PREFERRING OBJECT ORIENTATION TO METAPROGRAMMING

Steven G. Harms / **@sgharms**

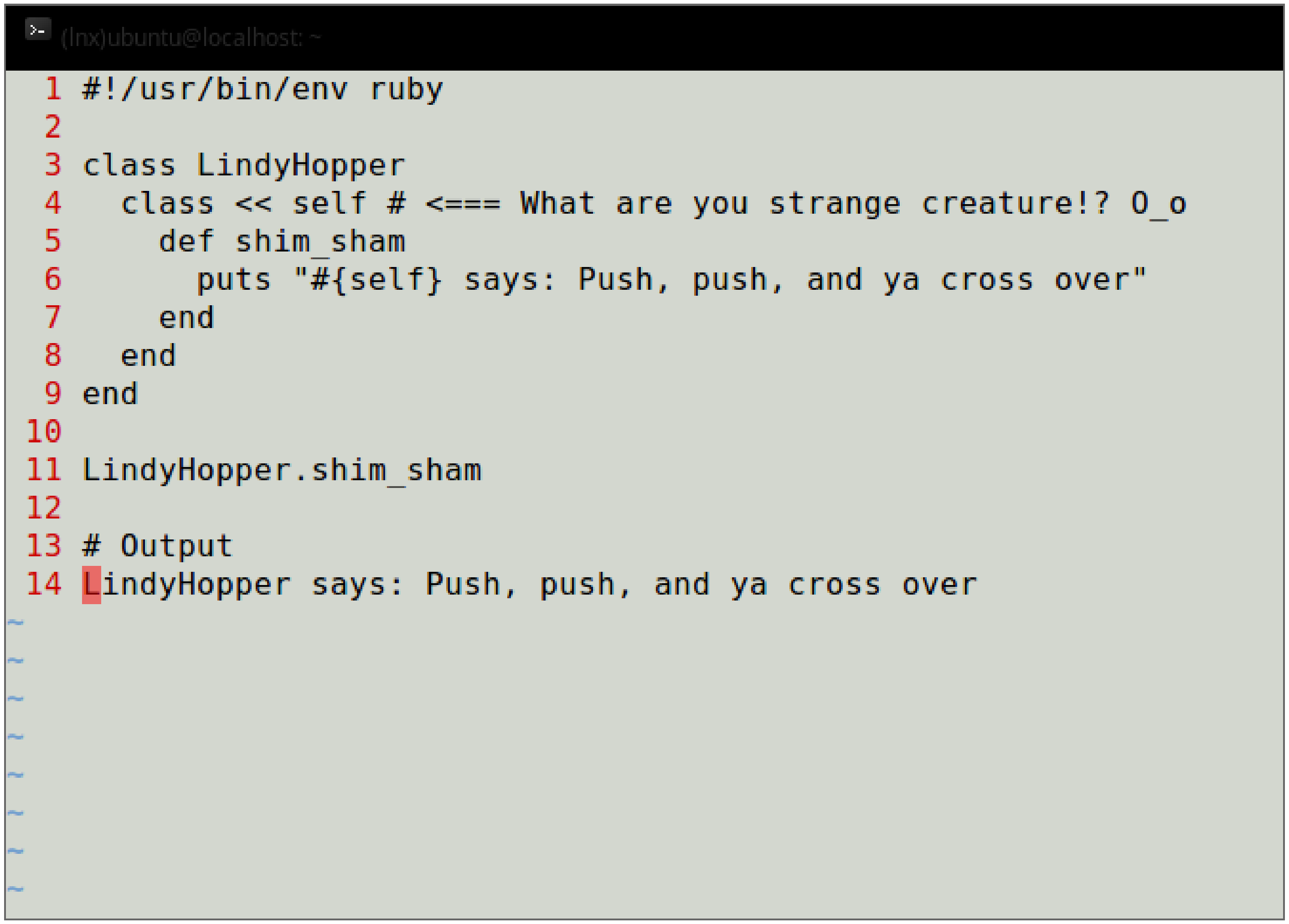
ABOUT ME

```
{
  "name": "Steven Harms",
  "email": "rubyconfXIII@sgharms.oib.com",
  "provenance": "San Francisco, CA",
  "chirp": "@sgharms"
}
```

PREFERRING OBJECT ORIENTATION TO METAPROGRAMMING

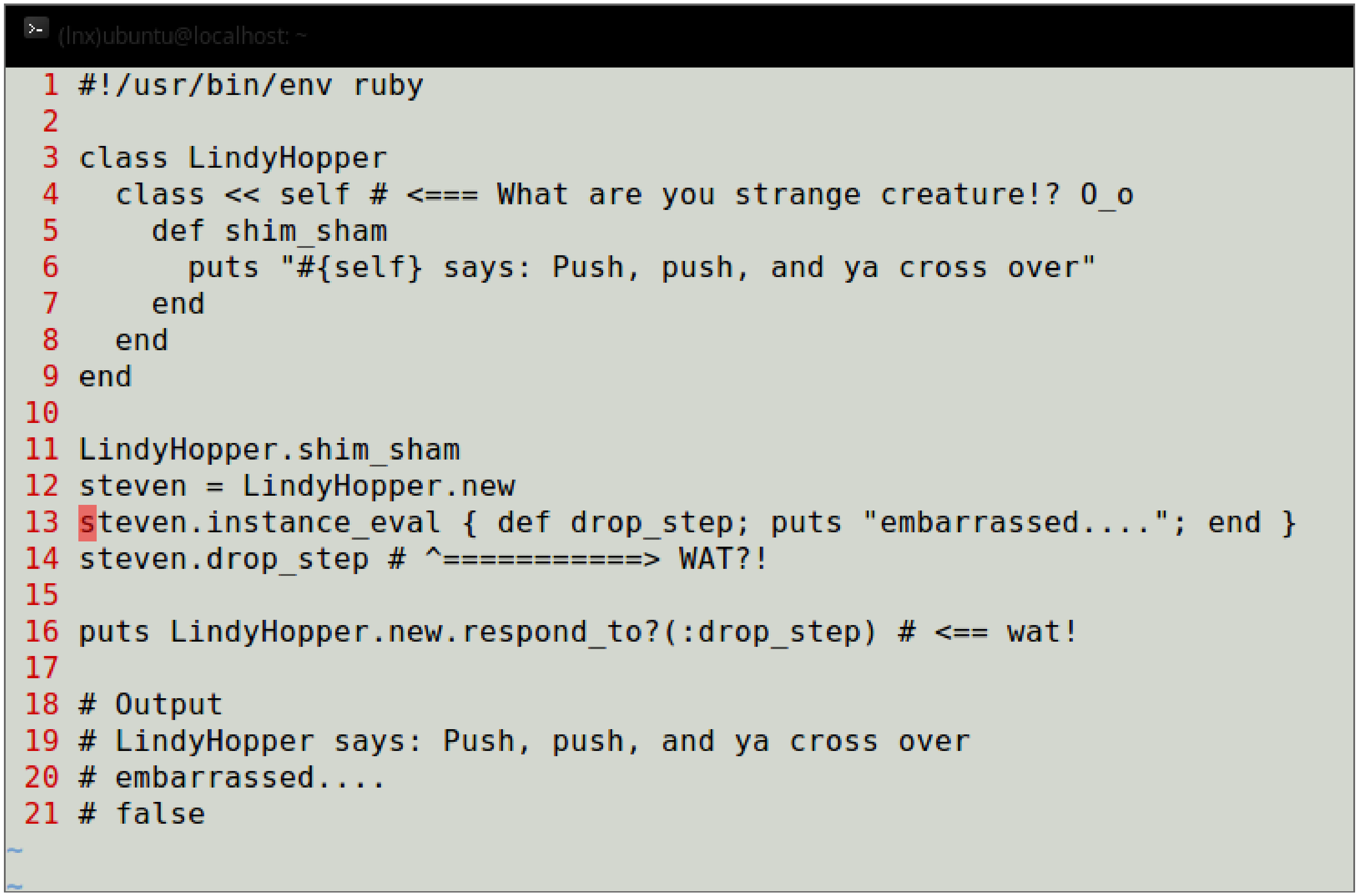
Steven G. Harms / **@sgharms**

I LOVE METAPROGRAMMING!


(lnx)ubuntu@localhost: ~

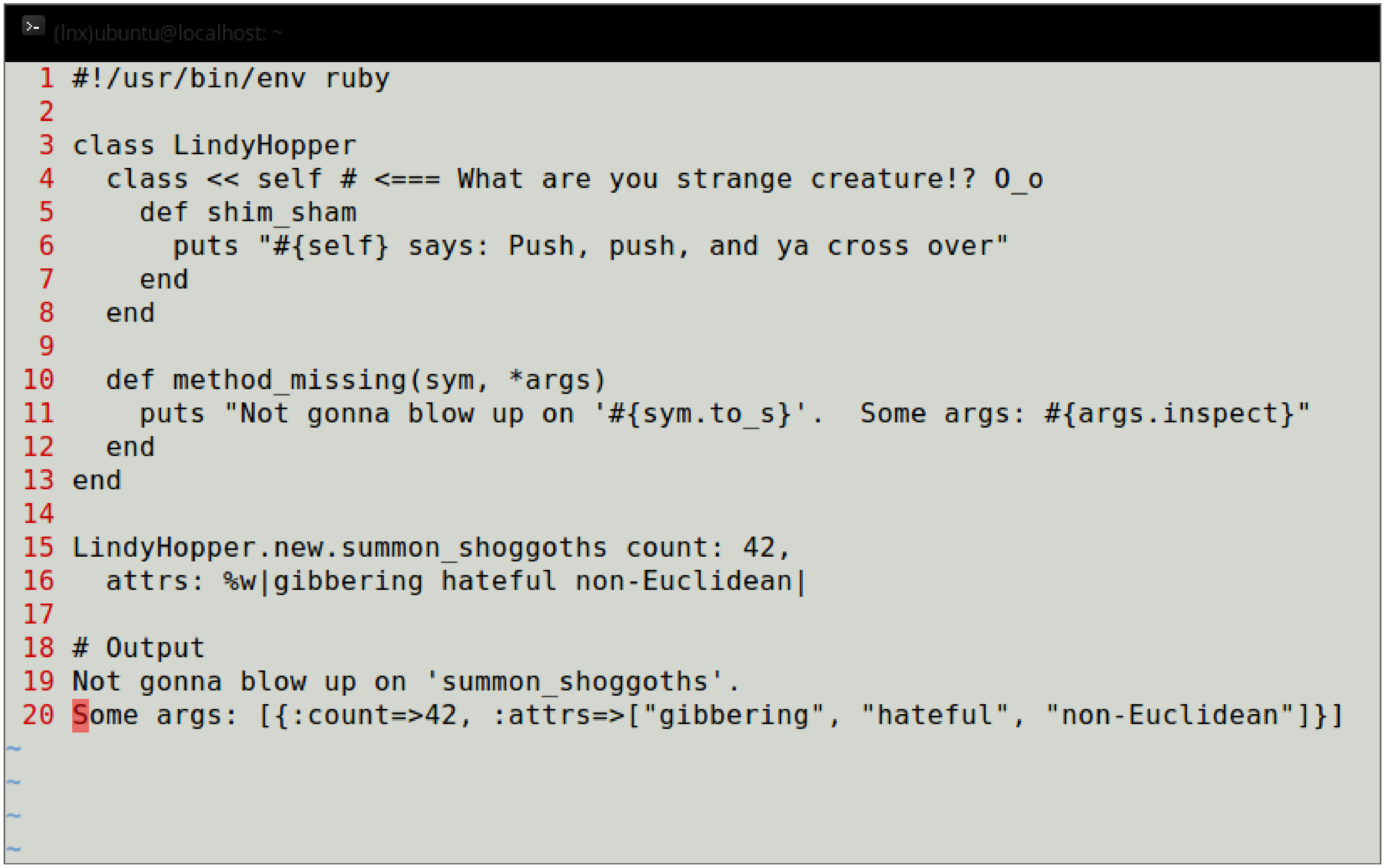
```
#!/usr/bin/env ruby

class LindyHopper
  class << self # <=== What are you strange creature!? O_o
    def shim_sham
      puts "#{self} says: Push, push, and ya cross over"
    end
  end
end

LindyHopper.shim_sham

# Output
LindyHopper says: Push, push, and ya cross over
```

AND THEN I SAW....



```
#!/usr/bin/env ruby

class LindyHopper
  class << self # <=== What are you strange creature!? O_o
    def shim_sham
      puts "#{self} says: Push, push, and ya cross over"
    end
  end
end

LindyHopper.shim_sham
steven = LindyHopper.new
steven.instance_eval { def drop_step; puts "embarrassed...."; end }
steven.drop_step # ^===========> WAT?!

puts LindyHopper.new.respond_to?(:drop_step) # <== wat!

# Output
# LindyHopper says: Push, push, and ya cross over
# embarrassed....
# false
```

AND **THEN** I SAW....



```
#!/usr/bin/env ruby

class LindyHopper
  class << self # <=== What are you strange creature!? O_o
    def shim_sham
      puts "#{self} says: Push, push, and ya cross over"
    end
  end

  def method_missing(sym, *args)
    puts "Not gonna blow up on '#{sym.to_s}'.  Some args: #{args.inspect}"
  end
end

LindyHopper.new.summon_shoggoths count: 42,
  attrs: %w|gibbering hateful non-Euclidean|

# Output
Not gonna blow up on 'summon_shoggoths'.
Some args: [{:count=>42, :attrs=>["gibbering", "hateful", "non-Euclidean"]}]
~
~
~
~
```

TIME PASSED…

THREE QUESTIONS

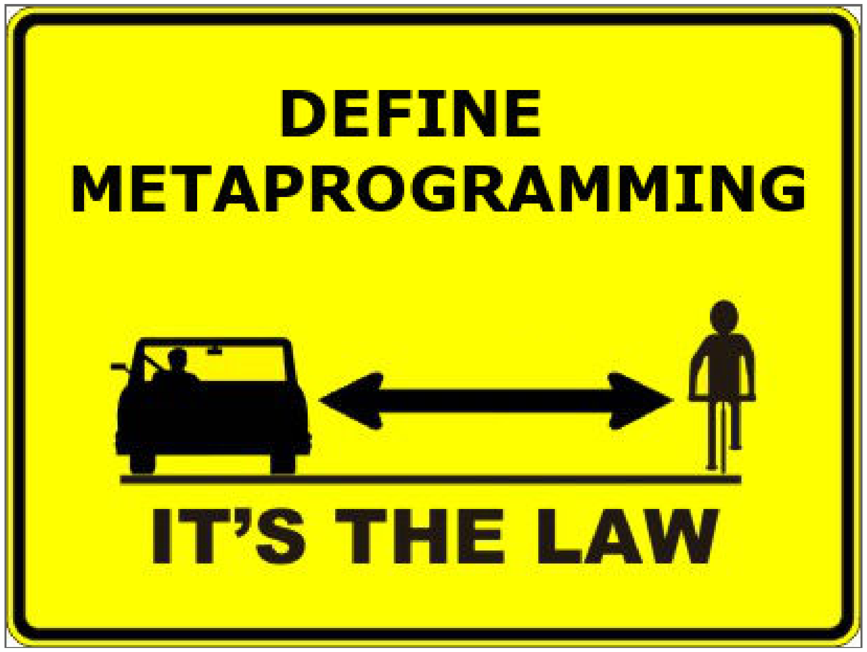
"WHEN SHOULD I USE METAPROGRAMMING?"

"WHEN SHOULD I PREFER SOMETHING ELSE?"

"IF SOMETHING ELSE, WHAT SHOULD IT BE?"

OBLIGATORY "DEFINE METAPROGRAMMING" SLIDE

To talk about metaprogramming you must first spend too much time defining it



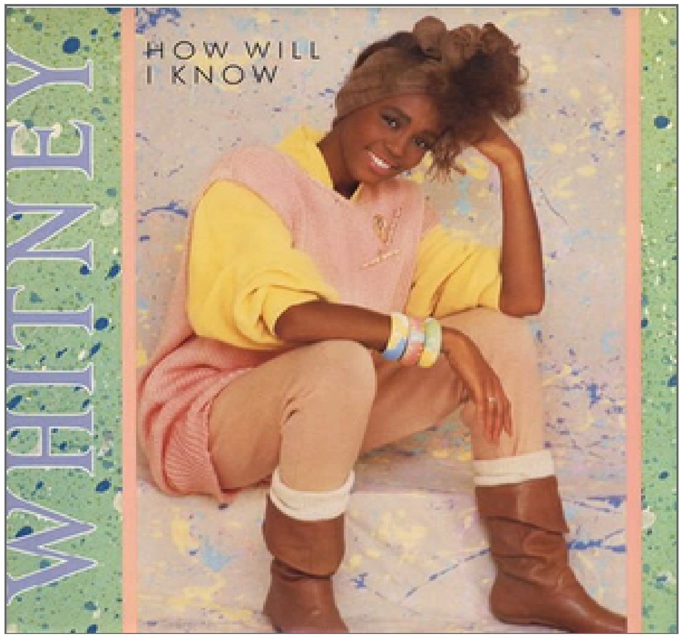
Writing code that:

1. redirects passed messages at runtime
2. provides or alters the structures that do said passing at runtime

At Runtime

1. redirects passed messages *at runtime*
2. provides or alters the structures that do said passing *at runtime*

HOW WILL I KNOW (CIRCA 2011)



"WHEN SHOULD I USE METAPROGRAMMING?"

1. Never
2. (Practically) Never

"WHEN SHOULD I USE SOMETHING ELSE?"

1. Always
2. (Practically) Always

UNCLEAR GUIDANCE (2013 EDITION)

TIM CONNOR

```
The first rule of ruby metaprogramming is
that you don't talk about Ruby
metaprogramming (because you'll lead
people new to it to over use it). ;)
```

TIM CONNOR

The second rule is "don't do it", and that might be part of why all the materials on it don't give the OP a good next step, because the next step should be "don't". If you can possibly get away with it, then use something simpler and clearer. Unfortunately, you cannot simple be told not to MP, you must see it yourself. ;)

THE ULTIMATE "NO" TO METAPROGRAMMING

NO TO THE TERM METAPROGRAMMING *PER SE*

1. Hard to define
2. Greek envy

NO TO CODING WITH METAPROGRAMMING

1. It has a bad conceptual framework
2. The Ruby code *itself* doesn't want you to
3. Normative social standards

DEFINITION RECALLED

Code that:

1. redirects passed messages at runtime
2. provides or alters the structures that do said passing at runtime

DEFINITION ANNOTATED

Code that:

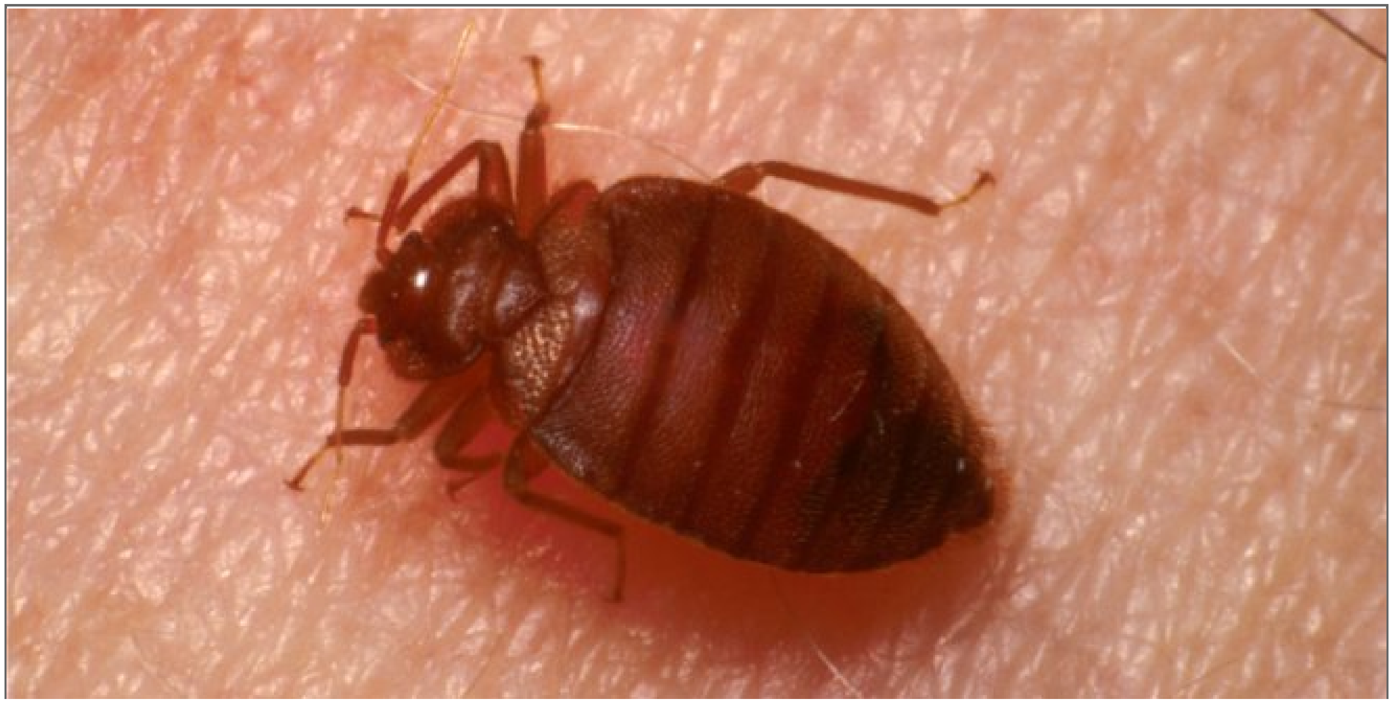
1. **redirects** passed messages **at runtime**
2. provides or **alters** the structures that do said passing **at runtime**

METAPROGRAMMING IS...

Willfully writing code that does something:

1. *contrary* to expectations communicated by the code's contextual conventions established at parse-time
2. at runtime

BUG

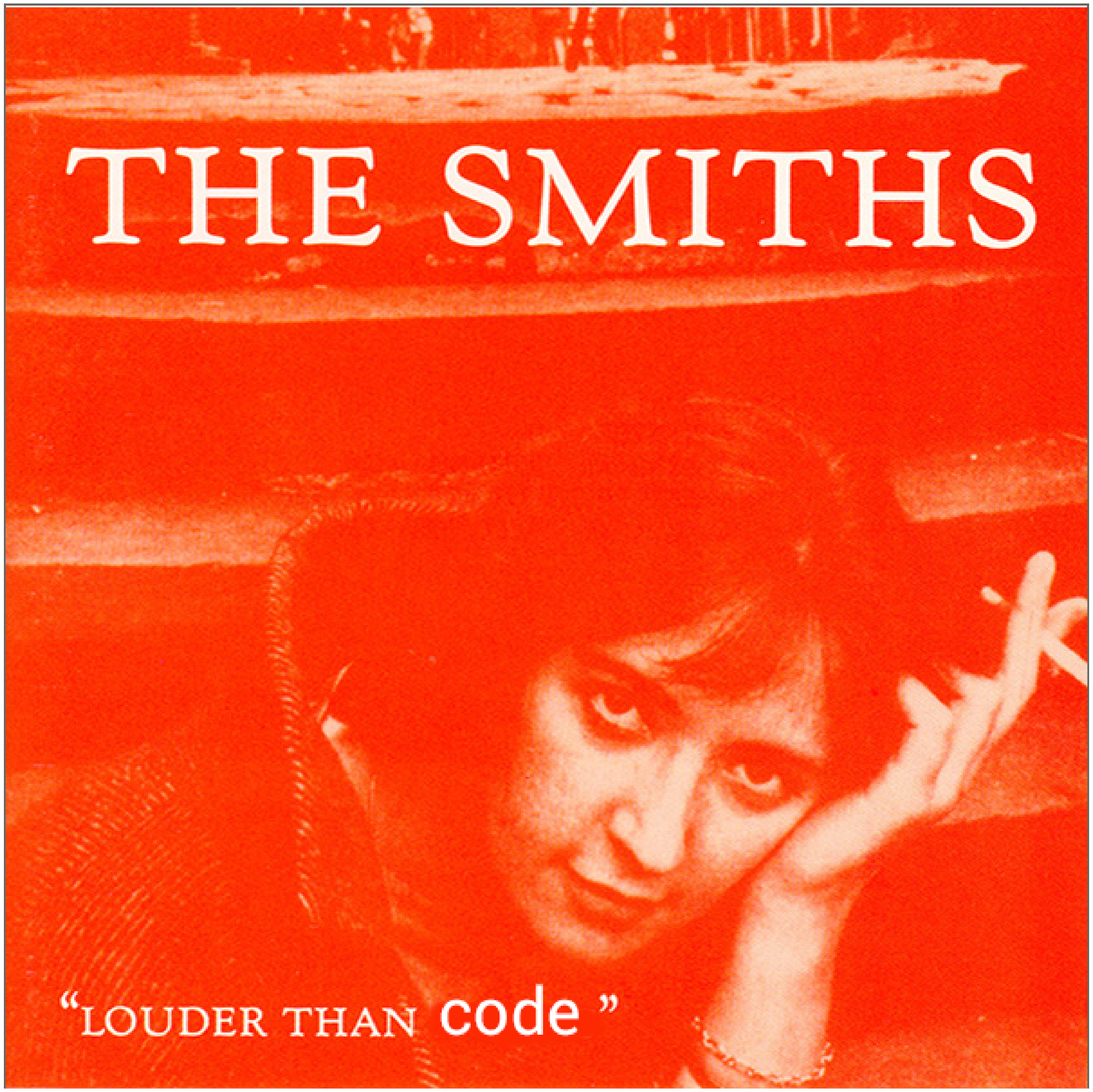


METAPROGRAMMING IS THEREFORE...

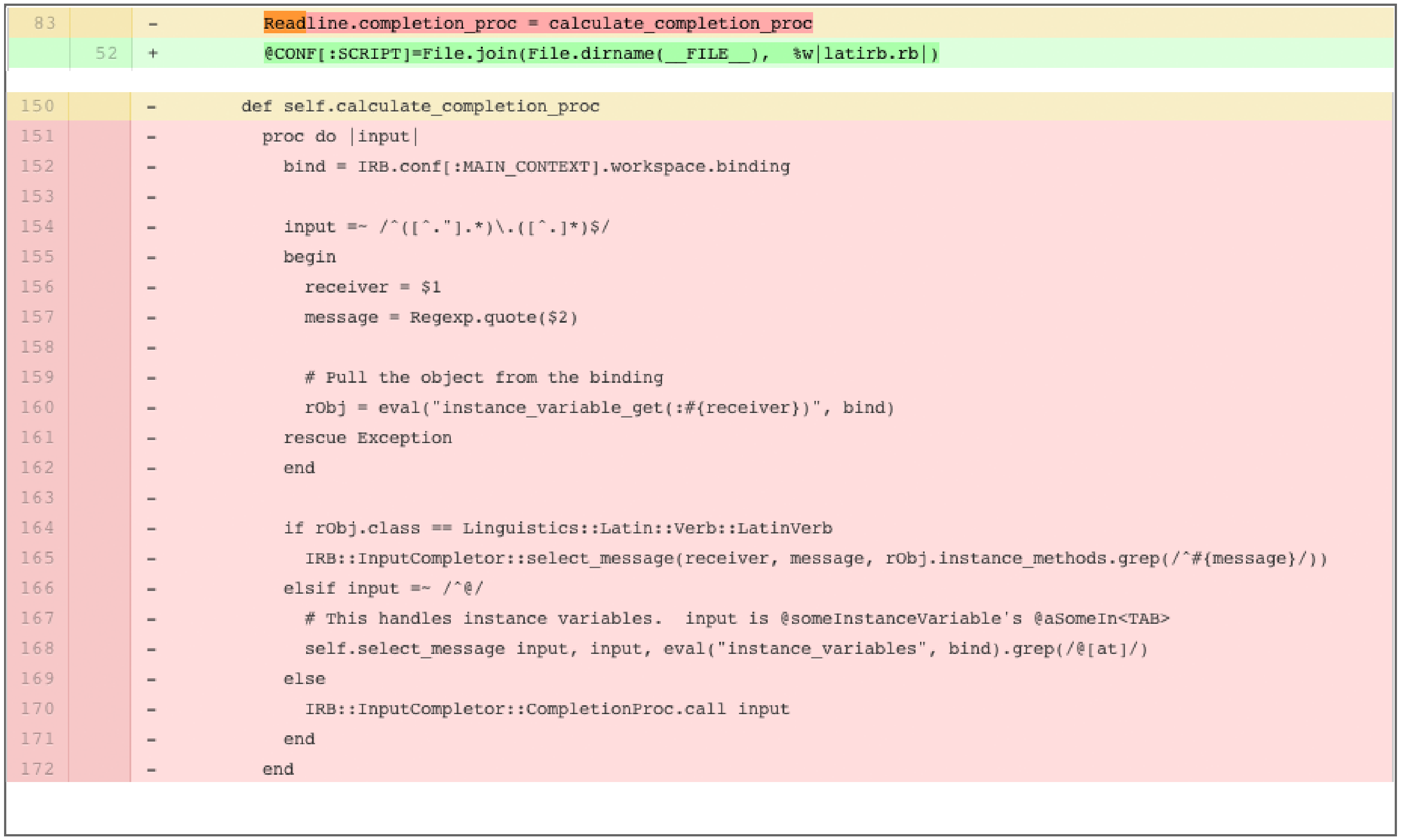
Writing bugs that have adventitious, beneficial side-effects

> *Adventitous (adj): coming from another source and not inherent or innate*

A series of aberrations that *just happen* to produce the right output

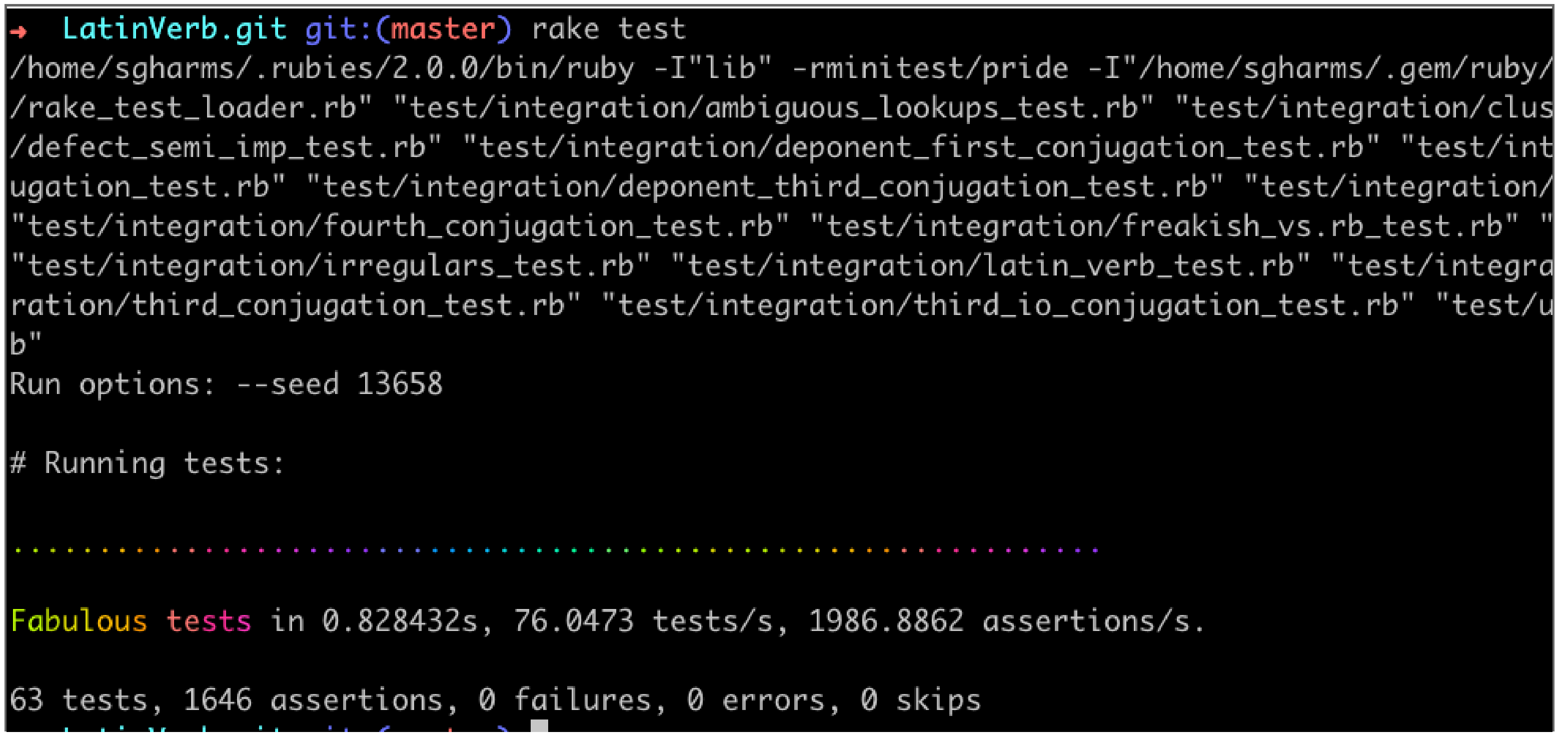

THE SMITHS
"LOUDER THAN code"

BROKEN TAB-COMPLETION IN IRB



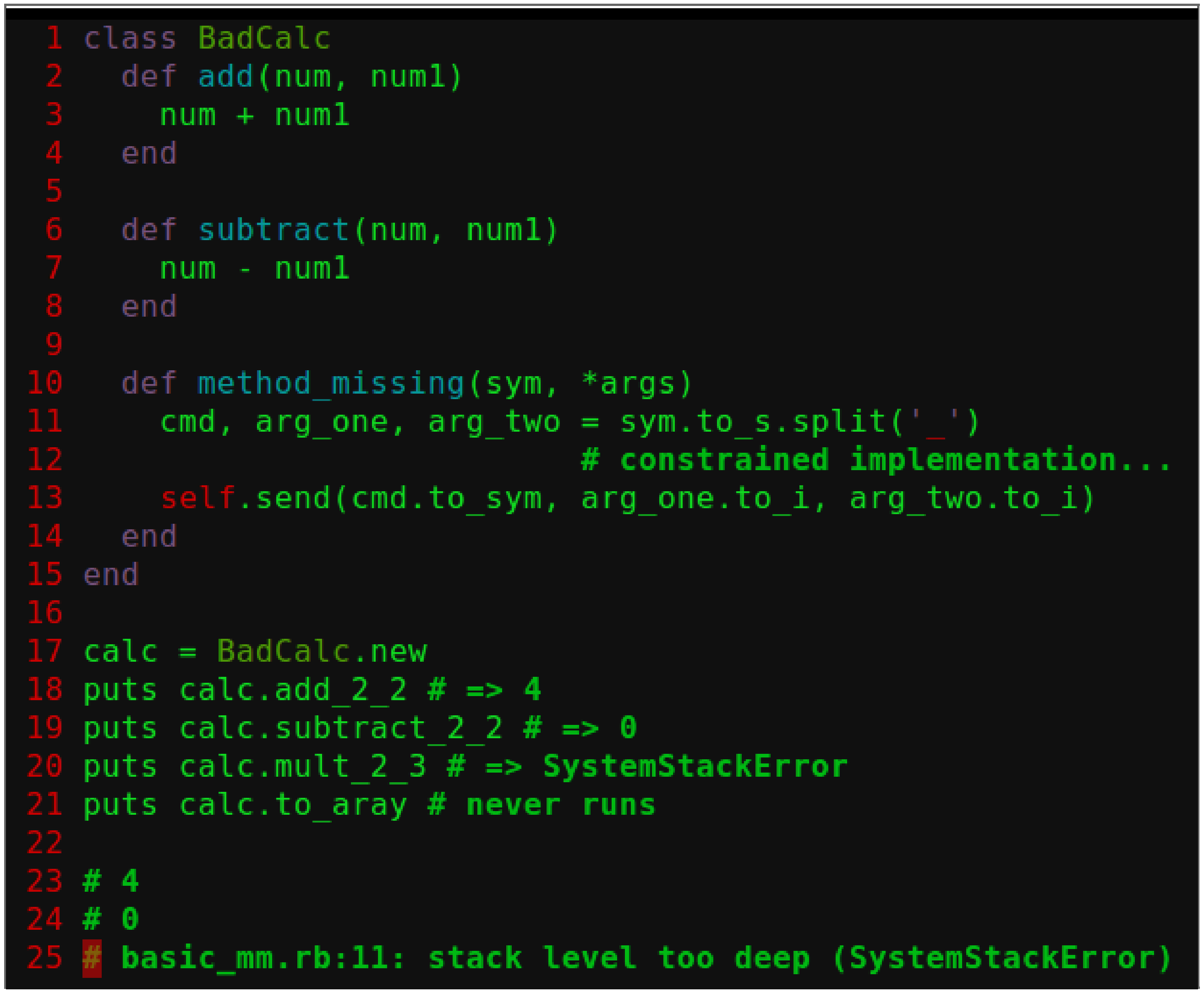
```
-         Readline.completion_proc = calculate_completion_proc
+         @CONF[:SCRIPT]=File.join(File.dirname(__FILE__),  %w|latirb.rb|)

-       def self.calculate_completion_proc
-         proc do |input|
-           bind = IRB.conf[:MAIN_CONTEXT].workspace.binding
-
-           input =~ /^([^."].*)\.([^.]*)$/
-           begin
-             receiver = $1
-             message = Regexp.quote($2)
-
-             # Pull the object from the binding
-             rObj = eval("instance_variable_get(:#{receiver})", bind)
-           rescue Exception
-           end
-
-           if rObj.class == Linguistics::Latin::Verb::LatinVerb
-             IRB::InputCompletor::select_message(receiver, message, rObj.instance_methods.grep(/^#{message}/))
-           elsif input =~ /^@/
-             # This handles instance variables.  input is @someInstanceVariable's @aSomeIn<TAB>
-             self.select_message input, input, eval("instance_variables", bind).grep(/@[at]/)
-           else
-             IRB::InputCompletor::CompletionProc.call input
-           end
-         end
```

MANY, MANY TESTS



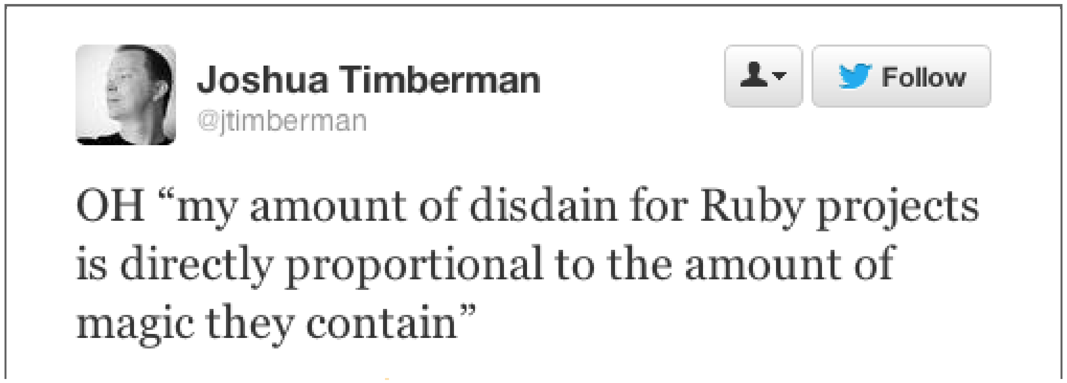
```
➜  LatinVerb.git git:(master) rake test
/home/sgharms/.rubies/2.0.0/bin/ruby -I"lib" -rminitest/pride -I"/home/sgharms/.gem/ruby/
/rake_test_loader.rb" "test/integration/ambiguous_lookups_test.rb" "test/integration/clus
/defect_semi_imp_test.rb" "test/integration/deponent_first_conjugation_test.rb" "test/int
ugation_test.rb" "test/integration/deponent_third_conjugation_test.rb" "test/integration/
"test/integration/fourth_conjugation_test.rb" "test/integration/freakish_vs.rb_test.rb" "
"test/integration/irregulars_test.rb" "test/integration/latin_verb_test.rb" "test/integra
ration/third_conjugation_test.rb" "test/integration/third_io_conjugation_test.rb" "test/u
b"
Run options: --seed 13658

# Running tests:

...............................................................

Fabulous tests in 0.828432s, 76.0473 tests/s, 1986.8862 assertions/s.

63 tests, 1646 assertions, 0 failures, 0 errors, 0 skips
```

HORRIBLE DEBUG-ABILITY



```ruby
class BadCalc
  def add(num, num1)
    num + num1
  end

  def subtract(num, num1)
    num - num1
  end

  def method_missing(sym, *args)
    cmd, arg_one, arg_two = sym.to_s.split('_')
                          # constrained implementation...
    self.send(cmd.to_sym, arg_one.to_i, arg_two.to_i)
  end
end

calc = BadCalc.new
puts calc.add_2_2 # => 4
puts calc.subtract_2_2 # => 0
puts calc.mult_2_3 # => SystemStackError
puts calc.to_aray # never runs

# 4
# 0
# basic_mm.rb:11: stack level too deep (SystemStackError)
```

LISTEN TO TWITTER



Joshua Timberman
@jtimberman

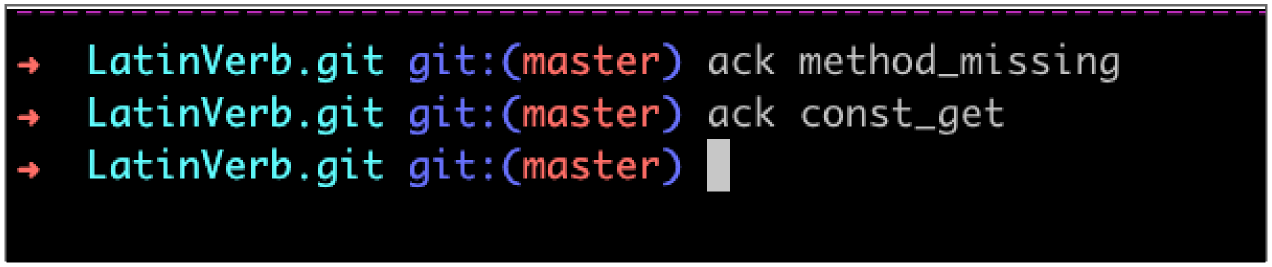
OH “my amount of disdain for Ruby projects is directly proportional to the amount of magic they contain”

THE END OF QUOTE-METAPROGRAMMING

- Term is bad
- Style is bad

Yes, this is horrible, this idea

PART II: HOW TO REPLACE QUOTE-METAPROGRAMMING



```
➜  LatinVerb.git git:(master) ack method_missing
➜  LatinVerb.git git:(master) ack const_get
➜  LatinVerb.git git:(master) 
```

ANTI-METAPROGRAMMING CODE

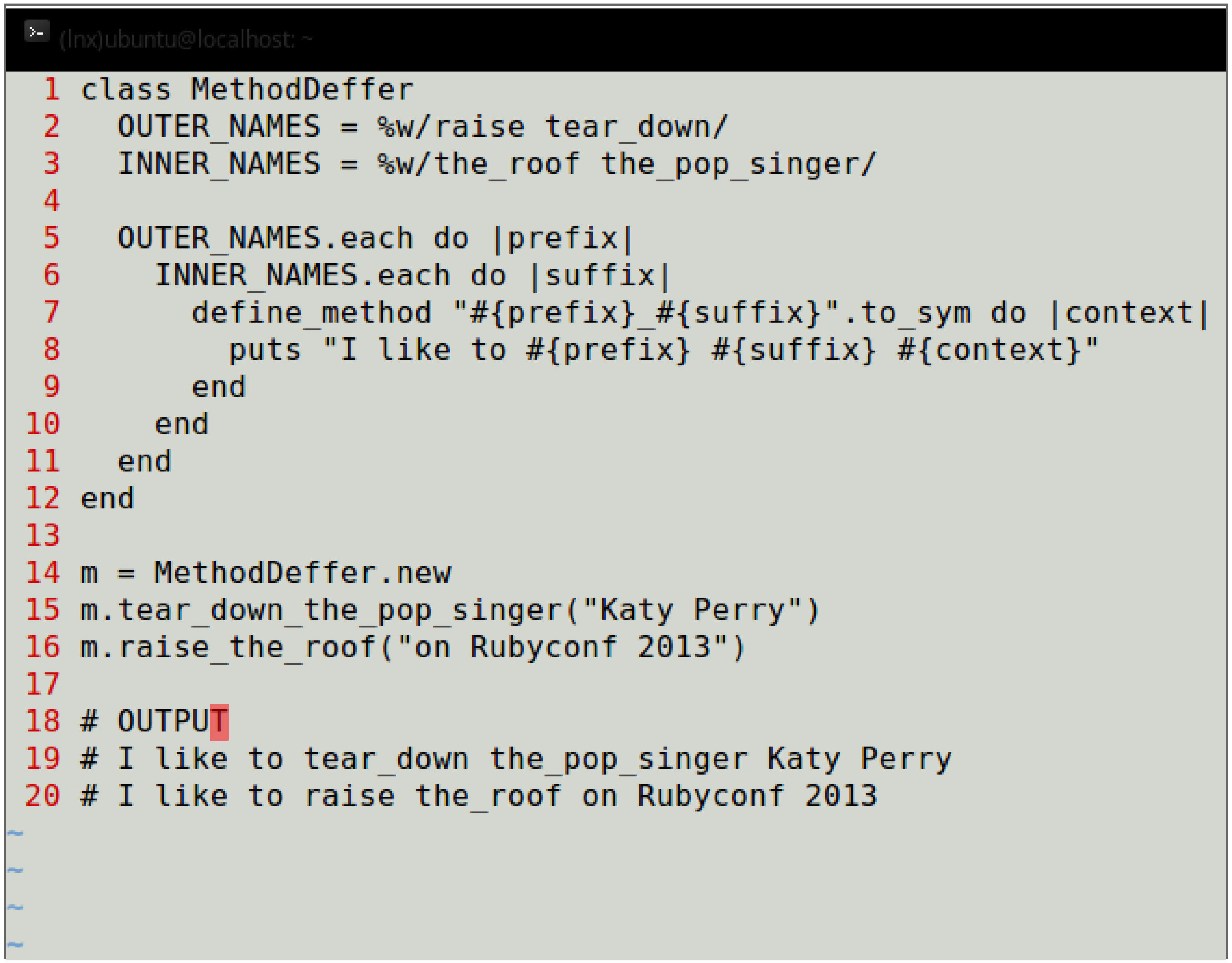
1. deducible from parse-time state
2. demonstrates reason and purpose *in itself*

Obvious

OBVIOUS

1. Greppable*
2. Graspable

ANTI-PATTERN: DEFINING METHODS WITH `define_method`



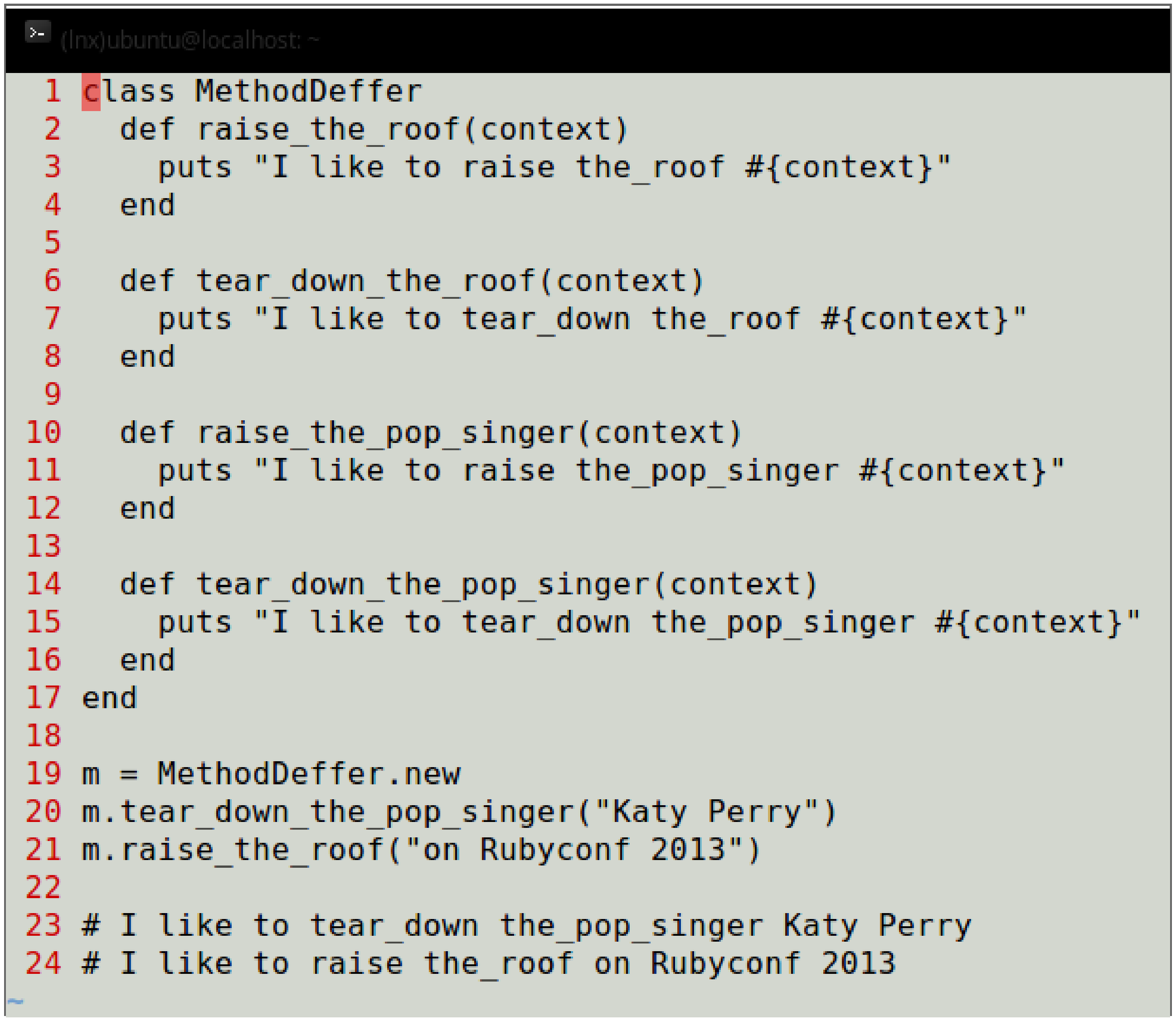
```ruby
class MethodDeffer
  OUTER_NAMES = %w/raise tear_down/
  INNER_NAMES = %w/the_roof the_pop_singer/

  OUTER_NAMES.each do |prefix|
    INNER_NAMES.each do |suffix|
      define_method "#{prefix}_#{suffix}".to_sym do |context|
        puts "I like to #{prefix} #{suffix} #{context}"
      end
    end
  end
end

m = MethodDeffer.new
m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")

# OUTPUT
# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
~
~
~
~
```

OBVIOUSNESS

1. Lowest greppability
2. Mid graspability, trending down

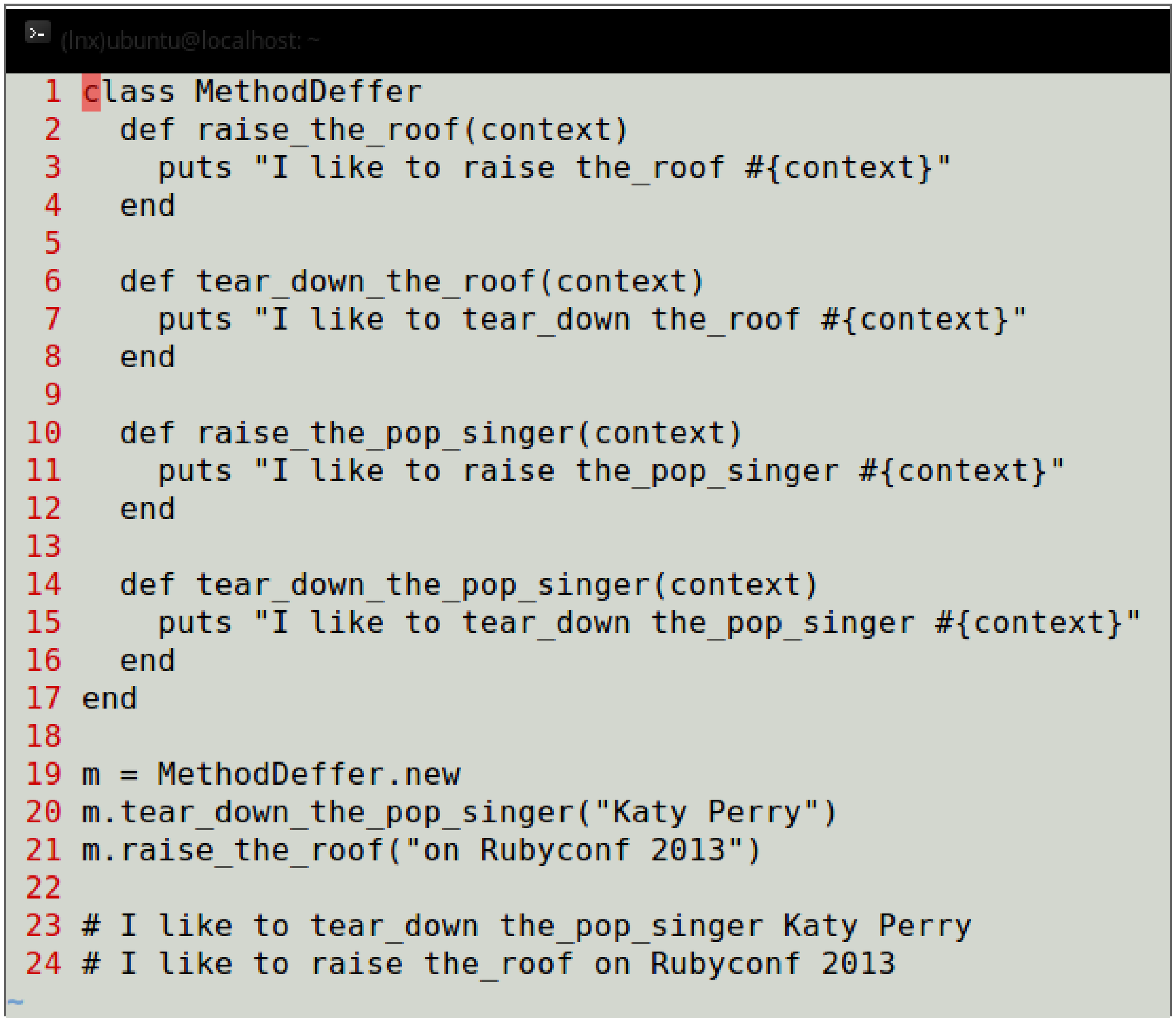
ALTERNATIVES

Use `def`

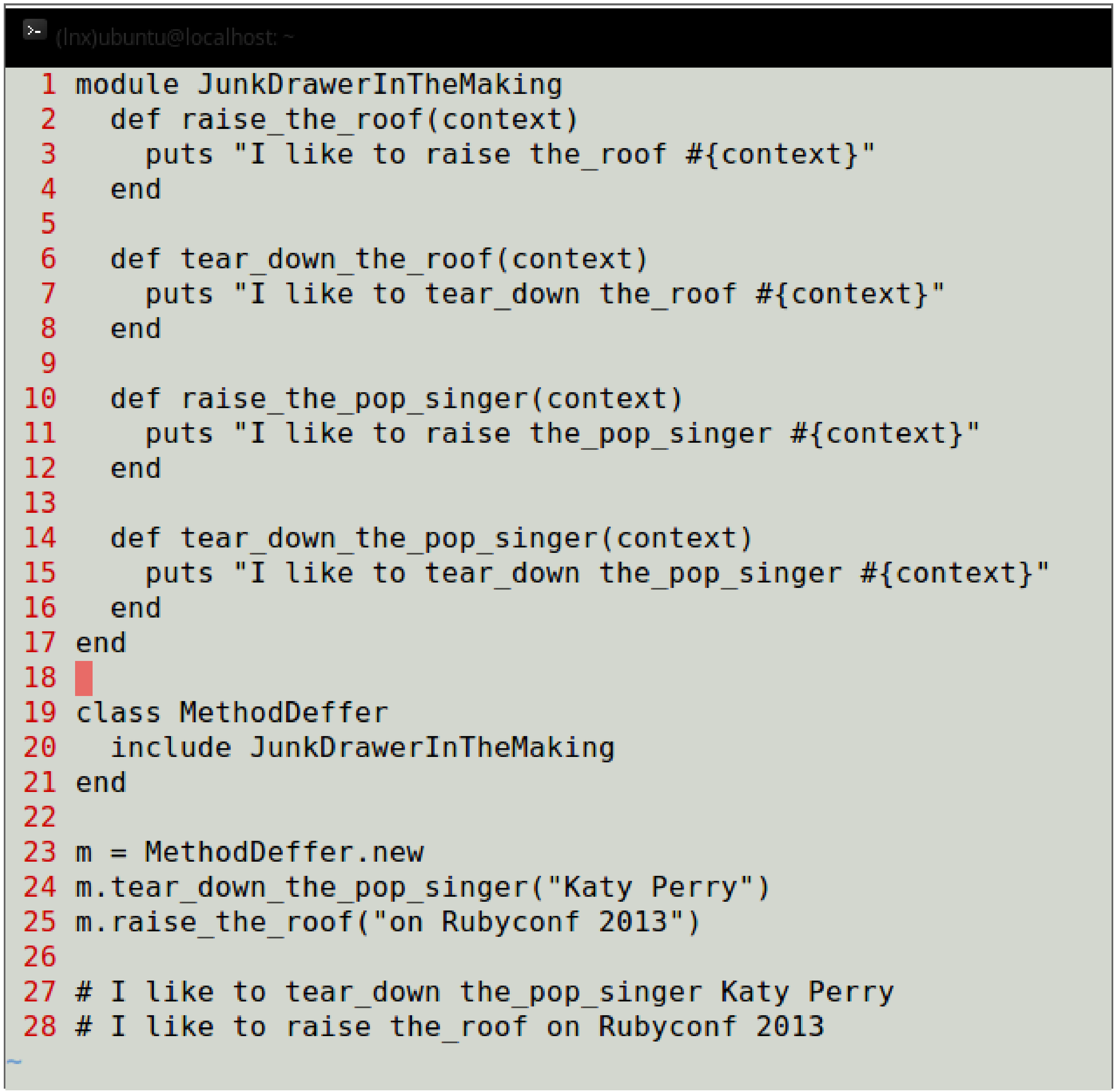


```ruby
class MethodDeffer
  def raise_the_roof(context)
    puts "I like to raise the_roof #{context}"
  end

  def tear_down_the_roof(context)
    puts "I like to tear_down the_roof #{context}"
  end

  def raise_the_pop_singer(context)
    puts "I like to raise the_pop_singer #{context}"
  end

  def tear_down_the_pop_singer(context)
    puts "I like to tear_down the_pop_singer #{context}"
  end
end

m = MethodDeffer.new
m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")

# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
```

ANTI-PATTERN: JUNK-DRAWER MODULES



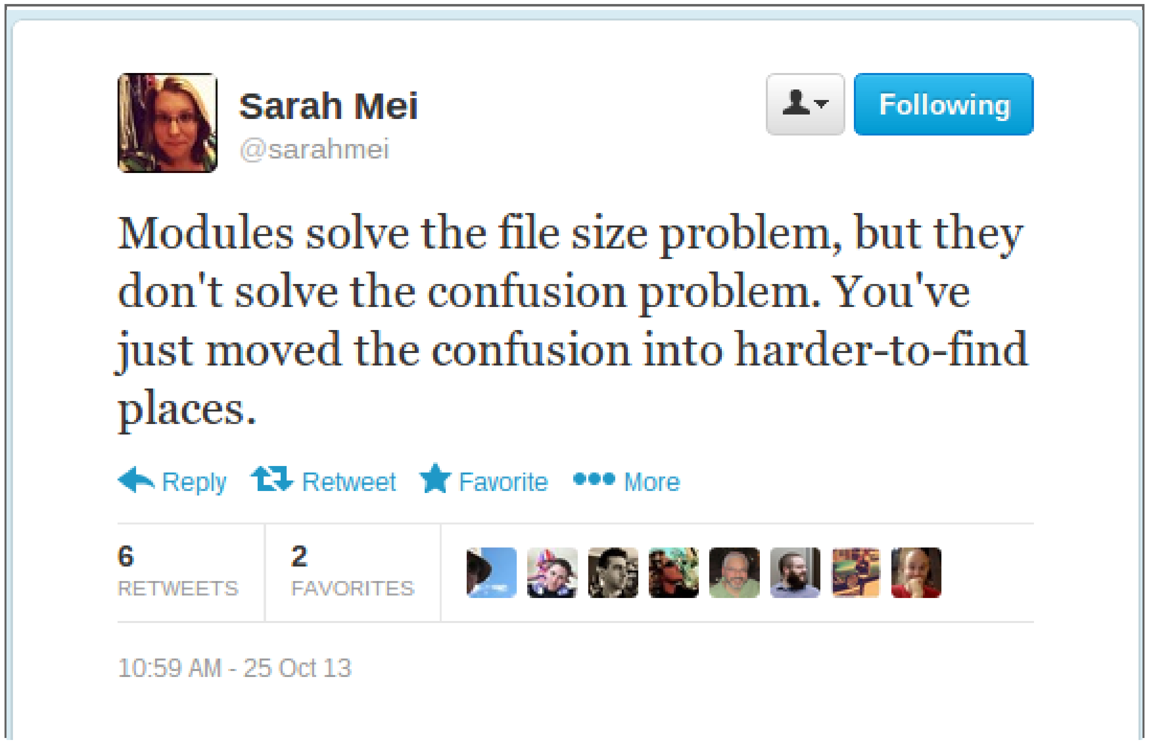
```ruby
class MethodDeffer
  def raise_the_roof(context)
    puts "I like to raise the_roof #{context}"
  end

  def tear_down_the_roof(context)
    puts "I like to tear_down the_roof #{context}"
  end

  def raise_the_pop_singer(context)
    puts "I like to raise the_pop_singer #{context}"
  end

  def tear_down_the_pop_singer(context)
    puts "I like to tear_down the_pop_singer #{context}"
  end
end

m = MethodDeffer.new
m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")

# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
~
```



```ruby
module JunkDrawerInTheMaking
  def raise_the_roof(context)
    puts "I like to raise the_roof #{context}"
  end

  def tear_down_the_roof(context)
    puts "I like to tear_down the_roof #{context}"
  end

  def raise_the_pop_singer(context)
    puts "I like to raise the_pop_singer #{context}"
  end

  def tear_down_the_pop_singer(context)
    puts "I like to tear_down the_pop_singer #{context}"
  end
end

class MethodDeffer
  include JunkDrawerInTheMaking
end

m = MethodDeffer.new
m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")

# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
```

OBVIOUSNESS

1. Highest greppability
2. Middle graspability, trending down

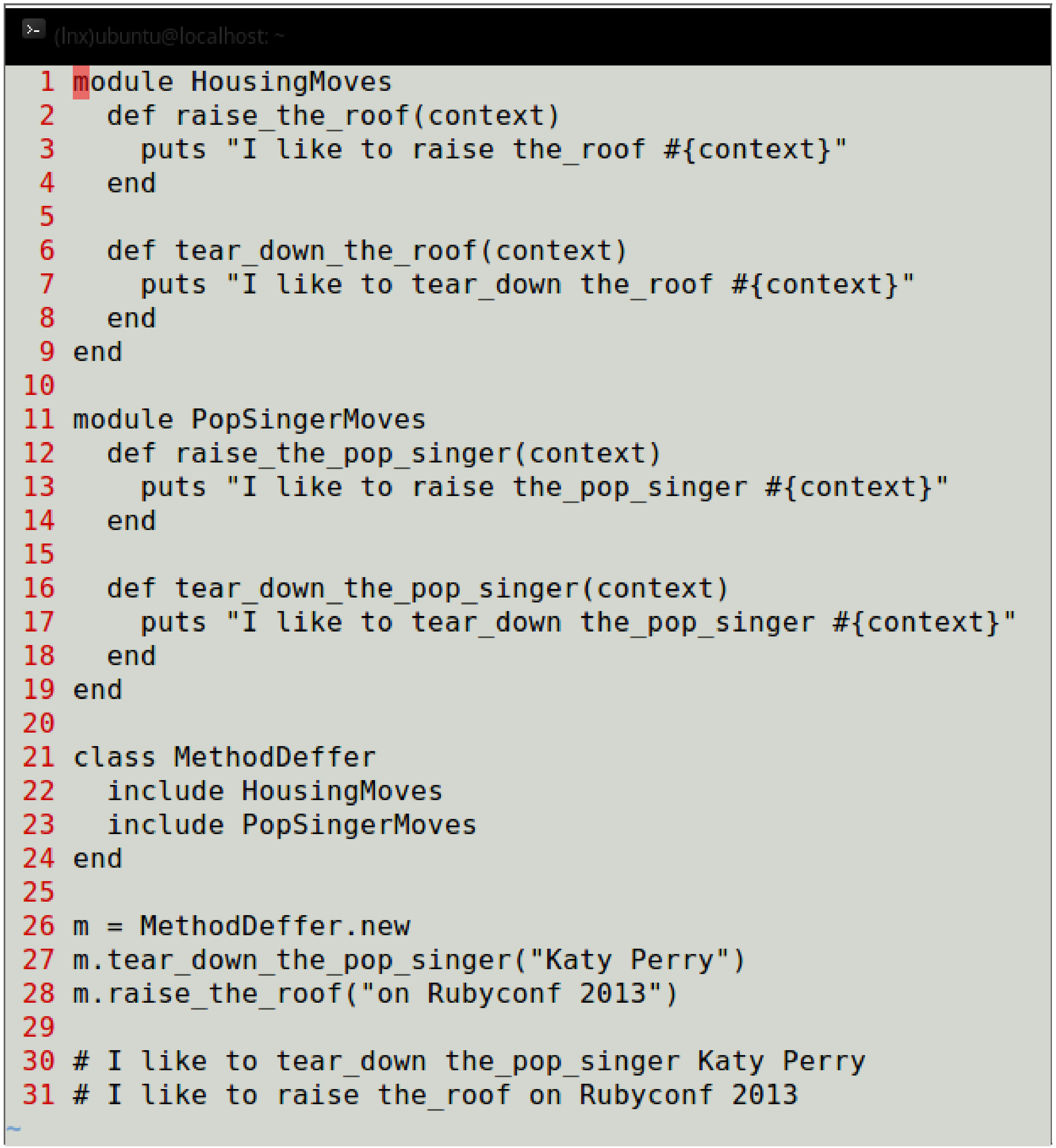


Sarah Mei
@sarahmei

Modules solve the file size problem, but they don't solve the confusion problem. You've just moved the confusion into harder-to-find places.

6 RETWEETS 2 FAVORITES

10:59 AM - 25 Oct 13

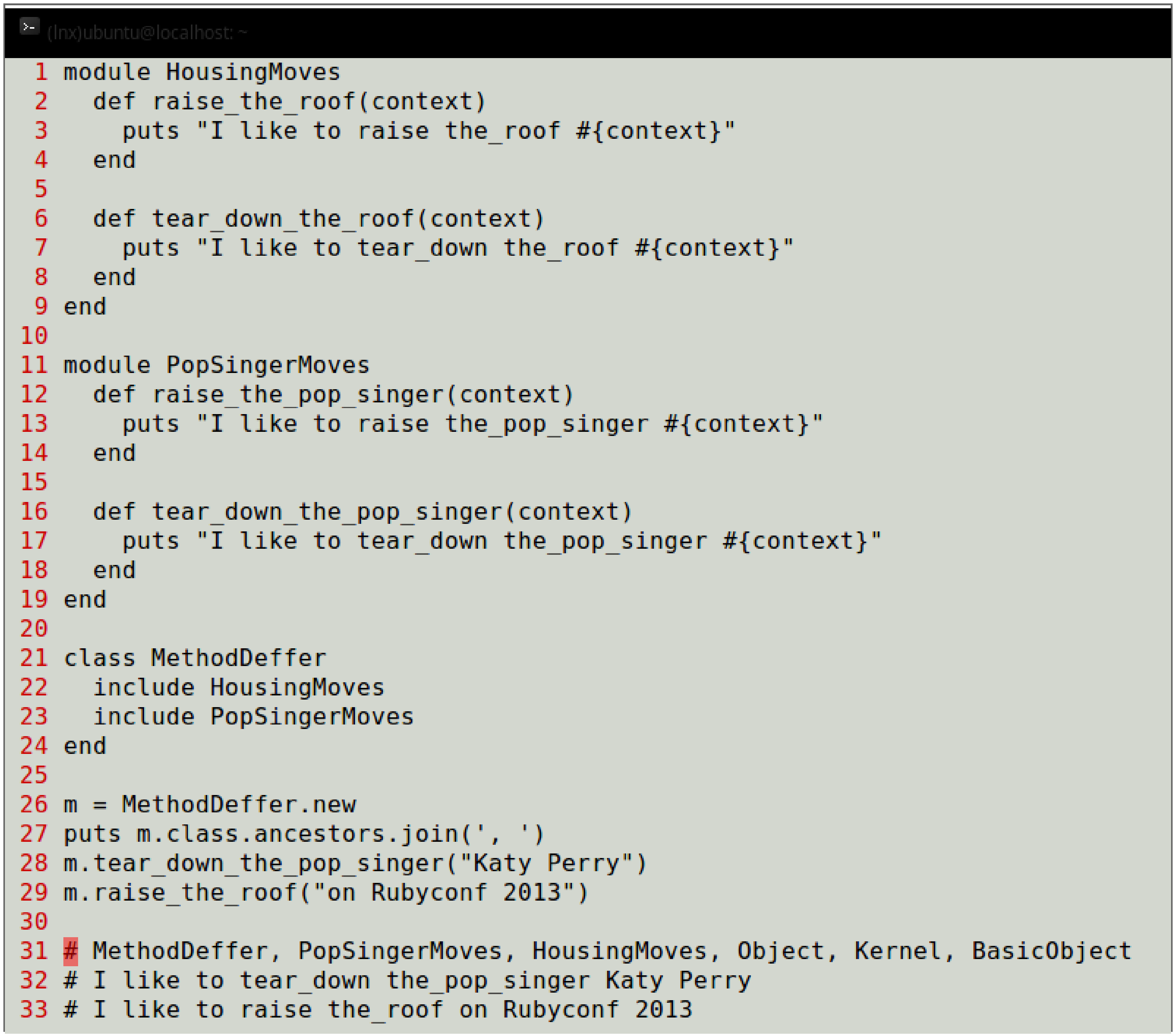


```ruby
module HousingMoves
  def raise_the_roof(context)
    puts "I like to raise the_roof #{context}"
  end

  def tear_down_the_roof(context)
    puts "I like to tear_down the_roof #{context}"
  end
end

module PopSingerMoves
  def raise_the_pop_singer(context)
    puts "I like to raise the_pop_singer #{context}"
  end

  def tear_down_the_pop_singer(context)
    puts "I like to tear_down the_pop_singer #{context}"
  end
end

class MethodDeffer
  include HousingMoves
  include PopSingerMoves
end

m = MethodDeffer.new
m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")

# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
```

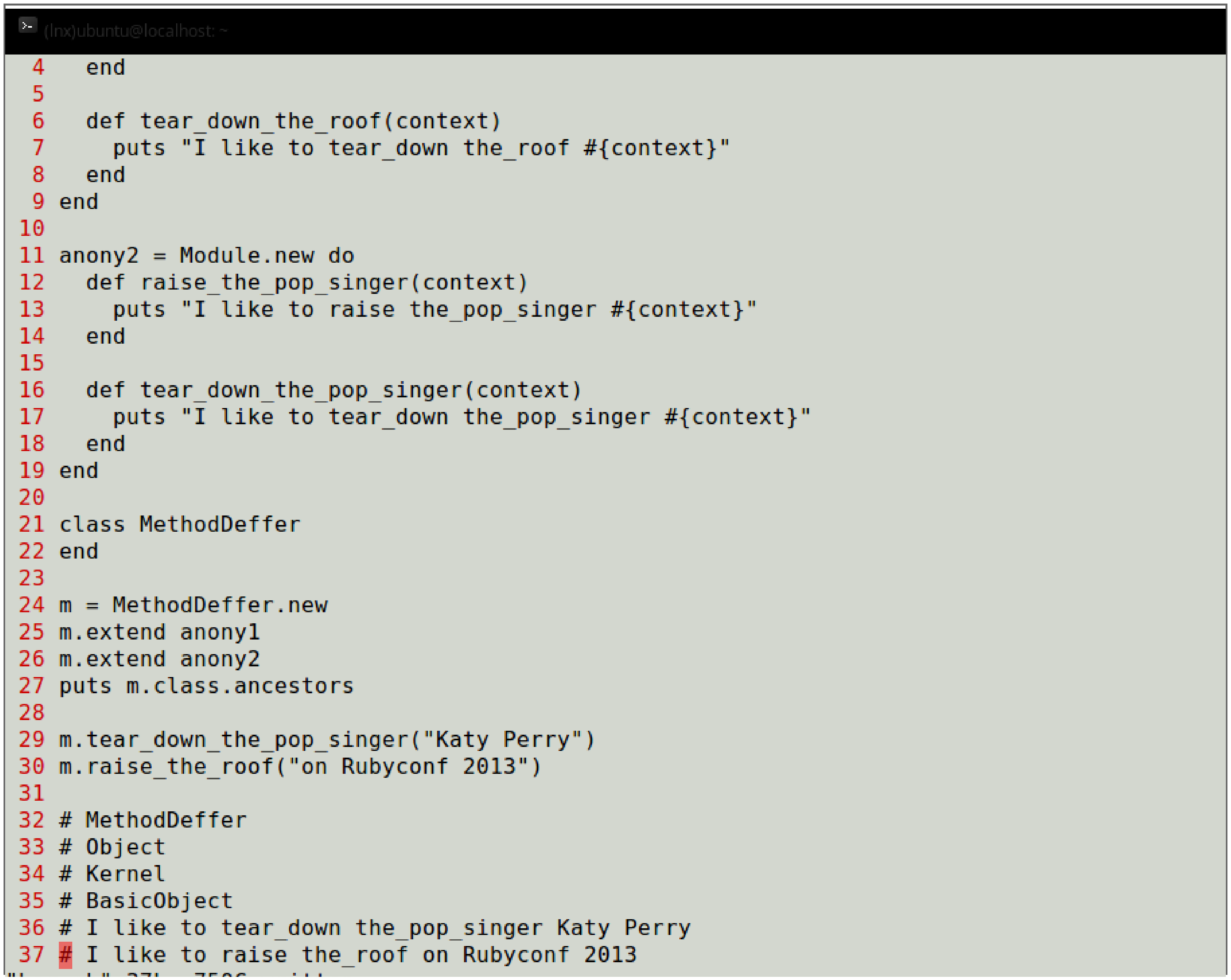
OBVIOUSNESS

1. Highest greppability
2. Middle-High graspability, trending down

ANTI-PATTERN: ANONYMOUS MODULES



```ruby
module HousingMoves
  def raise_the_roof(context)
    puts "I like to raise the_roof #{context}"
  end

  def tear_down_the_roof(context)
    puts "I like to tear_down the_roof #{context}"
  end
end

module PopSingerMoves
  def raise_the_pop_singer(context)
    puts "I like to raise the_pop_singer #{context}"
  end

  def tear_down_the_pop_singer(context)
    puts "I like to tear_down the_pop_singer #{context}"
  end
end

class MethodDeffer
  include HousingMoves
  include PopSingerMoves
end

m = MethodDeffer.new
puts m.class.ancestors.join(', ')
m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")

# MethodDeffer, PopSingerMoves, HousingMoves, Object, Kernel, BasicObject
# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
```



```
  end

  def tear_down_the_roof(context)
    puts "I like to tear_down the_roof #{context}"
  end
end

anony2 = Module.new do
  def raise_the_pop_singer(context)
    puts "I like to raise the_pop_singer #{context}"
  end

  def tear_down_the_pop_singer(context)
    puts "I like to tear_down the_pop_singer #{context}"
  end
end

class MethodDeffer
end

m = MethodDeffer.new
m.extend anony1
m.extend anony2
puts m.class.ancestors

m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")

# MethodDeffer
# Object
# Kernel
# BasicObject
# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
```

OBVIOUSNESS

1. Mid-low greppability, rapidly trending down
2. Mid-low graspability, rapidly trending down

ANTI-PATTERN: MIXINS TO APPLY BEHAVIOR / STATE TO INCLUSOR, HOOK METHOD ABUSE

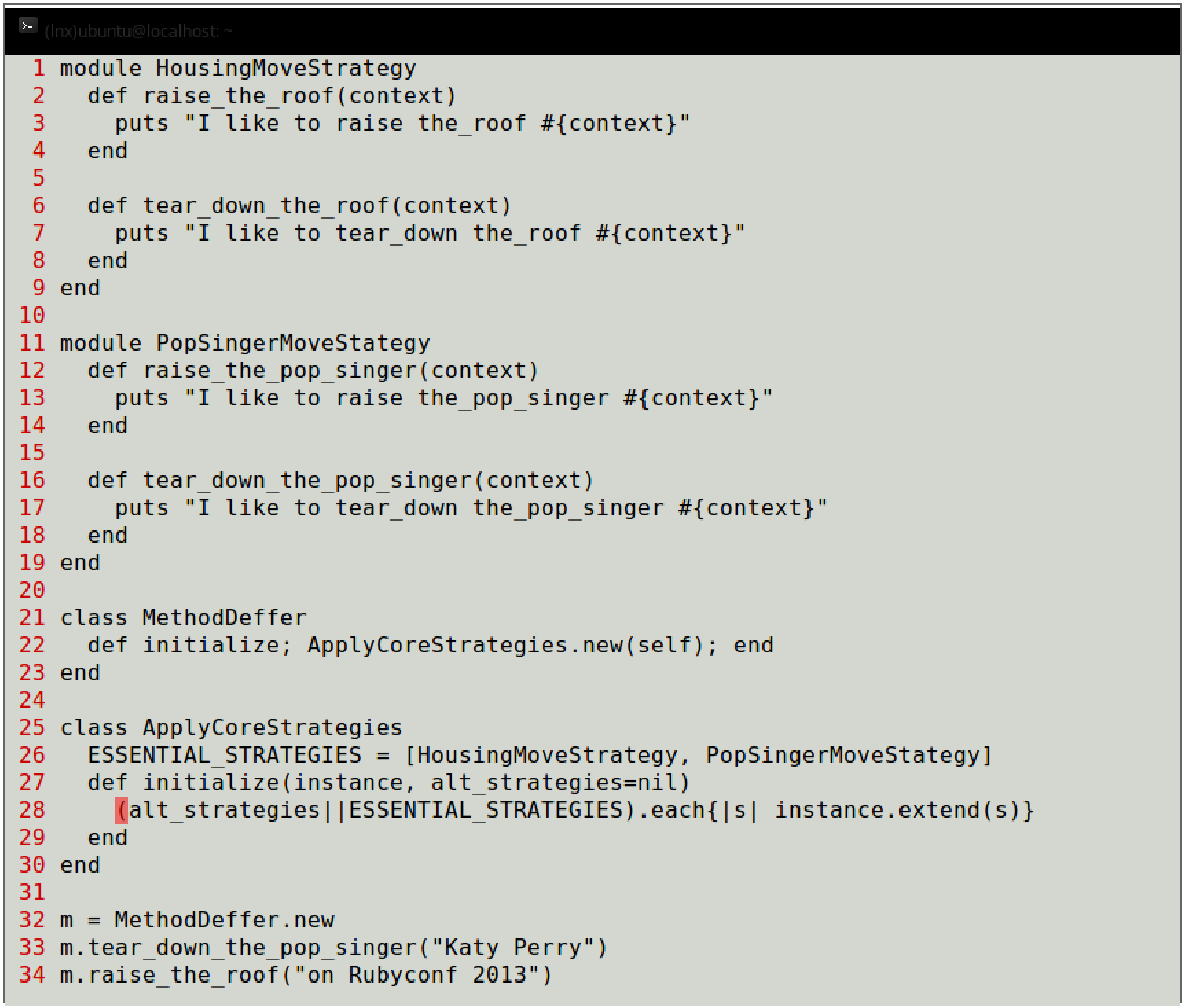
```
def self.extended(extending_instance)
  extending_instance.instance_eval do
    @proxy_verb_string = Linguistics::Latin::Verb::LatinVerb.create_pseudo_active_mask_
    @proxyVerb         = Linguistics::Latin::Verb::LatinVerb.new @proxy_verb_string
    apply_deponent_masking
  end
end
```

OBVIOUSNESS

1. Low greppability, abandon all hope
2. Low graspability, abandon all hope

1. Better pattern already exists!
2. Use strategy pattern to to change behavior of the inclusor

PATTERN: USE STRATEGIES TO APPLY BEHAVIOR / STATE TO INVERTED CLASS



```
module HousingMoveStrategy
  def raise_the_roof(context)
    puts "I like to raise the_roof #{context}"
  end

  def tear_down_the_roof(context)
    puts "I like to tear_down the_roof #{context}"
  end
end

module PopSingerMoveStategy
  def raise_the_pop_singer(context)
    puts "I like to raise the_pop_singer #{context}"
  end

  def tear_down_the_pop_singer(context)
    puts "I like to tear_down the_pop_singer #{context}"
  end
end

class MethodDeffer
  def initialize; ApplyCoreStrategies.new(self); end
end

class ApplyCoreStrategies
  ESSENTIAL_STRATEGIES = [HousingMoveStrategy, PopSingerMoveStategy]
  def initialize(instance, alt_strategies=nil)
    (alt_strategies||ESSENTIAL_STRATEGIES).each{|s| instance.extend(s)}
  end
end

m = MethodDeffer.new
m.tear_down_the_pop_singer("Katy Perry")
m.raise_the_roof("on Rubyconf 2013")
```

OBVIOUSNESS

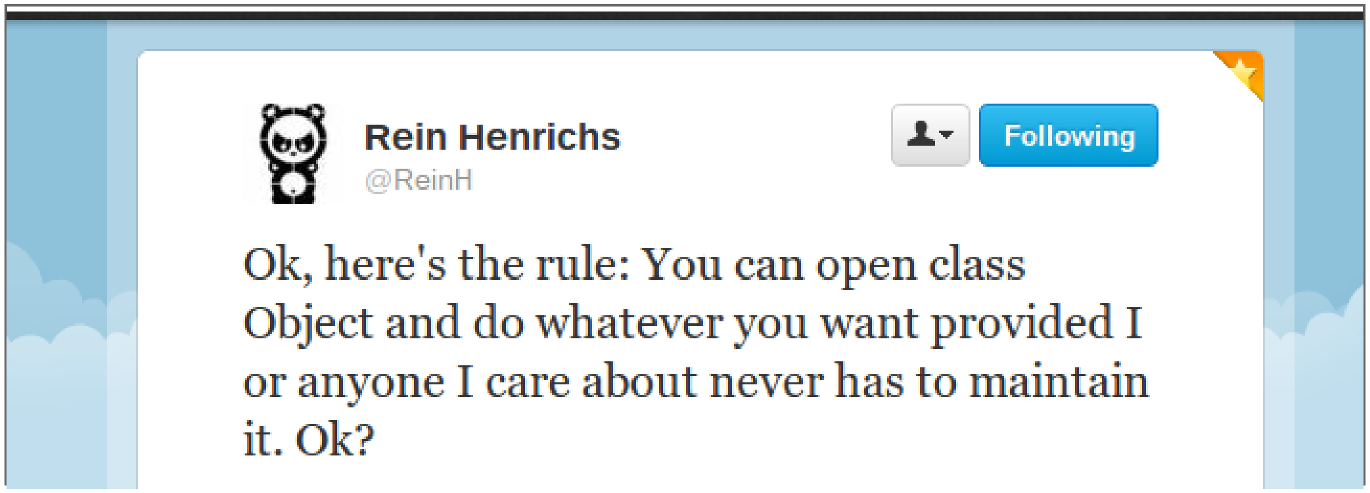
1. High greppability, easy refactors
2. High graspability, easy refactors
3. Verging on good OO!

ON THE TOPIC OF GOOD OO…

ANTI-PATTERN: MIND OF GOD CONSTRUCTS

Many quote-metaprogramming techniques go against OO

1. Method Missing
2. Open Class
3. Kernel method



Rein Henrichs
@ReinH

Ok, here's the rule: You can open class Object and do whatever you want provided I or anyone I care about never has to maintain it. Ok?

THEME OF RUBYCONF 2013

MATURING RUBY ECOSYSTEM

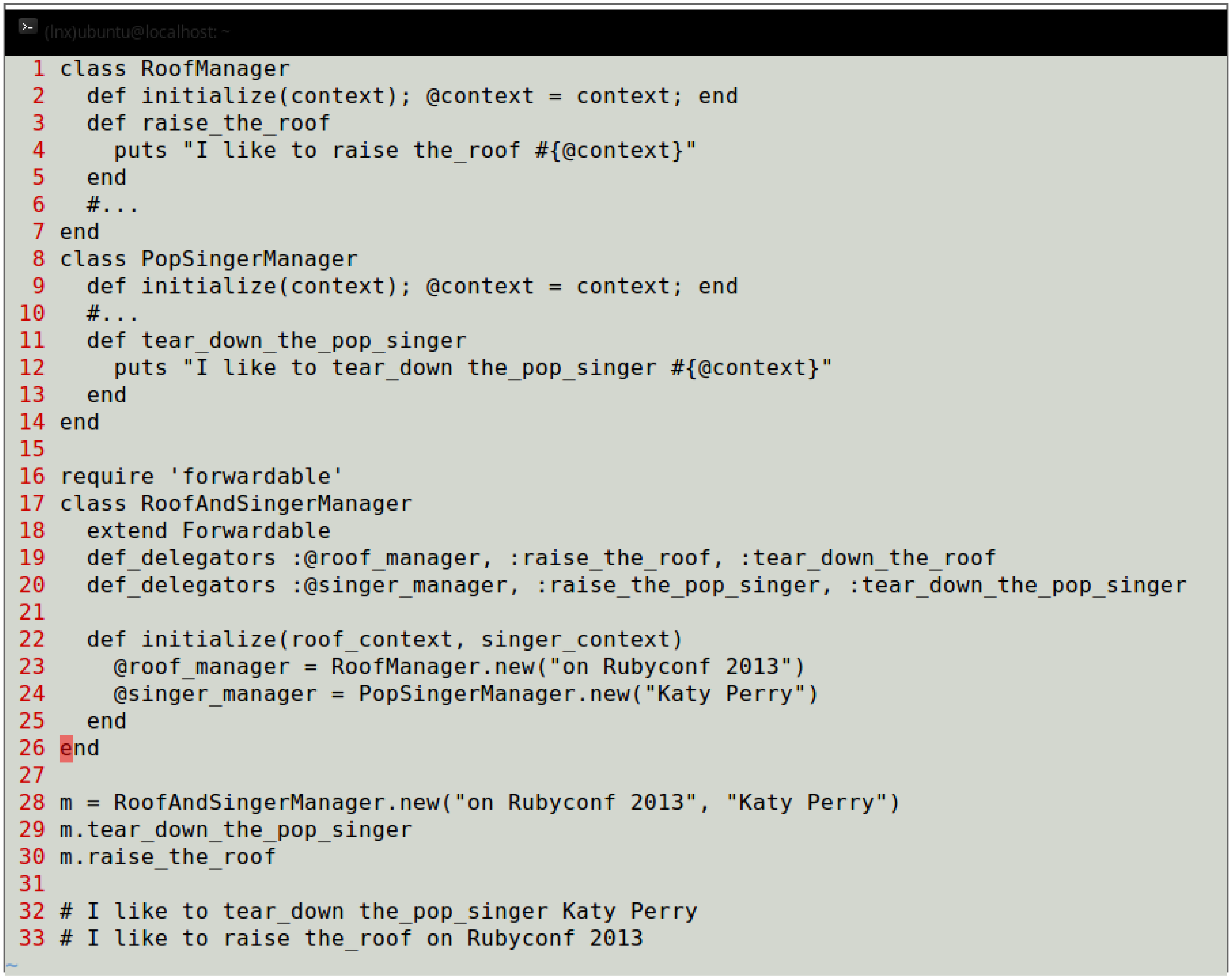
- Jason Clark: Extending Gems - Patterns and Anti-Patterns of Making Your Gem Pluggable — *Which mind of God wins?*
- Emily Stolfo: API design for gem authors (and users)

1. Method Missing
2. Open Class
3. Kernel method

1. Greppable? Kinda, sometimes mayyyybe?
2. Graspable? Not readily without comments

PREFER INSTEAD…

1. Strategies
2. Delegates



```ruby
class RoofManager
  def initialize(context); @context = context; end
  def raise_the_roof
    puts "I like to raise the_roof #{@context}"
  end
  #...
end
class PopSingerManager
  def initialize(context); @context = context; end
  #...
  def tear_down_the_pop_singer
    puts "I like to tear_down the_pop_singer #{@context}"
  end
end

require 'forwardable'
class RoofAndSingerManager
  extend Forwardable
  def_delegators :@roof_manager, :raise_the_roof, :tear_down_the_roof
  def_delegators :@singer_manager, :raise_the_pop_singer, :tear_down_the_pop_singer

  def initialize(roof_context, singer_context)
    @roof_manager = RoofManager.new("on Rubyconf 2013")
    @singer_manager = PopSingerManager.new("Katy Perry")
  end
end

m = RoofAndSingerManager.new("on Rubyconf 2013", "Katy Perry")
m.tear_down_the_pop_singer
m.raise_the_roof

# I like to tear_down the_pop_singer Katy Perry
# I like to raise the_roof on Rubyconf 2013
~
```

ANTI-PATTERN: LAZINESS LOOKUPS

`const_get`, `constantize`

1. Low greppability
2. Low graspability

1. Type it out
2. What's wrong with a simple `MAPPING` hash?

PRINCIPAL TECHNIQUES FOR HAPPIER PROGRAMMING

1. Make lots of small, greppable classes
2. Prefer composition and delegation
    1. Inject, inject, inject
    2. Love inversion
3. Strategies for the unnamable or that which requires comment

FURTHER REFERENCE

1. POODR: Start here
2. Refactoring Ruby Edition
3. Design Patterns in Ruby

WINDING UP: METAPROGRAMMING

1. **DON'T** write more of it
2. **DO** understand it
    - Lotta legacy metaprogrammatic code out there needing refactoring
    - Very helpful as a debugging technique with IRB, Pry, Byebug, Debugger

THANKS

1. Shop It To Me Engineering: Past and Present: Rafael, Alex, Kenneth, Jeremy, Josh, Rich
2. A-W Authors who made this path of transition possible: Sandi Metz, Russ Olsen, Jay Fields & Shane Harvie
3. Lauren: Who always supports my dreams
4. YOU! And **our** awesome ~~Ruby Community~~Team of Garbage Collectors

VALEDICTION: ALL IS WELL

***Rancho**: That day I understood that this heart scares easily.*
You have to trick it, however big the problem is. Tell your heart,
'Pal, all is well. All is well.'
***Raju**: Does that solve the problem?*
***Rancho**: No, but you gain courage to face it*

"3 Idiots"

END