



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

L. May
"The duke
Shakspeare's
grandee or gre
versity.
mag-ni-fi-er (T
nifies; a mag
lenses.
mag-ni-f



Extra effort



Save time



Your code base?



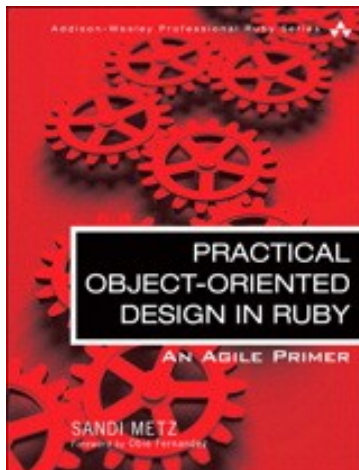
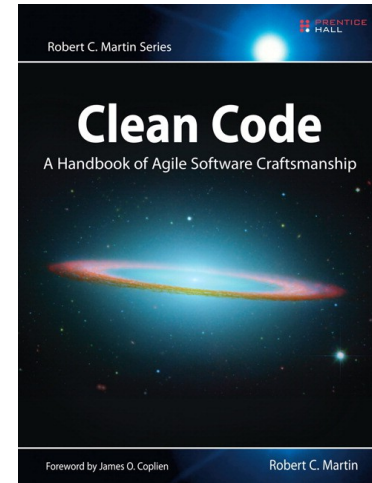
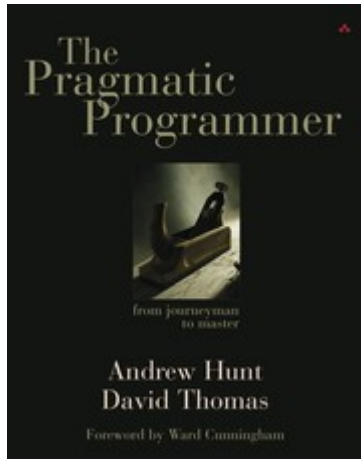


It's about joy!

Code is read many more times
than written

Tobias Pfeiffer
@PragTob
pragtob.wordpress.com

Sources





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Outdated comments are the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

Argument order dependency

No **magic** numbers

Explanatory variable

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
```

```
# 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
```

Query method

Intention revealing method

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

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

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

```
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
```

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

Tom Stuart (Berlin)

Same level of abstraction in a
method

```
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
```

prepare_drink requested_drink
prepare_check requested_drink

DRY

Nice code formatting

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

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



```

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 x app.rb x logger.rb x scrolling.rb x p.rb x point.rb x
move.rb x dsl.rb x key_event.rb x key_listener_spec.rb x

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 end
93
94 it ':ctrl_a' do
95   test_ctrl_character_press 'a'
96 end
97
98 it 'ctrl_z' do
99   test_ctrl_character_press 'z'
100 end
101
102 describe 'and it works with the shift key' do
103   it ':ctrl_A' do
104     test_ctrl_character_press 'A', SHIFT
105   end
106
107   it ':ctrl_Z' do
108     test_ctrl_character_press 'Z', SHIFT
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 end
117
118 it ':alt_A' do
119   test_alt_shift_character_press 'A'
120 end
121
122 it ':alt_Z' do
123   test_alt_shift_character_press 'Z'

```

80 character
Width limit



```
key_event_spec.rb
builtin_methods.rb
builtin_methods_spec.rb
expert-irb-adjusted.rb
image_spec.rb
key_listener.rb
image_pattern.rb
line_spec.rb
line.rb
```

```

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: '/'
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 end
93
94 it ':ctrl_a' do
95   test_ctrl_character_press 'a'
96 end
97
98 it 'ctrl_z' do
99   test_ctrl_character_press 'z'
100 end
101
102 describe 'and it works with the shift key' do
103   it ':ctrl_A' do
104     test_ctrl_character_press 'A', SHIFT
105   end
106
107   it ':ctrl_Z' do
108     test_ctrl_character_press 'Z', SHIFT
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 end
117
118 it ':alt_A' do
119   test_alt_shift_character_press 'A'
120 end
121
122 it ':alt_Z' do
123   test_alt_shift_character_press 'Z'

```

80 character
Width limit



```

key_event_spec.rb
  builtin_methods.rb
  builtin_methods_spec.rb
  expert-irb-adjusted.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 it works with 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



```

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 x app.rb x logger.rb x scrolling.rb x p.rb x point.rb x
move.rb x dsl.rb x key_event.rb x key_listener_spec.rb x

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 end
93
94 it ':ctrl_a' do
95   test_ctrl_character_press 'a'
96 end
97
98 it 'ctrl_z' do
99   test_ctrl_character_press 'z'
100 end
101
102 describe 'and it works with the shift key' do
103   it ':ctrl_A' do
104     test_ctrl_character_press 'A', SHIFT
105   end
106
107   it ':ctrl_Z' do
108     test_ctrl_character_press 'Z', SHIFT
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 end
117
118 it ':alt_A' do
119   test_alt_shift_character_press 'A'
120 end
121
122 it ':alt_Z' do
123   test_alt_shift_character_press 'Z'

```

80 character
Width limit

```

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

```

```

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

```

**80 character
Width limit**

Don't program by **coincidence**

Code bases deteriorate



No broken windows







Magical time?



The Boy Scout Rule

REFACTOR



quickmeme.com

TDD

80% Code Coverage

20% is never executed

*„**Incoming** messages should be **tested** for the state they return. **Outgoing command messages** should be tested to ensure they get sent. **Outgoing query messages** should **not be tested**.“*

Sandi Metz



**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.wordpress.com

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/\(CC BY-ND 2.0\)](http://www.flickr.com/photos/47833351@N02/5488791911/(CC%20BY-ND%202.0))
- (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>