Test-Driven Development training

Are you ready?

- Install mob tool.

  See https://github.com/remotemobprogramming/mob under "how to install."

  (Under Windows? Use git bash to execute the curl command.)
- Set up codebase:
  - `git clone https://github.com/jlangr/jan2021_ruby_tdd`
  - `git clone git@github.com:jlangr/jan2021_ruby_tdd.git`
  - Make sure you can run the tests
- **Pull the latest source**
- Install Kahoot! on your phone (or not, and wait for browser links)
- Email me your cell #
- Send me your GitHub user id / email address
  - Accept the invitation to the `jan2021_ruby_tdd` repo
- Take note of my contact info below

Jeff Langr / @jlangr / jeff@Langrsoft.com / +1-719-287-GEEK

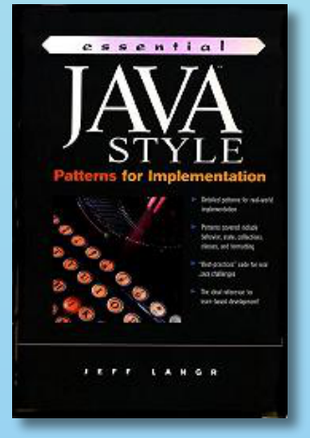
TDD Jump-Start in Ruby

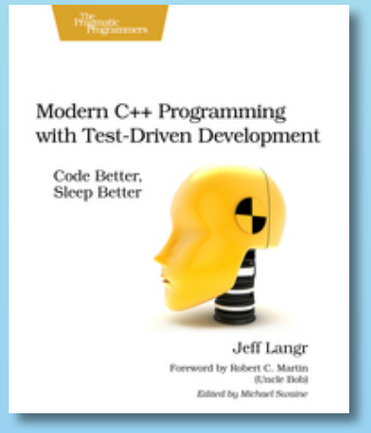
Jeff Langr

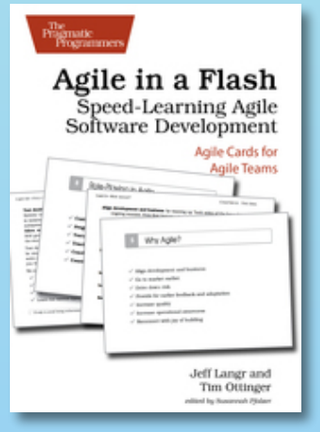
jeff@langrsoft.com
@jlangr

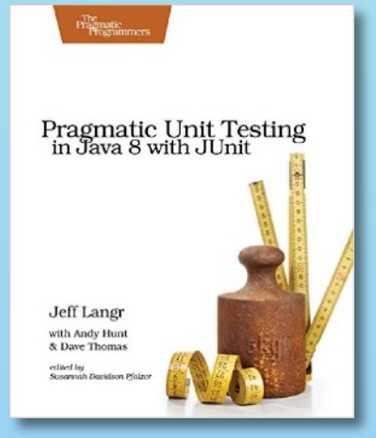
LANGR | SOFTWARE SOLUTIONS

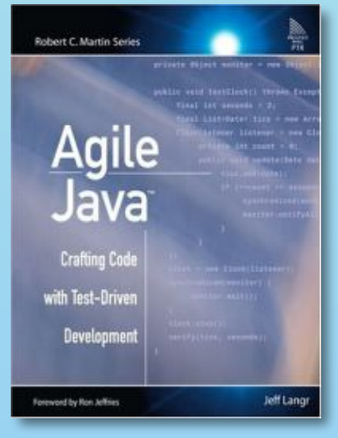
http://langrsoft.com

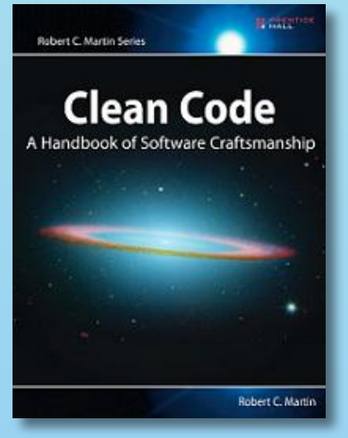












Exercise: Reading code

Describe the behaviors in ./lib/misc/n_normalizer_horror.rb

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Kahoot!

What's a unit test (“spec”) look like?

```
describe TargetClass do
  it "provides some behavior" do
    # Arrange

    # Act

    # Assert
  end
end
```

The AAA mnemonic is courtesy of Bill Wake.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

A sample unit test

```
describe Auto do
  it "idles engine when started" do
    subject.depress_brake

    subject.press_start_button

    expect(subject.RPM).to be_between(950, 1100)
  end
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

A sample unit test

```
describe Auto do
  subject(:auto) { Auto.new }

  it "idles engine when started" do
    auto.depress_brake

    auto.press_start_button

    expect(auto.RPM).to be_between(950, 1100)
  end
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

What do assertions look like?

```
expect(condition).to be true
expect(something).to eq “expected text"
expect(condition).to be true
expect(something).to eq "expected text"
expect(something).not_to eq "unexpected text"
expect("technology").to include "tech"
expect("something").to match /.*thing$/
expect { 4 / 0 }.to raise_error(ZeroDivisionError)
expect(some_collection).to be_empty
expect(obj).to be_instance_of(Auto)
expect([12, 1, 1, 1, 2, 1, 3]).to include(12, 3)
expect([2, 4, 8, 16, 32]).to all(be_even.and be < 64)
```

Where’re the docs?

- https://github.com/rspec/rspec-expectations
- https://relishapp.com/rspec/rspec-expectations/docs/built-in-matchers
- Use your editor’s intellisense
- Browse source (matchers.rb)

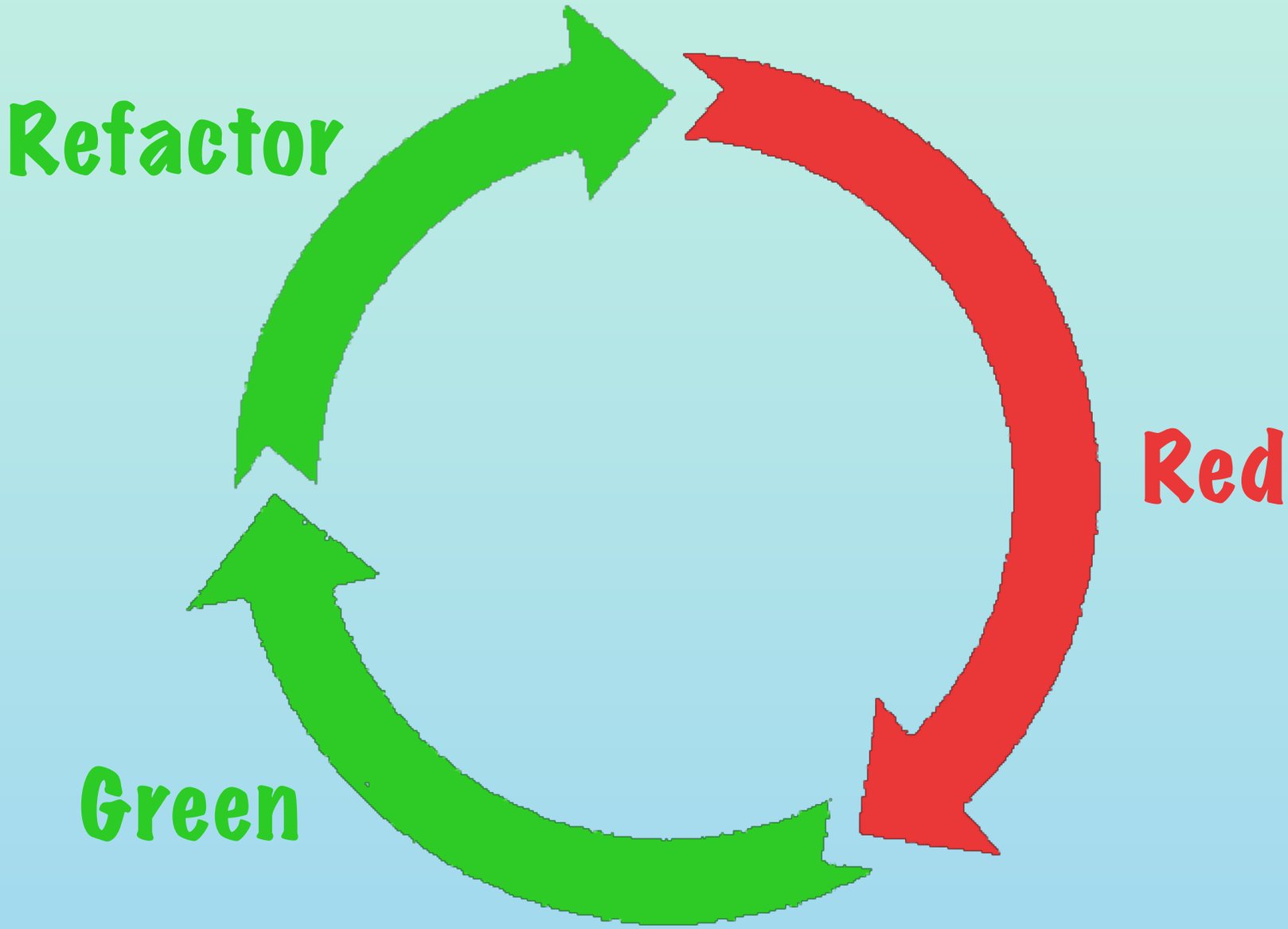
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: Simple outcomes

Flesh out the tests in ./spec/lib/misc/basics_spec.rb

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Test-Driven Development



An incremental design technique

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Mob programming

"All the brilliant people working on the same thing,
at the same time,
in the same place,
and on the same computer."

Woody Zuill

http://mobprogramming.org

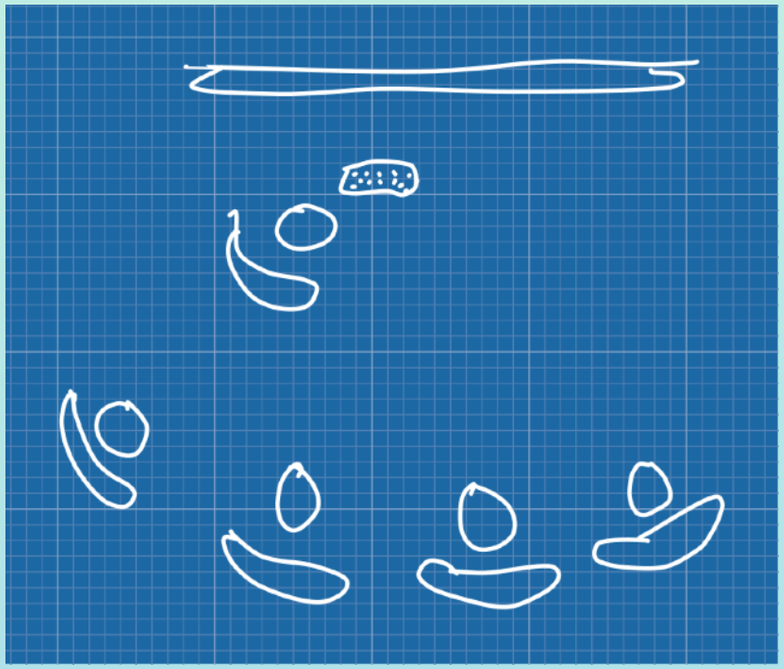
Short focused, timed rotations

Strong-style navigation

"We go faster."

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

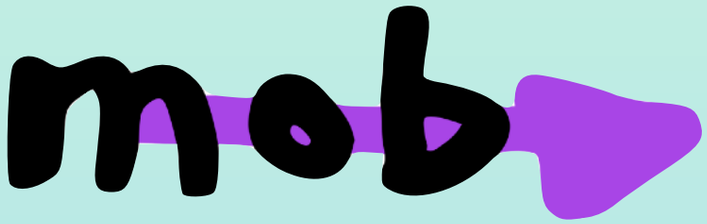
A tale of times past



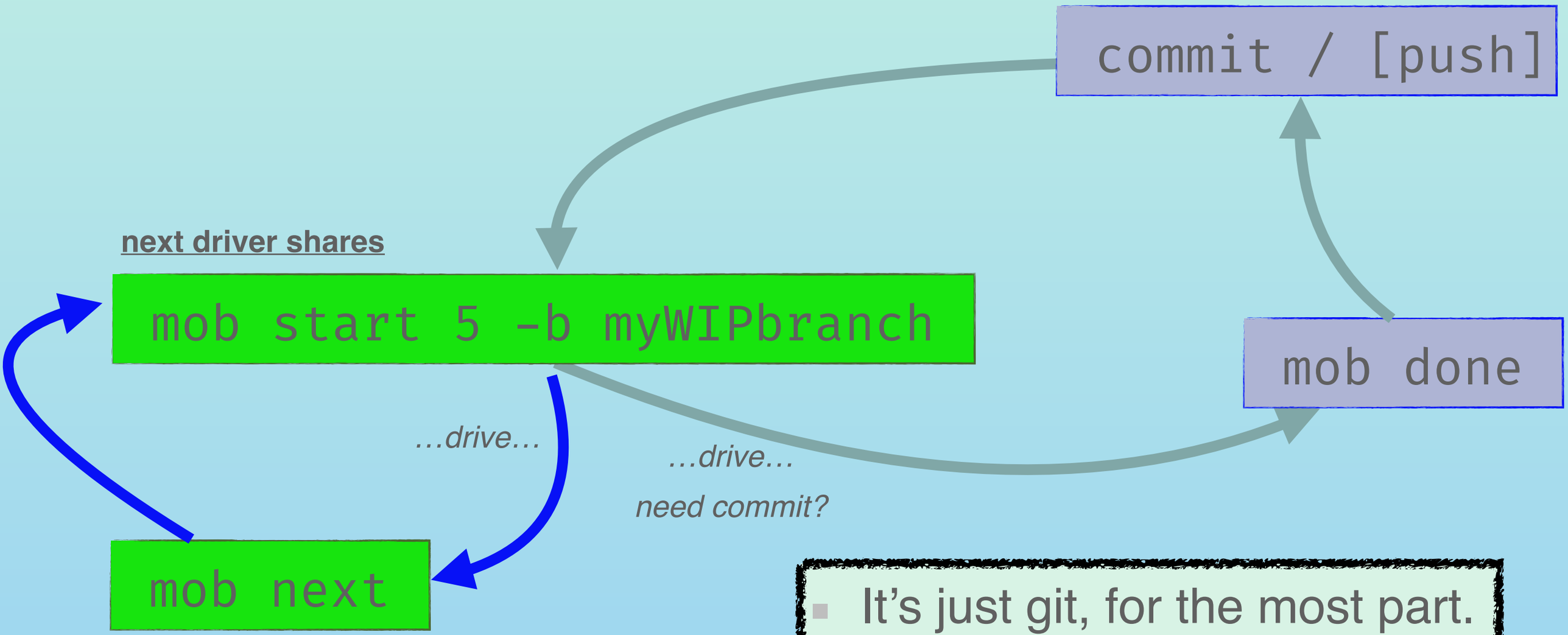
*Once upon a time, there existed a quaint notion of an "**office**," a physical place where people could un-distance and work physically face-to-face…*

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Remote mob programming via Git handover



http://github.com/remotemobprogramming/mob

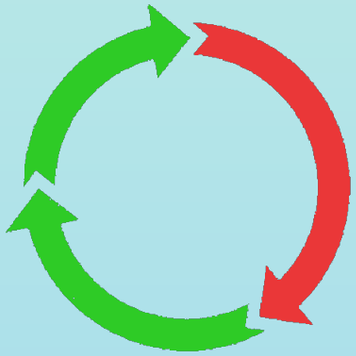


- It's just git, for the most part.
- `git status` often

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

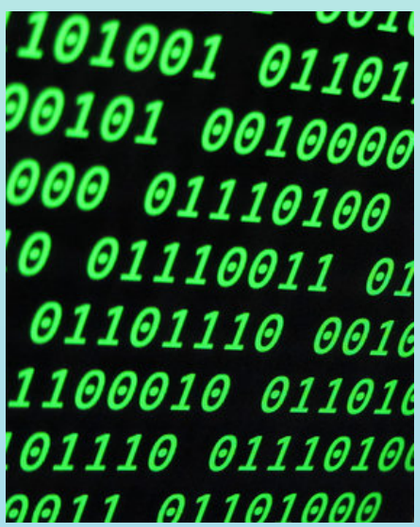
Exercise: (Test)-Code-Refactor

See ./spec/lib/misc/name_normalizer_spec.rb



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Why TDD?



Keep code malleable

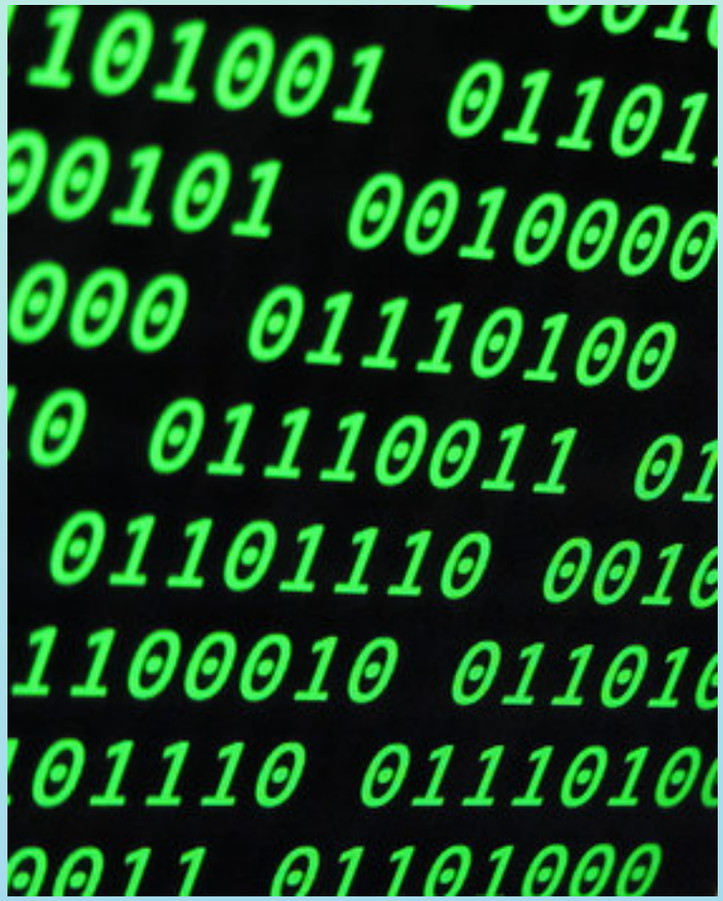
Document choices

Gate defects

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"BinaryCode," courtesy https://www.flickr.com/photos/132889348@N07/
License: https://creativecommons.org/licenses/by-sa/2.0/

Core TDD themes



- Drive *behavior*
- Specify by example
- Grow in small increments
- Stick to the cycle
  - Always see red
  - Always refactor

It's just code!

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"BinaryCode," courtesy https://www.flickr.com/photos/132889348@N07/
License: https://creativecommons.org/licenses/by-sa/2.0/

Kahoot!

DHH

Photo courtesy of @DHH's Twitter profile

"My name is David, and I do not write software test-first.

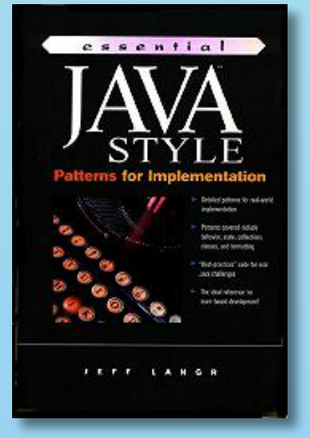
"I refuse to apologize for that any more, much less hide it. I'm grateful for what TDD did to open my eyes to automated regression testing, but I've long since moved on from the design dogma."

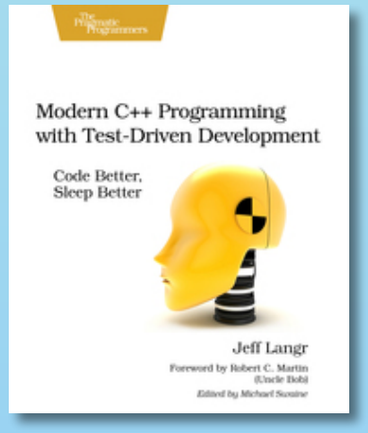
https://dhh.dk/2014/tdd-is-dead-long-live-testing.html

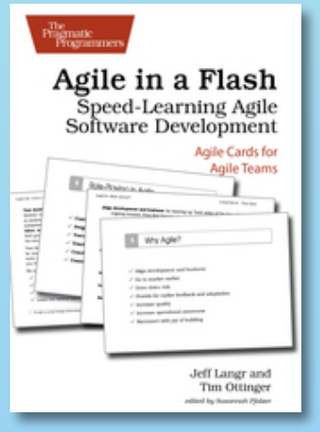
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

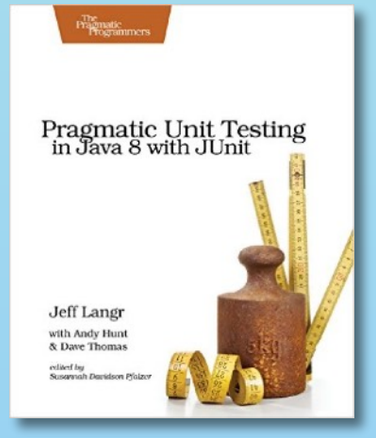
Stepping back: Writing tests

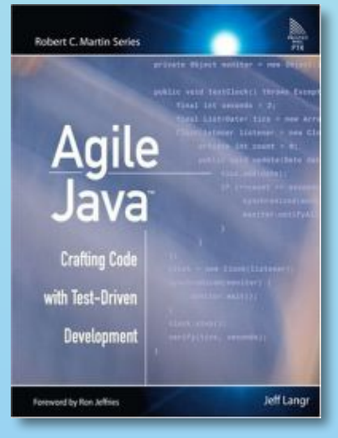
LANGR | SOFTWARE SOLUTIONS

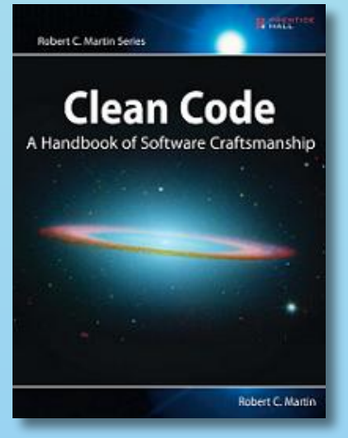












Recent file list

Create a list of recently-opened files (max: 5) for use in a GUI application.

What tests do we need?

```
class RecentFileList
  attr_reader :filenames
  def open(filename)
    # ...
  end
end
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Ordering tests

- **Z**ero
- **O**ne
- **M**any
- **B**oundaries
- **I**nterface definition
- **E**xceptional behavior
- **S**imple Solutions & Scenarios

ZOMBIES!

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

http://blog.wingman-sw.com/archives/677

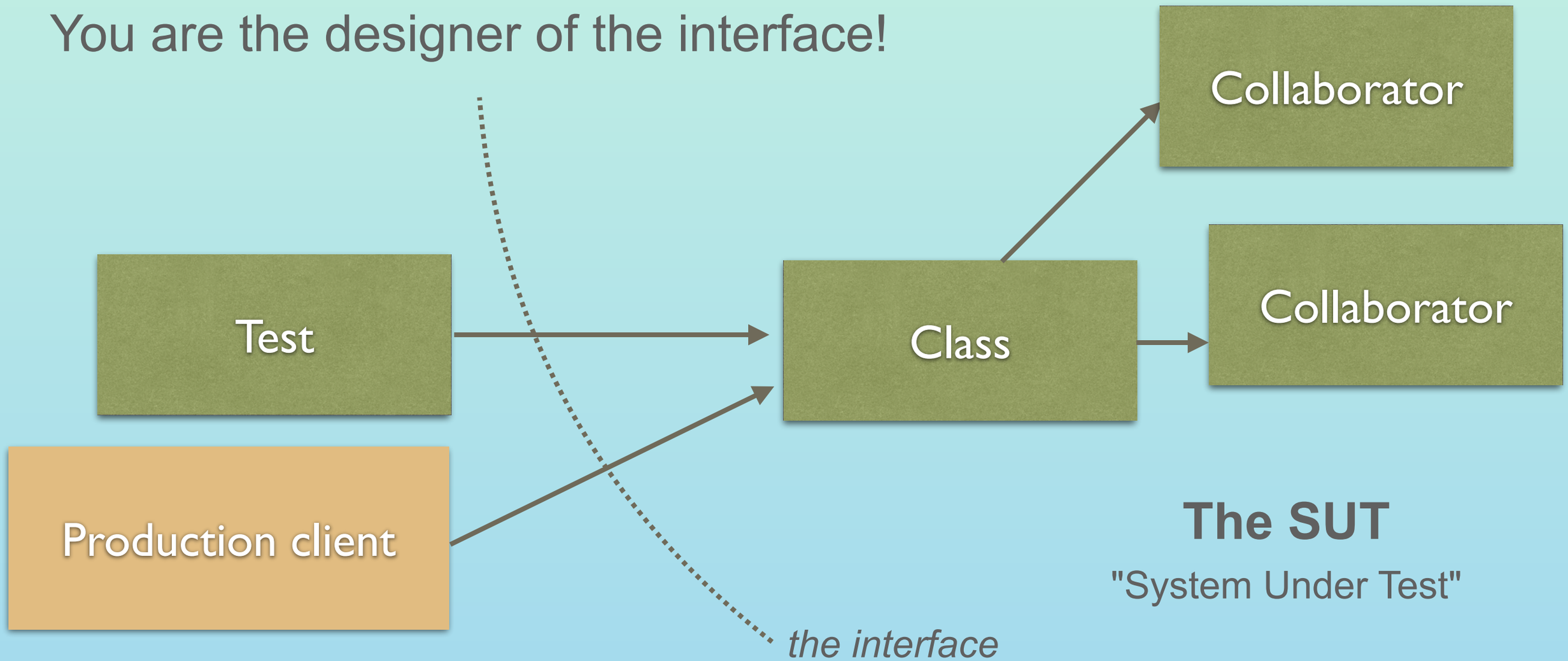
Test lists

- **Z**: No files opened
- **O**: One file opened
- **M**: Multiple files opened
- **B**: One more than capacity opened
- **E**: File opened twice

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Your test is a client

You are the designer of the interface!



The SUT

"System Under Test"

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Structuring the test

"sifts duplicate filename to top of recently-used list"

Arrange	Add file with name **opened-A.txt** Add file with name **opened-B.txt** Add file with name **opened-C.txt**
Act	Add file with name **opened-A.txt**
Assert	Expect recently-used list to equal: **opened-A.txt** **opened-C.txt** **opened-B.txt**

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

The test

```
it "sifts duplicate filename to top" do
  list.open("A")
  list.open("B")
  list.open("C")

  list.open("A")

  expect(list.filenames).to eq(%w[A C B])
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Assert-first

Consider working from the outcome.

"sifts duplicate filename to top of recently-used list"

```
expect(list.filenames).to eq([duplicate_of_first_opened, "C", "B"]
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Assert-first

Consider working from the outcome.

"sifts duplicate filename to top of recently-used list"

```
list.open(duplicate_of_first_opened)

expect(list.filenames).to eq([duplicate_of_first_opened, "C", "B"])
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Assert-first

Consider working from the outcome.

"sifts duplicate filename to top of recently-used list"

```
list.open("A")
list.open("B")
list.open("C")
duplicate_of_first_opened = "A"

list.open(duplicate_of_first_opened)

expect(list.filenames).to eq([duplicate_of_first_opened, "C", "B"])
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

AAA is a visual mnemonic

Don't do this:

```
it "sifts duplicate filename to top" do
  # arrange
  list.open("opened-A")
  list.open("opened-B")
  list.open("opened-C")

  # act
  list.open("opened-A")

  # assert
  expect(list.filenames).to eq(%w[opened-A opened-C opened-B])
end
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Let space do the talking

```
it "sifts duplicate filename to top" do
  list.open("opened-A")
  list.open("opened-B")
  list.open("opened-C")

  list.open("opened-A")

  expect(list.filenames).to eq(%w[opened-A opened-C opened-B])
end
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: Portfolio

- is it empty or not?
- what is the count of unique symbols?
- given a symbol and # of shares, make a purchase
- how many shares exist for a given symbol?
- given a symbol and # of shares, sell the shares
- throw an exception when selling too many shares

"Tokyo Stock Exchange," courtesy https://www.flickr.com/photos/31029865@N06/
License: https://creativecommons.org/licenses/by/2.0/

***** Remember! *****

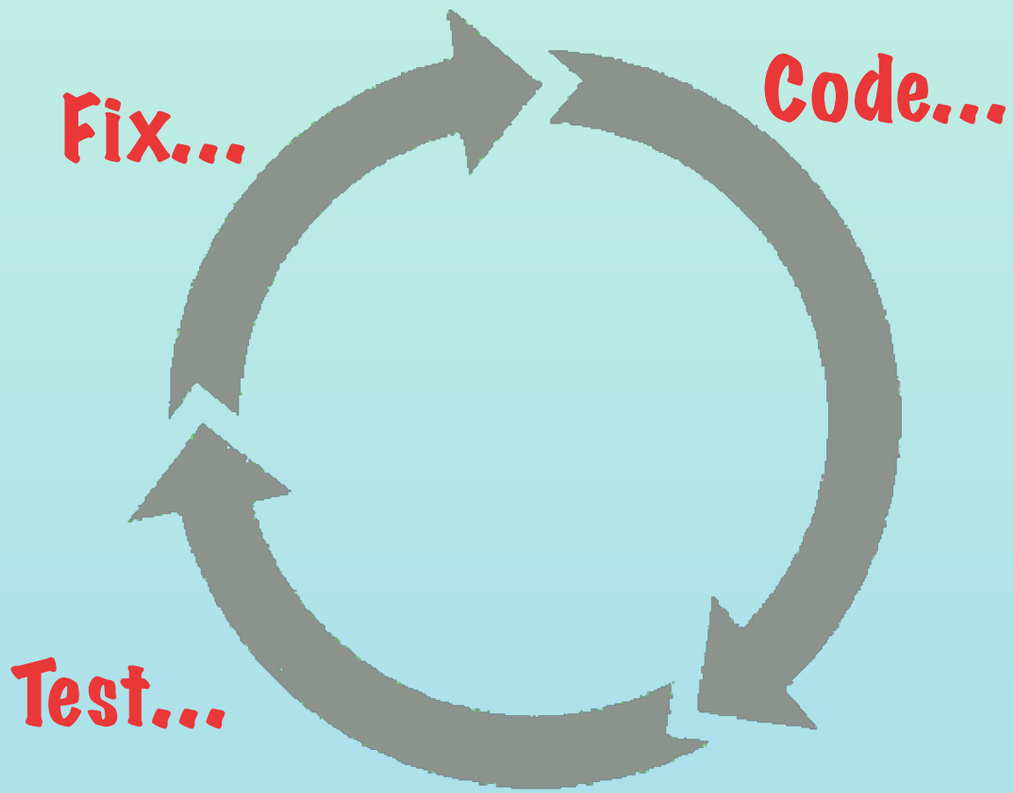
Red: Ensure test fails

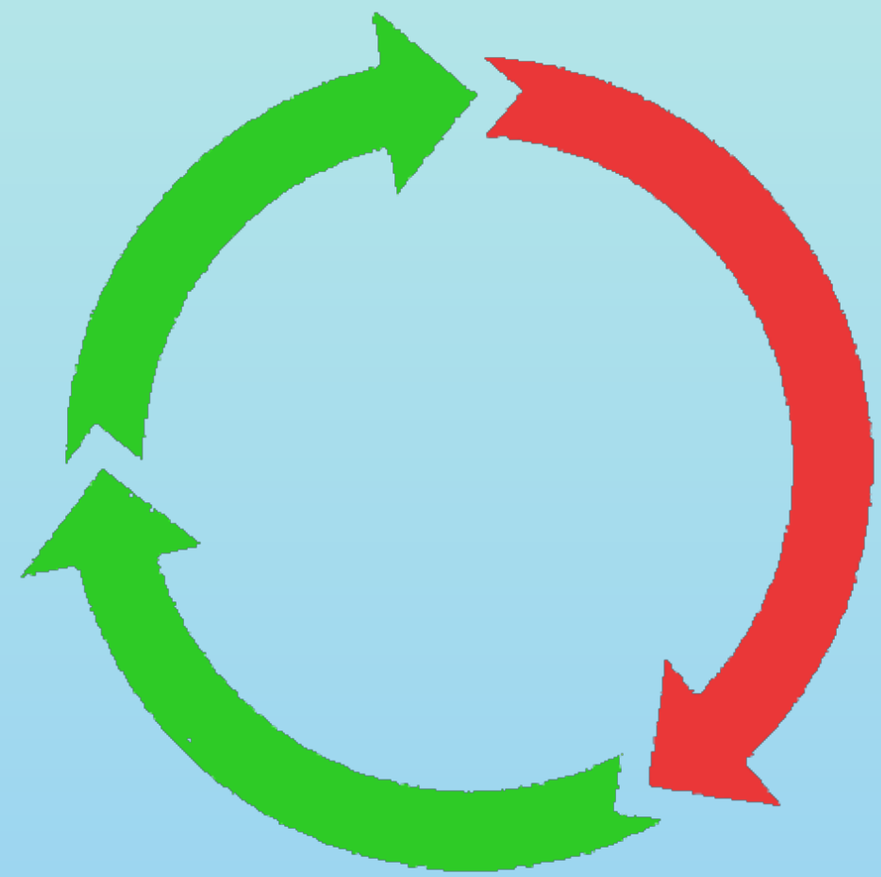
Green: Implement no more code than necessary

Refactor: Clarify and eliminate all duplication

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

TAD** and TDD: What's the difference?

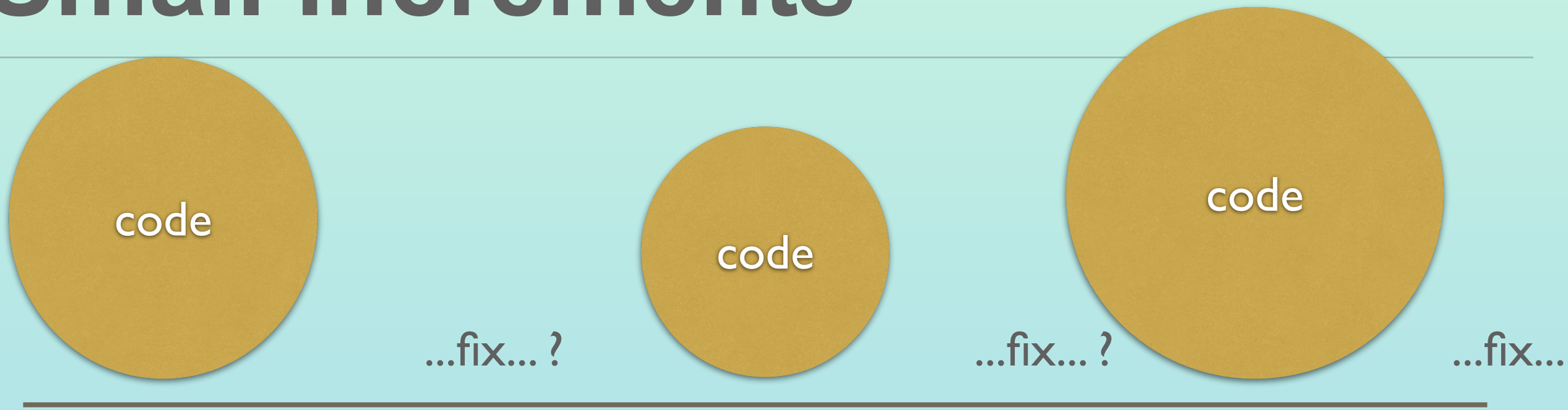




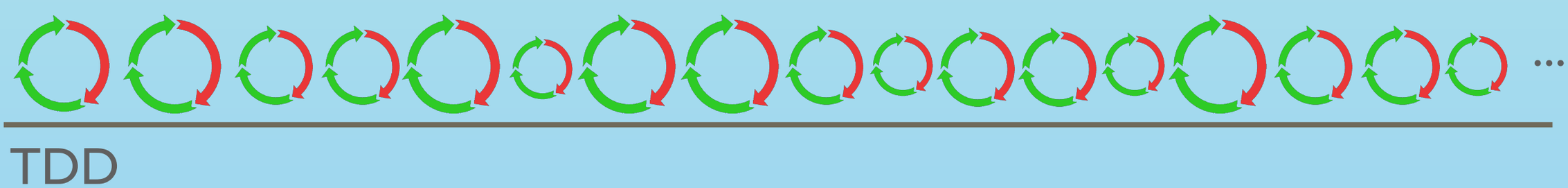
"Test-After Development"

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Small increments



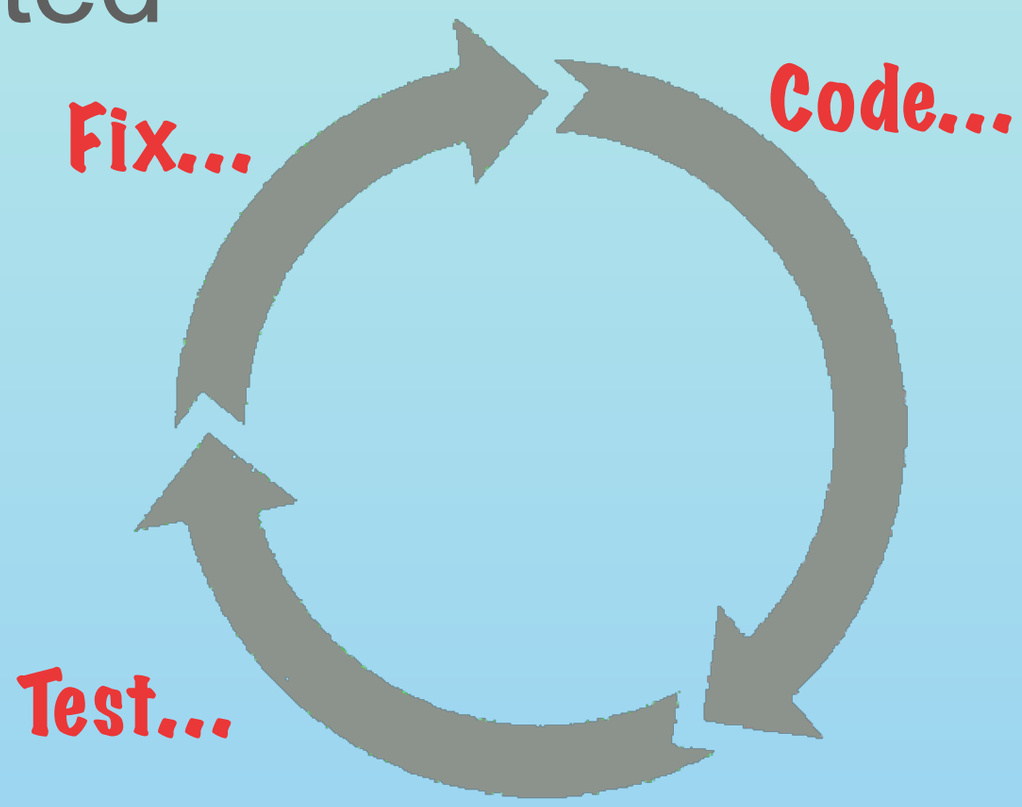
vs.



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Test-After Development (TAD)

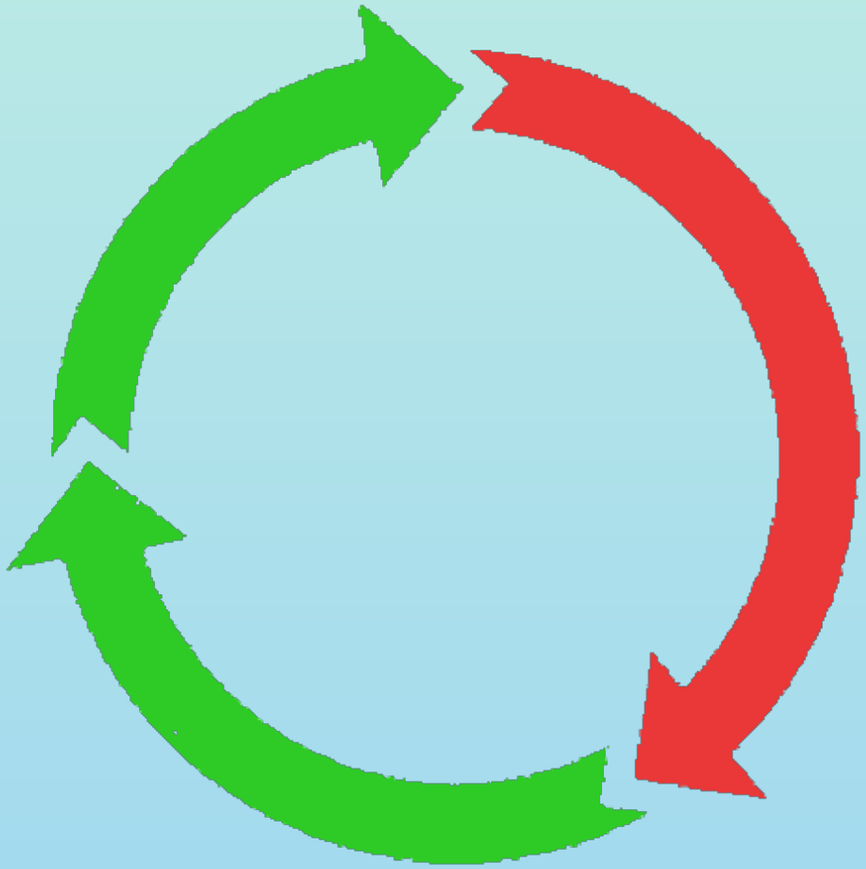
- Some refactoring accommodated
- Coverage: ~70%
- Design not usually impacted
- Some defect reduction
- Separate task?



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Test-Driven Development (TDD)

- Continual refactoring
- Coverage for all intended features
- Incremental design shaping
- Significant defect reduction
- Minimized debugging
- Integral part of coding process
- Clarify / document needs / choices
- Continual forward progress
- Consistent pacing
- Continual feedback / learning
- Sustainable



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Kahoot!

Katas

http://codekata.com

Other sites:

https://exercism.io

http://cyber-dojo.org

https://sites.google.com/site/tddproblems/

https://www.codewars.com

https://github.com/jlangr/name-normalizer

https://github.com/emilybache

https://projecteuler.net/archives

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Kata," courtesy https://www.flickr.com/photos/superwebdeveloper/
License: https://creativecommons.org/licenses/by/2.0/

Roman numeral converter

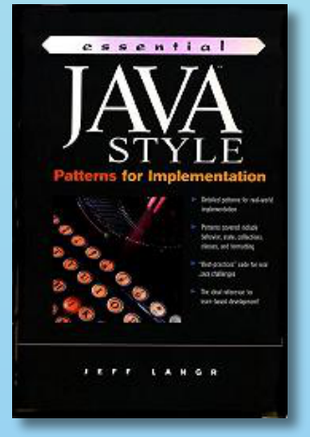
Given a positive integer
from 1 up to 4000,
answer its Roman equivalent

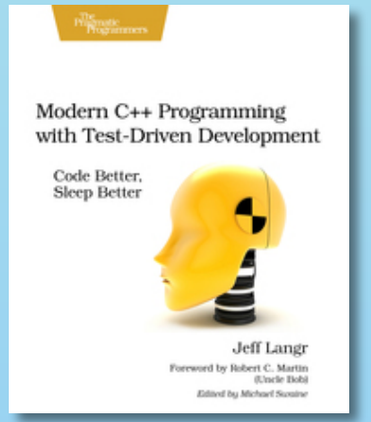
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

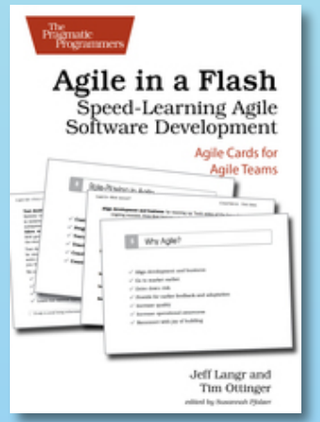
"Kata," courtesy https://www.flickr.com/photos/superwebdeveloper/
License: https://creativecommons.org/licenses/by/2.0/

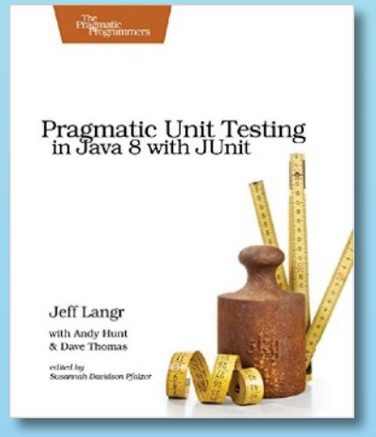
Where does TDD fit?

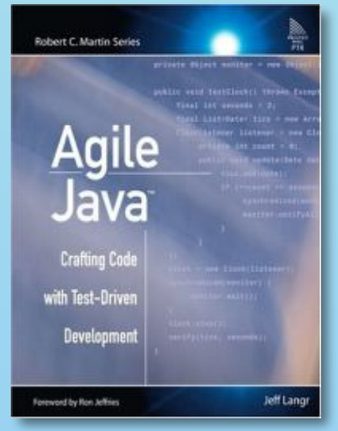
LANGR | SOFTWARE SOLUTIONS

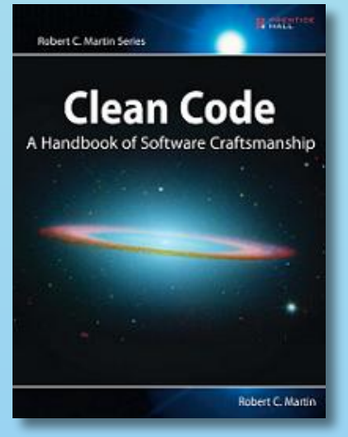




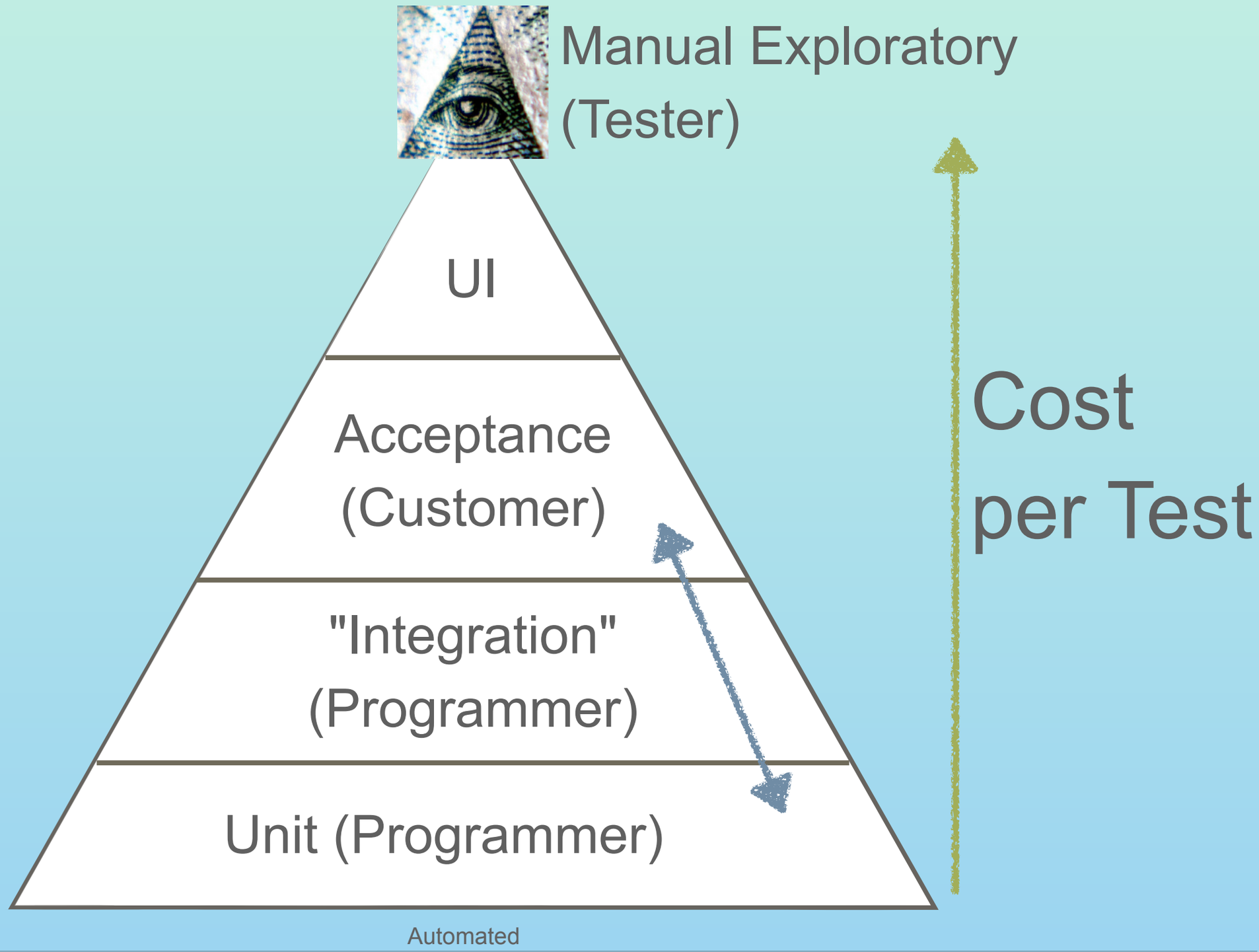








Unit testing is but a slice of the whole

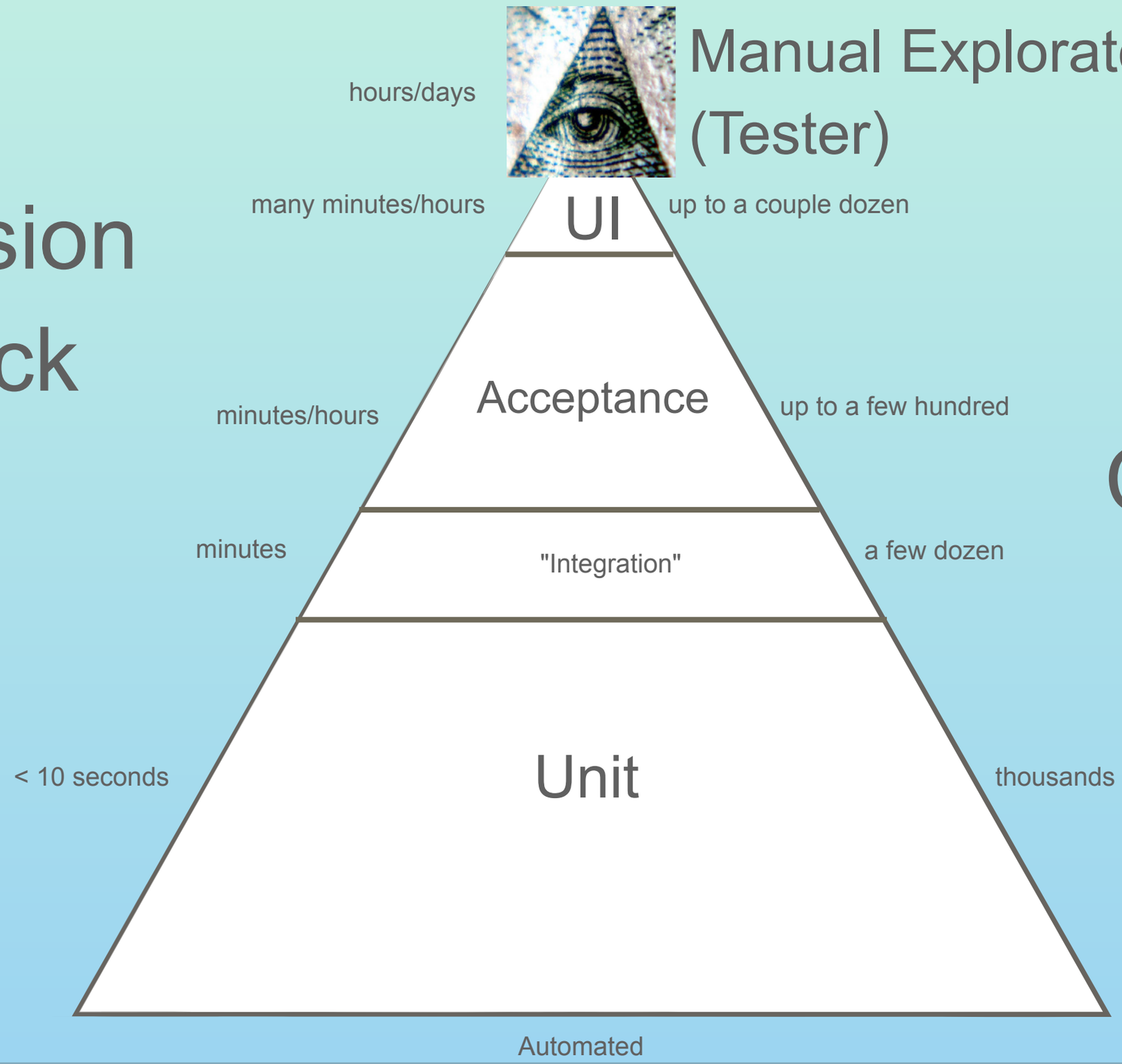


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Eye image: Frank Boston, https://www.flickr.com/photos/fixersphotos/401979
License: https://creativecommons.org/licenses/by/2.0/

Right-sizing the pyramid

Regression Feedback



Count

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Eye image: Frank Boston, https://www.flickr.com/photos/fixersphotos/401979
License: https://creativecommons.org/licenses/by/2.0/

Feedback questions

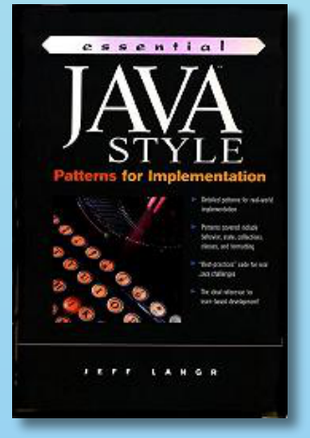
:-)	What have I learned of value?
%-|	What, if anything, am I unclear about regarding the discussion / exercises?
?:-/	What, if anything, did I find disturbing / disconcerting?

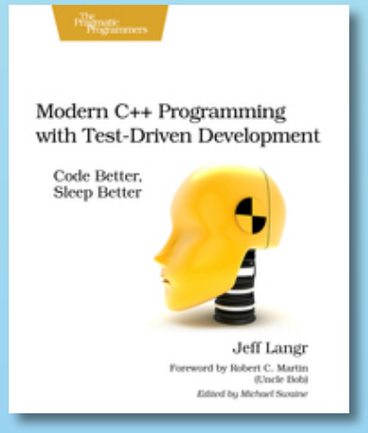
https://form.jotform.com/203163609669058

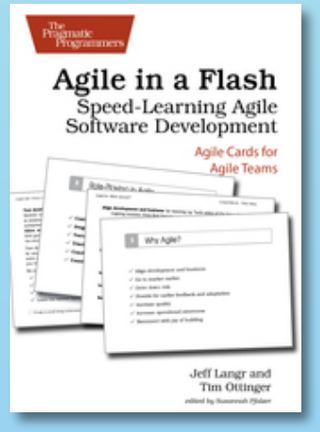
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

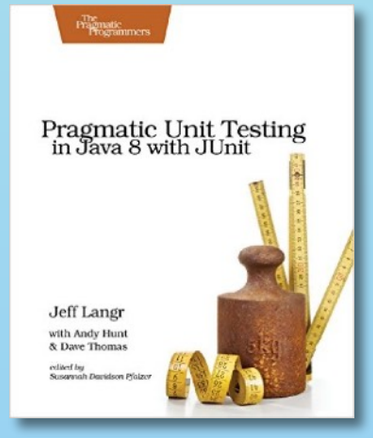
Documenting unit behaviors

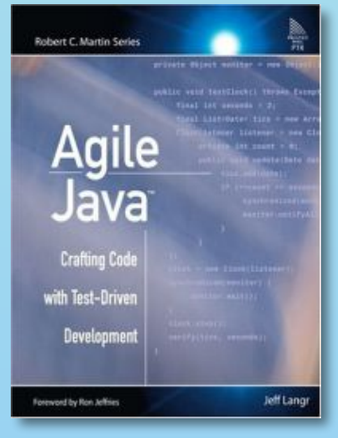
LANGR | SOFTWARE SOLUTIONS

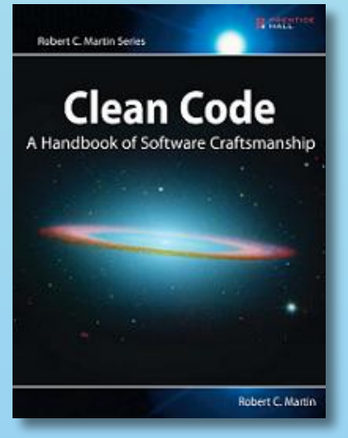












Specs as documentation

```
describe Auto do
  let(:auto) { Auto.new }

  it "idles engine when started" do
    auto.depress_brake

    auto.press_start_button

    expect(auto.RPM)
      .to be_between(950, 1100)
  end
end
```

what we're describing

the generalized behavior it supports

an example of that behavior

Seek to first understand through the tests.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Behavior-driven structuring

```
describe "given some context" do
  context "when some event occurs" do
    it "then can be verified" do

    end
  end
end
```

Copyright © Langr Software Solutions, Inc. 2020. All rights reserved.

BDD Structuring

```
describe('given a checked-out material') do
  let(:library) { Library.new }

  before(:each) do
    @due_date = library.borrow(Agile_Java, Patron_Id)
  end

  context 'when returned late' do
    before(:each) do
      library.return_material(Agile_Java, @due_date + 1)
    end

    it 'is marked as available' do
      expect(library.available? Agile_Java).to be true
    end

    it 'generates a late fine' do
      # ...
    end

    it 'notifies patrons with hold' do
      # ...
    end
  end
end
```

Copyright © Langr Software Solutions, Inc. 2020. All rights reserved.

Iterative naming

- Re-consider name continually

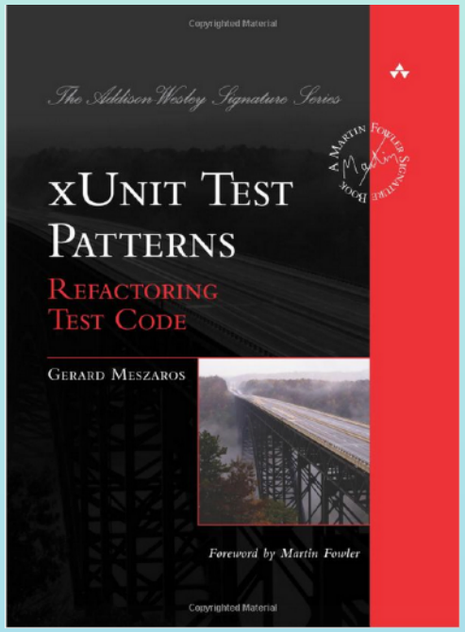
```
it 'does something' do

it 'starts engine' do

it 'sets idle speed on engine start' do

it 'has low idle on engine start' do
```

- Review often and holistically!

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Sustainable tests

The long answer:



A quicker route:

- Single behavior tests
- AAA
- Correlate results with context
- Test abstraction

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: Test smells

Find and fix test smells.
Paraphrase cleaned tests
to your pair / mob.

In: ./spec/controllers/checkouts_controller_spec.rb

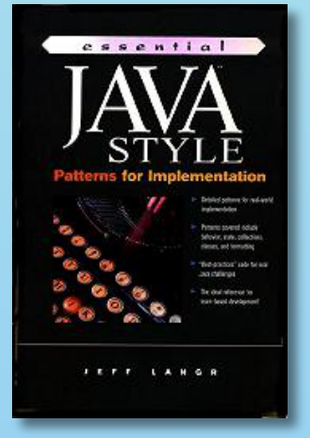
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

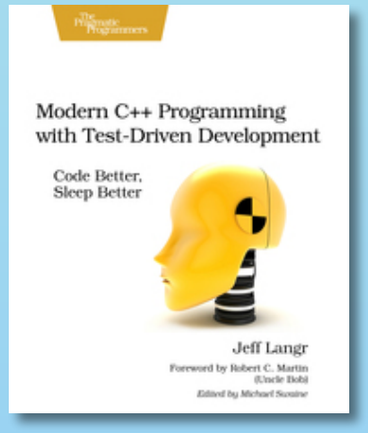
"skunk_dog," courtesy https://www.flickr.com/photos/istolethet
License: https://creativecommons.org/licenses/by/2.0/

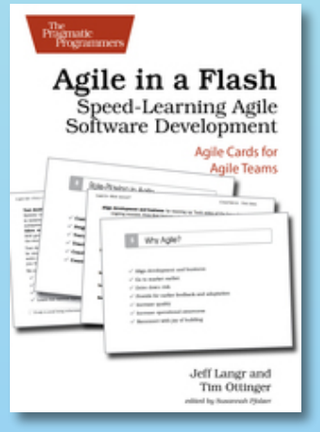
Kahoot!

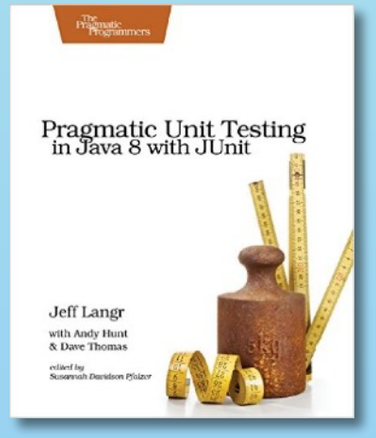
Continual design

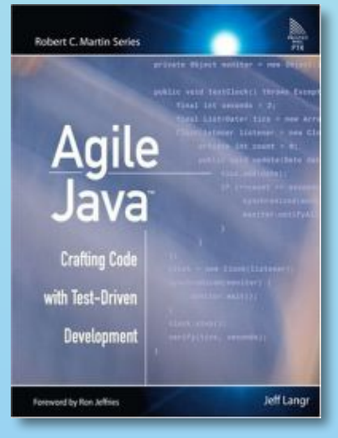
LANGR | SOFTWARE SOLUTIONS

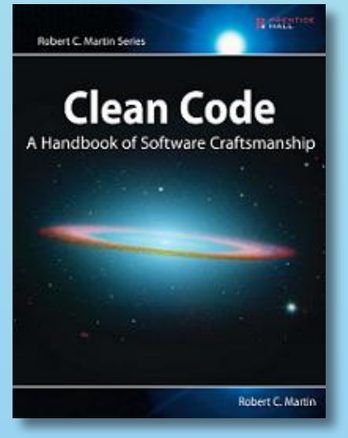






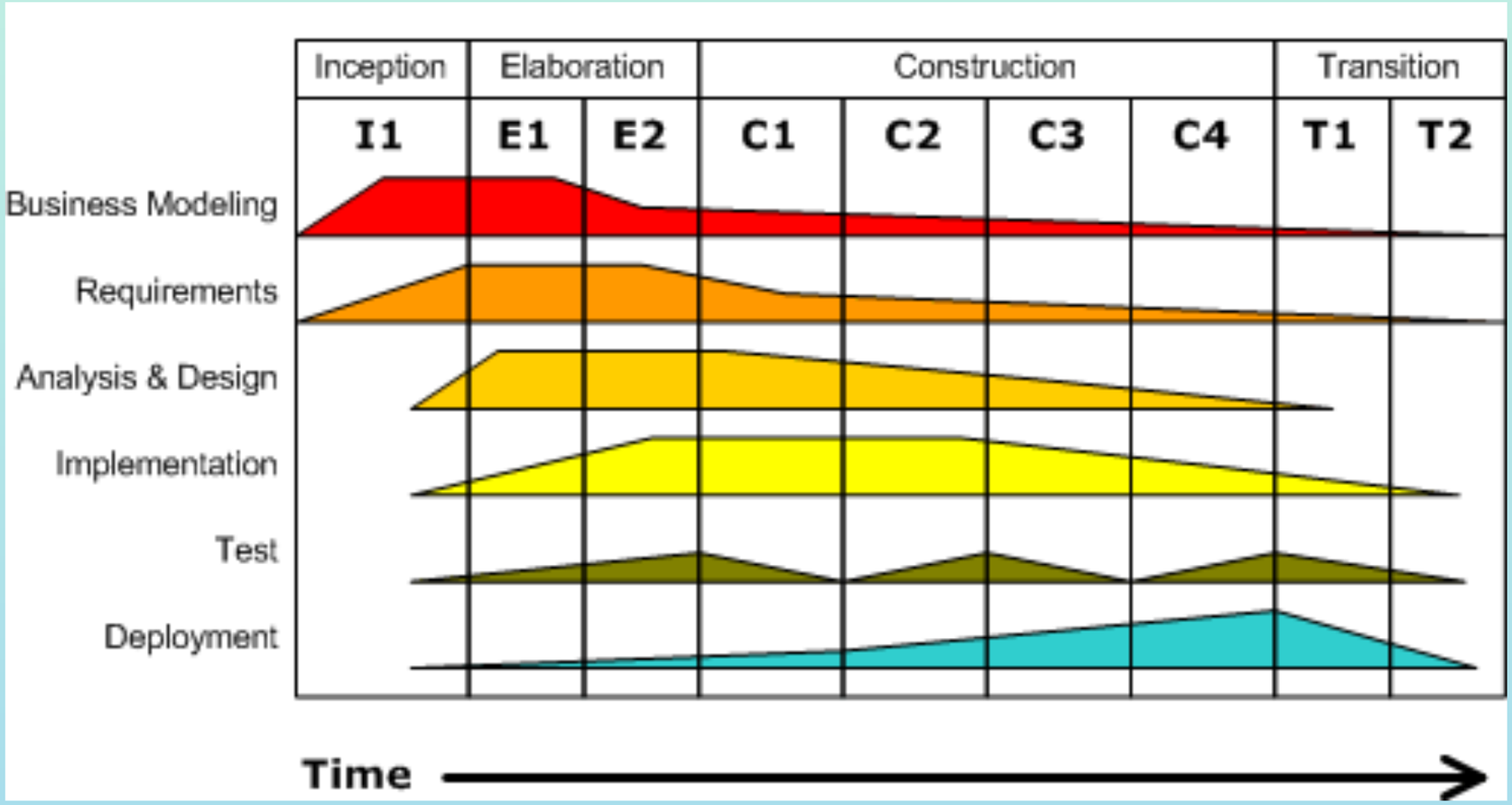






Software development activities

- Planning
- Analysis
- Design
- Coding
- Testing
- Review
- Documentation
- Deployment
- …

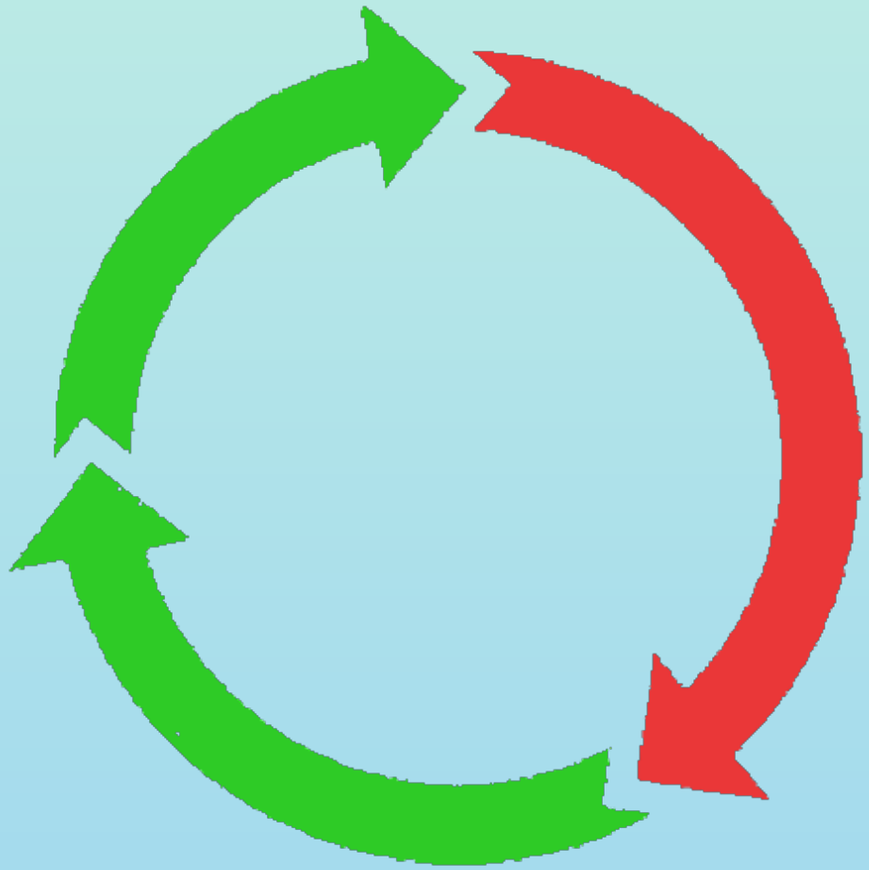


- Agile: *All activities all the time*

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

TDD is continual ...

- Testing
- Coding
- Documentation
- Review
- Design



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Design and TDD

What is design?

What aspects of building a solution involve design?

When / how does design occur in the TDD cycle?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

“What Is Design?”—Jack W. Reeves

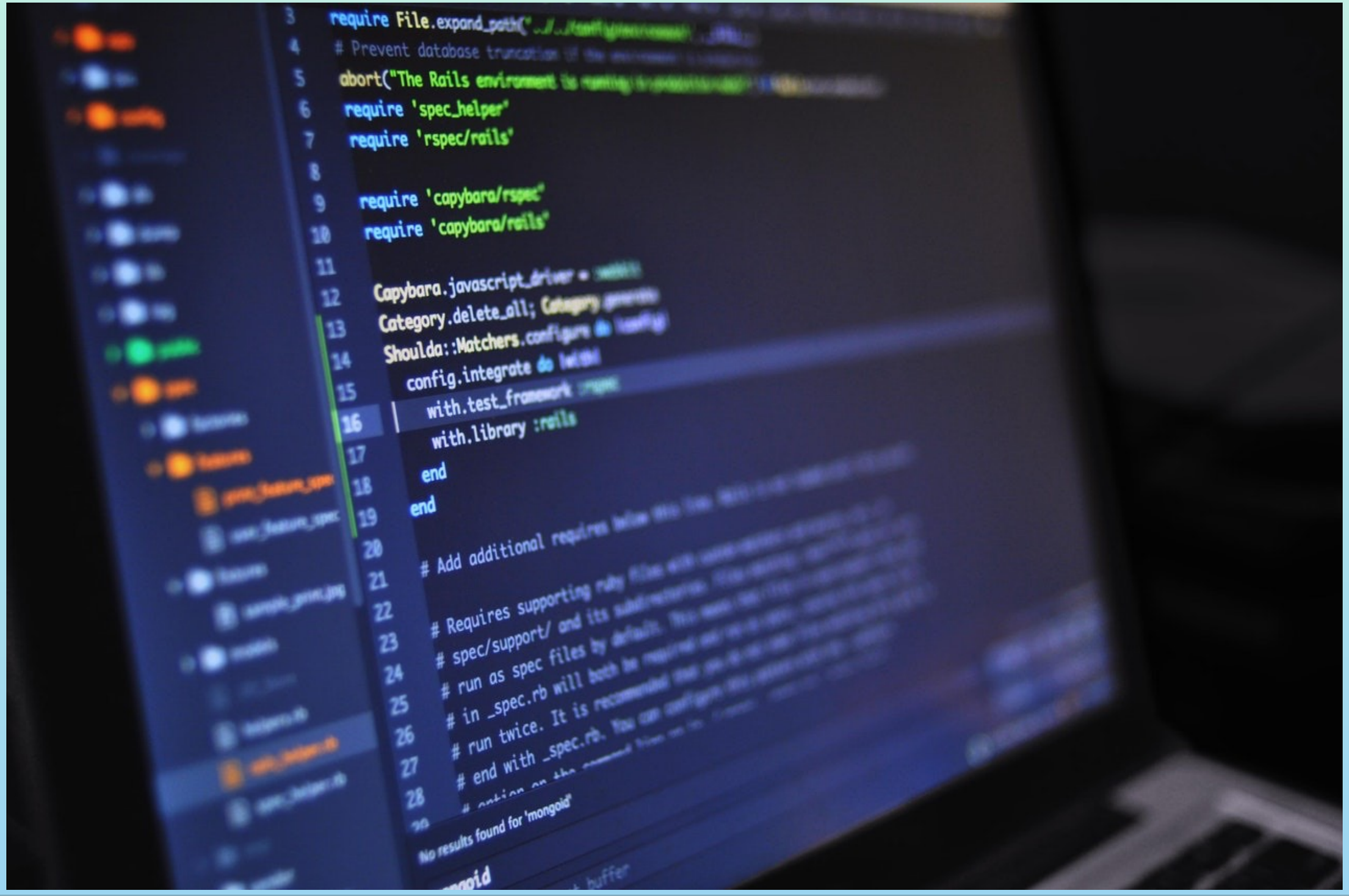
- A program listing is a document that represents a software design.
- Programming is a design activity.
- Testing [is] the software equivalent of the design validation & refinement processes of other engineering disciplines.
- Other design activities exist—top level design, module design, structural design, architectural design, whatever.

“All design activities interact. A good software design process recognizes this and allows the design to change, sometimes radically, as various design steps reveal the need.”

https://www.developerdotstar.com/mag/articles/reeves_design.html

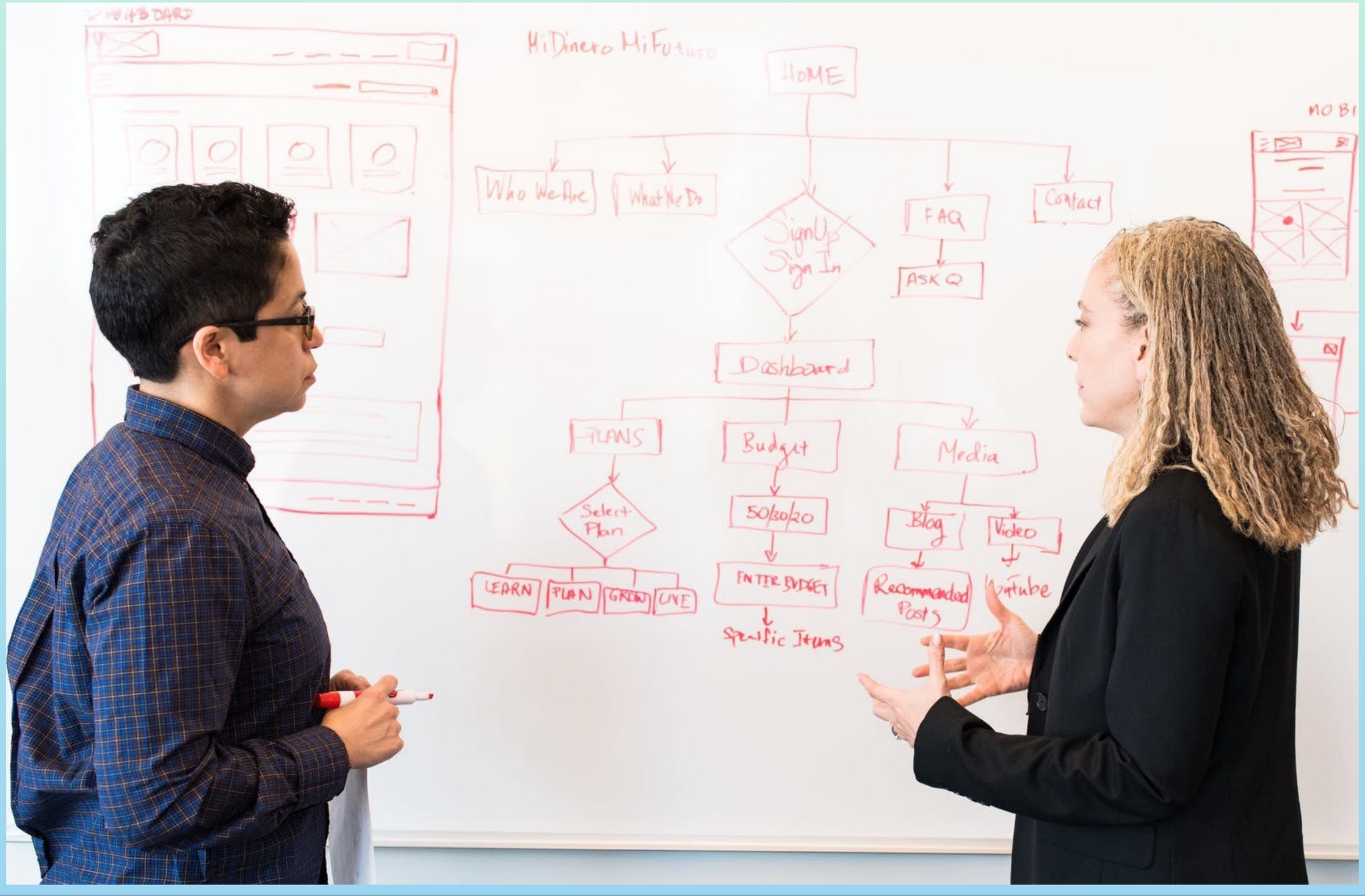
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

This is design



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

So is this



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

And this

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

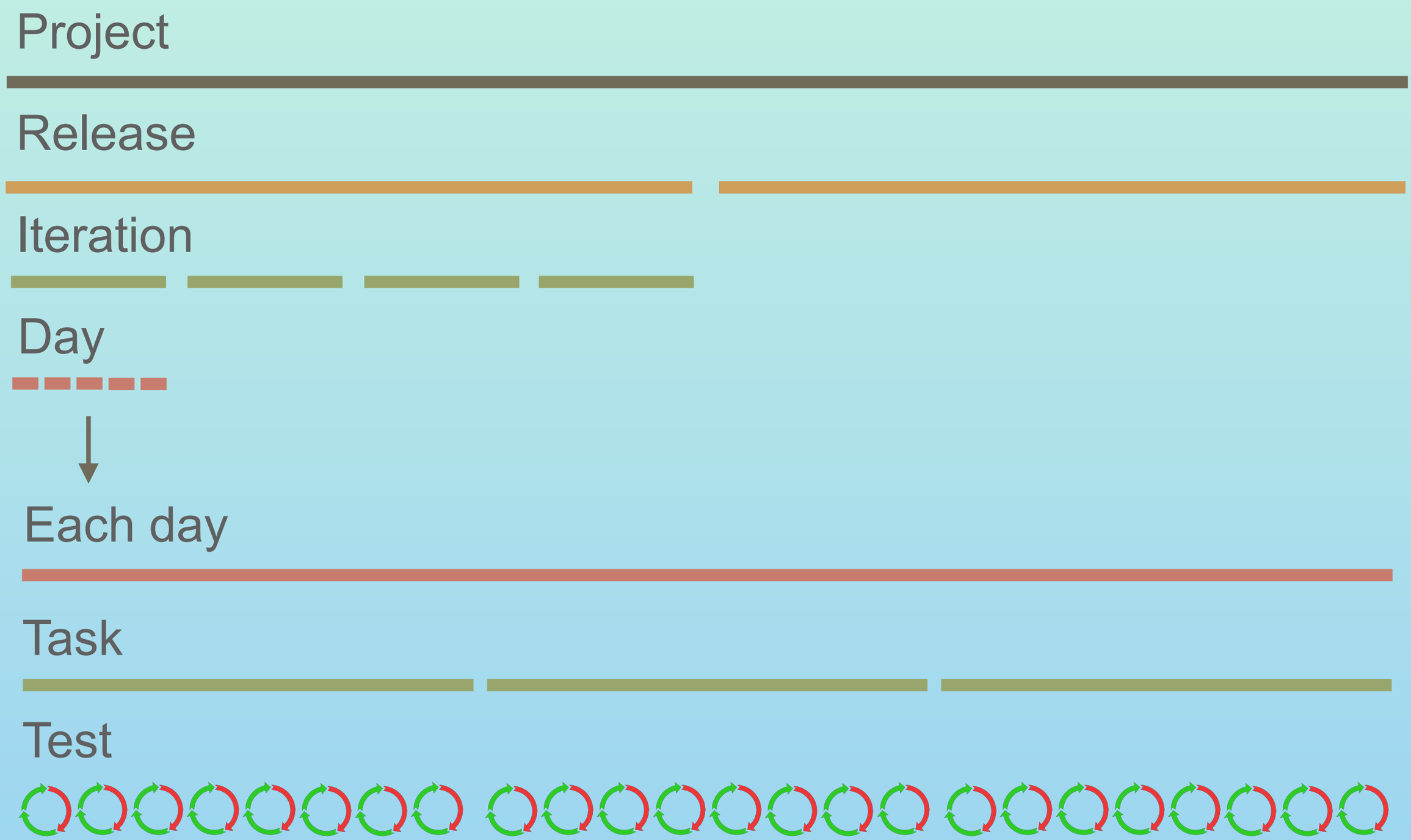
Old-school design planning

“Big Design Up-Front” (BDUF)

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"UML Diagram," courtesy https://www.flickr.com/photos/acehoward/
License: https://creativecommons.org/licenses/by/2.0/

Design occurs at every possible juncture

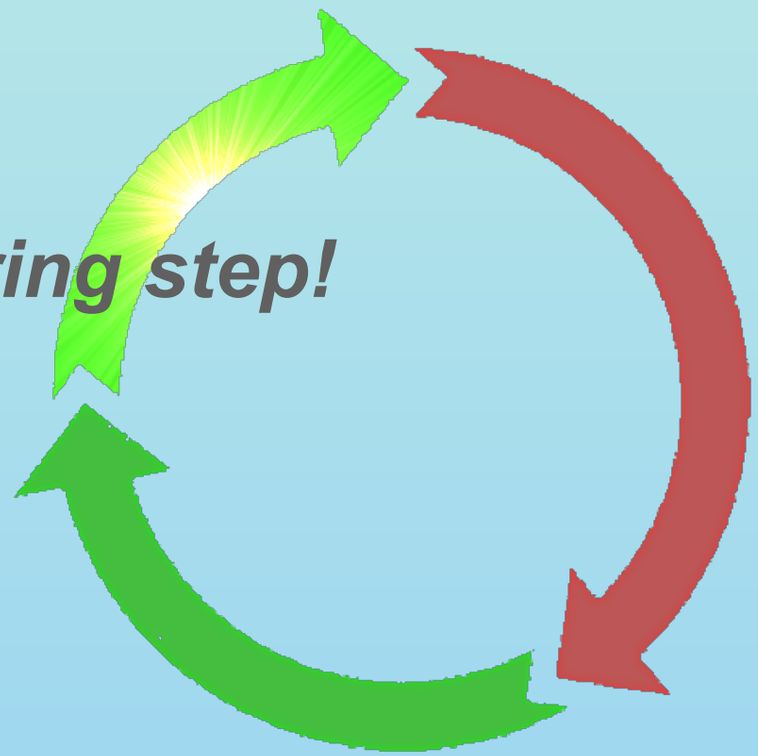


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Continual design in TDD

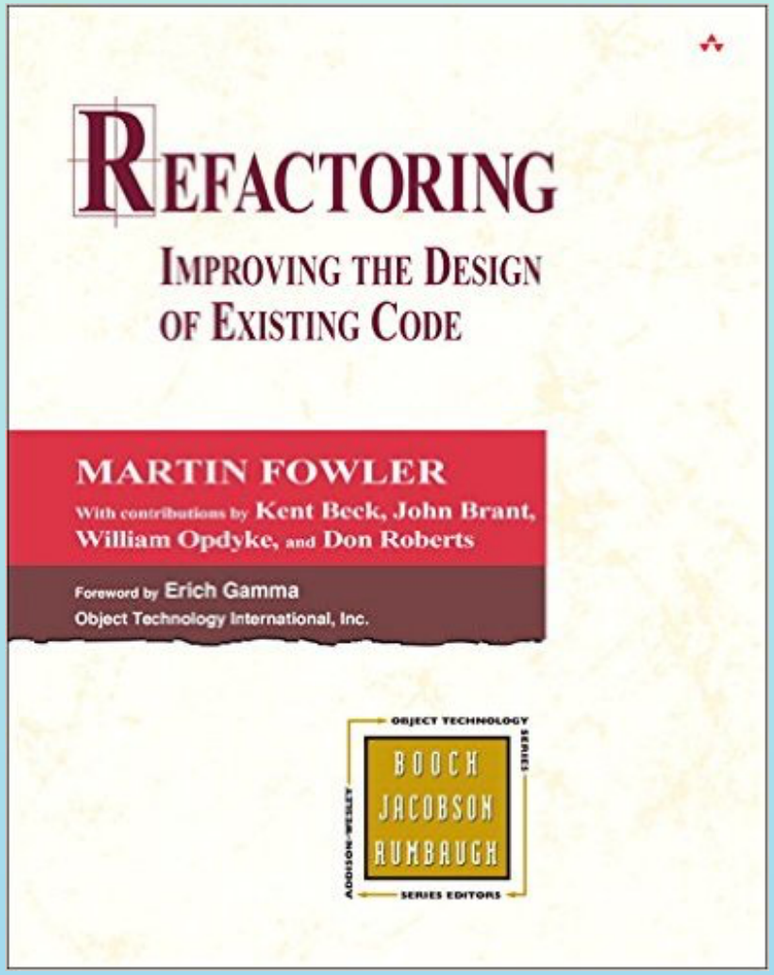
- Always retain an optimal design
- Entropy is unavoidable

Never skimp on the refactoring step!

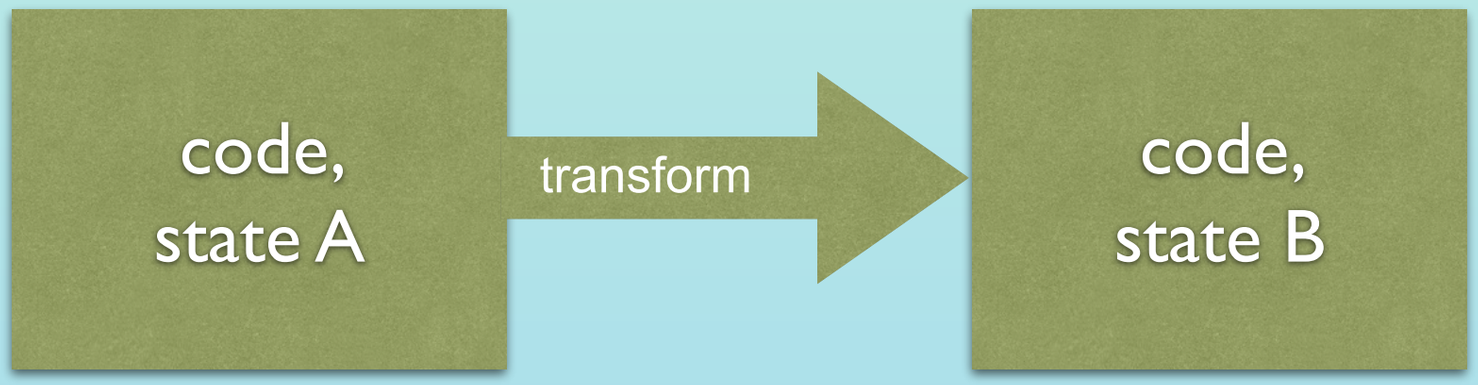


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Refactoring



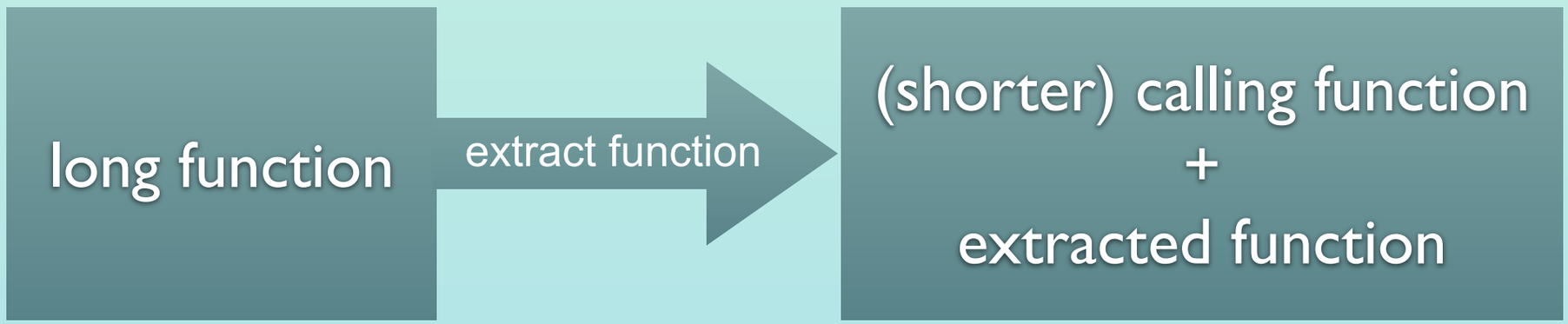
1999 / 2019



Same "externally recognized" behavior

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

For *(almost)* every transform...



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Code smells

An aroma?

- Comments
- Duplicate Code
- Feature Envy
- Large Module
- Long Function
- Long Parameter List
- Primitive Obsession
- Shotgun Surgery
- Speculative Generality
- Switch Statements
- **. . . and many more**

…or a stink?

"A complete code smells reference:"
https://github.com/lee-dohm/code-smells

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: Code smells

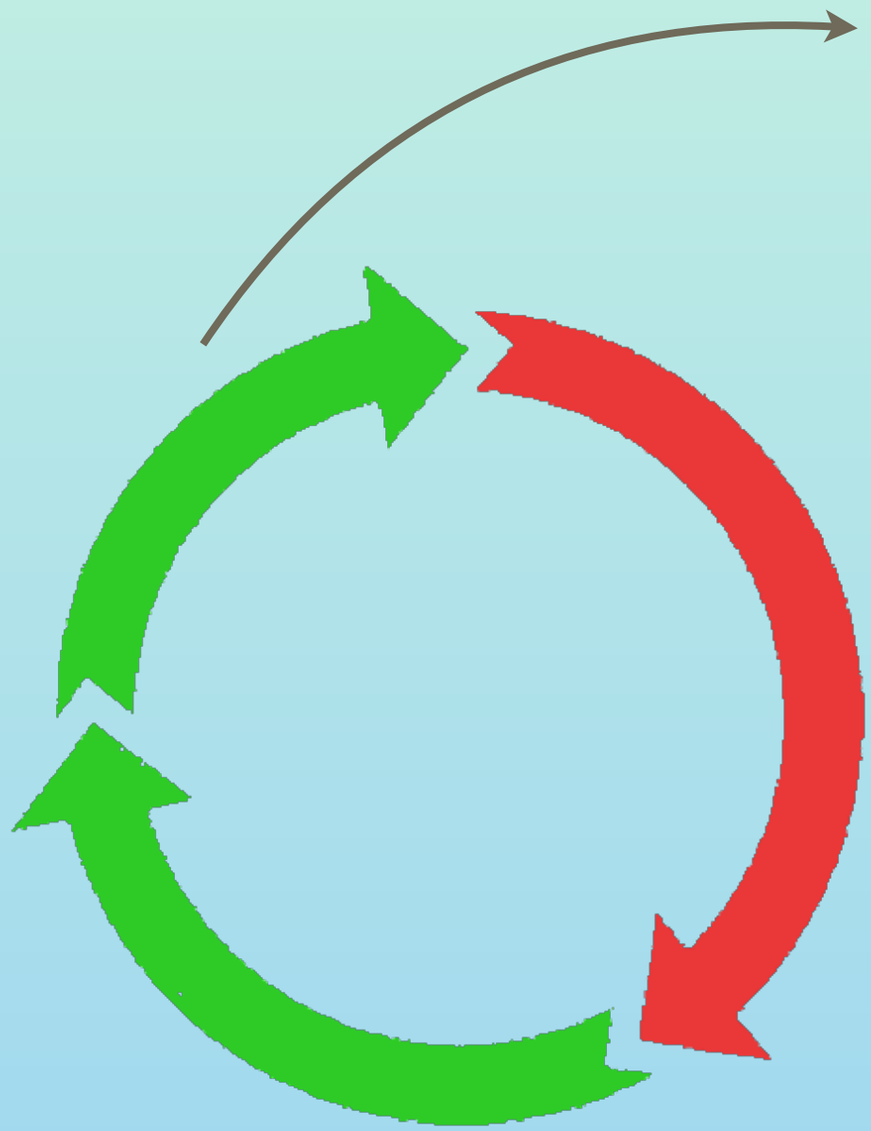
Identify code smells.

In ./app/controllers/checkouts_controller.rb

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"skunk_dog," courtesy https://www.flickr.com/photos/istolethetv/
License: https://creativecommons.org/licenses/by/2.0/

Refactoring opportunities



small cleanup
verify
small cleanup
verify
...

One thing at a time!

Break things once in a while.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Renaming

"Lake Louise Train Station Restaurant - Bread and Butter," courtesy https://www.flickr.com/photos/elsiehui/
License: https://creativecommons.org/licenses/by/2.0/

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Refactoring: Extract method

```
def ugly_method
  # stuff a
  # ...

  # stuff b
  # ...

  # some smaller behavior
  x = X.new
  x.do_stuff

  # stuff c
end
```

```
def ugly_method
  # stuff a
  # ...

  # stuff b
  # ...

  do_some_smaller_behavior

  # stuff c
end

def do_some_smaller_behavior
  x = X.new
  x.do_stuff
end
```

Why?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Single-line extract?

```
rescue DiagError => e
  errMsg = "#{Time.now}: #{System}-#{Module} ERROR #{truncate(e.msg, 80)}”
  log(errMsg)
  raise StandardError.new(errMsg)
end
```

```
  rescue DiagError => e
    errMsg = formatted_error_message(exception)
    log(errMsg)
    raise StandardError.new(errMsg)
  end
# ...

def formatted_error_message(exception)
  "#{Time.now}: #{System}-#{Module} ERROR #{truncate(e.msg, 80)}”
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Minimize duplication, simplify testing

```
  rescue DiagError => e
    log_and_reraise(e.message, Formats.method(:log_for_file))
  end
  ...

def log_and_reraise(message, format_fn)
  err_msg = format_fn.call(message)
  log(err_msg)
  raise StandardError.new(err_msg)
 end

class Formats
 def self.log_for_file(msg)
   "#{Time.now}: #{System}-#{Module} ERROR #{truncate(msg, 80)}"
 end
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

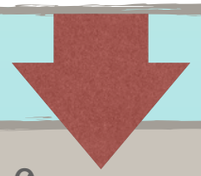
Refactoring-inhibiting temp

```
if not is_exempt and discount > 0
  discount_amount = discount * price
  discounted_price = price * (1.0 - discount)

  total_of_discounted_items += discounted_price

  text = item.description
  # format percent
  amount = sprintf("%.2f", price.round(2))
  amount_width = amount.length

  text_width = LINE_WIDTH - amount_width
  messages << text.ljust(text_width) + amount

  total += discounted_price
  # ...
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Refactoring: Replace temp with query

```
if not is_exempt and discount > 0
  discount_amount = discount * price
  discounted_price = price * (1.0 - discount)
  total_of_discounted_items += discounted_price
  total += discounted_price
```



```
  if not is_exempt and discount > 0
    discount_amount = discount * price
    discounted_price = price * (1.0 - discount)
    total_of_discounted_items += discounted_price(price, discount)
    total += discounted_price(price, discount)
    # ...
end

def discounted_price(price, discount)
  item.Amount * (1.0m - MemberDiscount)
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Replace temp with query

Steps

- Extract Method on the right-hand-side
- Replace temp references with the query
- Remove temp

Why?

Concerns?

Can we go the other way?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: Extract method

In ./app/controllers/checkouts_controller.rb, focus on extracts, renames, and eliminating lies in `#checkout_total`

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Code smell: Feature envy

```
class WrongPlace
  def some_public_behavior
    result = some_envious_method
    # ...
  end

  private

  def some_envious_method
    there = GreenerGrass.new
    result = there.do_stuff(@some_self_value)
    there.do_more_stuff(result)
    there.answer
  end
end
```

What's the problem?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Refactoring: move method

```
class WrongPlace
  def some_public_behavior
    result = GreenerGrass.new
                          .homebody_method(@some_self_value)
    # ...
  end
end

class GreenerGrass
  def homebody_method(some_value)
    result = do_stuff(@some_self_value)
    do_more_stuff(result)
    @answer
  end
  # ...
```

What must you also do when you move a method?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: Move method

- Create new homes for checkouts_controller code as appropriate.
- Write tests to document the moved behavior.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Premature Generalization

- Details change.
- Need never materializes.
- A better solution becomes obvious.

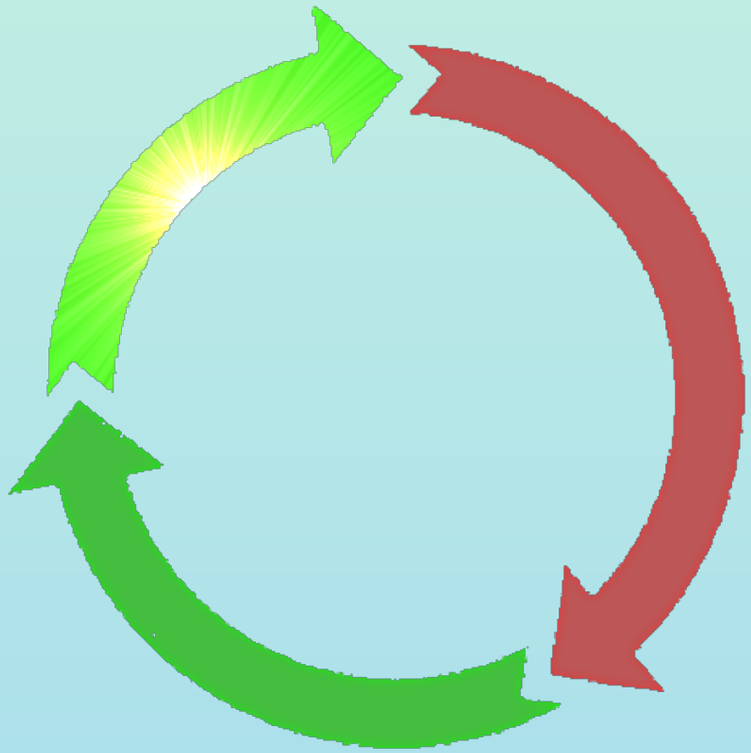
In the meantime, we all pay for the speculative complexity.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"crystal," courtesy https://www.flickr.com/photos/bb_matt/
License: https://creativecommons.org/licenses/by/2.0/

Refactoring Guidelines

Never skip!



Single-goal

Run *all* tests

It's your responsibility

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Design Drivers / Guidelines

- Code smells
- SOLID
- Design patterns
- GRASP
- Simple design
- ...

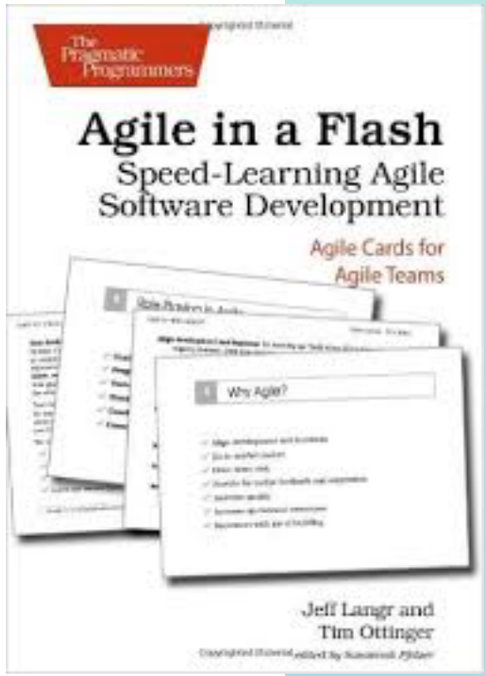
It's all good!

Does everything point to the same place?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

41 Build Superior Systems with Simple Design

- All tests must pass
- No code is duplicated
- Code is self-explanatory
- No superfluous parts exist



Kent Beck's (ordered) rules for emergent design

Exercise: Simple design rules

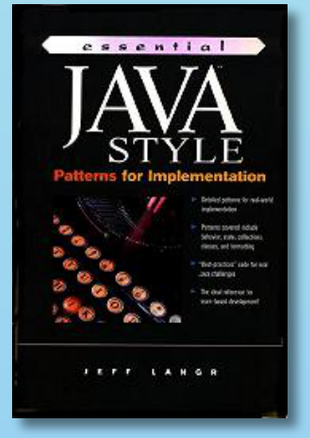
- In checkouts_controller
  - Stamp out duplication!
  - Strive for rapid readability
  - Have you gone too far?

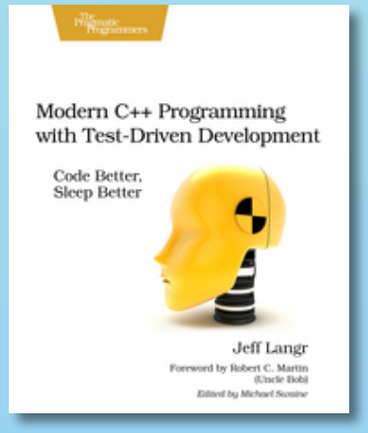
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

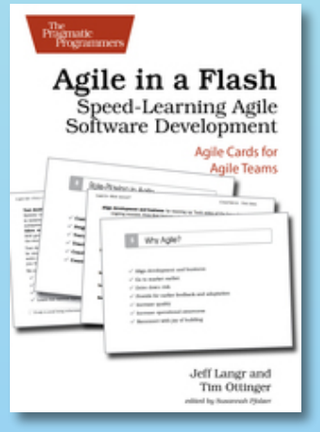
Kahoot!

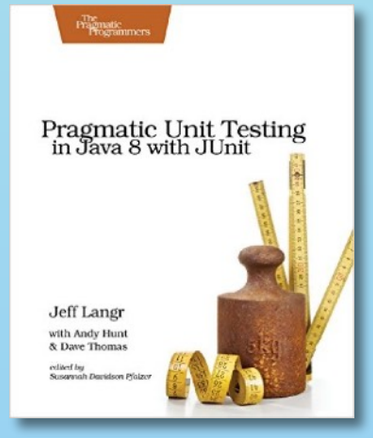
Test doubles

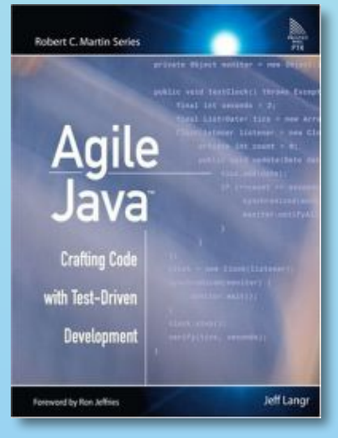
LANGR | SOFTWARE SOLUTIONS

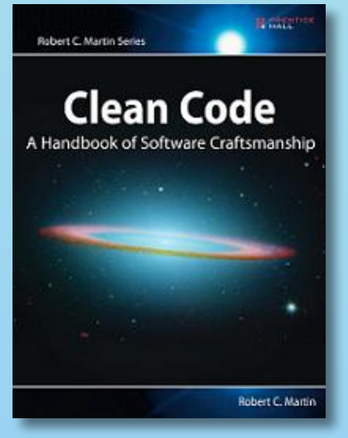




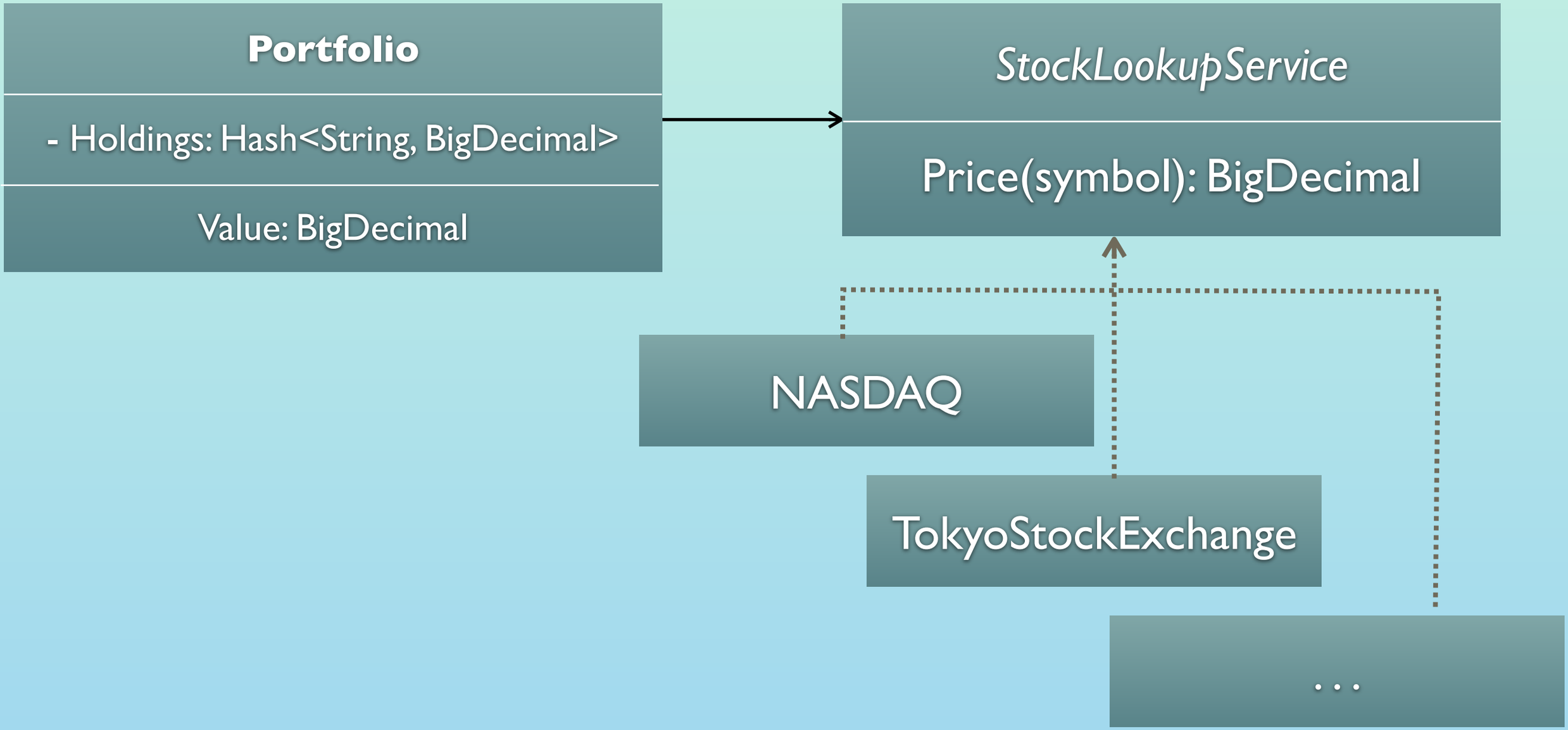






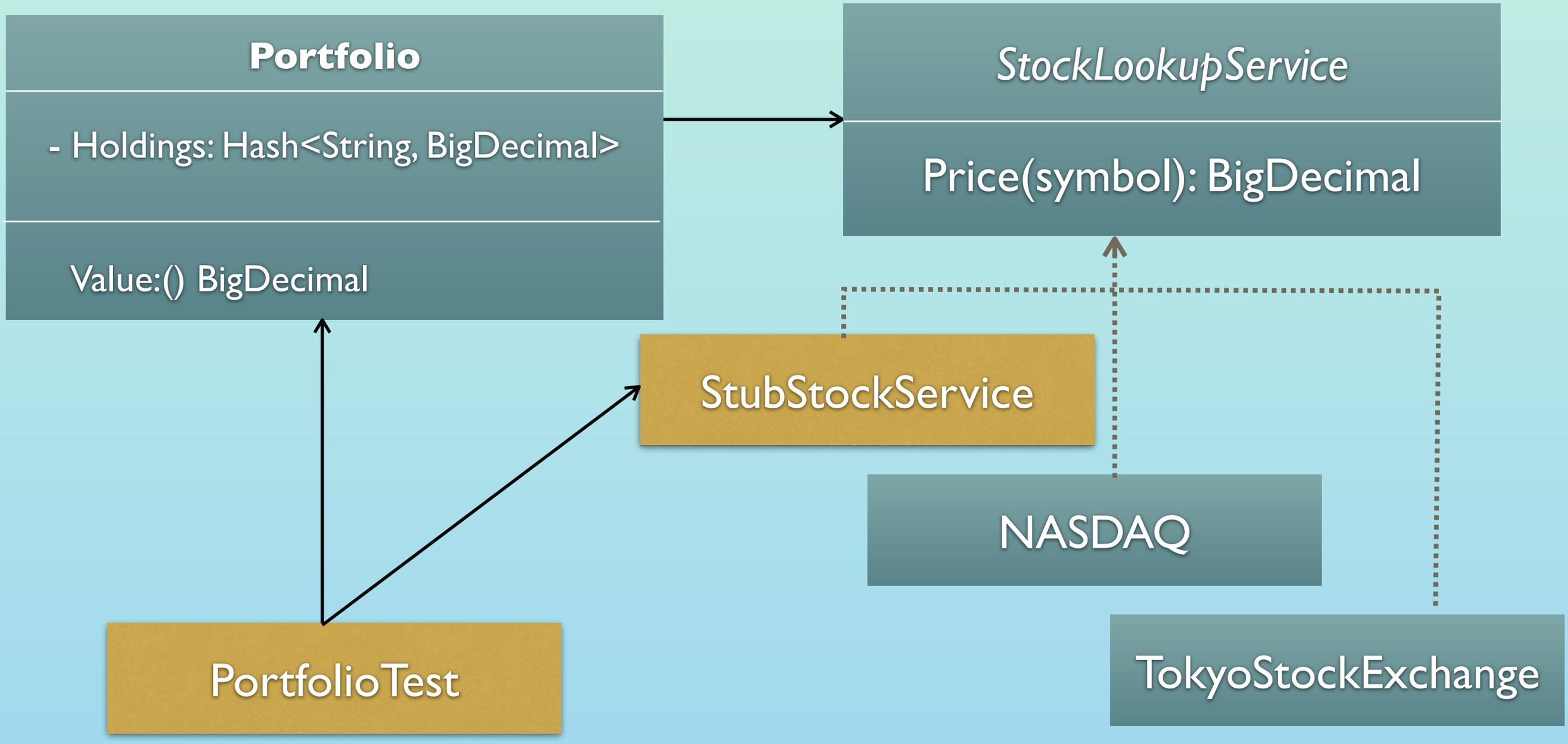


Testing challenge: Portfolio



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

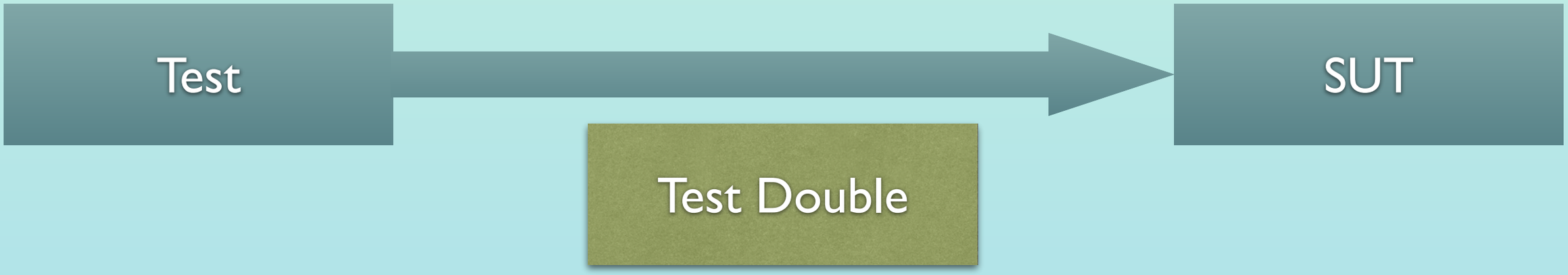
Using a test double



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Substituting a test double

… for the production behavior



Various techniques:

- Constructor / setter / parameter injection
- Factory / FactoryBot
- Method redefinition
- Injection tool (e.g. dry-auto_inject)
- …

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Demo: portfolio value

Exercise: Portfolio value

Drive the full value behavior into the Portfolio class.
Create a double for StockLookupService.

```
class StockLookupService
  def price(symbol)
    raise "service unavailable"
  end
end
```

`StockLookupService` makes a live call.

You shouldn't be changing or instantiating this production class.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

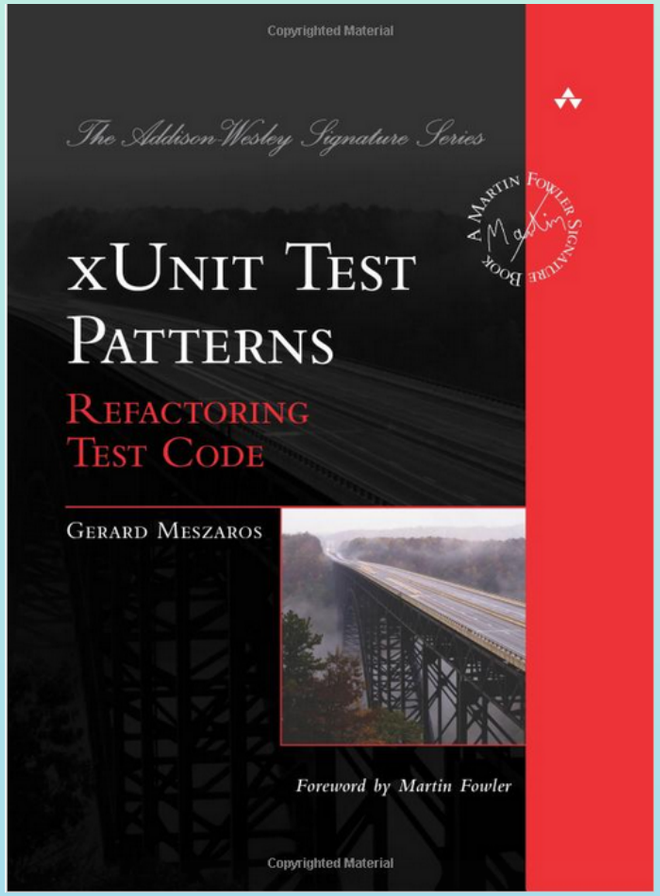
Test double terms

Stub: dumb emulation

Spy: captures values to verify*

Mock: self-verifies expectations

Fake: whole collaborator emulation



* Caution: Tool implementation of spies vary as to whether the real method gets called or not.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Mock tools

RSpec Mocks

RR

Flexmock

freerange / mocha

Flexmock

History suggests more & changes coming!

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

RSpec Mocks: A simple stub

https://rspec.info/documentation/3.9/rspec-mocks/

a method stub

```
weather_service = double
allow(weather_service)
 .to receive(:temperature).and_return(34)
expect(weather_service.temperature).to eq(34)
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

RSpec Mocks Method Stubs: Alt. Forms

```
allow(weather_service)
 .to receive(:temperature).and_return(34)


allow(weather_service).to receive(:temperature) { 34 }


allow(weather_service)
  .to receive_messages(:temperature ⇒ 34,
                       :pressure ⇒ 29.04)
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

RSpec Mocks Method Stubs: Shortcut

```
weather_service = double(:temperature ⇒ 34)
```

What’s not to like?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Instance doubles

Verify that the double is in sync with the real object

```
subject(:stock_service) { instance_double(StockService) }
```

Oops:

```
expect(stock_service).to receive(:prlce).and_return(100)
```

the StockService class does not implement the instance method: prlce

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: RSpec Stubs

Replace your hand-crafted stub with an RSpec method stub, for the portfolio value spec involving purchase of only one symbol.

Use an instance double for StockService. Experiment with breaking things.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

A message expectation

https://rspec.info/documentation/3.9/rspec-mocks/

an expectation

```
weather_service = double
expect(weather_service)
 .to receive(:temperature).and_return(34)
expect(weather_service.temperature).to eq(34)
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Expectations

Replace `allow` with `expect`.

Remove the code that got the test to pass in the first place; run tests.

Discuss.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

RSpec Mocks: Expectations with args

```
expect(service)
  .to receive(:temperature).with("Miami") { 67 }
expect(service)
  .to receive(:temperature).with("Des Moines") { 34 }

expect(service.temperature("Miami")).to eq(34)
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

RSpec: Argument matchers

```
expect(service)
  .to receive(:pressure).with(kind_of(String)) { 67 }
```

Also match via:

- `any_args`,
- regex,
- duck typing,
- "`array_including`," etc.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Rspec Mocks: Miscellany

Expect:

- twice, n times, at least n times, at most n times
- messages received in specific order
- argument capture via blocks (“arbitrary handling”)
- `and_call_original`
- `and_raise(error)`
- `and_yield(value1, value2, …)`

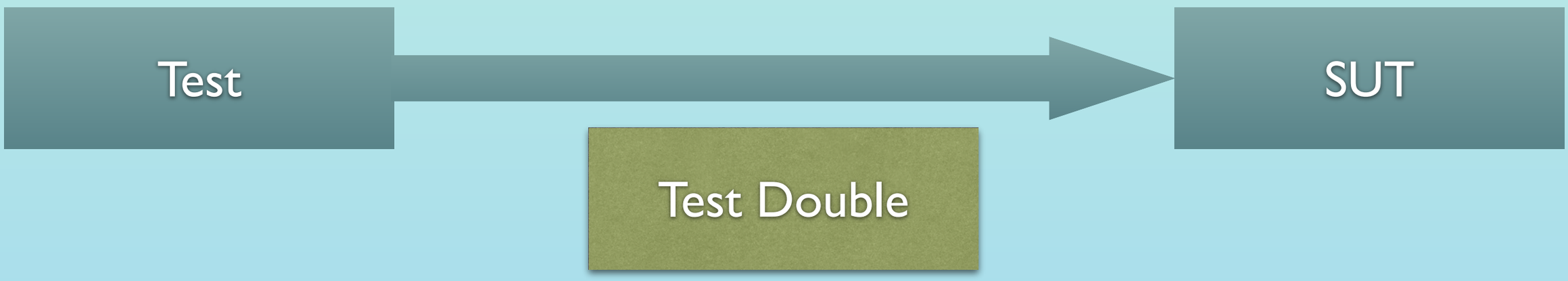
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: RSpec

Use argument matchers for specs as appropriate for portfolio value specs

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Engaging the double: Dependency Injection



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Engaging the double: Constructor injection

```
let(:stock_service) { double }
let(:portfolio) { Portfolio.new(stock_service) }

context "after purchase" do
  before() do
    portfolio.purchase(APPLE, 4
  end

  it "has value of share price" do
    expect(stock_service)
      .to receive(:price).and_return(CURRENT_APPLE_PRICE)
    expect(portfolio.value).to equal CURRENT_APPLE_PRICE * 42
  end
end
```

```
def initialize(service = StockService.new)
  @holdings = Hash.new
  @service = service
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Engaging the double: Method redefinition

```
let(:portfolio) { Portfolio.new }

describe Portfolio do
  it "has value of share price" do
    expect_any_instance_of(StockService)
      .to receive(:price).and_return(100)

    portfolio.purchase("QCOM", 1)

    expect(portfolio.value).to equal 100
  end
end
```

```
class Portfolio
  def initialize
    @holdings = Hash.new
    @service = StockService.new
  end
  # …
end
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Engaging the double: Stubbing new

```
let(:stock_service) { double }
before() do
  expect(StockService).to receive(:new).and_return(stock_service)
  @portfolio = Portfolio.new
end


describe "Portfolio", :focus do
  it "has value of share price" do
    expect(stock_service).to receive(:pric

    @portfolio.purchase("QCOM", 1)

    expect(@portfolio.value).to equal 100
  end
end
```

```
class Portfolio
  def initialize
    @holdings = Hash.new
    @service = StockService.new
  end
  # …
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: hand-crafted test double

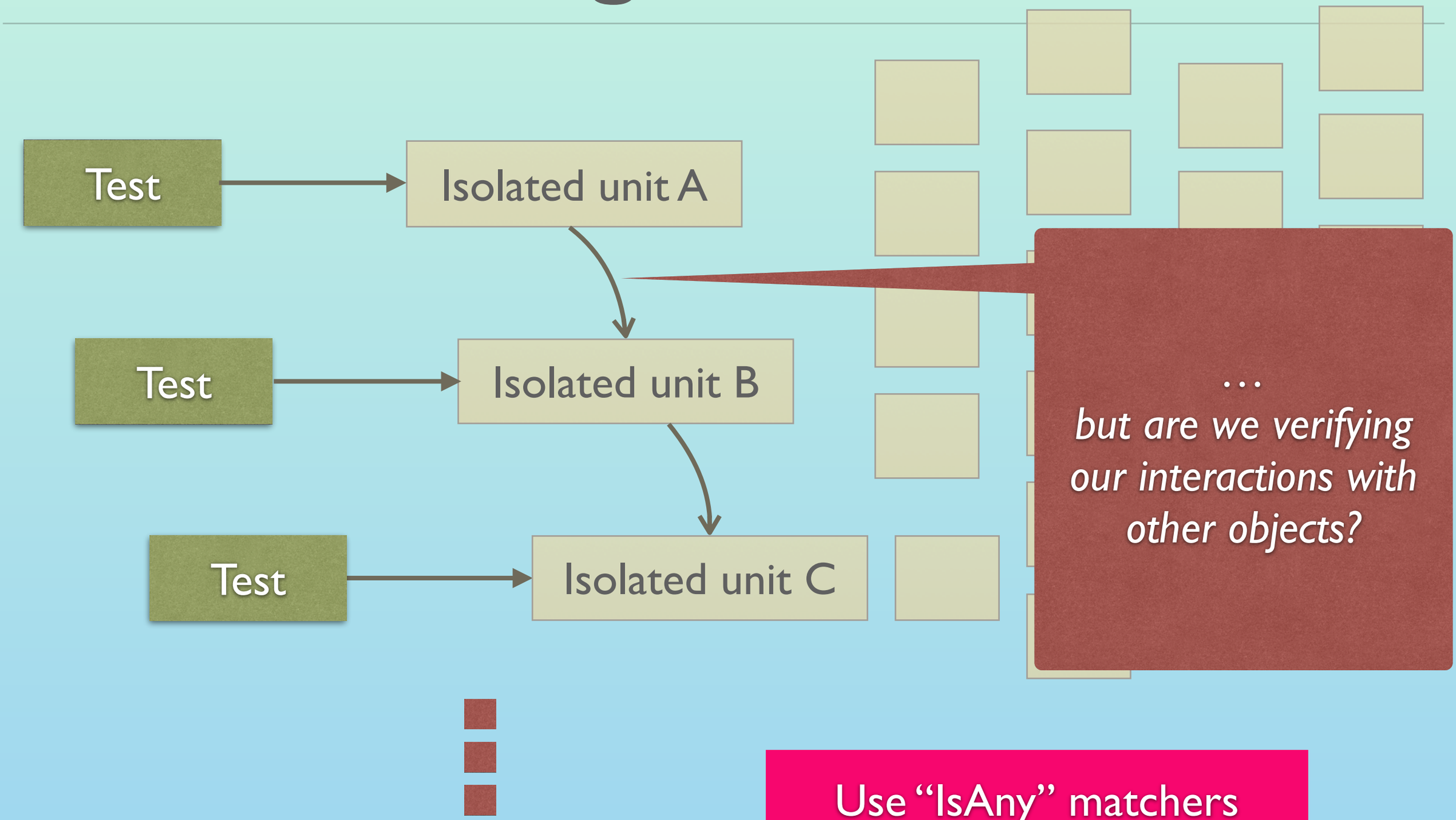
Test-drive posting a charge to the checkouts controller.

```
class CardProcessor
  def charge(name, card, exp, amount)
end
```

- `CardProcessor#charge` returns “Approved” or “Declined”
  - CardProcessor makes a call to 3rd party validation. Do not change this class.
- `./config/routes.rb` declares the route.
- `CheckoutsController#charge` already exists
  - It only takes on parameters so far.
- Return a 200 from the controller if approved, 406 otherwise
- Use an alternate injection technique

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

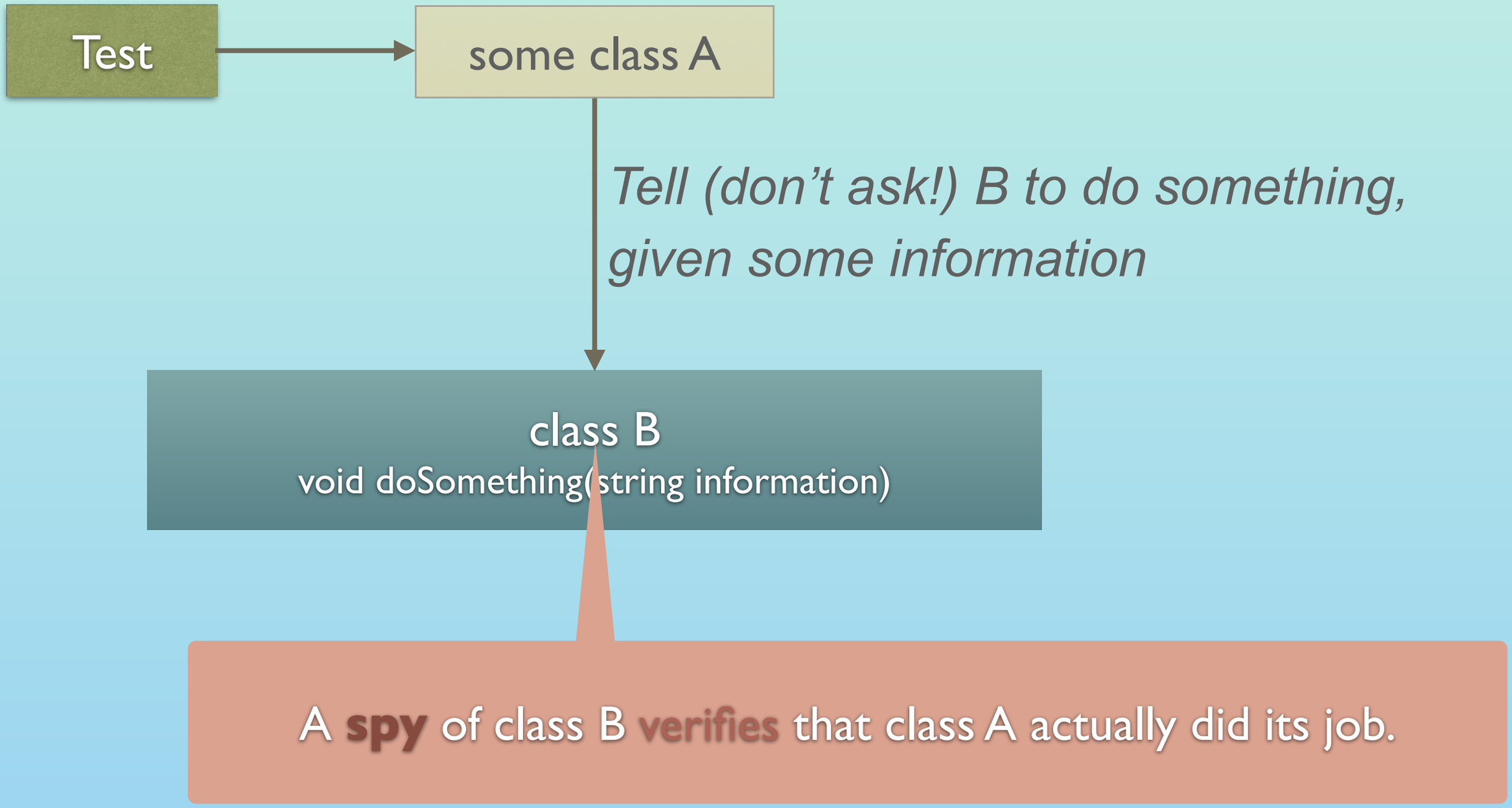
“Unit” testing



Use “IsAny” matchers sparingly!

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

“Tell, Don’t Ask”



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Test-driving interactions

Given this code:

```
def add(word, definition)
  @definitions[word] = definition
  @auditor.log("added to dictionary”)
end
```

We want to test-drive this.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

RSpec Mocks: Spy

```
it "logs audit record on add" do
  auditor_spy = spy("auditor")
  dictionary.auditor = auditor_spy

  dictionary.add("smelt", "a small fish")

  expect(auditor_spy).to have_received(:log)
    .with("added to dictionary")
end
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Spy / verify exercise

Verify that each portfolio purchase is audited via a call to this interface:

```
class Auditor
  log_purchase(symbol, shares, timestamp)
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

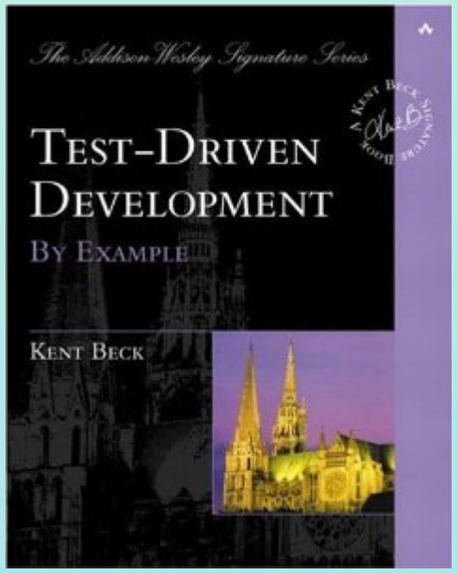
Schools of mock

Classic ("Detroit")

algorithmic approach

verification of state

tests drive code from specific to general

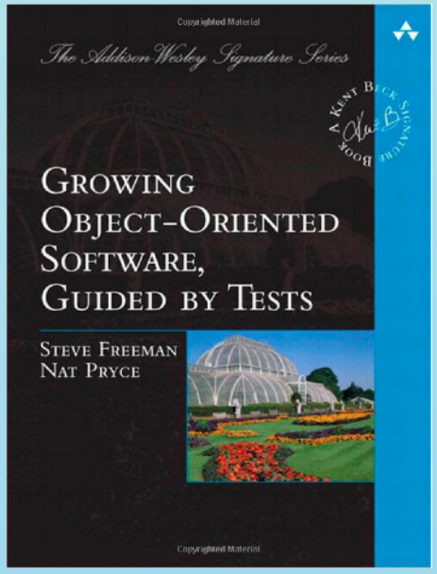


London

roles, responsibilities, and interactions

"Fake it until you make it"

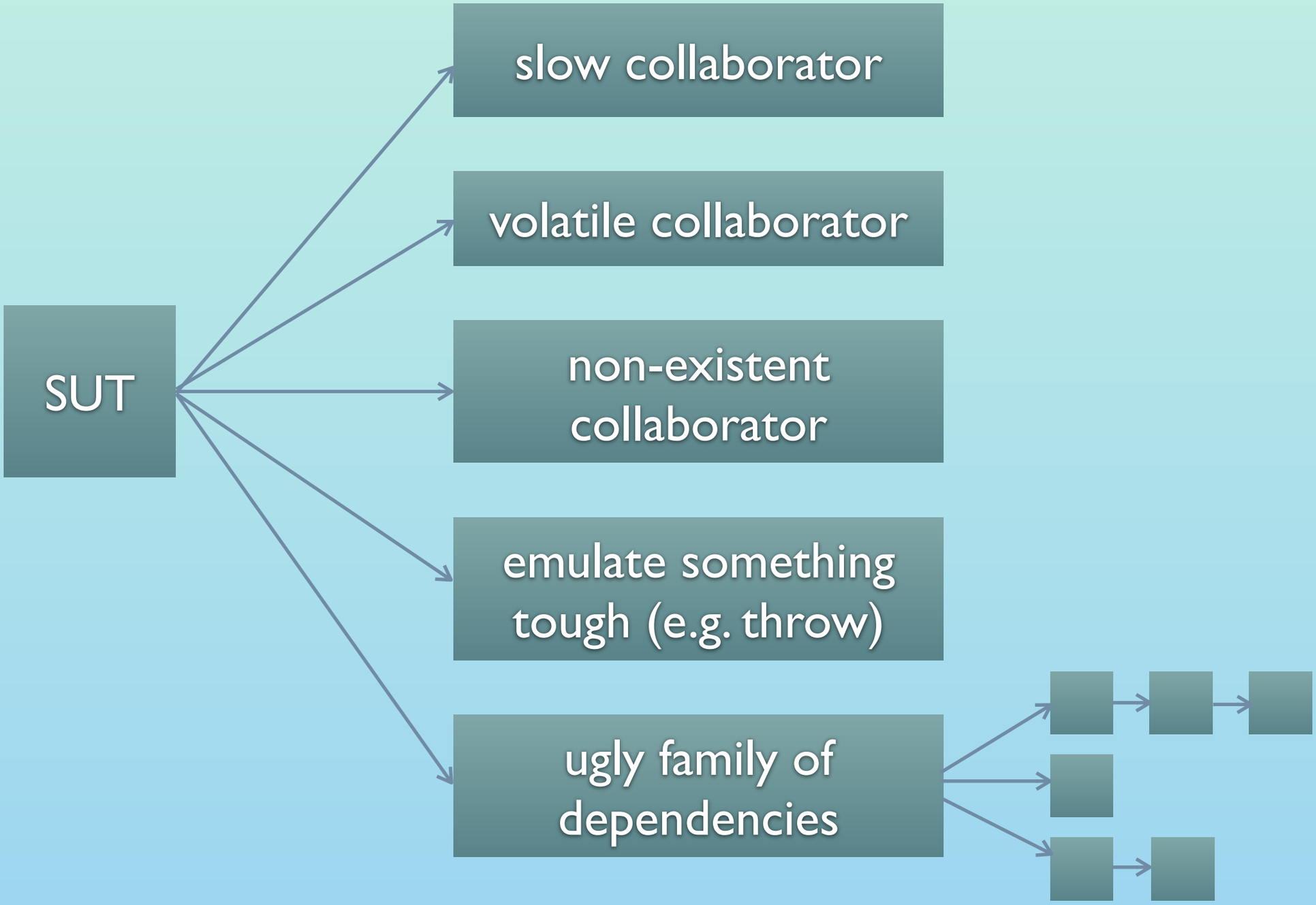
drive incrementally from the outside in



Summarized well in Martin Fowler's "Mocks Aren't Stubs"

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Classic school: When to mock



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

The pragmatic mocker

- Writes integration tests where mocks are used
- Watches coupling between tests & implementation
- Seeks to mock only direct collaborators
- Avoids fakes (and if necessary, test-drives 'em!)
- Isolates, minimizes the use of test doubles
  - But prefers testability

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Jester," courtesy https://www.flickr.com/photos/mrpolyonymous/
License: https://creativecommons.org/licenses/by/2.0/

Simplifying testing

How would you redesign the system so that the portfolio `value` function (or any of Portfolio) didn't need to involve test doubles?

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Jester," courtesy https://www.flickr.com/photos/mrpolyonymous/
License: https://creativecommons.org/licenses/by/2.0/

TDD wrap-up exercise

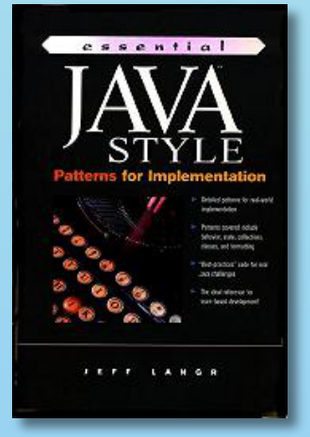
Test-drive that the POS fully handles multi-item discounts.

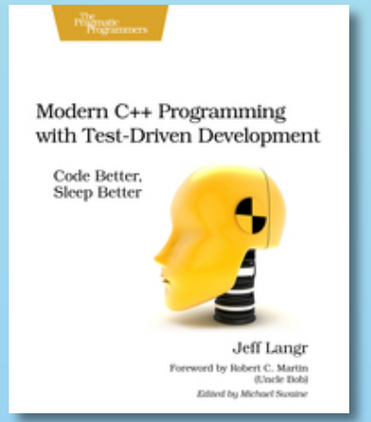
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

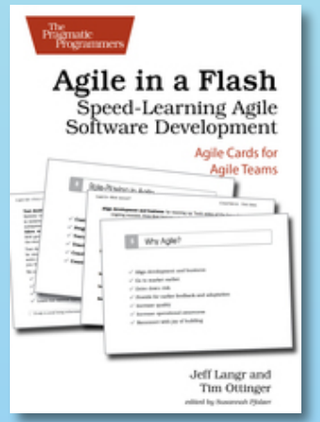
"barcode_scan3_code128.png," courtesy https://www.flickr.com/photos/mdornseif/
License: https://creativecommons.org/licenses/by-sa/2.0/

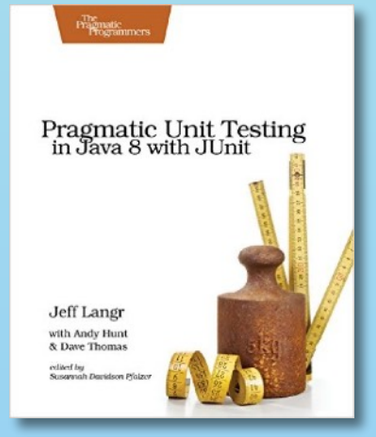
Sustaining TDD

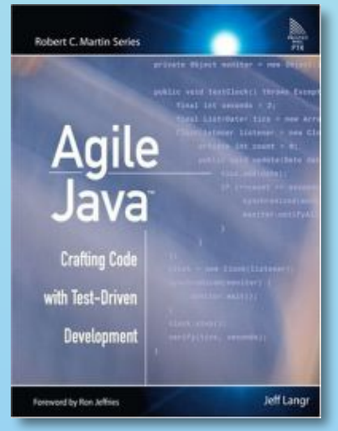
LANGR | SOFTWARE SOLUTIONS

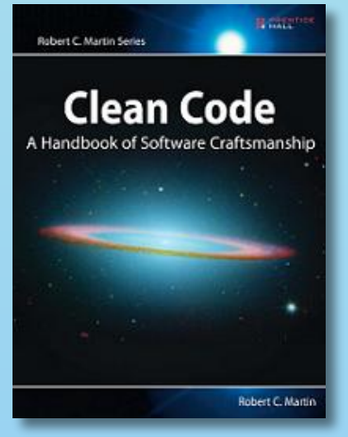












Next week

- TDD policies / standards
- CI build: “Stop the line”
- Reviews cover tests
- Kick off with mob programming

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Kickoff," courtesy https://www.flickr.com/photos/edrost88/
License: https://creativecommons.org/licenses/by/2.0/

Growing TDD

- Sharing sessions
- Daily stand-downs
- Challenges / contests
- Coaches & champions
- Influential, dynamic metrics
- Continued collaboration

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Kickoff," courtesy https://www.flickr.com/photos/10209472@N03/
License: https://creativecommons.org/licenses/by-sa/2.0/

Sustaining TDD

- "Why?"
- Review standards regularly
- Keep tests fast
- Use tests as an entry point
- Refactor, refactor, refactor
  - *Tests too*

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Infinity!" courtesy https://www.flickr.com/photos/thecampbell/
License: https://creativecommons.org/licenses/by/2.0/

Mastering TDD

TDD is a skill.

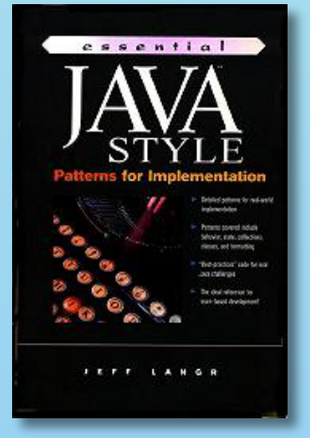
Practice it. **(Really.)**

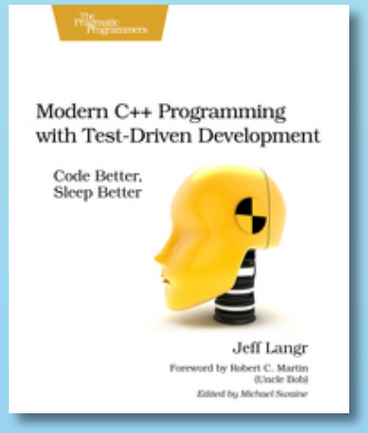
Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

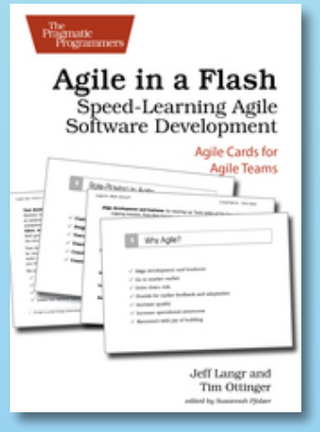
"Beautiful and dusty," courtesy https://www.flickr.com/photos/80435089@N08/
License: https://creativecommons.org/licenses/by/2.0/

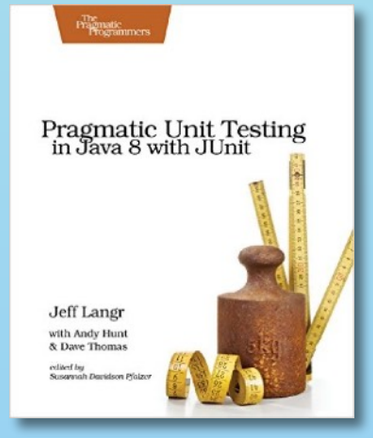
Miscellaneous Topics

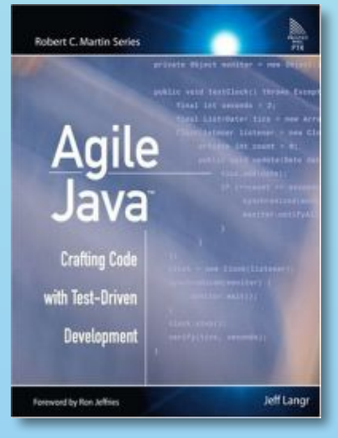
LANGR | SOFTWARE SOLUTIONS

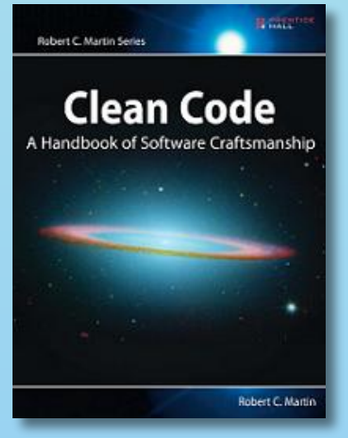












Rails: Model Testing

Using shoulda-matchers:

```
RSpec.describe Item, type: :model do
  it { should belong_to(:checkout) }
  it { should validate_presence_of(:name) }
  it { should validate_presence_of(:upc) }
end
```

Zzz.

```
class Item < ApplicationRecord
  belongs_to :checkout
  validates_presence_of :description
  validates_presence_of :upc
end
```


Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Creating dependent objects

```
describe("discounting BOGO") do
  subject(:discount) { BogoDiscount.create(upc: "123", reqd_for_discount: 2) }
  before(:context) do
    Item.create!(upc: "123", price: 2.00, description: "eggs")
  end
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

With FactoryBot + Faker

```
FactoryBot.define do
  factory :item do
    upc { Faker::Alphanumeric.alphanumeric(number: 8) }
    description { Faker::Lorem.word }
    is_exempt { Faker::Boolean.boolean(true_ratio: 0.05) }
    price { Faker::Number.between(from: 0.95, to: 8.95) }
  end
end
```

```
describe("discounting BOGO") do
  subject(:discount) { BogoDiscount.create(upc: "123", reqd_for_discount: 2) }
  before(:context) do
    create(:item, upc: "123", price: 2.00)
  end
  …
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

FactoryBot: Creating objects

`build`

instance that’s not saved (fastest)

`create`

instance that’s saved (slowest)

`build_stubbed`

instance w/ all defined attributes stubbed out.

will populate timestamps, id

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

build_stubbed and a mock

```
describe("discounting BOGO") do
  subject(:discount) { Discount.create(upc: "123", reqd_for_discount: 2 }
  let(:item) { build_stubbed(:item, upc: "123", price: 2.00) }
  before() { expect(Item).to receive(:find_by).and_return(item) }
```

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Exercise: Model test-driving

Items must be priced (and discounted) as they are scanned.

Support applying a multi-item discount to the latest scanned item in a checkout.

In the MultiItemDiscount model, test-drive a `price_last` method. Given a list of all item UPC scans (e.g. `[{upc: "123"}, {upc: "234"}]`, `price_last` should decorate the last scan with base & discounted prices:

```
{upc: "123", price: 2.00, discounted_price: 1.00}
```

- Price comes from the Item model, using the UPC that the MultiItemDiscount applies to.
- The last item by definition will be the UPC of the MultiItemDiscount.
- Discounted price is the regular item price unless the discount applies.
- The discount applies if this the ***nth*** item with the same scan code
  - ***n*** corresponds to `number_required_for_discount` (in the model)
- If the discount applies, the discounted price is price - (`discount_at_required` * price).
- A "BOGO:" `number_required_for_discount` => 2, `discount_at_required` => 1.0
- "Buy 2 get 3rd 50% off:" `number_required_for_discount` => 3, `discount_at_required` => 0.5
- Discounts continue to apply. For BOGO, every even item is free.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Controller Specs vs Request Specs

Request specs:

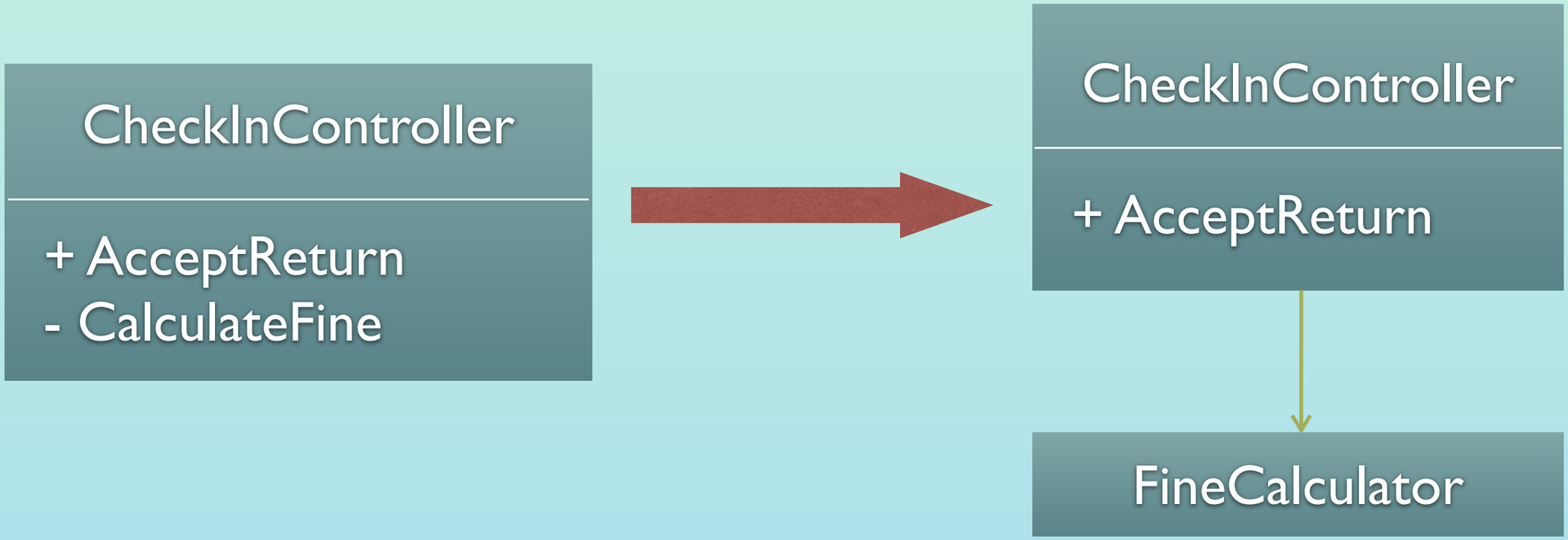
* Recommended by RSpec
* Test the entire stack, including routing

Controller specs:

* Simulate a single HTTP request per example
* Views not rendered
  * stubbed w/ template that renders “”

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Testing Private Behavior?



Or simply relax access.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Rules of Ten

Green & clean in < 10 (minutes)

Delete and repeat if you're late!

... taking smaller steps this time.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Egg Timer," courtesy https://www.flickr.com/photos/engendercdp/ spikeheap/
License: https://creativecommons.org/licenses/by/2.0/

Rules of Ten

Debate stops at 10 (minutes)

"Show me."

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Egg Timer," courtesy https://www.flickr.com/photos/engendercdp/ spikeheap/
License: https://creativecommons.org/licenses/by/2.0/

Rules of Ten

All unit tests run in < 10 (seconds).

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

"Egg Timer," courtesy https://www.flickr.com/photos/engendercdp/ spikeheap/
License: https://creativecommons.org/licenses/by/2.0/

Run Tagged Tests

```
it “ … ”, :focus do
```

Command line:

```
rspec --tag focus
```

Remember: Running a subset of tests increases risk.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Fixing a Slow Test Run

Warn on tests < ***n*** *ms*

Fail on tests > ***n*** ms

Running a subset of tests increases risk.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Uncle Bob's TPP

Transformations have a preferred ordering, which if maintained by ordering of tests, prevents "long outages" in TDD.

The Priority List

(**{}->nil)** no code at all->code that employs nil
(nil->constant)
(constant->constant+) a simple constant to a more complex constant
(constant->scalar) replacing a const. with a variable or an argument
(statement->statements) adding more unconditional statements.
(unconditional->if) splitting the execution path
(scalar->array)
(array->container)
(statement->tail-recursion)
(if->while)
(statement->recursion)
(expression->function) replacing expression w/ a function or algorithm
(variable->assignment) replacing the value of a variable.
(case) adding a case (or else) to an existing switch or if

"As the tests get more specific, the code gets more generic."

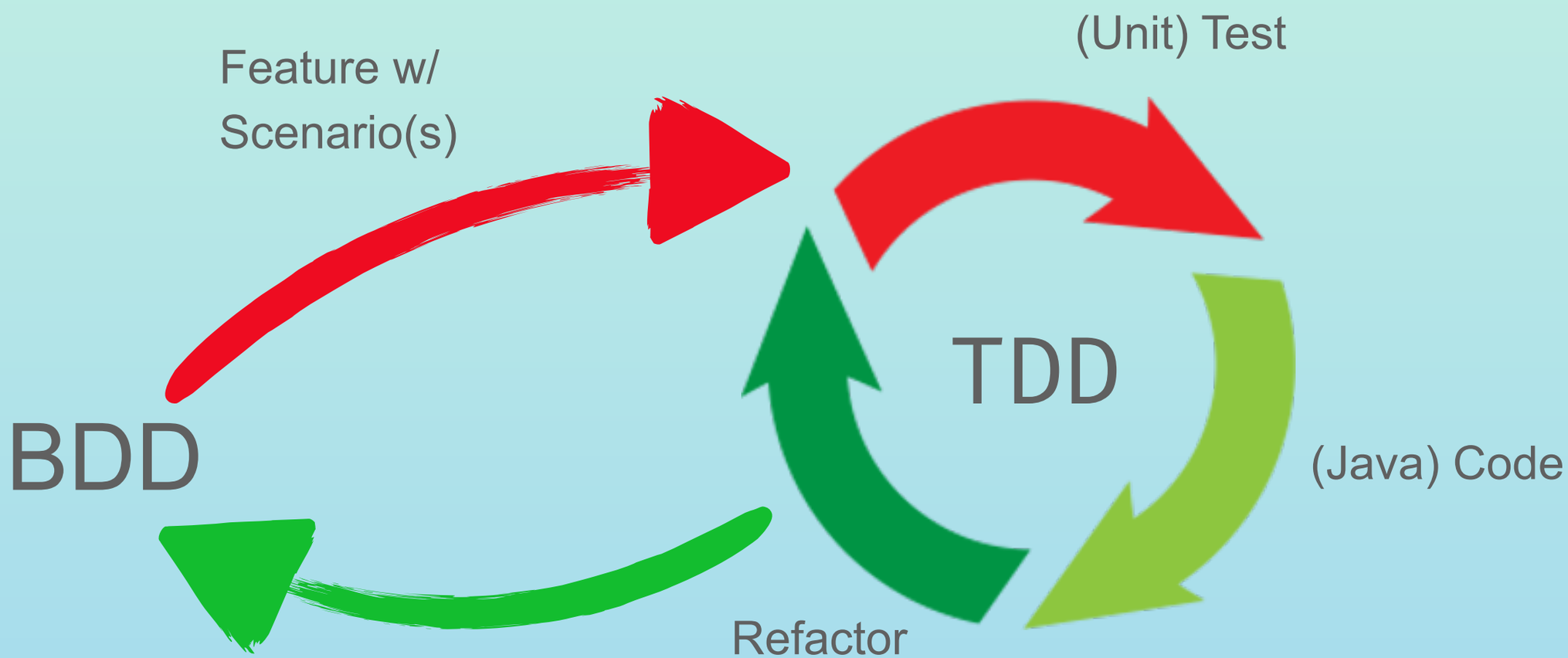
http://thecleancoder.blogspot.com/2011/02/fib-t-p-premise.html

https://8thlight.com/blog/uncle-bob/2013/05/27/TheTransformationPriorityPremise.html

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Per and updated by Robert C. Martin.

Behavior-Driven Development (BDD)



Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

References / Reading List

[Beck2002] Beck, Kent. **Test-Driven Development: By Example**. Addison-Wesley, 2002.

[Feathers2005] Feathers, Michael. **Working Effectively With Legacy Code**. Prentice Hall, 2005.

[Fowler1999] Fowler, Martin. **Refactoring: Improving the Design of Existing Code**. Addison-Wesley, 1999.

[Freeman, 2009]. Freeman, S.; Pryce, N. **Growing Object-Oriented Software, Guided By Tests**. Addison-Wesley, 2009.

[Langr2005] Langr, Jeff. **Agile Java: Crafting Code With Test-Driven Development**. Prentice Hall, 2005.

[Langr2011] Langr, Jeff and Ottinger, Tim. **Agile in a Flash**. Pragmatic Programmers, 2011.

[Langr2013] Langr, Jeff. **Modern C++ Programming With Test-Driven Development**, Pragmatic Programmers, 2013.

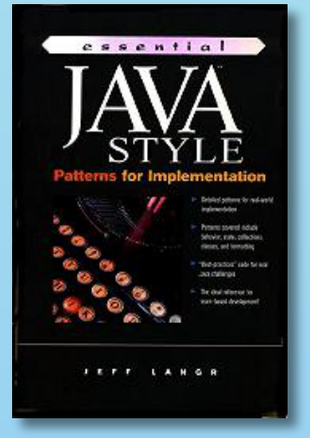
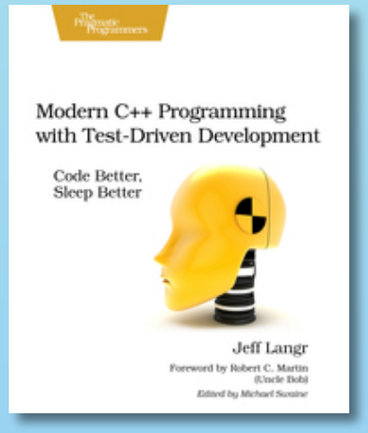
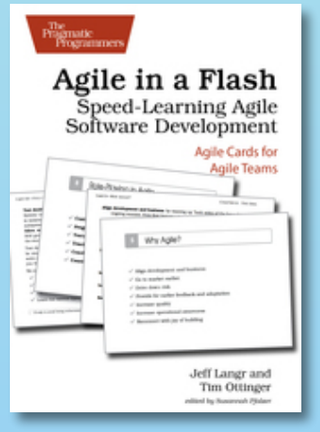
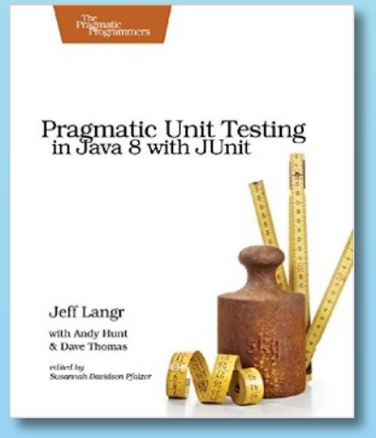
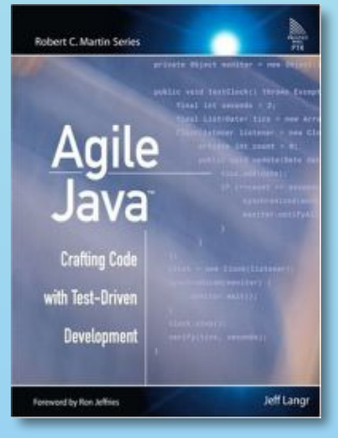
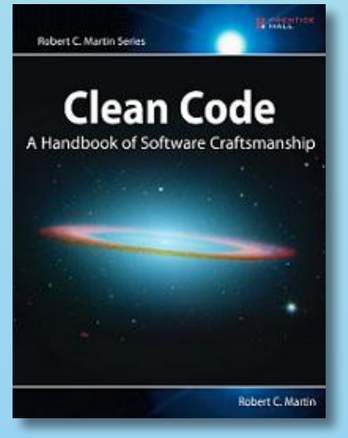
[Langr2015] Langr, Jeff, et. al. **Pragmatic Unit Testing in Java 8 with JUnit.** Pragmatic Programmers, 2015.

[Martin2002] Martin, Robert C. **Agile Software Development: Principles, Patterns, and Practices**. Prentice Hall, 2002.

[Metz2012] Metz, Sandi. **Practical Object-Oriented Design in Ruby: An Agile Primer**. Addison-Wesley, 2012.

[Meszaros2007] Meszaros, Gerard. **xUnit Test Patterns: Refactoring Test Code**. Addison-Wesley, 2007.

Copyright © Langr Software Solutions, Inc. 2021. All rights reserved.

Thank you!

LANGR | SOFTWARE SOLUTIONS