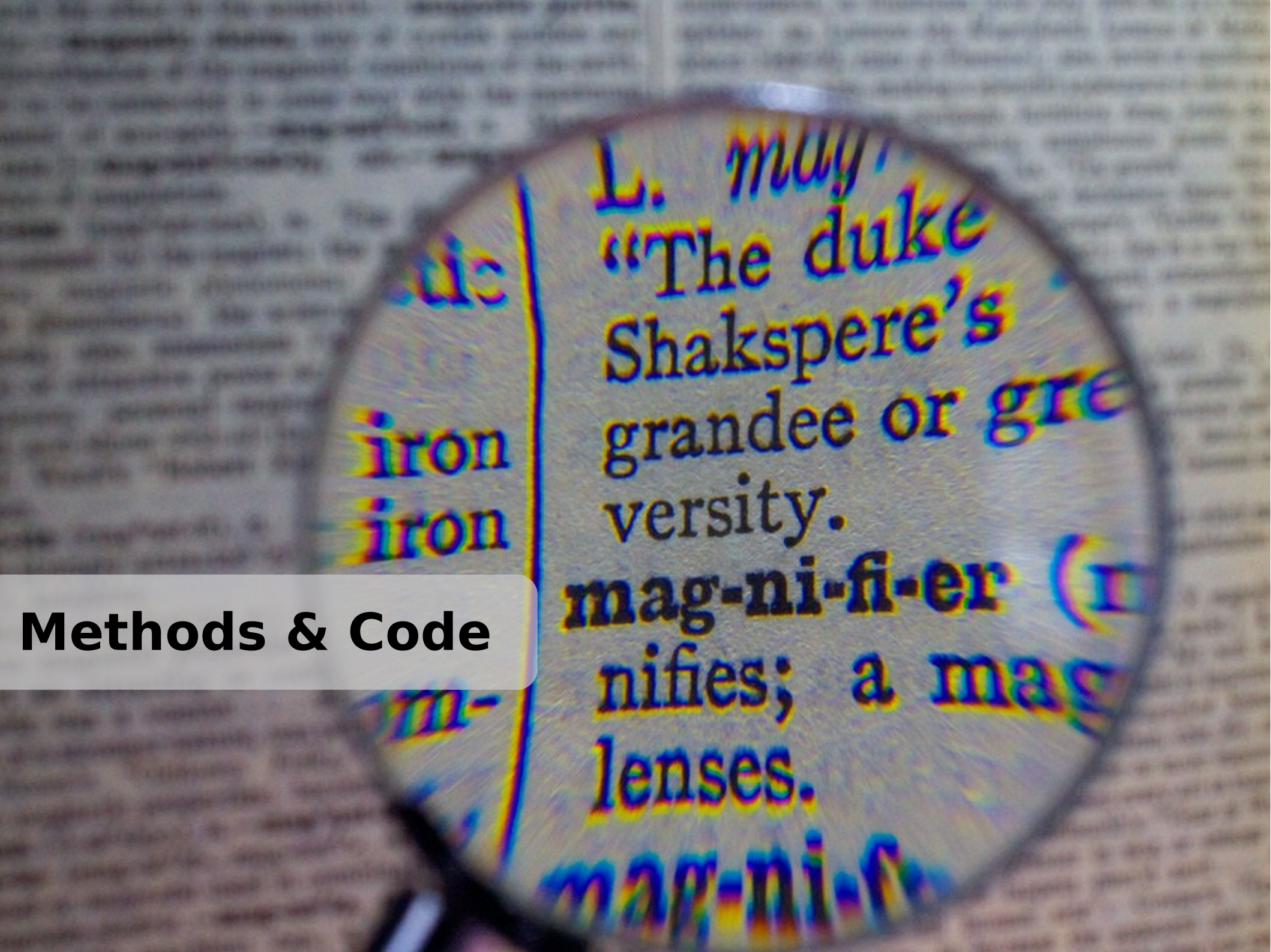

We write code


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


Not about
Architecture


Methods & Code


Extra effort


Save time

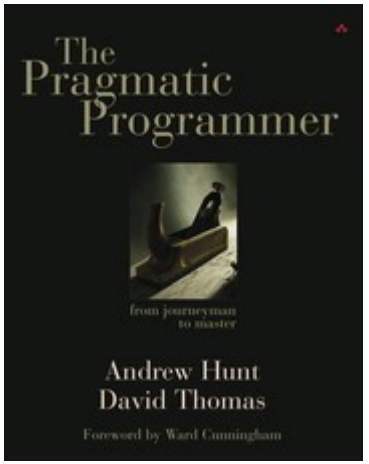

Your code base?


It's about joy!

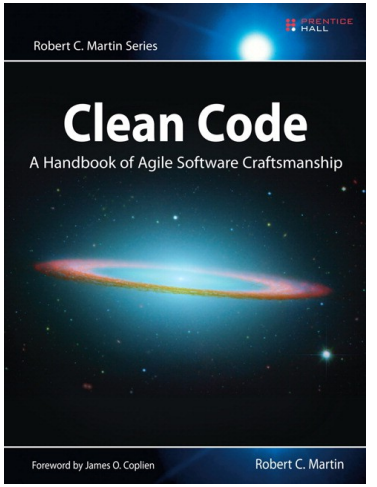
Code is read many more times than written

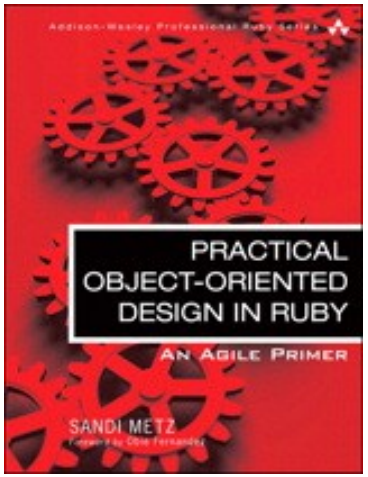
Tobias Pfeiffer
@PragTob
pragtob.info

Sources


The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham


SMALLTALK
BEST PRACTICE
PATTERNS
KENT BECK


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin


PRACTICAL
OBJECT-ORIENTED
DESIGN IN RUBY
AN AGILE PRIMER
SANDI METZ


Crazy?

Are comments a **smell**?

Outdated comments are the **worst**

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

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

```
# do one thing
…
…
…


# do another thing
…
…
…

# do something more
…
…
```

Extract Methods

```
do_one_thing
do_another_thing
do_something_more
```

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

Explanatory and meaningful names

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

Try to keep it to 2 parameters

Short Methods (<= 8 LOC)

```
# allowed to drink?
if customer.age > 18
  say 'Okay'
  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 customer.age > 18
  say 'Okay'
  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 customer.age > 18
  say 'Okay'
  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 customer.age > 18
  say 'Okay'
  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
```

Nice code formatting

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

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

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

key_event_spec.rb

builtin_methods.rb | builtin_methods_spec.rb

expert-irb-adjusted.rb | line_spec.rb | line.rb

image_spec.rb | key_listener.rb | image_pattern.rb | line.rb

```
25      def initialize(blk)
26        @block = blk
27      end
28
29      # implemented by subclasses
30      def key_pressed(event)
31      end
32
33      def key_released(event)
34      end
35
36      private
37
38      def handle_key_event(event)
39        modifiers = modifier_keys(event)
40        character = character_key(event)
41        key_string = modifiers + character
42        key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43        eval_block key_string unless character.empty?
44      end
45
46      def eval_block(key_string)
47        @block.call key_string
48      end
49
50      def modifier_keys(event)
51        modifier_keys = ''
52        modifier_keys += 'control_' if control?(event)
53        modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54        modifier_keys += 'alt_' if alt?(event)
55        modifier_keys
56      end
57
58      def alt?(event)
59        is_this_modifier_key?(event, ::Swt::SWT::ALT)
60      end
61
62      # NOTE: state_mask and key_code error for me so the java version is used
63      def is_this_modifier_key?(event, key)
64        (event.stateMask & key) == key
65      end
66
67      def shift?(event)
68        is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
69      end
70
71      def control?(event)
72        is_this_modifier_key?(event, ::Swt::SWT::CTRL)
73      end
74
```

line.rb | app.rb | logger.rb | scrolling.rb | p.rb | point.rb

move.rb | dsl.rb | key_event.rb | key_listener_spec.rb

```
70      it ':alt_/' do
71        test_alt_character_press '/'
72      end
73
74      it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
75        block.should_receive(:call).with(:'alt_/')
76        event = double  character: '÷'.ord,
77                        stateMask: ALT,
78                        keyCode: '/'.ord
79        subject.key_pressed(event)
80      end
81    end
82
83    describe 'works with the ctrl key pressed such as' do
84      def test_ctrl_character_press(character, modifier = 0)
85        result_char = ('control_' + character).to_sym
86        block.should_receive(:call).with(result_char)
87        event = double character: 'something weird like \x00',
88                       stateMask: CTRL | modifier,
89                       keyCode:   character.downcase.ord
90        subject.key_pressed(event)
91      end
92
93      it ':ctrl_a' do
94        test_ctrl_character_press 'a'
95      end
96
97      it 'ctrl_z' do
98        test_ctrl_character_press 'z'
99      end
100
101     describe 'and [illegible] the shift key' do
102       it ':ctrl_A' do
103         test_ctrl_character_press 'A', SHIFT
104       end
105
106       it ':ctrl_Z' do
107         test_ctrl_character_press 'Z', SHIFT
108       end
109     end
110   end
111
112   describe 'works with shift combined with alt yielding capital letters' do
113     def test_alt_shift_character_press(character)
114       test_alt_character_press(character, SHIFT)
115     end
116
117     it ':alt_A' do
118       test_alt_shift_character_press 'A'
119     end
120
121     it ':alt_Z' do
122       test_alt_shift_character_press 'Z'
```

80 character Width limit

94:36 LF UTF-8 Git: master

Menu | Pictures | shoes4 - [~/github/... | code_is_read_man... | Terminal | Displays | Thu Sep 5, 13:45

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

key_event_spec.rb

builtin_methods.rb | builtin_methods_spec.rb

expert-irb-adjusted.rb | line_spec.rb | line.rb

image_spec.rb | key_listener.rb | image_pattern.rb | line.rb

```ruby
    @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
```

line.rb | app.rb | logger.rb | scrolling.rb | p.rb | point.rb

move.rb | dsl.rb | key_event.rb | key_listener_spec.rb

```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 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
```

80 character Width limit

94:36 LF UTF-8 Git: master

Menu | Pictures | shoes4 - [~/github/... | code_is_read_man... | Terminal | Displays | Thu Sep 5, 13:45

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

key_event_spec.rb

builtin_methods.rb | builtin_methods_spec.rb

expert-irb-adjusted.rb | line_spec.rb | line.rb

image_spec.rb | key_listener.rb | image_pattern.rb | line.rb

```
25      def initialize(blk)
26        @block = blk
27      end
28
29      # implemented by subclasses
30      def key_pressed(event)
31      end
32
33      def key_released(event)
34      end
35
36      private
37
38      def handle_key_event(event)
39        modifiers = modifier_keys(event)
40        character = character_key(event)
41        key_string = modifiers + character
42        key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43        eval_block key_string unless character.empty?
44      end
45
46      def eval_block(key_string)
47        @block.call key_string
48      end
49
50      def modifier_keys(event)
51        modifier_keys = ''
52        modifier_keys += 'control_' if control?(event)
53        modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54        modifier_keys += 'alt_' if alt?(event)
55        modifier_keys
56      end
57
58      def alt?(event)
59        is_this_modifier_key?(event, ::Swt::SWT::ALT)
60      end
61
62      # NOTE: state_mask and key_code error for me so the java version is used
63      def is_this_modifier_key?(event, key)
64        (event.stateMask & key) == key
65      end
66
67      def shift?(event)
68        is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
69      end
70
71      def control?(event)
72        is_this_modifier_key?(event, ::Swt::SWT::CTRL)
73      end
```

line.rb | app.rb | logger.rb | scrolling.rb | p.rb | point.rb

move.rb | dsl.rb | key_event.rb | key_listener_spec.rb

```
70      it ':alt_/' do
71        test_alt_character_press '/'
72      end
73
74      it 'works with what Nacs seem to produce for opt + / (should be alt_/)' do
75        block.should_receive(:call).with(:'alt_/')
76        event = double  character: '÷'.ord,
77                        stateMask: ALT,
78                        keyCode: '/'.ord
79        subject.key_pressed(event)
80      end
81    end
82
83    describe 'works with the ctrl key pressed such as' do
84      def test_ctrl_character_press(character, modifier = 0)
85        result_char = ('control_' + character).to_sym
86        block.should_receive(:call).with(result_char)
87        event = double character: 'something weird like \x00',
88                       stateMask: CTRL | modifier,
89                       keyCode:   character.downcase.ord
90        subject.key_pressed(event)
91      end
92
93      it ':ctrl_a' do
94        test_ctrl_character_press 'a'
95      end
96
97      it 'ctrl_z' do
98        test_ctrl_character_press 'z'
99      end
100
101     describe 'and [illegible] the shift key' do
102       it ':ctrl_A' do
103         test_ctrl_character_press 'A', SHIFT
104       end
105
106       it ':ctrl_Z' do
107         test_ctrl_character_press 'Z', SHIFT
108       end
109     end
110   end
111
112   describe 'works with shift combined with alt yielding capital letters' do
113     def test_alt_shift_character_press(character)
114       test_alt_character_press(character, SHIFT)
115     end
116
117     it ':alt_A' do
118       test_alt_shift_character_press 'A'
119     end
120
121     it ':alt_Z' do
```

80 character Width limit

94:36 LF UTF-8 Git: master

Menu | Pictures | shoes4 - [~/github/... | code_is_read_man... | Terminal | Displays | Thu Sep 5, 13:45

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

key_event_spec.rb

builtin_methods.rb | builtin_methods_spec.rb

expert-irb-adjusted.rb | line_spec.rb | line.rb

image_spec.rb | key_listener.rb | image_pattern.rb | line.rb

```
25    def initialize(blk)
26      @block = blk
27    end
28
29    # implemented by subclasses
30    def key_pressed(event)
31    end
32
33    def key_released(event)
34    end
35
36    private
37
38    def handle_key_event(event)
39      modifiers = modifier_keys(event)
40      character = character_key(event)
41      key_string = modifiers + character
42      key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43      eval_block key_string unless character.empty?
44    end
45
46    def eval_block(key_string)
47      @block.call key_string
48    end
49
50    def modifier_keys(event)
51      modifier_keys = ''
52      modifier_keys += 'control_' if control?(event)
53      modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54      modifier_keys += 'alt_' if alt?(event)
55      modifier_keys
56    end
57
58    def alt?(event)
59      is_this_modifier_key?(event, ::Swt::SWT::ALT)
60    end
61
62    # NOTE: state_mask and key_code error for me so the java version is used
63    def is_this_modifier_key?(event, key)
64      (event.stateMask & key) == key
65    end
66
67    def shift?(event)
68      is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
69    end
70
71    def control?(event)
72      is_this_modifier_key?(event, ::Swt::SWT::CTRL)
73    end
```

line.rb | app.rb | logger.rb | scrolling.rb | p.rb | point.rb

move.rb | dsl.rb | key_event.rb | key_listener_spec.rb

```
70      it ':alt_/' do
71        test_alt_character_press '/'
72      end
73
74      it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
75
76        event = double  character: '÷'.ord,
77                        stateMask: ALT,
78                        keyCode: '/'.ord
79        subject.key_pressed(event)
80      end
81    end
82
83    describe 'works with the ctrl key pressed such as' do
84      def test_ctrl_character_press(character, modifier = 0)
85        result_char = ('control_' + character).to_sym
86        block.should_receive(:call).with(result_char)
87        event = double character: 'something weird like \x00',
88                       stateMask: CTRL | modifier,
89                       keyCode:   character.downcase.ord
90        subject.key_pressed(event)
91      end
92
93      it ':ctrl_a' do
94        test_ctrl_character_press 'a'
95      end
96
97      it 'ctrl_z' do
98        test_ctrl_character_press 'z'
99      end
100
101     describe 'and [illegible] the shift key' do
102       it ':ctrl_A' do
103         test_ctrl_character_press 'A', SHIFT
104       end
105
106       it ':ctrl_Z' do
107         test_ctrl_character_press 'Z', SHIFT
108       end
109     end
110   end
111
112   describe 'works with shift combined with alt yielding capital letters' do
113     def test_alt_shift_character_press(character)
114       test_alt_character_press(character, SHIFT)
115     end
116
117     it ':alt_A' do
118       test_alt_shift_character_press 'A'
119     end
120
121     it ':alt_Z' do
122       test_alt_shift_character_press 'Z'
```

80 character Width limit

94:36 LF UTF-8 Git: master

Menu | Pictures | shoes4 - [~/github/... | code_is_read_man... | Terminal | Displays | Thu Sep 5, 13:45

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

key_event_spec.rb

builtin_methods.rb | builtin_methods_spec.rb

expert-irb-adjusted.rb | line_spec.rb | line.rb

image_spec.rb | key_listener.rb | image_pattern.rb | line.rb

```
25      def initialize(blk)
26        @block = blk
27      end
28
29      # implemented by subclasses
30      def key_pressed(event)
31      end
32
33      def key_released(event)
34      end
35
36      private
37
38      def handle_key_event(event)
39        modifiers = modifier_keys(event)
40        character = character_key(event)
41        key_string = modifiers + character
42        key_string = key_string.to_sym if should_be_symbol?(event, modifiers)
43        eval_block key_string unless character.empty?
44      end
45
46      def eval_block(key_string)
47        @block.call key_string
48      end
49
50      def modifier_keys(event)
51        modifier_keys = ''
52        modifier_keys += 'control_' if control?(event)
53        modifier_keys += 'shift_' if shift?(event) && special_key?(event)
54        modifier_keys += 'alt_' if alt?(event)
55        modifier_keys
56      end
57
58      def alt?(event)
59        is_this_modifier_key?(event, ::Swt::SWT::ALT)
60      end
61
62      # NOTE: state_mask and key_code error for me so the java version is used
63      def is_this_modifier_key?(event, key)
64        (event.stateMask & key) == key
65      end
66
67      def shift?(event)
68        is_this_modifier_key?(event, ::Swt::SWT::SHIFT)
69      end
70
71      def control?(event)
72        is_this_modifier_key?(event, ::Swt::SWT::CTRL)
73      end
74
```

line.rb | app.rb | logger.rb | scrolling.rb | p.rb | point.rb

move.rb | dsl.rb | key_event.rb | key_listener_spec.rb

```
70      it ':alt_/' do
71        test_alt_character_press '/'
72      end
73
74      it 'works with what Macs seem to produce for opt + / (should be alt_/)' do
75        block.should_receive(:call).with(:'alt_/')
76        event = double  character: '÷'.ord,
77                        stateMask: ALT,
78                        keyCode: '/'.ord
79        subject.key_pressed(event)
80      end
81    end
82
83    describe 'works with the ctrl key pressed such as' do
84      def test_ctrl_character_press(character, modifier = 0)
85        result_char = ('control_' + character).to_sym
86        block.should_receive(:call).with(result_char)
87        event = double character: 'something weird like \x00',
88                       stateMask: CTRL | modifier,
89                       keyCode:   character.downcase.ord
90        subject.key_pressed(event)
91      end
92
93      it ':ctrl_a' do
94        test_ctrl_character_press 'a'
95      end
96
97      it 'ctrl_z' do
98        test_ctrl_character_press 'z'
99      end
100
101     describe 'and the shift key' do
102       it ':ctrl_A' do
103         test_ctrl_character_press 'A', SHIFT
104       end
105
106       it ':ctrl_Z' do
107         test_ctrl_character_press 'Z', SHIFT
108       end
109     end
110   end
111
112   describe 'works with shift combined with alt yielding capital letters' do
113     def test_alt_shift_character_press(character)
114       test_alt_character_press(character, SHIFT)
115     end
116
117     it ':alt_A' do
118       test_alt_shift_character_press 'A'
119     end
120
121     it ':alt_Z' do
122       test_alt_shift_character_press 'Z'
```

80 character Width limit

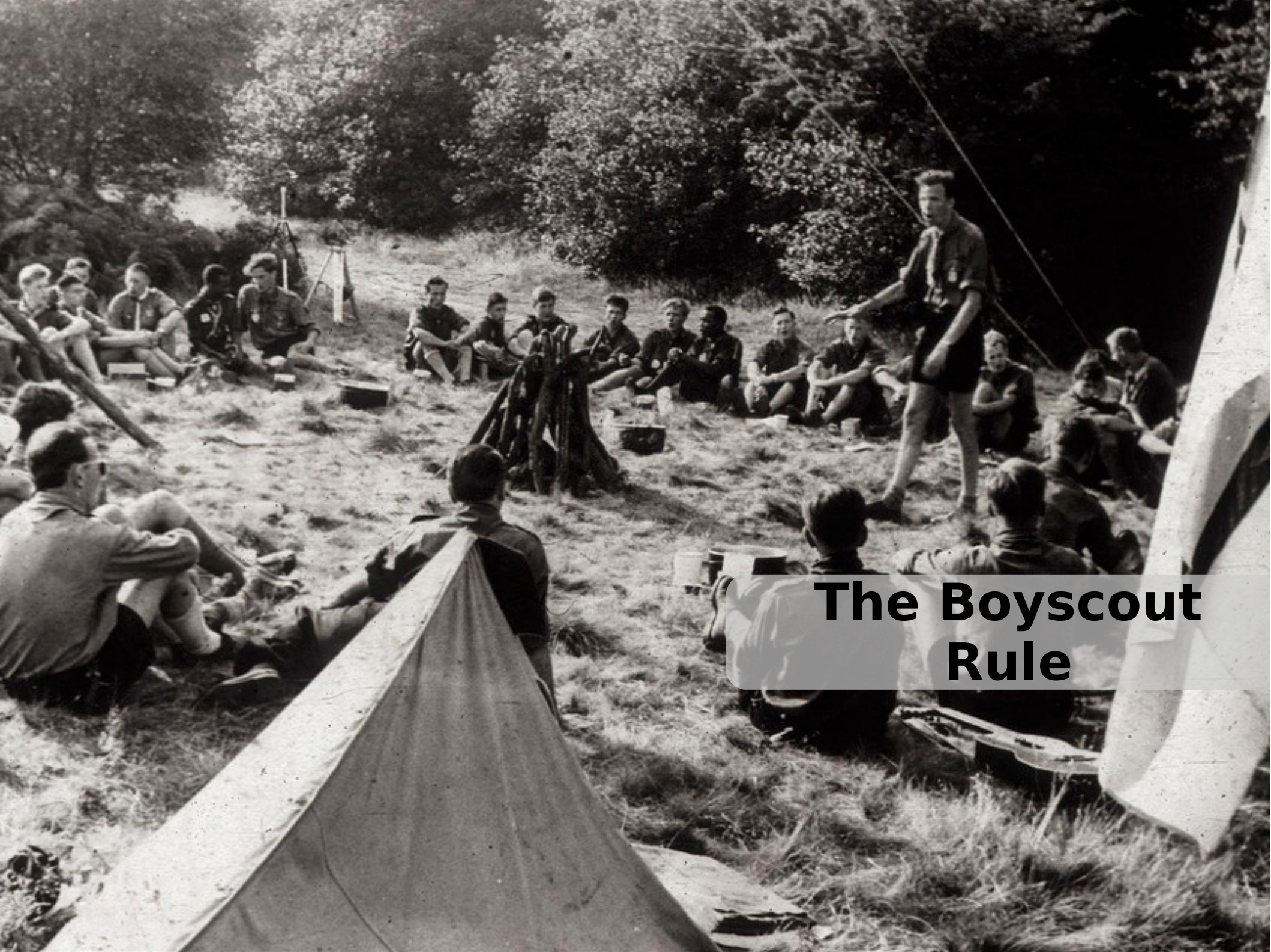
94:36 LF UTF-8 Git: master

Menu | Pictures | shoes4 - [~/github/... | code_is_read_man... | Terminal | Displays | Thu Sep 5, 13:45

Code bases detoriate

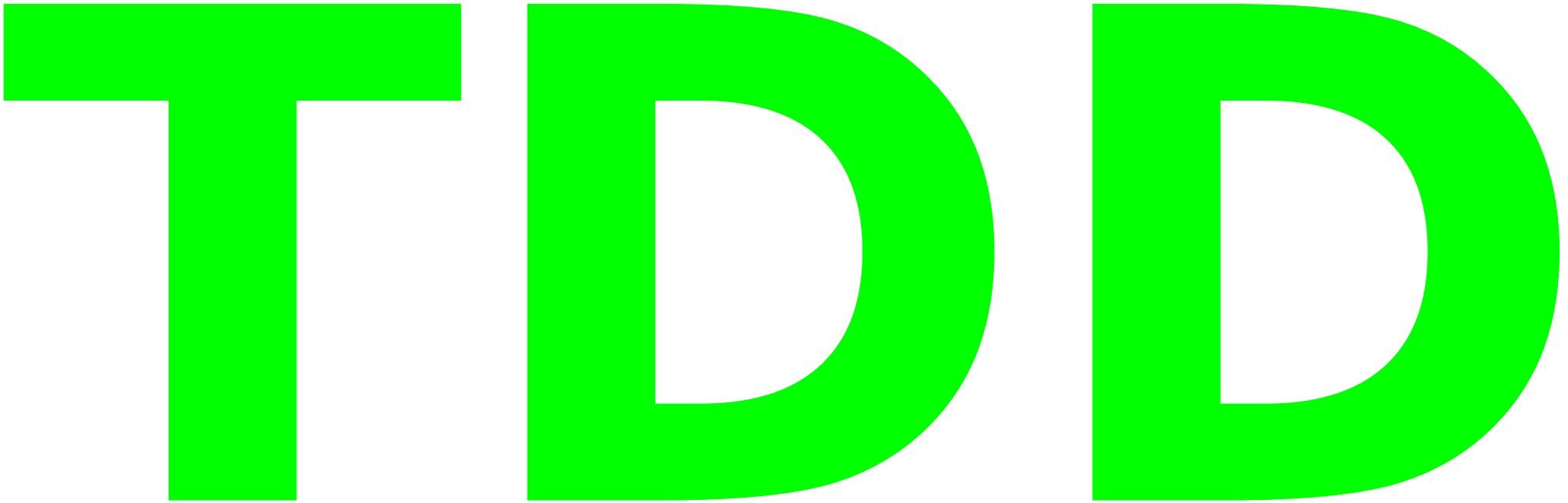

No broken windows


Magical time?



The Boyscout Rule


REFACTOR
ALL THE THINGS
quickmeme.com


TDD

Know when to break the rules

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

Enjoy Coding!

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
- (CC BY-SA 2.0)
  - http://www.flickr.com/photos/83633410@N07/7658272558/in/photostream/
  - http://www.flickr.com/photos/83633410@N07/7658165122/
  - http://en.wikipedia.org/wiki/File:Kent_Beck_no_Workshop_Mapping_XP.jpg
- (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/
- 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/
- (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/
- (CC BY-NC 2.0)
  - http://www.flickr.com/photos/37996583811@N01/5757983532/
  - http://www.flickr.com/photos/sevendead/5650065458/

Source pictures

- http://imagery.pragprog.com/products/59/tpp.jpg
- http://www.informit.com/ShowCover.aspx?isbn=9780132852111&type=f
- http://coding-in.net/blog/wp-content/uploads/clean_code.jpg
- http://www.informit.com/ShowCover.aspx?isbn=9780321721334&type=f
- http://mendicantuniversity.org/images/logo.png