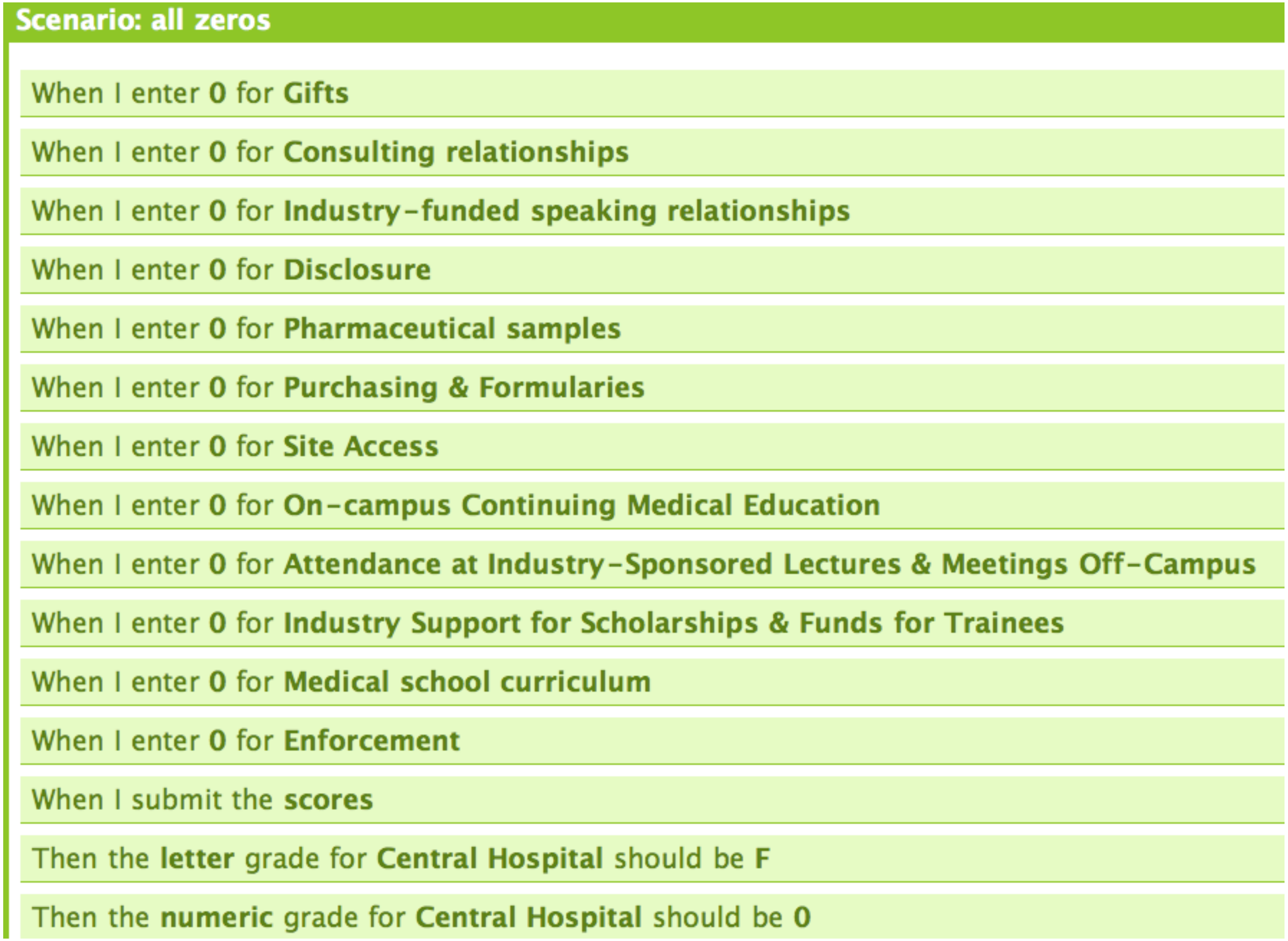
The Enforcement domain is not incorporated into the final score.

Final score is calculated as the total cumulative score for all domains (excluding Enforcment) divided by total possible score x 100.

(Total possible score = 15, unless a domain is classified as N/A.)

Grades are assigned as follows:

A >= 85%
B >= 70%
C >= 60%
D >= 40%
F < 40%



Scenario: all zeros

When I enter **0** for **Gifts**

When I enter **0** for **Consulting relationships**

When I enter **0** for **Industry-funded speaking relationships**

When I enter **0** for **Disclosure**

When I enter **0** for **Pharmaceutical samples**

When I enter **0** for **Purchasing & Formularies**

When I enter **0** for **Site Access**

When I enter **0** for **On-campus Continuing Medical Education**

When I enter **0** for **Attendance at Industry-Sponsored Lectures & Meetings Off-Campus**

When I enter **0** for **Industry Support for Scholarships & Funds for Trainees**

When I enter **0** for **Medical school curriculum**

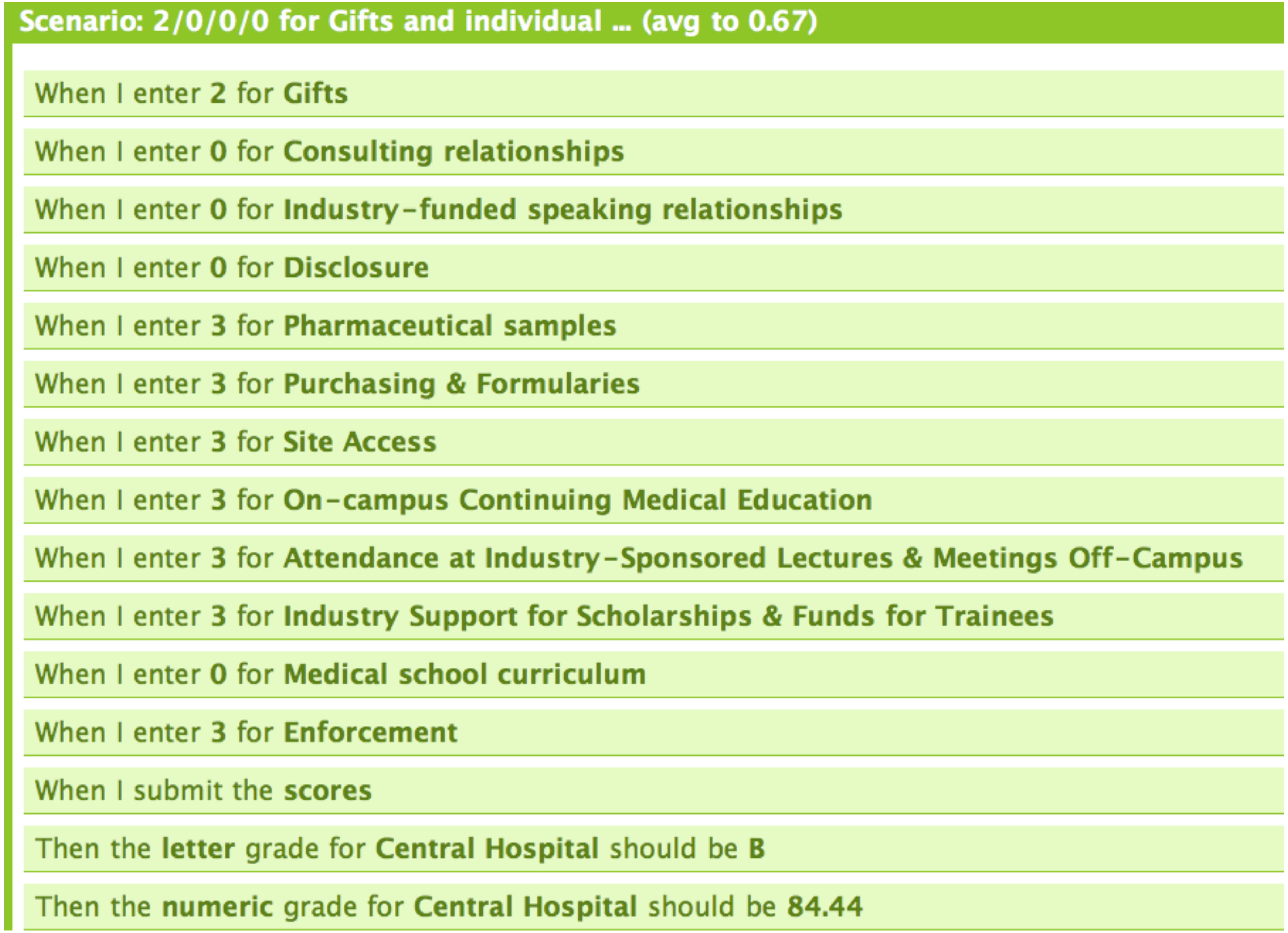
When I enter **0** for **Enforcement**

When I submit the **scores**

Then the **letter** grade for **Central Hospital** should be **F**

Then the **numeric** grade for **Central Hospital** should be **0**

```ruby
Then(/the (letter|numeric) grade for (.*) should be (.*)/) do
  |grade_type, institution_name, grade|
  institution = get_ivar("institution", institution_name)
  case grade_type
  when "letter"
    institution.letter_grade.should == grade
  when "numeric"
    institution.numeric_grade.should be_close(grade.to_f, 0.01)
  end
end
```



Scenario: 2/0/0/0 for Gifts and individual ... (avg to 0.67)

When I enter **2** for **Gifts**

When I enter **0** for **Consulting relationships**

When I enter **0** for **Industry-funded speaking relationships**

When I enter **0** for **Disclosure**

When I enter **3** for **Pharmaceutical samples**

When I enter **3** for **Purchasing & Formularies**

When I enter **3** for **Site Access**

When I enter **3** for **On-campus Continuing Medical Education**

When I enter **3** for **Attendance at Industry-Sponsored Lectures & Meetings Off-Campus**

When I enter **3** for **Industry Support for Scholarships & Funds for Trainees**

When I enter **0** for **Medical school curriculum**

When I enter **3** for **Enforcement**

When I submit the **scores**

Then the **letter** grade for **Central Hospital** should be **B**

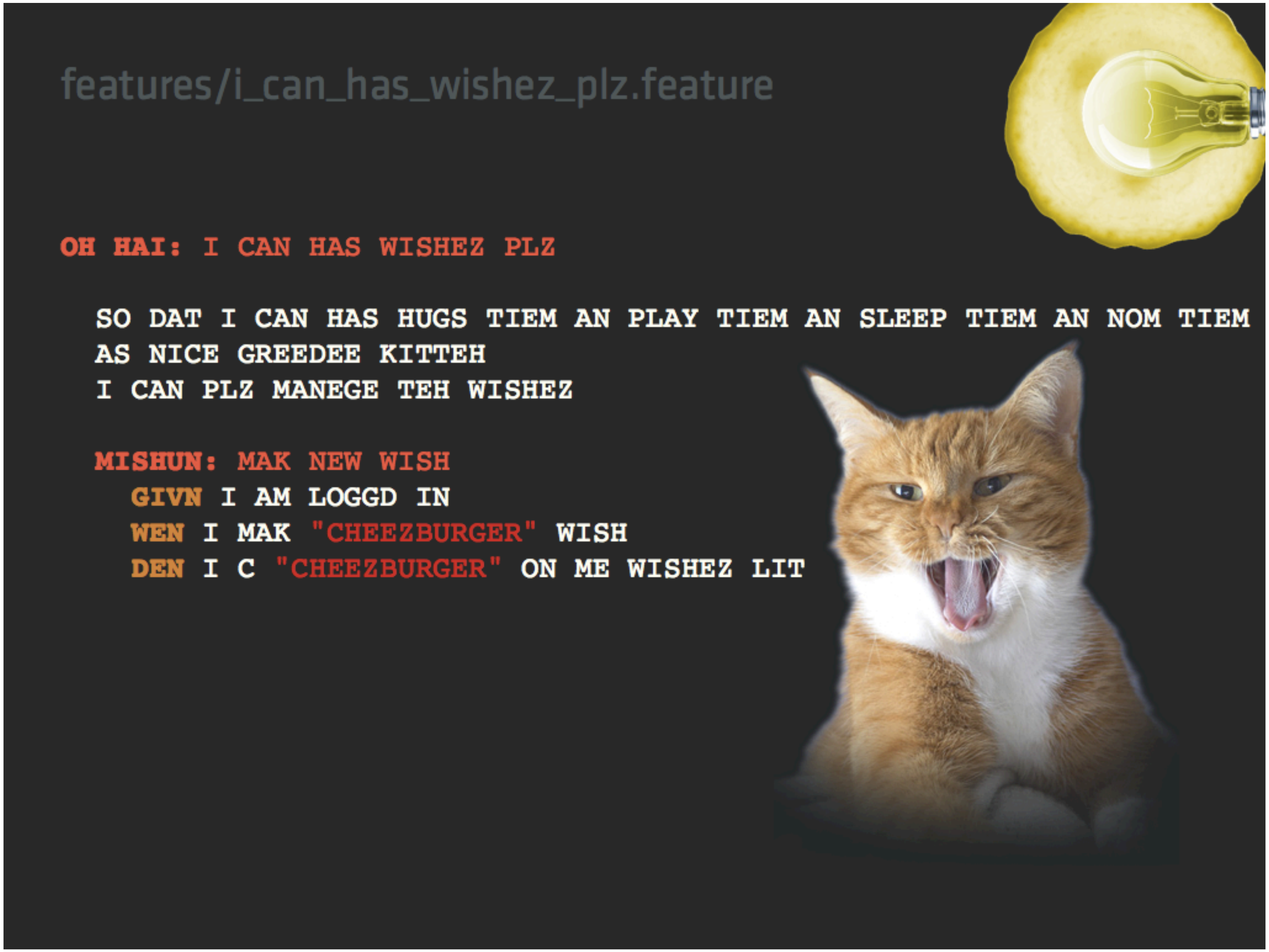
Then the **numeric** grade for **Central Hospital** should be **84.44**

Cucumber

- *Aslak Hellesøy's rewrite of RBehave/ Story Runner*
  - *Built on Treetop, a Ruby Parser Library*
    - *very developer-friendly*
    - *supports multiple spoken languages*
    - *languages are easily added*

```
Feature: code-breaker submits guess

  The code-breaker submits a guess of four colored
  pegs. The mastermind game marks the guess with black
  and white "marker" pegs.

  For each peg in the guess that matches color
  and position of a peg in the secret code, the
  mark includes one black peg. For each additional
  peg in the guess that matches the color but not
  the position of a color in the secret code, a
  white peg is added to the mark.

  Scenario Outline: submit guess
    Given the secret code is <code>
    When I guess <guess>
    Then the mark should be <mark>

  Scenarios: all colors correct
    | code    | guess   | mark |
    | r g y c | r g y c | bbbb |
    | r g y c | r g c y | bbww |
    | r g y c | y r g c | bwww |
    | r g y c | c r g y | wwww |
Given /^the secret code is (. . . .)$/ do |code|
  game.start(code.split)
end

When /^I guess (. . . .)$/ do |code|
  game.guess(code.split)
end

Then /^the mark should be (.*)$/ do |mark|
  messages_should_include(mark)
end
```

```
Característica: mede evolução das inscrições em relação ao objetivo final
  Como um organizador de conferéncias
  Eu quero um relatorio de inscrições
  Para que eu possa medir evolução

  Cenário: um inscrição mostra 1%
    Dado um alvo de 200 inscrições
    Quando 1 participante se inscreve
    Então 1% do alvo foi atingido

  Cenário: um inscrição menos que o alvo mostra 99%
    Dado um alvo de 200 inscrições
    Quando 199 participantes se inscrevem
    Então 99% do alvo foi atingido
```



features/i_can_has_wishez_plz.feature

```
OH HAI: I CAN HAS WISHEZ PLZ

  SO DAT I CAN HAS HUGS TIEM AN PLAY TIEM AN SLEEP TIEM AN NOM TIEM
  AS NICE GREEDEE KITTEH
  I CAN PLZ MANEGE TEH WISHEZ

  MISHUN: MAK NEW WISH
    GIVN I AM LOGGD IN
    WEN I MAK "CHEEZBURGER" WISH
    DEN I C "CHEEZBURGER" ON ME WISHEZ LIT
```

http://mwrc2009.confreaks.com/14-mar-2009-15-00-bdd-with-cucumber-ben-mabey.html

Environmental Factors

Who writes the step definitions?

Team Structure

- *QC or Testing Team?*
  - *Can they code?*
- *All developers?*

```
Feature: attendee registers

  As a potential attendee
  I want to register for a conference
  So that I may attend and learn great stuff

  Scenario: successful registration
    Given a conference named ETEC
    And   I am on the registration form

    When  I fill in "Name" with "Joe Smith"
    And   I fill in "E-Mail" with "jsmith@site.com"
    And   I select "BDD" from "Conferences"
    And   I check "Tutorials"
    And   I press "Register"

    Then  I should see the Registration Confirmation
    And   I should see "Conference: BDD"
    And   I should see "Name: Joe Smith"
    And   I should see "E-Mail: jsmith@site.com"
    And   I should see "Tutorials: Yes"
```

```ruby
Given /^I am on (.+)$/ do |page_name|
  visit path_to(page_name)
end

When /^I fill in "([^\"]*)" with "([^\"]*)"$/ do |field, value|
  fill_in(field, :with => value)
end

When /^I select "([^\"]*)" from "([^\"]*)"$/ do |value, field|
  select(value, :from => field)
end

When /^I press "([^\"]*)"$/ do |button|
  click_button(button)
end

Then /^I should see "([^\"]*)"$/ do |text|
  response.should contain(text)
end
```

```
Feature: attendee registers
  As a potential attendee
  I want to register for a conference
  So that I may attend and learn great stuff

  Scenario: successful registration
    When I successfully register for a conference
    Then I should a registration confirmation
    And it should show the conference name and date
    And it should show my name and email
```

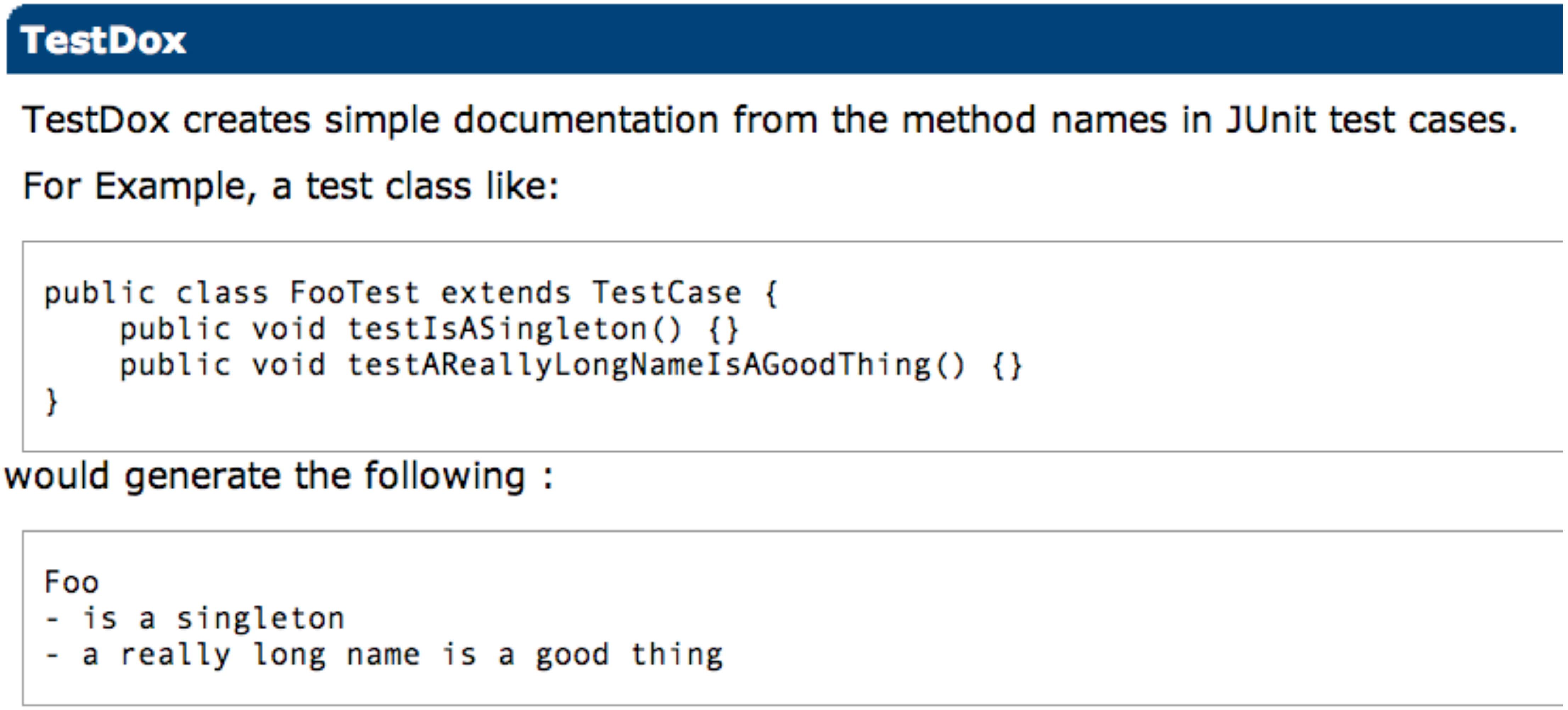
```ruby
When /^I successfully register for a conference$/ do
  Conference.create!(:name => "ETEC", :date => Date.new(2009, 3, 26))
  visit new_registration_path
  fill_in :name, :with => "Joe Smith"
  fill_in :email, :with => "joe@smith.com"
  select "ETEC", :from => "Conferences"
  click_button "Register"
end

Then /^I should a registration confirmation$/ do
  response.should contain("Registration Confirmation")
end

And /^it should show the conference name and date$/ do
  response.should have_tag(".conference_name", "ETEC")
  response.should have_tag(".conference_date", Date.new(2009, 3, 26).to_s)
end

And /^it should show my name and email$/ do
  response.should have_tag(".attendee_name", /Joe Smith/)
  response.should have_tag(".attendee_email", /joe@smith.com/)
end
```

Trust

Audience: Developers

TestDox

- *Turns java method names into sentences*
  - *by Chris Stevenson*



TestDox

TestDox creates simple documentation from the method names in JUnit test cases.

For Example, a test class like:

```
public class FooTest extends TestCase {
    public void testIsASingleton() {}
    public void testAReallyLongNameIsAGoodThing() {}
}
```

would generate the following :

```
Foo
- is a singleton
- a really long name is a good thing
```

http://agiledox.sourceforge.net/

DocTest

- *Runs python in-line documentation as tests*
  - *Tim Peters*

```python
def factorial(n):
    """Return the factorial of n, an exact integer >= 0.

    If the result is small enough to fit in an int, return an int.
    Else return a long.

    >>> [factorial(n) for n in range(6)]
    [1, 1, 2, 6, 24, 120]
    >>> [factorial(long(n)) for n in range(6)]
    [1, 1, 2, 6, 24, 120]
    >>> factorial(30)
    """


    import math
    if not n >= 0:
        raise ValueError("n must be >= 0")
    if math.floor(n) != n:
        raise ValueError("n must be exact integer")
    if n+1 == n:  # catch a value like 1e300
        raise OverflowError("n too large")
    result = 1
    factor = 2
    while factor <= n:
        result *= factor
        factor += 1
    return result
```

RSpec

RSpec

- *The Original Ruby BDD Framework*
  - *by Steven Baker*
- *inspired by Dave Astels' blog post "A New Look at Test Driven Development"*
- *Experiment in molding language to better support the process of TDD*

http://techblog.daveastels.com/2005/07/05/a-new-look-at-test-driven-development/

RSpec

- *Sapir/Whorf Hypothesis*
  - *postulates a systematic relationship between the grammatical categories of the language a person speaks and how that person both understands the world and behaves in it.*

the words we use impact the way we think

Sapir/Whorf-ish

```ruby
class EmptyMovieListTest < Test::Unit::TestCase

  def setup
    @list = MovieList.new
  end

  def test_should_have_size_of_0
    assert_equal 0, @list.size
  end

  def test_should_not_include_star_wars
    assert !@list.include?("Star Wars")
  end

end
```

```ruby
class EmptyMovieList < Spec::Context

  def setup
    @list = MovieList.new
  end

  def should_have_size_of_0
    @list.size.should_equal 0
  end

  def should_not_include_star_wars
    @list.should_not_include "Star Wars"
  end

end
```

```ruby
context MovieList, "when empty" do

  before(:each) do
    @list = MovieList.new
  end

  specify "should have a size of 0" do
    @list.size.should == 0
  end

  specify "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end

end
```

```
alias :describe, :context
alias :it, :specify
```

```ruby
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  it "should have a size of 0" do
    @list.size.should == 0
  end

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

```
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  it "should have a size of 0" do
    @list.size.should == 0
  end

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

```ruby
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  it "should be empty" do
    @list.size.should == 0
  end

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

```
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  it "should be empty" do
    @list.should be_empty
  end

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

```ruby
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  it "should be empty" do
    @list.should be_empty
    # => passes if @list.empty?
  end

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

```
MovieList when empty
  should be empty
  should not include 'Star Wars'

Finished in 0.001719 seconds

2 examples, 0 failures
```

```
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  it "should be empty" do
    @list.should be_empty
    # => passes if @list.empty?
  end

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

Duplication

```
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  specify { @list.should be_empty }

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

```
MovieList when empty
  should be empty
  should not include 'Star Wars'

Finished in 0.001719 seconds

2 examples, 0 failures
```

```ruby
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  specify { @list.should be_empty }

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

```
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  specify { @list.should be_empty }

  it "should not include 'Star Wars'" do
    @list.should_not include('StarWars')
  end
end
```

Duplication

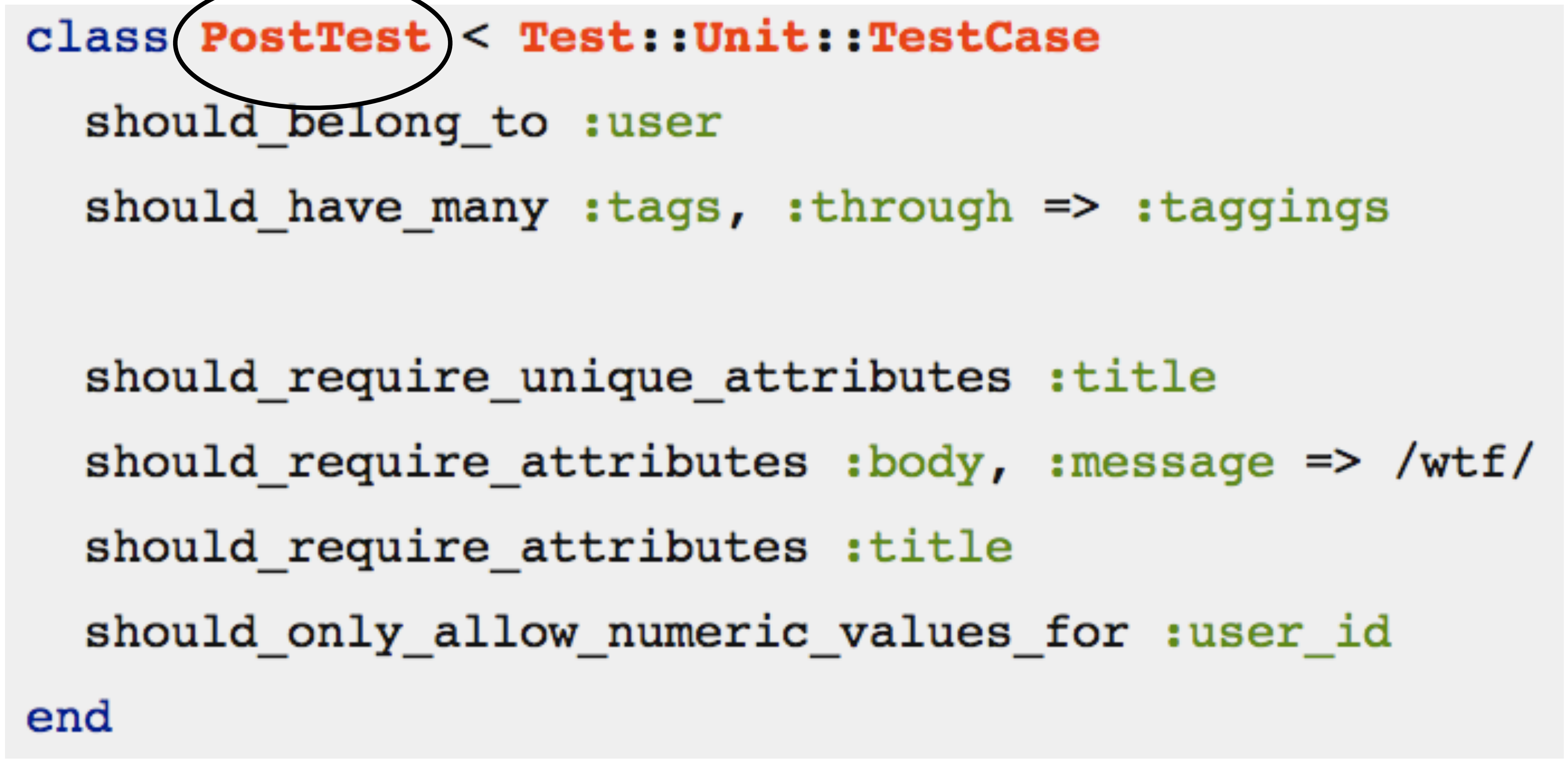
```ruby
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  specify { @list.should be_empty }

  specify { @list.should_not include('StarWars') }
end
```

```
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  specify { @list.should be_empty }

  specify { @list.should_not include('StarWars') }
end
```

```ruby
class PostTest < Test::Unit::TestCase
  should_belong_to :user
  should_have_many :tags, :through => :taggings

  should_require_unique_attributes :title
  should_require_attributes :body, :message => /wtf/
  should_require_attributes :title
  should_only_allow_numeric_values_for :user_id
end
```

http://www.thoughtbot.com/projects/shoulda/

Infers Post from PostTest



```
class PostTest < Test::Unit::TestCase
  should_belong_to :user
  should_have_many :tags, :through => :taggings

  should_require_unique_attributes :title
  should_require_attributes :body, :message => /wtf/
  should_require_attributes :title
  should_only_allow_numeric_values_for :user_id
end
```

http://www.thoughtbot.com/projects/shoulda/

```ruby
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  specify { @list.should be_empty }

  specify { @list.should_not include('StarWars') }
end
```

No inference necessary!

```
describe MovieList, "when empty" do
  before(:each) do
    @list = MovieList.new
  end

  specify { @list.should be_empty }

  specify { @list.should_not include('StarWars') }
end
```

No inference necessary!

```
describe MovieList, "when empty" do
  before(:each) do
   @list = MovieList.new
  end

  specify { @list.should be_empty }

  specify { @list.should_not include('StarWars') }
end
```

```
describe MovieList, "when empty" do
  specify { should be_empty }
  specify { should_not include('StarWars') }
end
```

Delegate to implicit subject

```
describe MovieList, "when empty" do
  specify { should be_empty }
  specify { should_not include('StarWars') }
end
```

```
describe MovieList, "when empty" do
  specify { should be_empty }
  specify { should_not include('StarWars') }
end
```

```
describe MovieList, "when empty" do
  it { should be_empty }
  it { should_not include('StarWars') }
end
```

```
describe MovieList do
  context "when empty" do
    it { should be_empty }
    it { should_not include('StarWars') }
  end
end
```

```
MovieList
  when empty
    should be empty
    should not include "StarWars"

Finished in 0.0021 seconds

2 examples, 0 failures
```

```
describe MovieList do
  context "when empty" do
    it { should be_empty }
    it { should_not include('StarWars') }
  end
end
```

exclude?

```ruby
describe MovieList do
  context "when empty" do
    it { should be_empty }
    it { should exclude('StarWars') }
  end
end
```

```
Spec::Matchers.create :exclude do |element|
  match do |list|
    !list.include?(element)
  end
end
```

```
MovieList
  when empty
    should be empty
    should exclude StarWars

Finished in 0.001994 seconds

2 examples, 0 failures
```

```
MovieList
  when empty
    should be empty
    should exclude StarWars (FAILED - 1)

1)
'MovieList when empty should exclude StarWars' FAILED
expected #<MovieList:0x1a5f4b4> to exclude StarWars
./movie_list_spec.rb:117:

Finished in 0.001975 seconds

2 examples, 1 failure
```

Impact on Process/Code

```ruby
describe Person do
  context "who is 18 years old" do
    subject {
      Person.new(:birthdate => 18.years.ago)
    }
    it { should be_eligible_to_vote }
  end
end
```

```
Person
  who is 18 years old
    should be eligible to vote (FAILED - 1)

1)
NoMethodError in 'Person who is 18 years old should be eligible to vote'
undefined method `eligible_to_vote?' for #<Person:0x17a4030>
./impact_spec.rb:17:

Finished in 0.005175 seconds

1 example, 1 failure
```

```
Person
  who is 18 years old
    should be eligible to vote (FAILED - 1)

1)
NoMethodError in 'Person who is 18 years old should be eligible to vote'
undefined method `eligible_to_vote?' for #<Person:0x17a4030>
./impact_spec.rb:17:

Finished in 0.005175 seconds

1 example, 1 failure
```

Encourages useful methods on objects

```ruby
def eligible_to_vote?
  Date.today - 18.years >= @birthdate
end
```

```
Person
  who is 18 years old
    should be eligible to vote

Finished in 0.004661 seconds

1 example, 0 failures
```

```
given an almost full stack
  when it receives #push
    then it adds the element to the top
  when it receives #pop
    then it returns the top element
    then it removes the top element
  when it receives #peek
    then it returns the top element
    then it leaves the top element
given a full stack
  when it receives #push
    then it raises a StackOverflow error
  when it receives #pop
    then it removes the top element
  when it receives #peek
    then it returns the top element
    then it leaves the top element
```

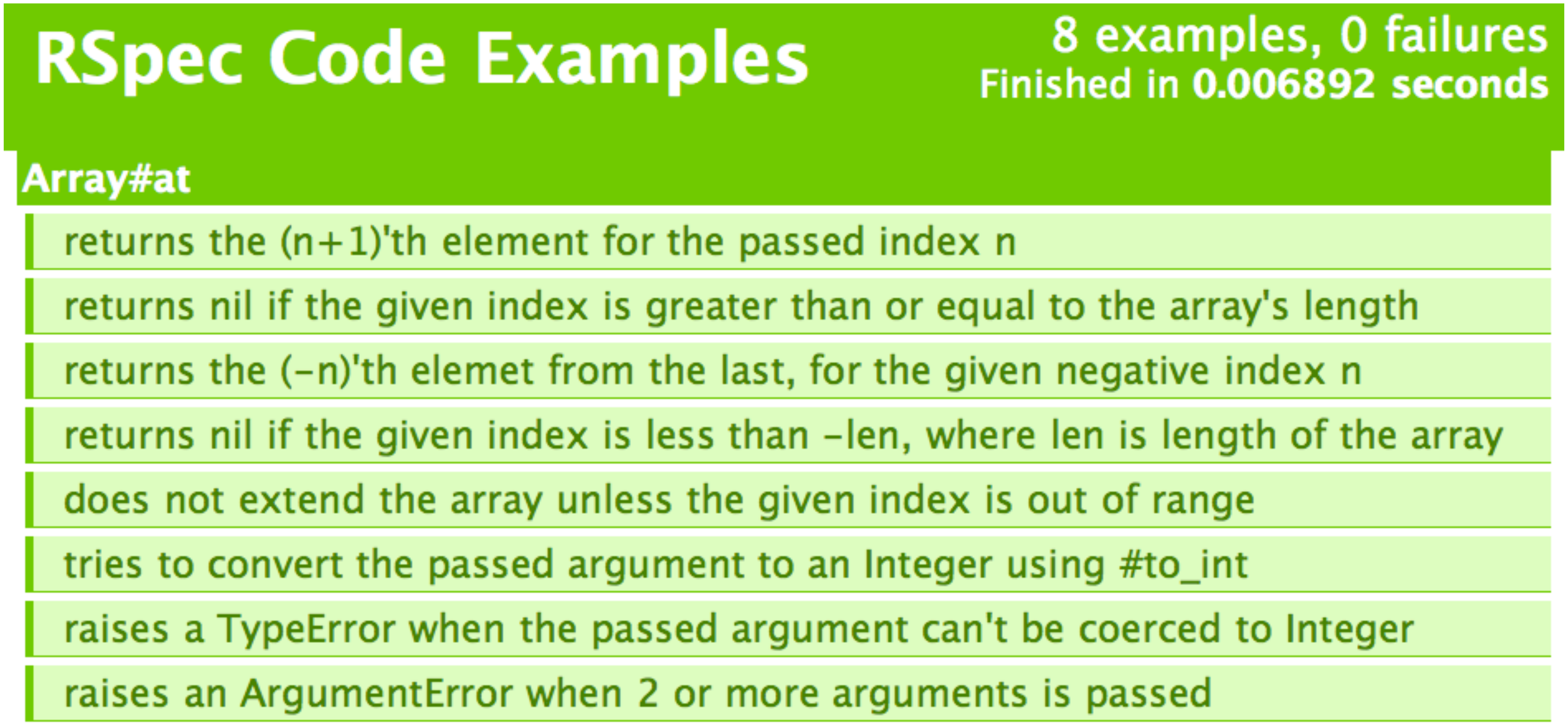
Rubinius

Ruby, written in Ruby

RubySpec

Executable Specification for the Ruby Programming Lanugage

```
describe "Array#at" do
  it "returns the (n+1)'th element for the passed index n" do
    a = [1, 2, 3, 4, 5, 6]
    a.at(0).should == 1
    a.at(1).should == 2
    a.at(5).should == 6
  end

  it "returns nil if the given index is greater than or equal to the array's length" do
    a = [1, 2, 3, 4, 5, 6]
    a.at(6).should == nil
    a.at(7).should == nil
  end

  it "returns the (-n)'th elemet from the last, for the given negative index n" do
    a = [1, 2, 3, 4, 5, 6]
    a.at(-1).should == 6
    a.at(-2).should == 5
    a.at(-6).should == 1
  end

  it "returns nil if the given index is less than -len, where len is length of the array"  do
    a = [1, 2, 3, 4, 5, 6]
    a.at(-7).should == nil
    a.at(-8).should == nil
  end
```



```
RSpec Code Examples                          8 examples, 0 failures
                                 Finished in 0.006892 seconds
Array#at
  returns the (n+1)'th element for the passed index n
  returns nil if the given index is greater than or equal to the array's length
  returns the (-n)'th elemet from the last, for the given negative index n
  returns nil if the given index is less than -len, where len is length of the array
  does not extend the array unless the given index is out of range
  tries to convert the passed argument to an Integer using #to_int
  raises a TypeError when the passed argument can't be coerced to Integer
  raises an ArgumentError when 2 or more arguments is passed
```

So What's Next?

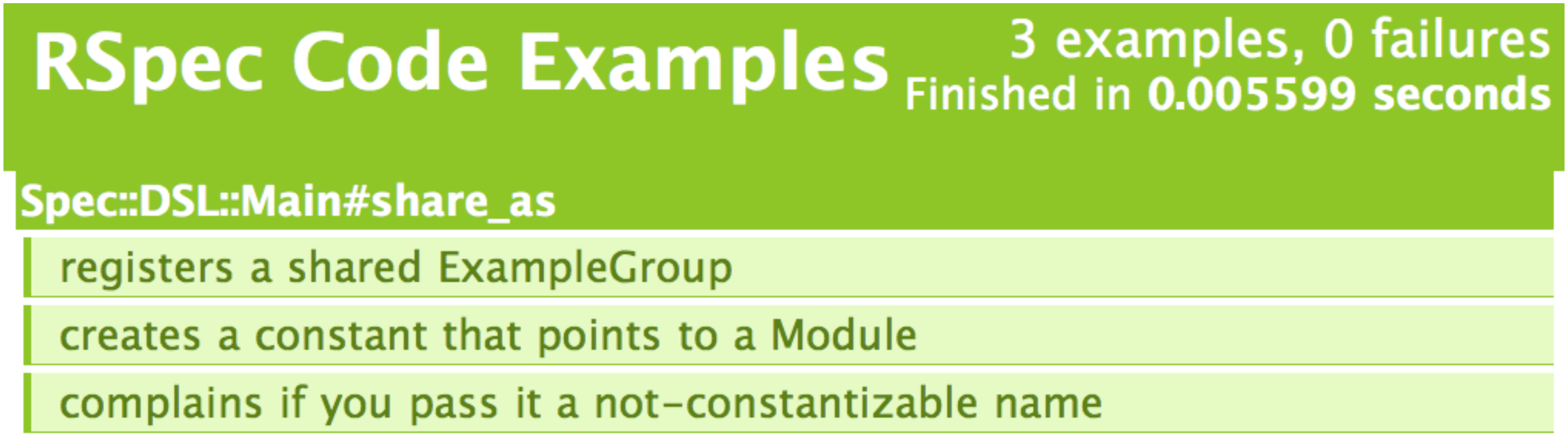
```
Expectations do

  # State based expectation where a value equals another value
  expect 2 do
    1 + 1
  end

  # State based expectation where an exception is expected...
  expect NoMethodError do
    Object.no_method
  end

  # Behavior based test using a traditional mock
  expect mock.to.receive(:dial).with("2125551212").times(2) do |phone|
    phone.dial("2125551212")
    phone.dial("2125551212")
  end
```

http://expectations.rubyforge.org/

```
Expectations F...................
Finished in 0.00173 seconds

Failure: 1 failed, 0 errors, 19 fulfilled

--Failures--
method expect in successes_test.rb at line 6
file
</Users/david/projects/ruby/expectations/trunk/test/successes_test.rb>
line <6>
expected: <2> got: <3>
```

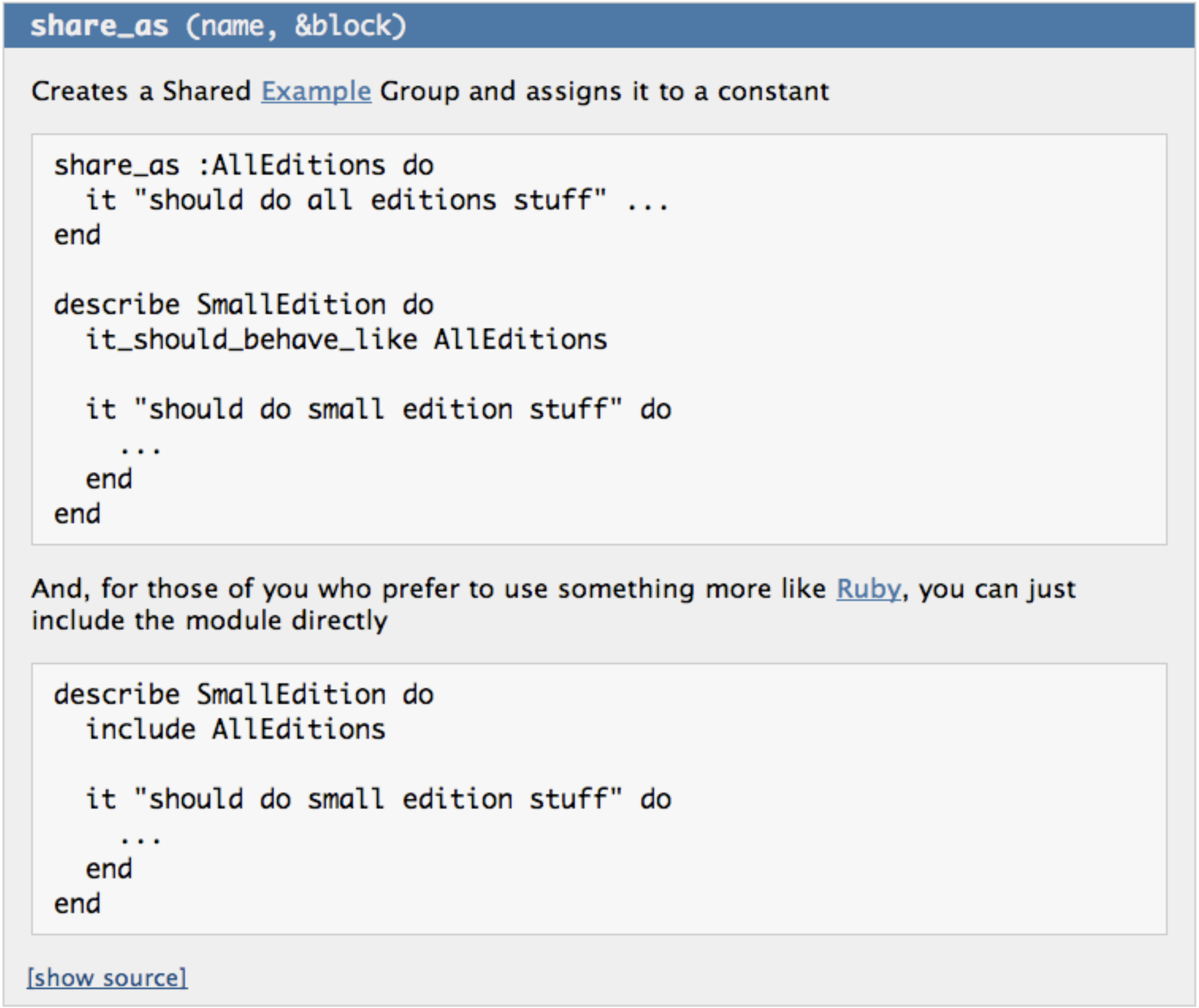
```
describe MovieList do
  context "when first created" do
    behaviour "should be empty" do

      expect true do
        list = List.new
        list.empty?
      end

      expect false do
        list = List.new
        list.include?('StarWars')
      end

    end
  end
end
```

```ruby
describe "#share_as" do
  def self.next_group_name
    @group_number ||= 0
    @group_number += 1
    "Group#{@group_number}"
  end

  def group_name
    @group_name ||= self.class.next_group_name
  end

  it "registers a shared ExampleGroup" do
    block = lambda {|a,b|}
    Spec::Example::ExampleGroupFactory.should_receive(:create_shared_example_group).with(
      group_name, hash_including(:location), &block
    )
    @main.share_as group_name, &block
  end

  it "creates a constant that points to a Module" do
    group = @main.share_as group_name do end
    Object.const_get(group_name).should equal(group)
  end

  it "complains if you pass it a not-constantizable name" do
    lambda do
      @group = @main.share_as "Non Constant" do end
    end.should raise_error(NameError, /The first argument to share_as must be a legal name for a constant/)
  end

end
```



```
RSpec Code Examples                3 examples, 0 failures
                      Finished in 0.005599 seconds
Spec::DSL::Main#share_as
  registers a shared ExampleGroup
  creates a constant that points to a Module
  complains if you pass it a not-constantizable name
```

```ruby
# Creates a Shared Example Group and assigns it to a constant
#
#  share_as :AllEditions do
#    it "should do all editions stuff" ...
#  end
#
#  describe SmallEdition do
#    it_should_behave_like AllEditions
#
#    it "should do small edition stuff" do
#      ...
#    end
#  end
#
# And, for those of you who prefer to use something more like Ruby, you
# can just include the module directly
#
#  describe SmallEdition do
#    include AllEditions
#
#    it "should do small edition stuff" do
#      ...
#    end
#  end
def share_as(name, &block)
  begin
    args = [name]
    Spec::Example::set_location(args, caller(0)[1])
    Object.const_set(name, Spec::Example::ExampleGroupFactory.create_shared_example_group(*args, &block))
  rescue NameError => e
    raise NameError.new(e.message + "\nThe first argument to share_as must be a legal name for a constant\n")
  end
end
```



share_as (name, &block)

Creates a Shared Example Group and assigns it to a constant

```
share_as :AllEditions do
  it "should do all editions stuff" ...
end

describe SmallEdition do
  it_should_behave_like AllEditions

  it "should do small edition stuff" do
    ...
  end
end
```

And, for those of you who prefer to use something more like Ruby, you can just include the module directly

```
describe SmallEdition do
  include AllEditions

  it "should do small edition stuff" do
    ...
  end
end
```

[show source]

```
rspec + rdoc == ???
```

???

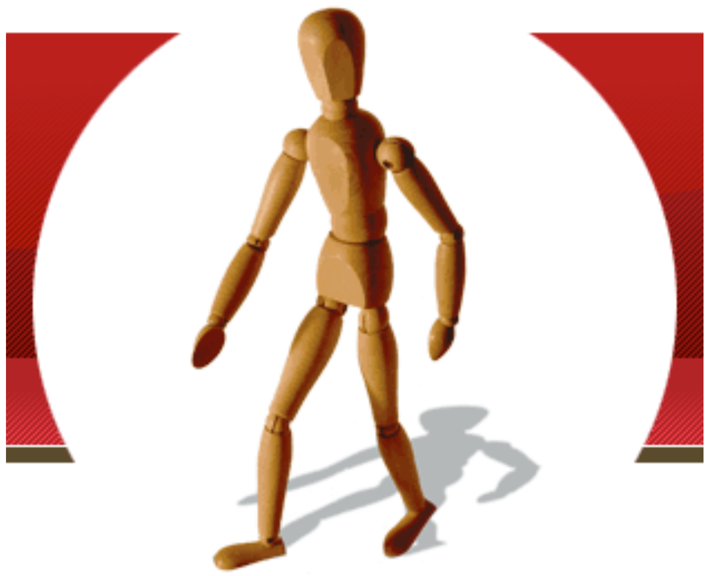
http://blog.davidchelimsky.net/

http://www.articulatedman.com/

http://rspec.info/

http://cukes.info/

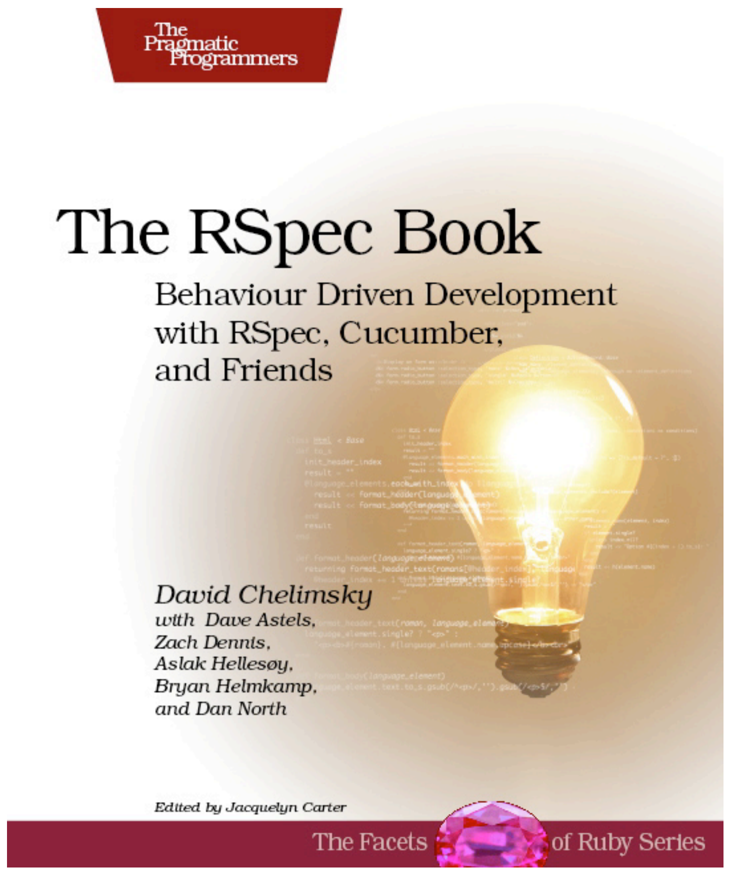
http://pragprog.com/titles/achbd/the-rspec-book



describe it with rspec

Cucumber

Behaviour Driven Development with elegance and joy



The Pragmatic Programmers

The RSpec Book

Behaviour Driven Development with RSpec, Cucumber, and Friends

David Chelimsky with Dave Astels, Zach Dennis, Aslak Hellesøy, Bryan Helmkamp, and Dan North

Edited by Jacquelyn Carter

The Facets of Ruby Series