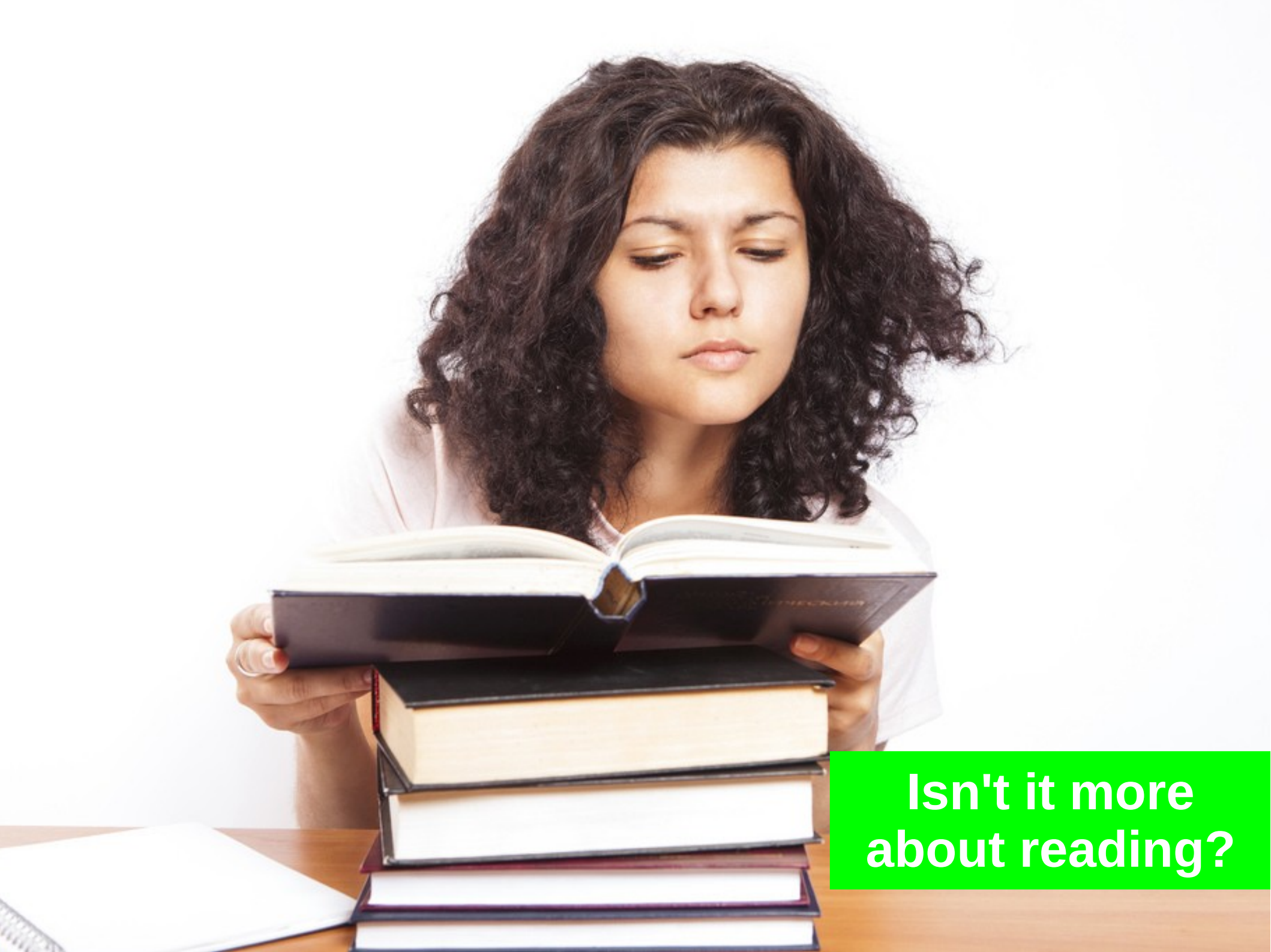
We write code


Isn't it more
about reading?

Written **once** – read **many** times

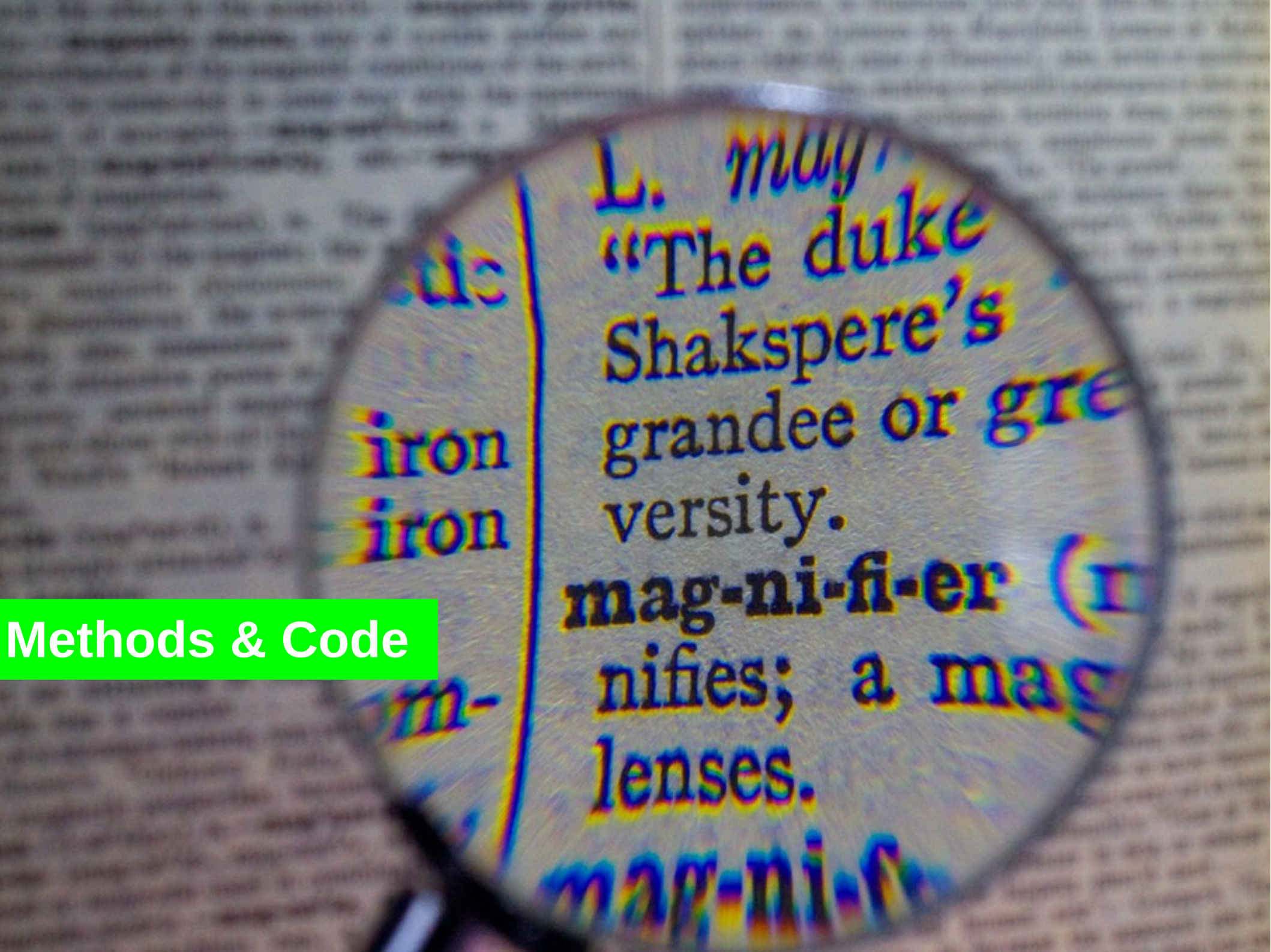
*„(...) when you program, you have to **think about** how someone **will read your code**, not just how a computer will interpret it.“*

Kent Beck

*„**Any fool** can write code that a **computer can understand**. **Good programmers** write code that **humans can understand**.“*

Martin Fowler


Not about architecture


Methods & Code


Nurturing a code base


Extra effort

Save time!

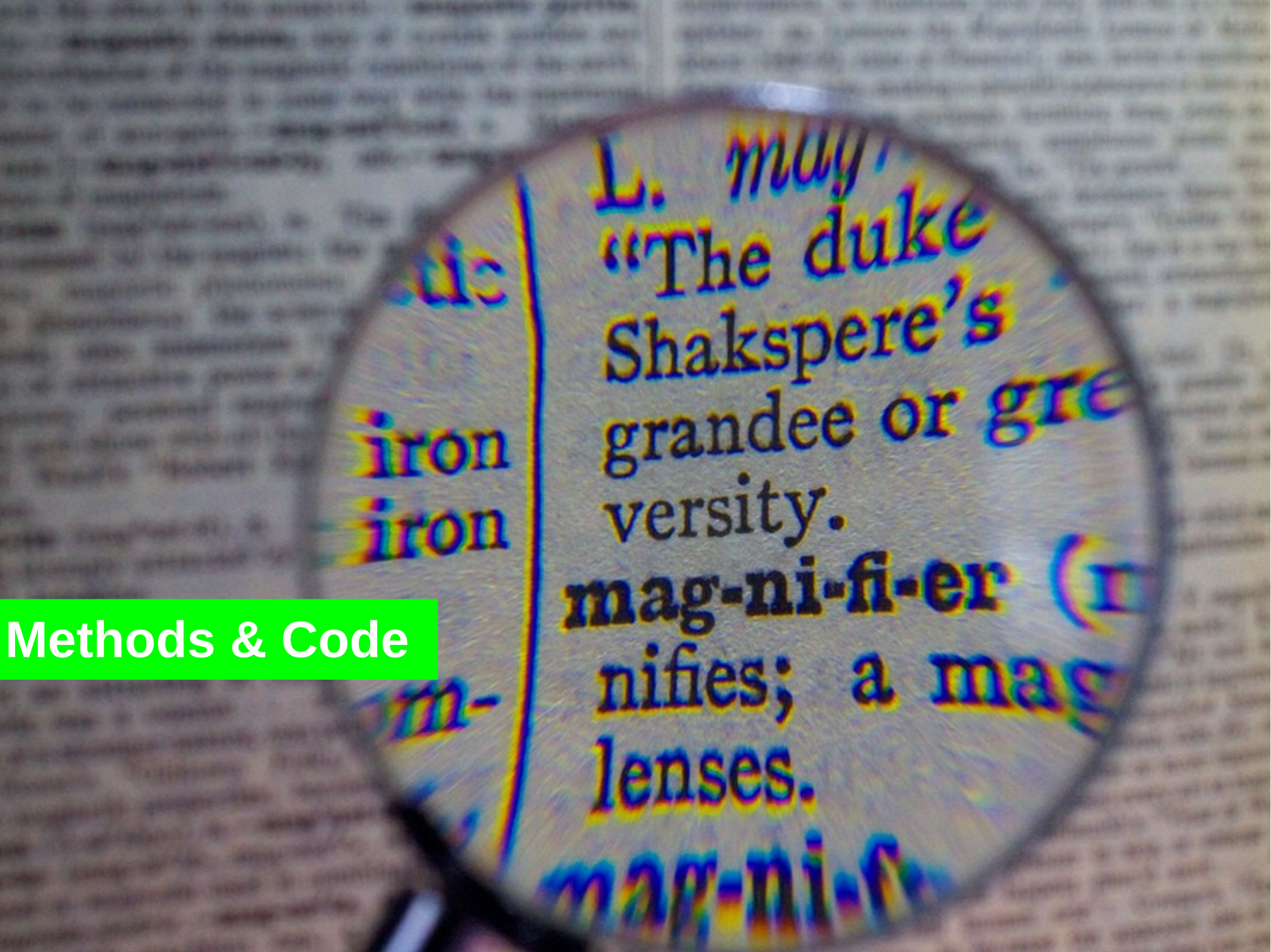

Your code base?


It's about joy!

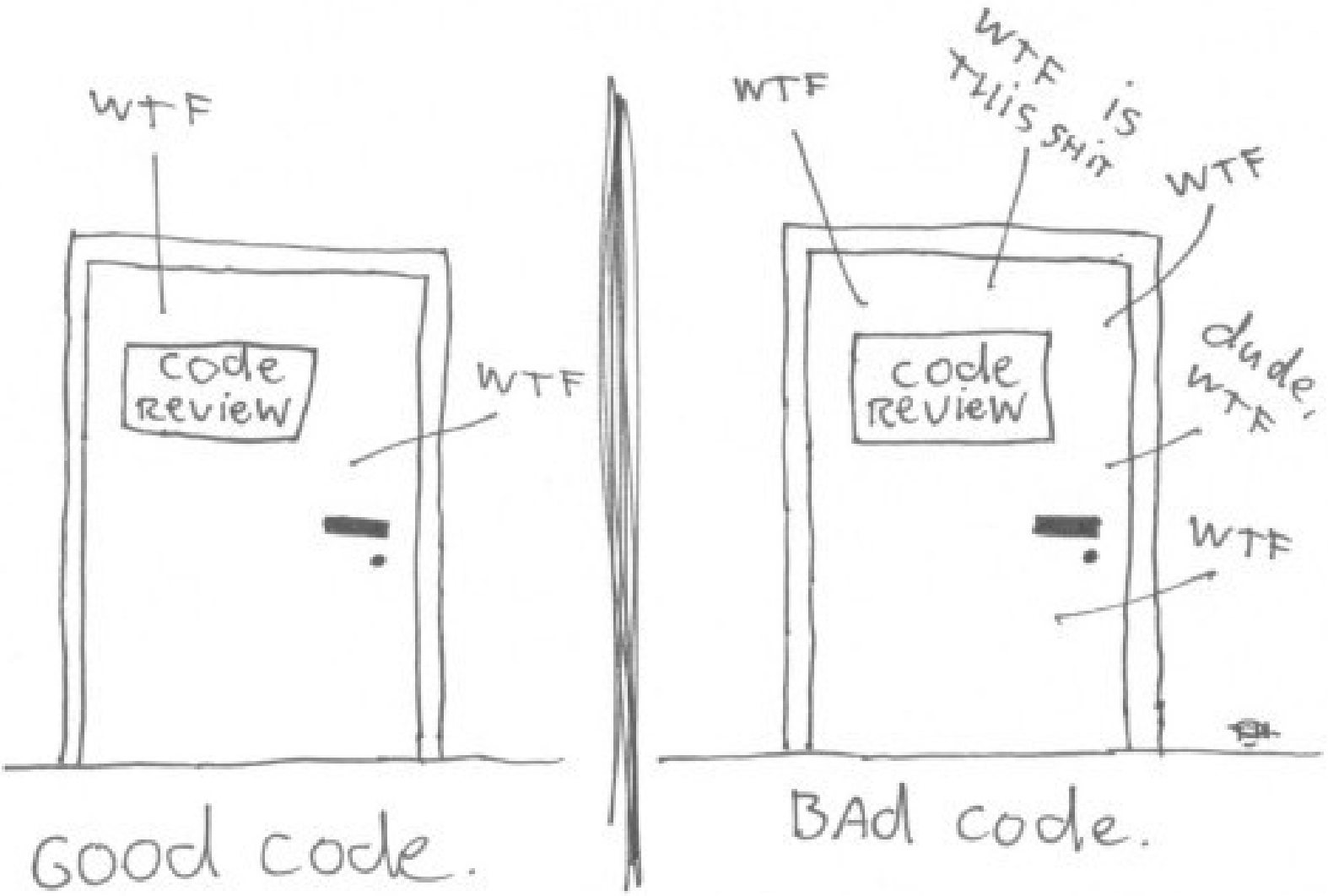
Optimizing for Readability

Tobias Pfeiffer
@PragTob
pragtob.info

Crazy?


Methods & Code

The ONLY VALID MEASUREMENT OF Code QUALITY: WTFs/MINUTE



Keep **I**t **S**imple **S**tupid

Are comments a **smell**?

Comments are an **excuse** of the code that it could not be **clearer**.

Outdated comments are the **worst**

The **why** not the **what**

```
def paint_control(event)
  # some painting code
rescue => e
  # Really important to rescue here.
    Failures that escape this method
  # cause odd-ball hangs with no
    backtraces. See #559 for an example.
  #
  puts "SWALLOWED PAINT EXCEPTION ON
#{@obj} - go take care of it: " + e.to_s
  puts 'Unfortunately we have to swallow
it because it causes odd failures :('
end
```

Also known as the smell that **tries** to make other **smells seem ok**

```
# do one thing
...
...
...
...
...

# do another thing
...
...
...
...

# do something more
...
...
```

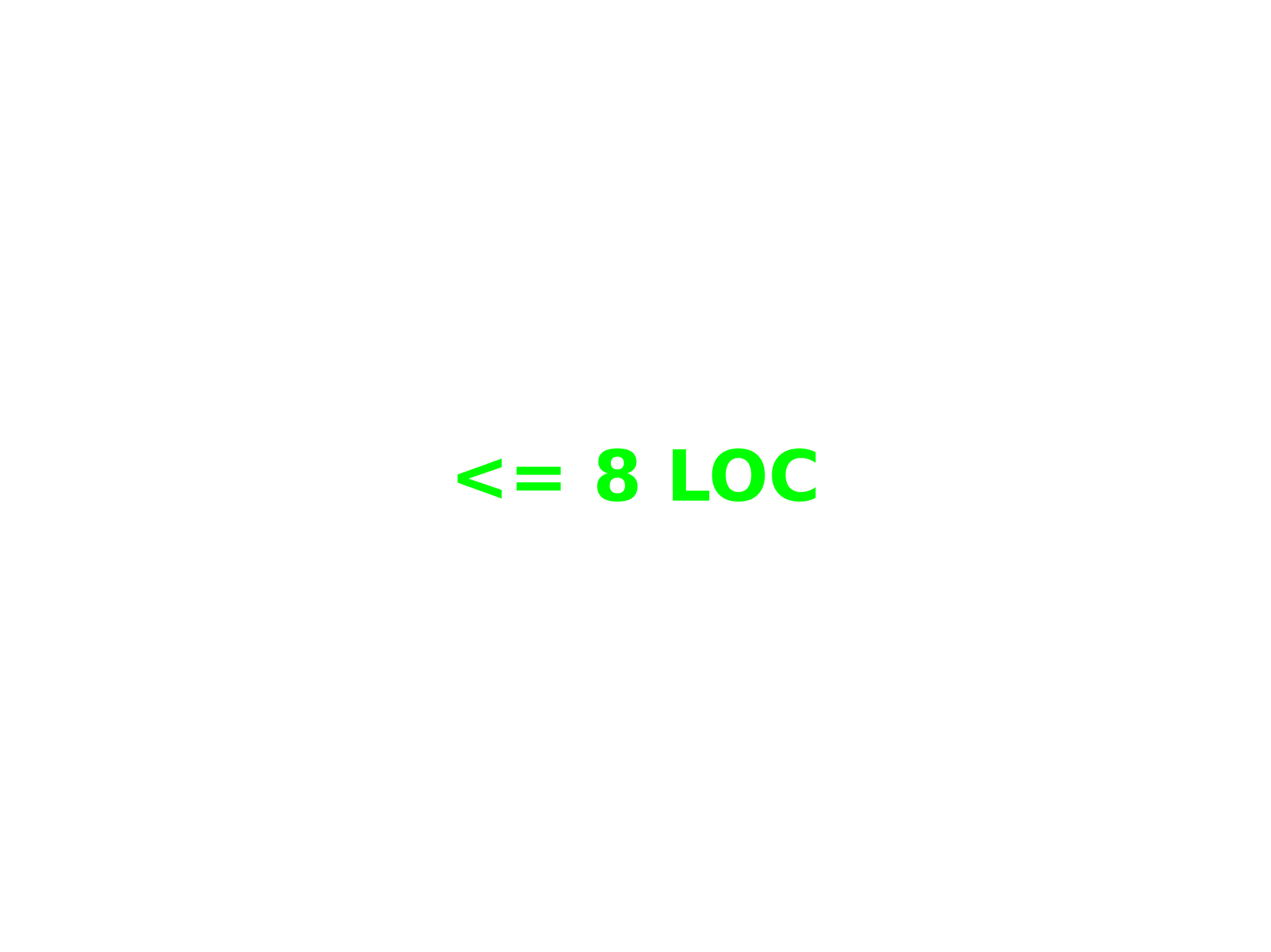
```
# do one thing
...
...
...
...
...

# do another thing
...
...
...
...

# do something more
...
...
```

```
# do one thing
...
...
...
...
...

# do another thing
...
...
...
...

# do something more
...
...
```

Cocepts

Method too long

Short Methods


<= 8 LOC

Extract Methods

```
do_one_thing
do_another_thing
do_something_more
```

Cocepts

```
# context, outlet, times, time per
step, state, data
def pattern(c, o, t, l, s, d)
  # ...
end
```

Incomprehensible names

```
# context, outlet, times, time per
step, state, data
def pattern(c, o, t, l, s, d)
  # ...
end
```

```
# context, outlet, times, time per
step, state, data
def pattern(c, o, t, l, s, d)
  # ...
end
```

Explanatory names

Naming is **hard**

```
def pattern(context, outlet, time,
            time_per_step, state,
            data)
  # ...
end
```

Argument **order dependency**

Try to keep it to **2 parameters**


Example

```ruby
# allowed to drink?
if customer.age >= 18
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{['cola', 'mate'].sample}?"
end
```

```
# allowed to drink?
if customer.age >= 18
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{['cola', 'mate'].sample}?"
end
```

No **magic numbers**

```
NON_ALCOHOLIC_DRINKS = ['cola', 'mate']
MIN_DRINKING_AGE     = 18
```

```ruby
# allowed to drink?
if customer.age >= MIN_DRINKING_AGE
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{NON_ALCOHOLIC_DRINKS.sample}?"
end
```

```ruby
# allowed to drink?
if customer.age >= MIN_DRINKING_AGE
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{NON_ALCOHOLIC_DRINKS.sample}?"
end
```

```
# allowed to drink?
if customer.age >= MIN_DRINKING_AGE
   say 'Okay'
   drink = prepare_drink requested_drink
   say 'here you go'
   hand_drink_over drink, customer
else
   say 'I am sorry you are not legally
        allowed rather to drink here'
   say "Would you rather have a
        #{NON_ALCOHOLIC_DRINKS.sample}?"
end
```

Query method

Intention revealing method

```
# ...
text.color = red
# ...
```

```
# ...
text.color = red
# ...
```

```
# ...
highlight(text)
# ...
```

```
def highlight(text)
   text.color = red
end
```

```
def highlight(text)
   text.color     = red
   text.underline = true
   update_highlights
end
```

```
# ...
text.color      = red
text.underline = true
update_highlights
# ...
```

```
# ...
highlight(text)
# ...
```

```ruby
# allowed to drink?
if customer.age >= MIN_DRINKING_AGE
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{NON_ALCOHOLIC_DRINKS.sample}?"
end
```

```ruby
# allowed to drink?
if customer.age >= MIN_DRINKING_AGE
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{NON_ALCOHOLIC_DRINKS.sample}?"
end
```

```
# allowed to drink?
if customer.age >= MIN_DRINKING_AGE
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{NON_ALCOHOLIC_DRINKS.sample}?"
end
```

```ruby
# allowed to drink?
if customer.age >= MIN_DRINKING_AGE
  say 'Okay'
  drink = prepare_drink requested_drink
  say 'here you go'
  hand_drink_over drink, customer
else
  say 'I am sorry you are not legally
       allowed rather to drink here'
  say "Would you rather have a
       #{NON_ALCOHOLIC_DRINKS.sample}?"
end
```

```
if allowed_to_drink_alcohol?(customer)
  serve_drink requested_drink,
              customer
else
  propose_non_alcoholic_drink
end
```

*„If you have a **good name** for a method you **don't need to look at the body**.“*

Martin Fowler

„The easiest code to ***understand*** *is the code you* ***don't have to read*** *at all.“*

Tom Stuart (Berlin)

```
prepare_drink requested_drink
price = requested_drink.price
check = Check.new
check.add_price price
say 'That whill be ' + check.total
```

```
prepare_drink requested_drink
price = requested_drink.price
check = Check.new
check.add_price price
say 'That whill be ' + check.total
```

```
prepare drink requested_drink
price = requested_drink.price
check = Check.new
check.add_price price
say 'That whill be ' + check.total
```

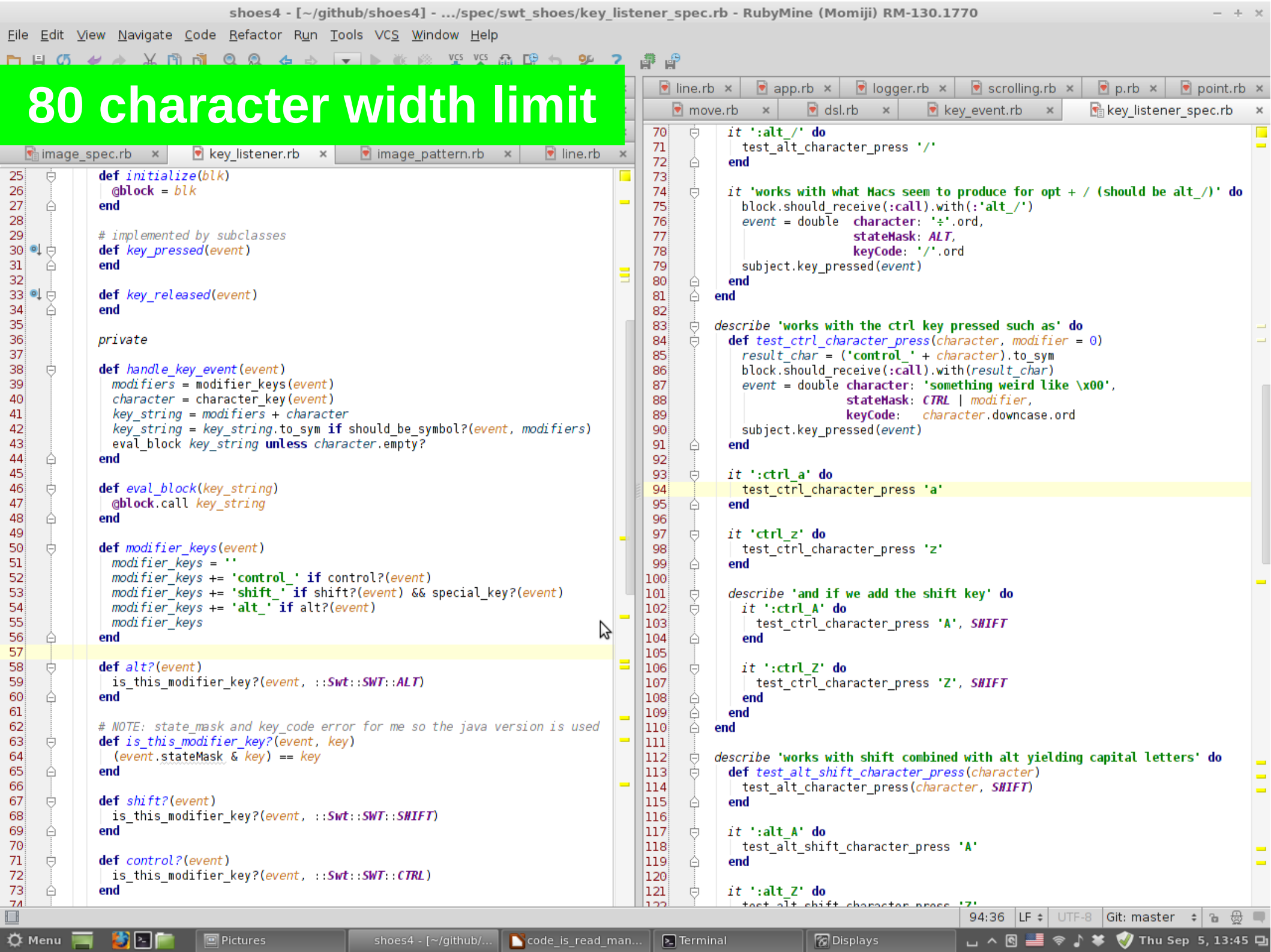
Same **level of abstraction** in a method

```
prepare_drink requested_drink
prepare_check requested_drink
```

Nice code **formatting**

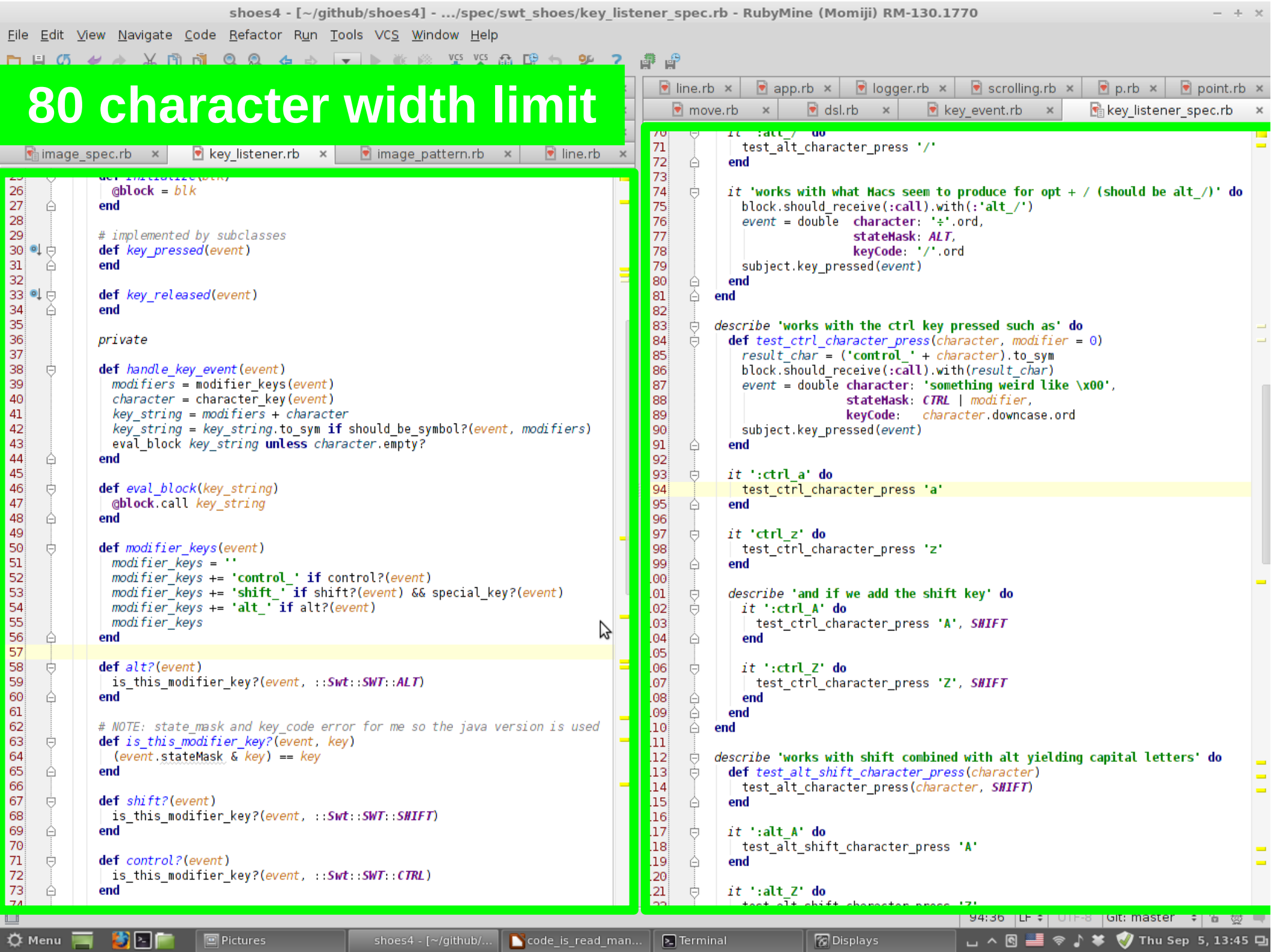
```
@left   ||= 0
@top    ||= 0
@width  ||= 1.0
@height ||= 0
```

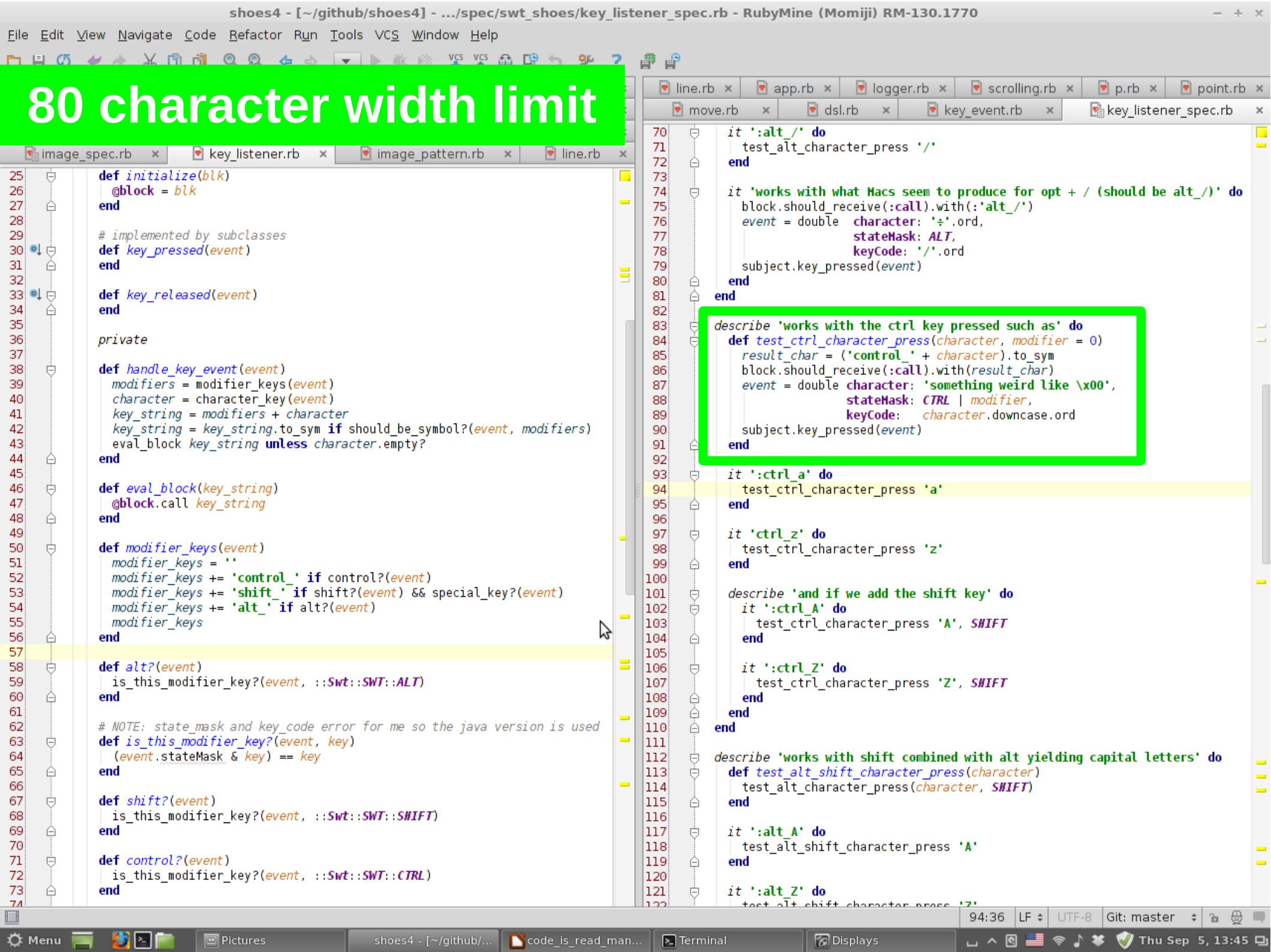
```
double character: 'something weird',
       stateMask: CTRL | modifier,
       KeyCode:   character.downcase.ord
```




```ruby
def initialize(blk)
  @block = blk
end

# implemented by subclasses
def key_pressed(event)
end

def key_released(event)
end

private

def handle_key_event(event)
  modifiers = modifier_keys(event)
  character = character_key(event)
  key_string = modifiers + character
  key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
  eval_block key_string unless character.empty?
end

def eval_block(key_string)
  @block.call key_string
end

def modifier_keys(event)
  modifier_keys = ''
  modifier_keys += 'control_' if control?(event)
  modifier_keys += 'shift_' if shift?(event) && special_key?(event)
  modifier_keys += 'alt_' if alt?(event)
  modifier_keys
end

def alt?(event)
  is_this_modifier_key?(event, ::Swt::SWT::ALT)
end

# NOTE: state_mask and key_code error for me so the java version is used
def is_this_modifier_key?(event, key)
  (event.stateMask & key) == key
end

def shift?(event)
  is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
end

def control?(event)
  is_this_modifier_key?(event, ::Swt::SWT::CTRL)
end
```

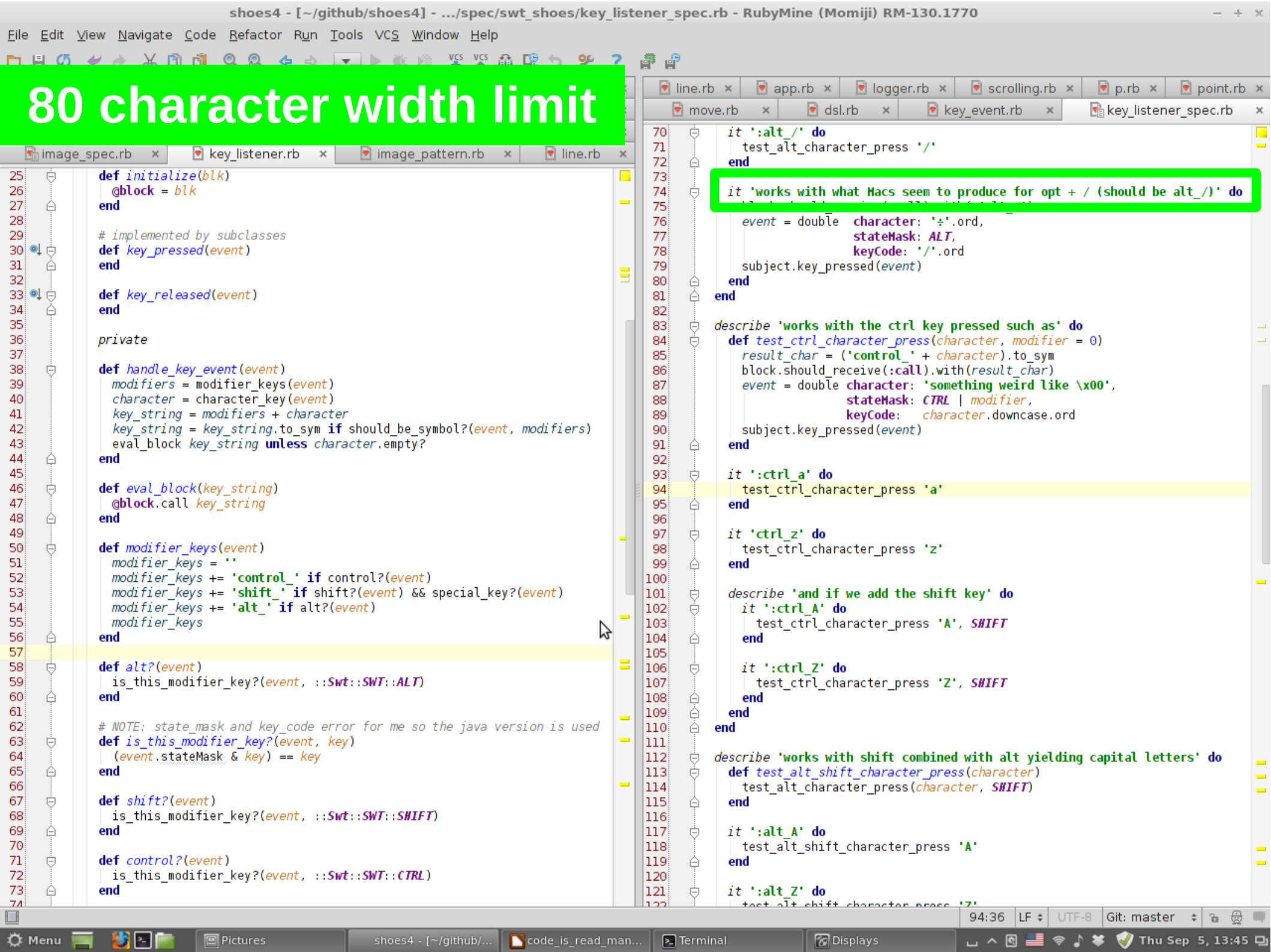
```ruby
  it ':alt_/' do
    test_alt_character_press '/'
  end

  it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
    block.should_receive(:call).with(:'alt_/')
    event = double  character: '÷'.ord,
                    stateMask: ALT,
                    keyCode: '/'.ord
    subject.key_pressed(event)
  end
end

describe 'works with the ctrl key pressed such as' do
  def test_ctrl_character_press(character, modifier = 0)
    result_char = ('control_' + character).to_sym
    block.should_receive(:call).with(result_char)
    event = double character: 'something weird like \x00',
                   stateMask: CTRL | modifier,
                   keyCode:   character.downcase.ord
    subject.key_pressed(event)
  end

  it ':ctrl_a' do
    test_ctrl_character_press 'a'
  end

  it 'ctrl_z' do
    test_ctrl_character_press 'z'
  end

  describe 'and if we add the shift key' do
    it ':ctrl_A' do
      test_ctrl_character_press 'A', SHIFT
    end

    it ':ctrl_Z' do
      test_ctrl_character_press 'Z', SHIFT
    end
  end
end

describe 'works with shift combined with alt yielding capital letters' do
  def test_alt_shift_character_press(character)
    test_alt_character_press(character, SHIFT)
  end

  it ':alt_A' do
    test_alt_shift_character_press 'A'
  end

  it ':alt_Z' do
```


shoes4 - [~/github/shoes4] - .../spec/swt_shoes/key_listener_spec.rb - RubyMine (Momiji) RM-130.1770
File Edit View Navigate Code Refactor Run Tools VCS Window Help
80 character width limit
image_spec.rb
key_listener.rb
image_pattern.rb
line.rb
line.rb
app.rb
logger.rb
scrolling.rb
p.rb
point.rb
move.rb
dsl.rb
key_event.rb
key_listener_spec.rb
@block = blk
end
implemented by subclasses
def key_pressed(event)
end
def key_released(event)
end
private
def handle_key_event(event)
modifiers = modifier_keys(event)
character = character_key(event)
key_string = modifiers + character
key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
eval_block key_string unless character.empty?
end
def eval_block(key_string)
@block.call key_string
end
def modifier_keys(event)
modifier_keys = ''
modifier_keys += 'control_' if control?(event)
modifier_keys += 'shift_' if shift?(event) && special_key?(event)
modifier_keys += 'alt_' if alt?(event)
modifier_keys
end
def alt?(event)
is_this_modifier_key?(event, ::Swt::SWT::ALT)
end
NOTE: state_mask and key_code error for me so the java version is used
def is_this_modifier_key?(event, key)
(event.stateMask & key) == key
end
def shift?(event)
is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
end
def control?(event)
is_this_modifier_key?(event, ::Swt::SWT::CTRL)
end
test_alt_character_press '/'
end
it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
block.should_receive(:call).with(:'alt_/')
event = double character: '÷'.ord,
stateMask: ALT,
keyCode: '/'.ord
subject.key_pressed(event)
end
end
describe 'works with the ctrl key pressed such as' do
def test_ctrl_character_press(character, modifier = 0)
result_char = ('control_' + character).to_sym
block.should_receive(:call).with(result_char)
event = double character: 'something weird like \x00',
stateMask: CTRL | modifier,
keyCode: character.downcase.ord
subject.key_pressed(event)
end
it ':ctrl_a' do
test_ctrl_character_press 'a'
end
it 'ctrl_z' do
test_ctrl_character_press 'z'
end
describe 'and if we add the shift key' do
it ':ctrl_A' do
test_ctrl_character_press 'A', SHIFT
end
it ':ctrl_Z' do
test_ctrl_character_press 'Z', SHIFT
end
end
end
describe 'works with shift combined with alt yielding capital letters' do
def test_alt_shift_character_press(character)
test_alt_character_press(character, SHIFT)
end
it ':alt_A' do
test_alt_shift_character_press 'A'
end
it ':alt_Z' do
94:36 LF UTF-8 Git: master
Menu Pictures shoes4 - [~/github/... code_is_read_man... Terminal Displays Thu Sep 5, 13:45



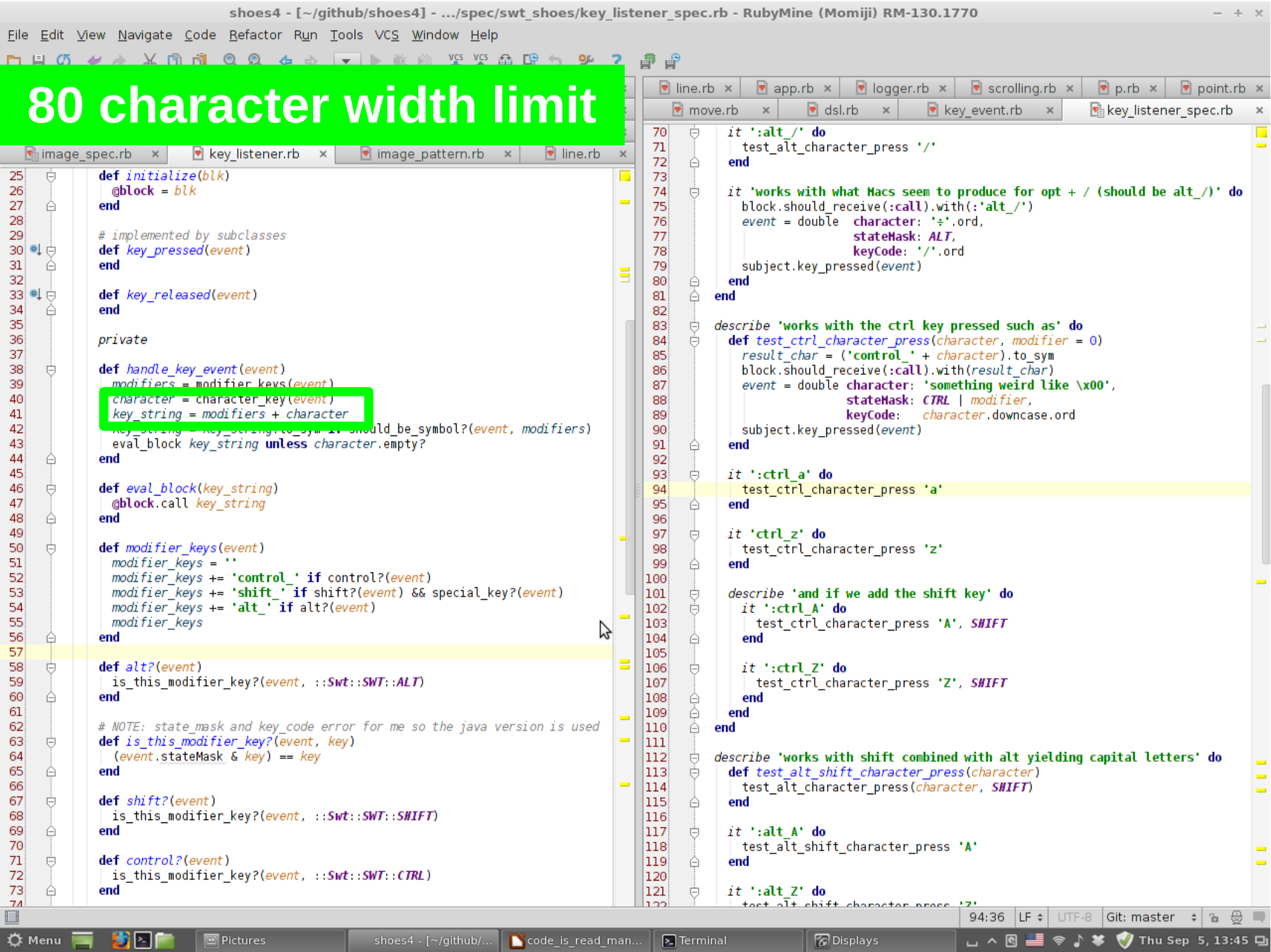

```ruby
def initialize(blk)
  @block = blk
end

# implemented by subclasses
def key_pressed(event)
end

def key_released(event)
end

private

def handle_key_event(event)
  modifiers = modifier_keys(event)
  character = character_key(event)
  key_string = modifiers + character
  key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
  eval_block key_string unless character.empty?
end

def eval_block(key_string)
  @block.call key_string
end

def modifier_keys(event)
  modifier_keys = ''
  modifier_keys += 'control_' if control?(event)
  modifier_keys += 'shift_' if shift?(event) && special_key?(event)
  modifier_keys += 'alt_' if alt?(event)
  modifier_keys
end

def alt?(event)
  is_this_modifier_key?(event, ::Swt::SWT::ALT)
end

# NOTE: state_mask and key_code error for me so the java version is used
def is_this_modifier_key?(event, key)
  (event.stateMask & key) == key
end

def shift?(event)
  is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
end

def control?(event)
  is_this_modifier_key?(event, ::Swt::SWT::CTRL)
end
```

```ruby
  it ':alt_/' do
    test_alt_character_press '/'
  end

  it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
    block.should_receive(:call).with(:'alt_/')
    event = double  character: '÷'.ord,
                    stateMask: ALT,
                    keyCode: '/'.ord
    subject.key_pressed(event)
  end
end

describe 'works with the ctrl key pressed such as' do
  def test_ctrl_character_press(character, modifier = 0)
    result_char = ('control_' + character).to_sym
    block.should_receive(:call).with(result_char)
    event = double character: 'something weird like \x00',
                   stateMask: CTRL | modifier,
                   keyCode:   character.downcase.ord
    subject.key_pressed(event)
  end

  it ':ctrl_a' do
    test_ctrl_character_press 'a'
  end

  it 'ctrl_z' do
    test_ctrl_character_press 'z'
  end

  describe 'and if we add the shift key' do
    it ':ctrl_A' do
      test_ctrl_character_press 'A', SHIFT
    end

    it ':ctrl_Z' do
      test_ctrl_character_press 'Z', SHIFT
    end
  end
end

describe 'works with shift combined with alt yielding capital letters' do
  def test_alt_shift_character_press(character)
    test_alt_character_press(character, SHIFT)
  end

  it ':alt_A' do
    test_alt_shift_character_press 'A'
  end

  it ':alt_Z' do
```




```ruby
    def initialize(blk)
      @block = blk
    end

    # implemented by subclasses
    def key_pressed(event)
    end

    def key_released(event)
    end

    private

    def handle_key_event(event)
      modifiers = modifier_keys(event)
      character = character_key(event)
      key_string = modifiers + character
      key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
      eval_block key_string unless character.empty?
    end

    def eval_block(key_string)
      @block.call key_string
    end

    def modifier_keys(event)
      modifier_keys = ''
      modifier_keys += 'control_' if control?(event)
      modifier_keys += 'shift_' if shift?(event) && special_key?(event)
      modifier_keys += 'alt_' if alt?(event)
      modifier_keys
    end

    def alt?(event)
      is_this_modifier_key?(event, ::Swt::SWT::ALT)
    end

    # NOTE: state_mask and key_code error for me so the java version is used
    def is_this_modifier_key?(event, key)
      (event.stateMask & key) == key
    end

    def shift?(event)
      is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
    end

    def control?(event)
      is_this_modifier_key?(event, ::Swt::SWT::CTRL)
    end
```

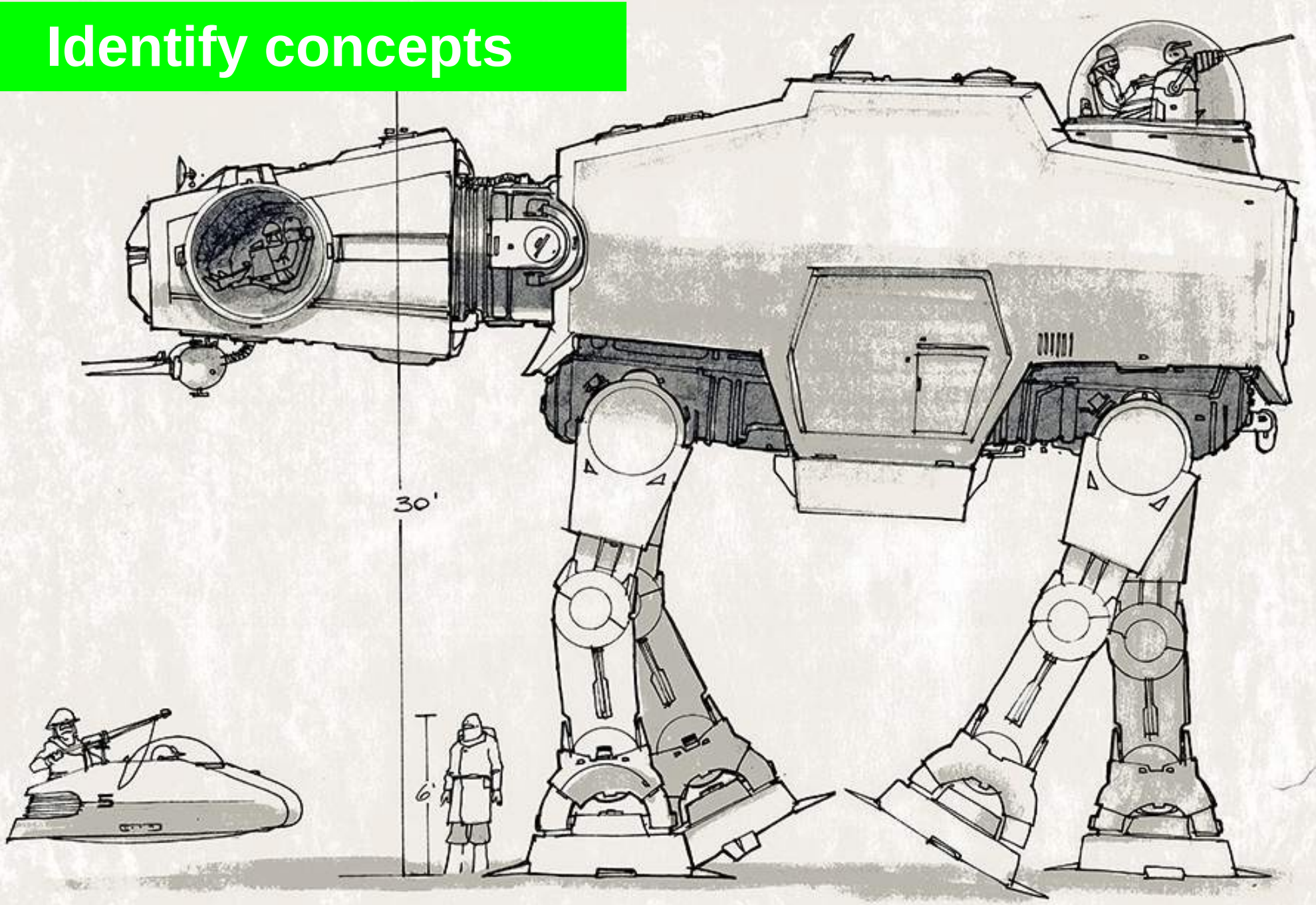
```ruby
  it ':alt_/' do
    test_alt_character_press '/'
  end

  it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
    event = double  character: '÷'.ord,
                    stateMask: ALT,
                    keyCode: '/'.ord
    subject.key_pressed(event)
  end
end

describe 'works with the ctrl key pressed such as' do
  def test_ctrl_character_press(character, modifier = 0)
    result_char = ('control_' + character).to_sym
    block.should_receive(:call).with(result_char)
    event = double character: 'something weird like \x00',
                   stateMask: CTRL | modifier,
                   keyCode:   character.downcase.ord
    subject.key_pressed(event)
  end

  it ':ctrl_a' do
    test_ctrl_character_press 'a'
  end

  it 'ctrl_z' do
    test_ctrl_character_press 'z'
  end

  describe 'and if we add the shift key' do
    it ':ctrl_A' do
      test_ctrl_character_press 'A', SHIFT
    end

    it ':ctrl_Z' do
      test_ctrl_character_press 'Z', SHIFT
    end
  end
end

describe 'works with shift combined with alt yielding capital letters' do
  def test_alt_shift_character_press(character)
    test_alt_character_press(character, SHIFT)
  end

  it ':alt_A' do
    test_alt_shift_character_press 'A'
  end

  it ':alt_Z' do
```




```ruby
def initialize(blk)
  @block = blk
end

# implemented by subclasses
def key_pressed(event)
end

def key_released(event)
end

private

def handle_key_event(event)
  modifiers = modifier_keys(event)
  character = character_key(event)
  key_string = modifiers + character
  key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
  eval_block key_string unless character.empty?
end

def eval_block(key_string)
  @block.call key_string
end

def modifier_keys(event)
  modifier_keys = ''
  modifier_keys += 'control_' if control?(event)
  modifier_keys += 'shift_' if shift?(event) && special_key?(event)
  modifier_keys += 'alt_' if alt?(event)
  modifier_keys
end

def alt?(event)
  is_this_modifier_key?(event, ::Swt::SWT::ALT)
end

# NOTE: state_mask and key_code error for me so the java version is used
def is_this_modifier_key?(event, key)
  (event.stateMask & key) == key
end

def shift?(event)
  is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
end

def control?(event)
  is_this_modifier_key?(event, ::Swt::SWT::CTRL)
end
```

```ruby
  it ':alt_/' do
    test_alt_character_press '/'
  end

  it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
    block.should_receive(:call).with(:'alt_/')
    event = double  character: '÷'.ord,
                    stateMask: ALT,
                    keyCode: '/'.ord
    subject.key_pressed(event)
  end
end

describe 'works with the ctrl key pressed such as' do
  def test_ctrl_character_press(character, modifier = 0)
    result_char = ('control_' + character).to_sym
    block.should_receive(:call).with(result_char)
    event = double character: 'something weird like \x00',
                   stateMask: CTRL | modifier,
                   keyCode:   character.downcase.ord
    subject.key_pressed(event)
  end

  it ':ctrl_a' do
    test_ctrl_character_press 'a'
  end

  it 'ctrl_z' do
    test_ctrl_character_press 'z'
  end

  describe 'and if we add the shift key' do
    it ':ctrl_A' do
      test_ctrl_character_press 'A', SHIFT
    end

    it ':ctrl_Z' do
      test_ctrl_character_press 'Z', SHIFT
    end
  end
end

describe 'works with shift combined with alt yielding capital letters' do
  def test_alt_shift_character_press(character)
    test_alt_character_press(character, SHIFT)
  end

  it ':alt_A' do
    test_alt_shift_character_press 'A'
  end

  it ':alt_Z' do
```

Identify concepts



One language

Don't **R**epeat **Y**ourself

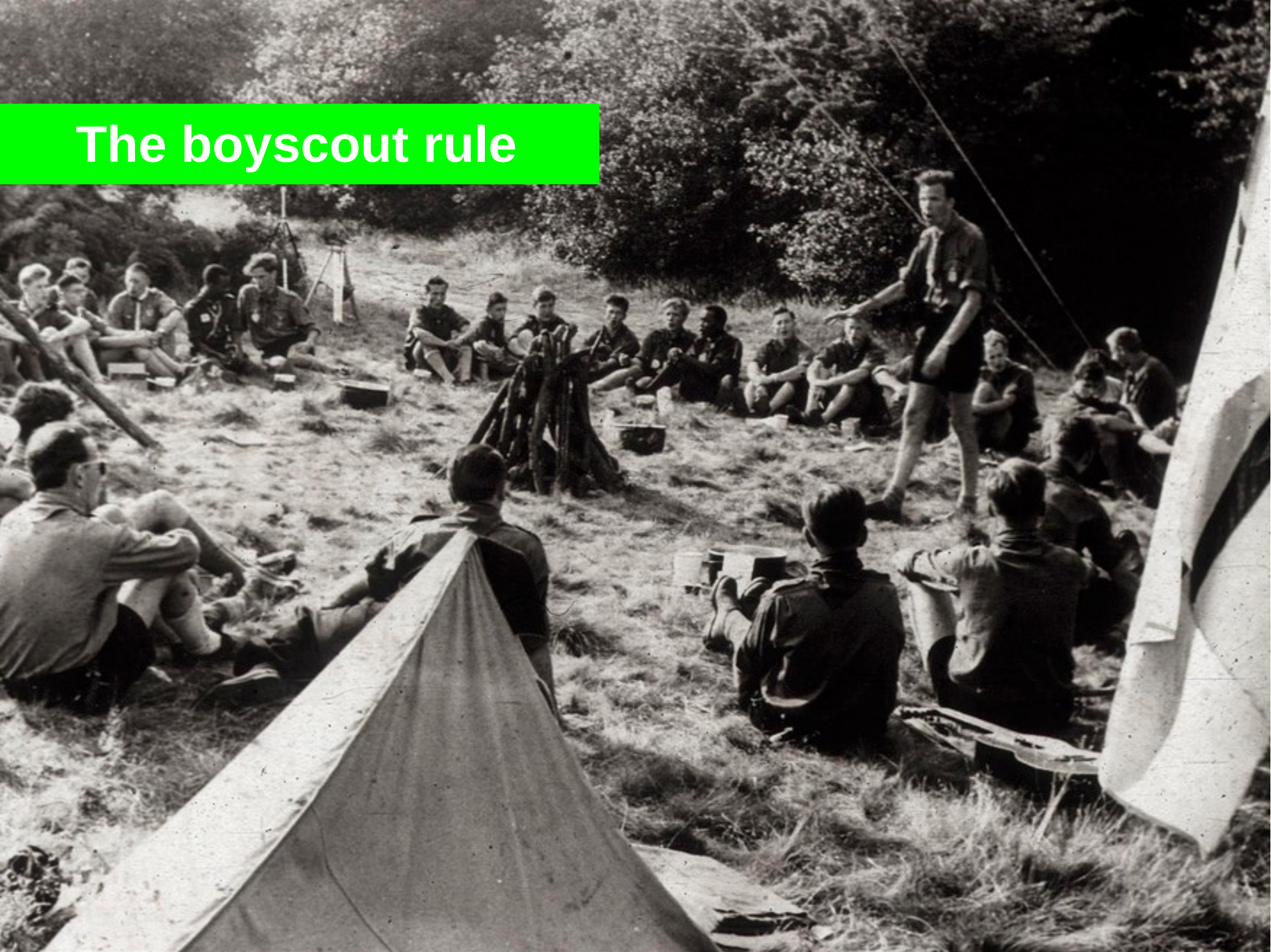

Nurturing a code base

Code bases **detoriate**


No broken windows!

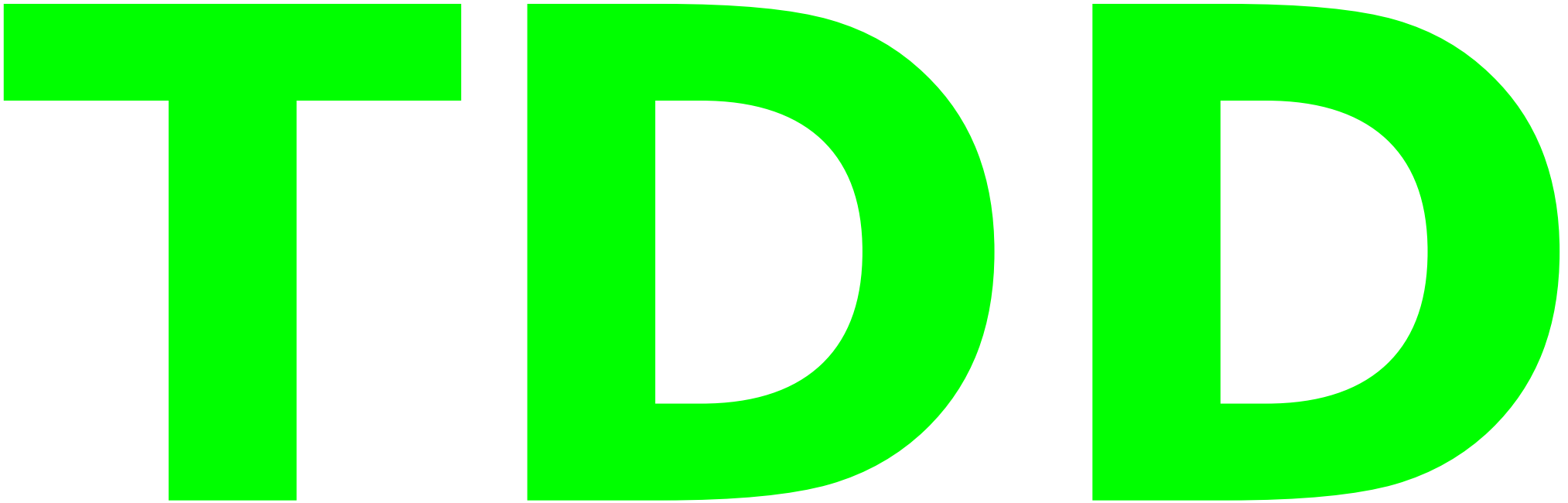

Magical time?

The boyscout rule




REFACTOR
ALL THE THINGS
quickmeme.com

Opportunistic Refactoring


TDD

80% Code Coverage

20% is never executed

jasonrclark commented on an outdated diff on Sep 29, 2014 — Hide outdated diff

spec/shoes/app_spec.rb

```
@@ -389,6 +389,20 @@
389 389         expect(subject.instance_variable_get(:@__additional_context__)).to be_r
390 390       end
391 391     end
392 +
393 +   describe '.new_dsl_method (for widget method notifications etc.)' do
394 +     before :each do
395 +       Shoes::App.new_dsl_method :widget_method do
396 +         # noop
397 +       end
398 +     end
399 +
400 +     [Shoes::App, Shoes::URL, Shoes::Widget].each do |klazz|
```

jasonrclark added a note on Sep 29, 2014 — Owner

If we add another class to `METHOD_SUBSCRIBER`, we won't be running the tests on them unless someone remembers to update this. Is it too self-referential for this to use `[Shoes::App] + METHOD_SUBSCRIBERS` as the basis for the `each`?

PragTob added a note on Sep 29, 2014 — Owner

It most likely is not - thanks for the input!

Might as well stay like this if I add the subscription (don't have the explicit list anywhere anymore).

Code Review Culture

Add a line note

PragTob added some commits on Oct 1, 2014

- Mechanism to subscribe classes/modules to DSL methods ··· f2d7fc7
- Moved DELEGATE_METHODS specs down to form a logical unit ✓ 878109a
- Open class forward references no longer needed ✓ ffae2fd

„Brown Bag“ lunches

Pair Programming

Reaping the benefits


Know when to break the rules

If you still like your code from
two years ago,
then you are not ***learning*** *fast enough.*

Enjoy writing readable code!

Tobias Pfeiffer
@PragTob
pragtob.info

Sources

- The Pragmatic Programmer
- Smalltalk Best Practice Patterns
- Clean Code
- Practical Object Oriented Design in Ruby

Photo Credit

- http://officeimg.vo.msecnd.net/en-us/images/MP900439313.jpg
- http://officeimg.vo.msecnd.net/en-us/images/MC900021328.wmf
- http://www.osnews.com/story/19266/WTFs_m
- (CC BY-SA 2.0)
  - http://www.flickr.com/photos/83633410@N07/7658272558/in/photostream/
  - http://www.flickr.com/photos/83633410@N07/7658165122/
  - https://www.flickr.com/photos/93425126@N00/313056379/
- (CC BY-NC-ND 2.0)
  - http://www.flickr.com/photos/andih/86577529/
  - http://www.flickr.com/photos/12584908@N08/3293117576/
  - http://www.flickr.com/photos/jasonlparks/4525188865/
  - http://www.flickr.com/photos/20714221@N04/2293045156/
  - https://www.flickr.com/photos/eyewash/2603717864/
  - https://www.flickr.com/photos/stevie_gill/3950697539/
  - https://www.flickr.com/photos/randar/15787696685/
- http://www.flickr.com/photos/47833351@N02/5488791911/(CC BY-ND 2.0)
- (CC BY 2.0)
  - http://www.flickr.com/photos/barry_b/76055201/
  - http://www.flickr.com/photos/25165196@N08/7725273678/
  - http://www.flickr.com/photos/29254399@N08/3187186308/
  - https://www.flickr.com/photos/garryknight/5650367750/
  - https://www.flickr.com/photos/alper/10742816123/
- (CC BY-NC-SA 2.0)
  - http://www.flickr.com/photos/dolescum/7380616658/
  - http://www.flickr.com/photos/antonkovalyov/5795281215/
  - http://www.flickr.com/photos/doug88888/2792209612/
  - https://www.flickr.com/photos/denverjeffrey/4392418334/
- (CC BY-NC 2.0)
  - http://www.flickr.com/photos/37996583811@N01/5757983532/
  - http://www.flickr.com/photos/sevendead/5650065458/
  - https://www.flickr.com/photos/whitecatsg/3146092196/