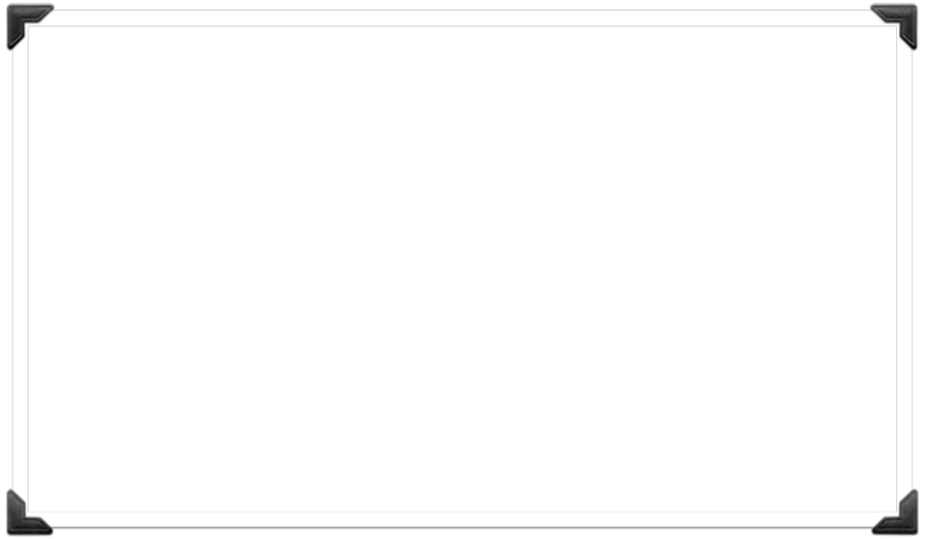
TDD & RSpec

Designing and Maintaining Software (DAMS)

Louis Rose

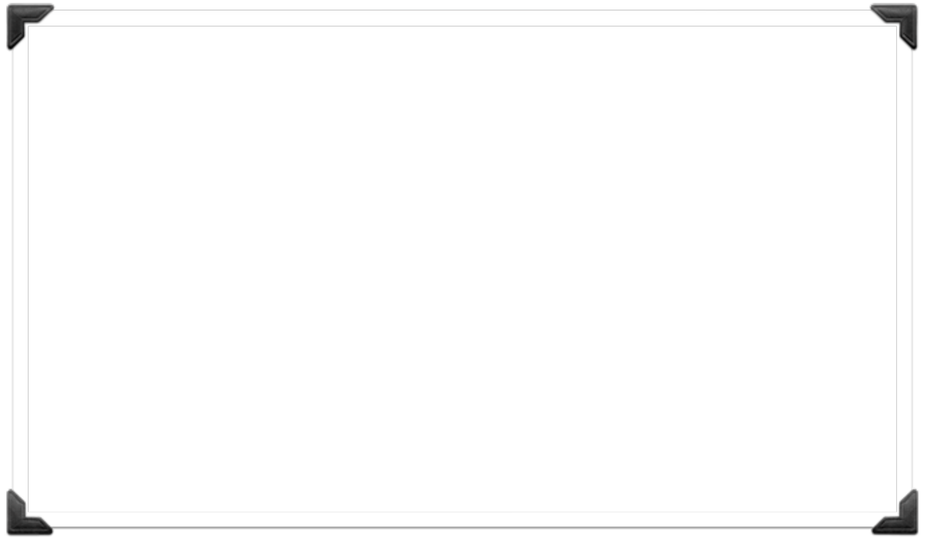


TDD in a nutshell

1. “Red” - Write a test that fails

2. “Green” - Write just enough code to pass that test

3. “Refactor” - Improve the habitability of the code

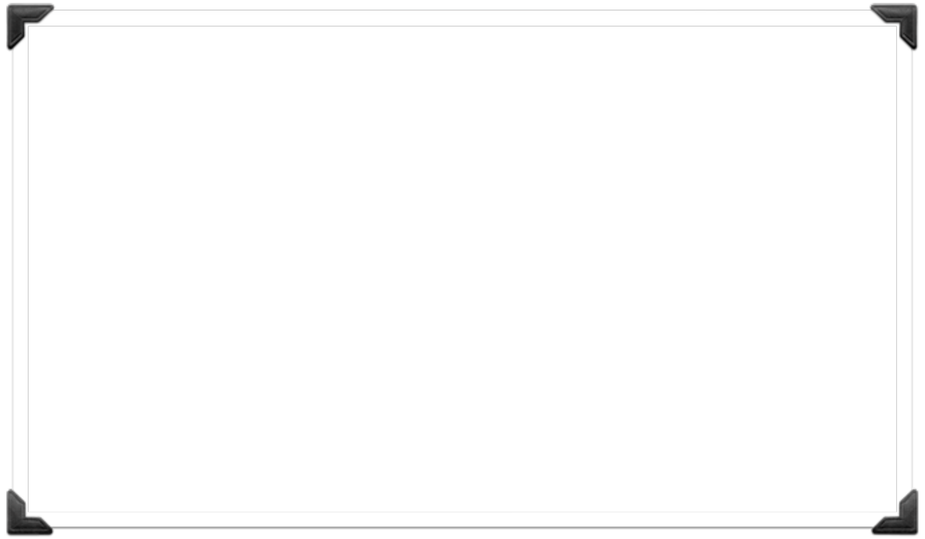


TDD as I do it today

1. “Red” - Write an **integration test** that fails

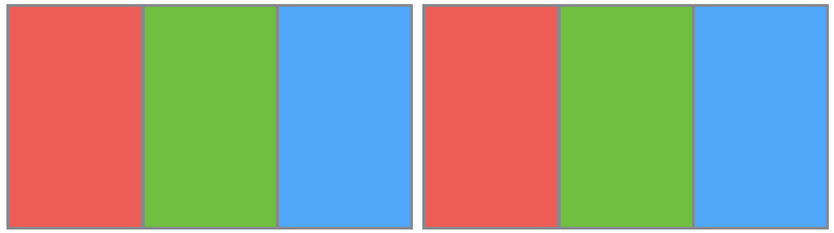
2. “Green” - Complete TDD loops until the test passes

3. “Refactor” - Improve the habitability of the code



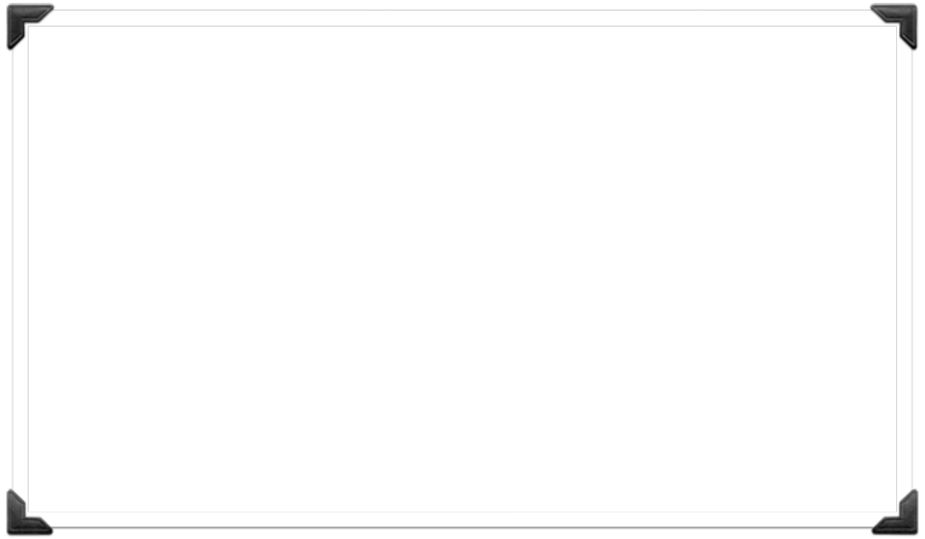
TDD as I do it today

1. “Red” - Write an **integration test** that fails



…

3. “Refactor” - Improve the habitability of the code



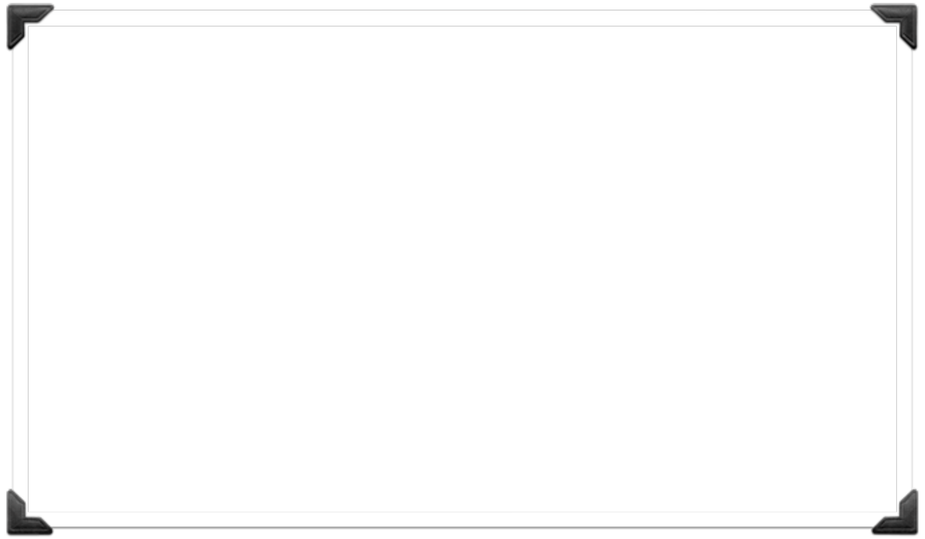
TDD benefits

We have a set of tests that we are confident in:

- Reduces defects
- Increases habitability

TDD is well aligned with some (of my) habitability factors:

- Encourages lean design
- Discourages complex and coupled code

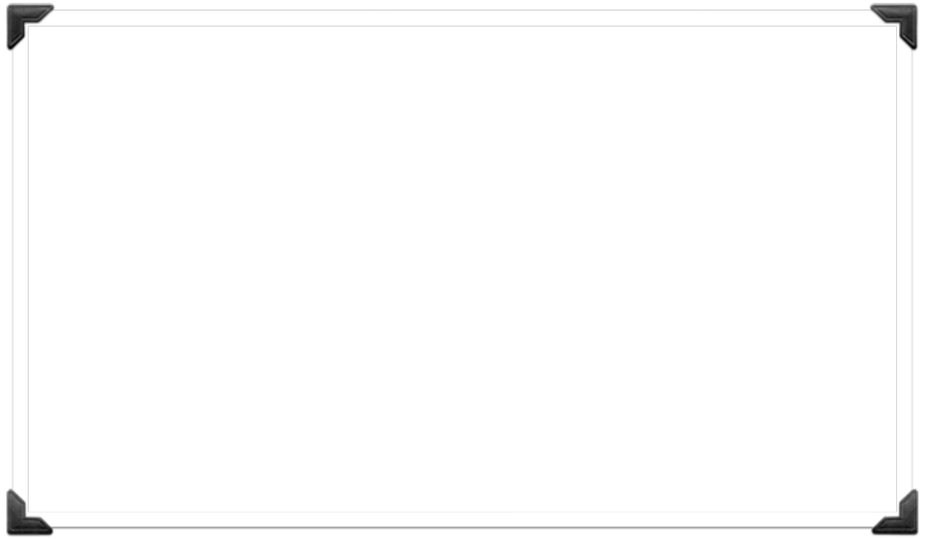


Ruby Testing Frameworks

3 popular options are: RSpec, Minitest and Test::Unit

We'll use RSpec, as it has the most comprehensive docs.

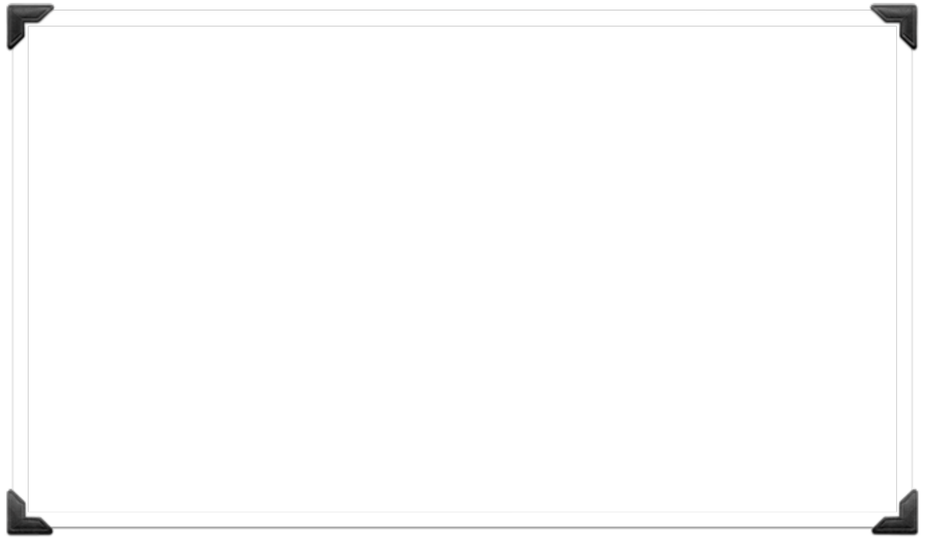
Introductory videos are at: http://rspec.info



An RSpec example

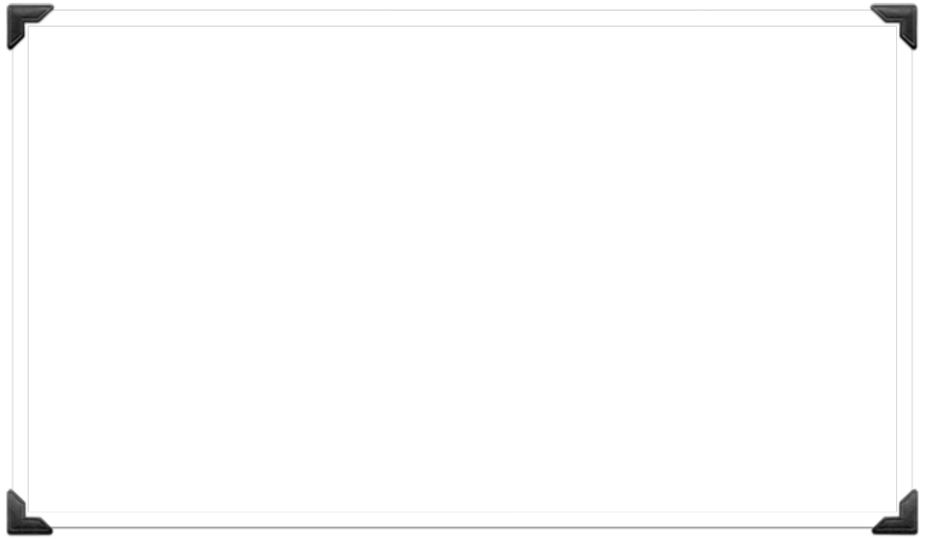
```
require "calculator/max"

module Calculator
  describe Max do
    it "returns correct answer for a tie" do
      expect(Max.new.run(4, 4)).to eq(4)
    end

    it "returns correct answer when first is larger" do
      expect(Max.new.run(4, 3)).to eq(4)
    end

    it "returns correct answer when last is larger" do
      expect(Max.new.run(3, 4)).to eq(4)
    end
  end
end
```

http://rspec.info/documentation/3.3/rspec-core/
http://rspec.info/documentation/3.3/rspec-expectations/



An RSpec example

```
% rspec
..F

Failures:

  1) Max returns correct answer when last is larger
     Failure/Error: expect(subject.run(3, 4)).to eq(4)

       expected: 4
            got: 3

       (compared using ==)
     # ./spec/calculator/max_spec.rb:14:in `block (2 levels) in <module:Calculator>'

Finished in 0.01108 seconds (files took 0.0841 seconds to load)
3 examples, 1 failure

Failed examples:

rspec ./spec/calculator/max_spec.rb:13 # Max returns correct answer when last is
larger
```



“We produce well-designed, well-tested, and well-factored code in small, verifiable steps.”

– James Shore
http://www.jamesshore.com/Agile-Book/test_driven_development.html