HOW TO BE A FUNCTIONAL PROGRAMMER

without being a jerk about it

NATHAN DOTZ • @NATHANDOTZ • CODEMASH 2014

SLEEPYNATE

MONAD

FUNCTOR

ZYGOHISTOMORPHIC PREPROMORPHISMS

You’re a jerk.

Nathan Dotz (sleepynate)


I F#
ON THE
FIRST
DATE

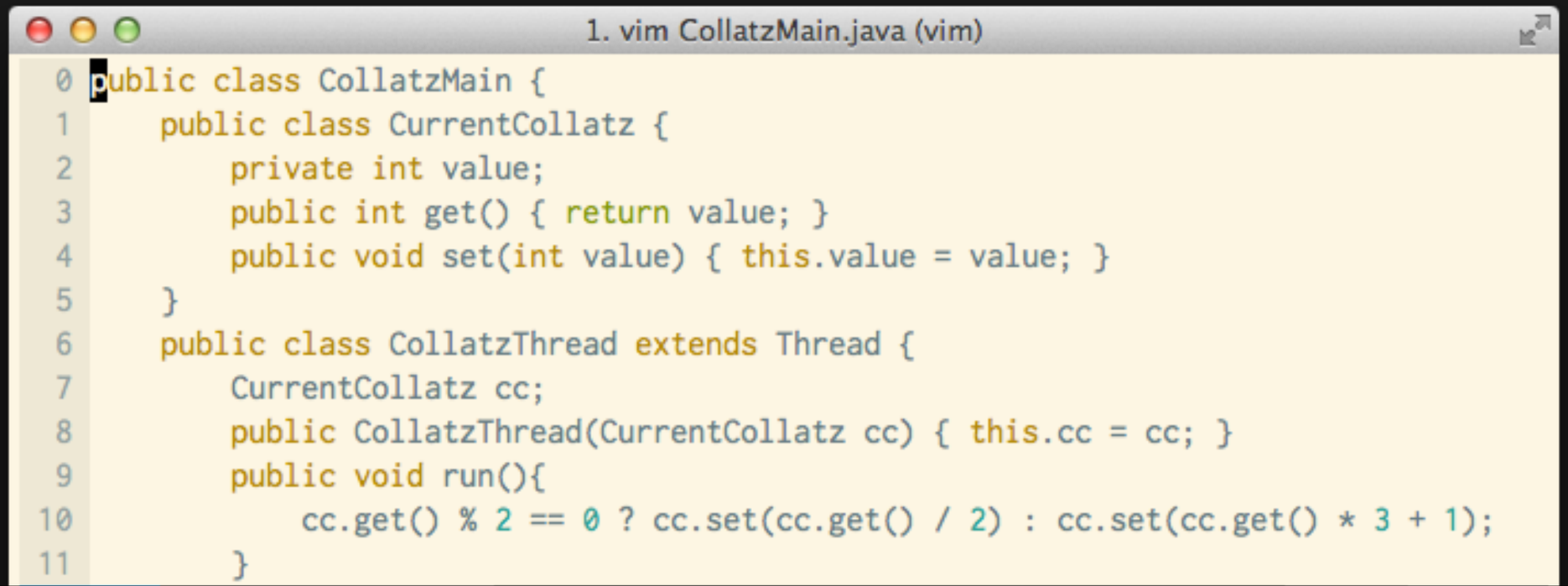
IMPERATIVE PROGRAMMING OBVIOUSLY SUCKS

Damn, this t-shirt is ugly

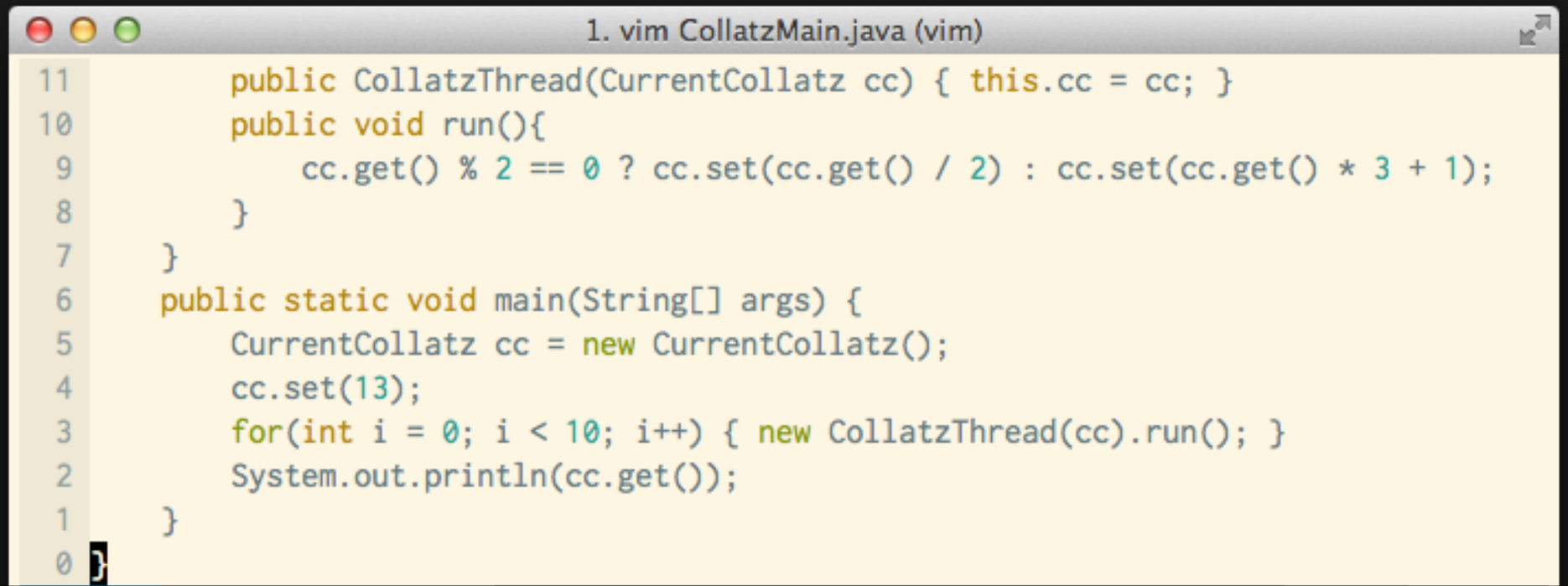
WHY DOES IMPERATIVE SUCK?

- Mutability
- Break, continue, goto. Where's the control?
- Assignment and dereferencing

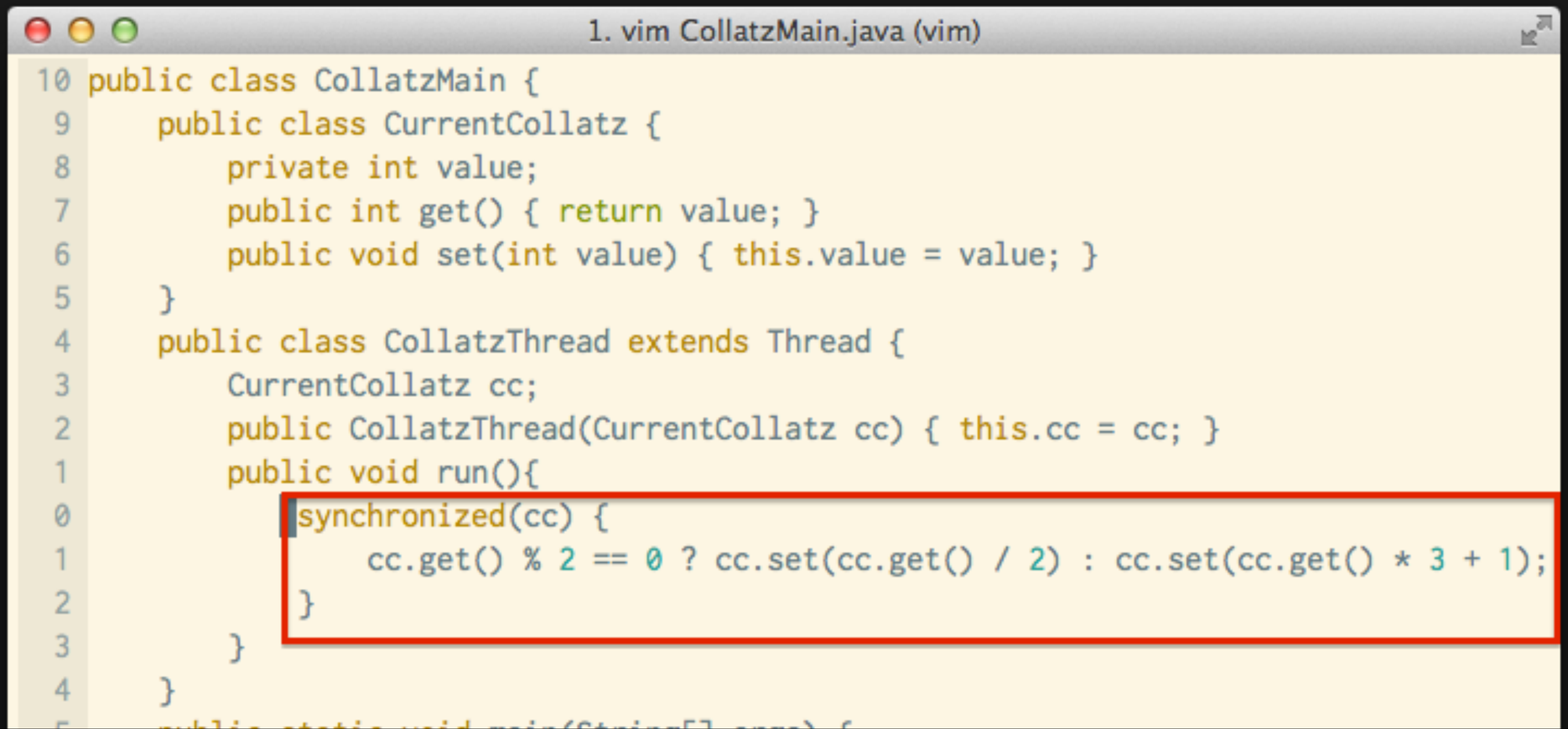
IMPERATIVE SUCKS ‘CAUSE MUTABILITY?



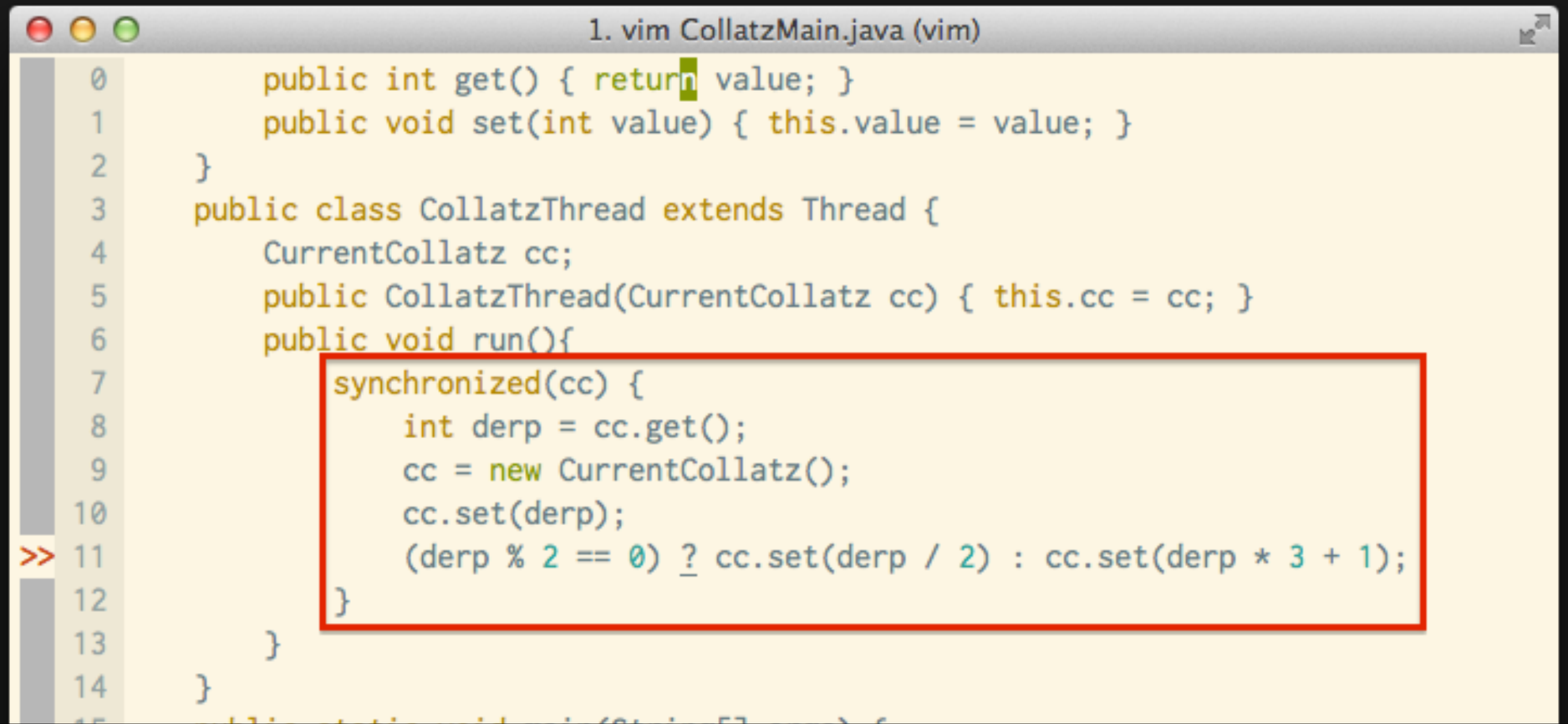
IMPERATIVE SUCKS ‘CAUSE MUTABILITY?



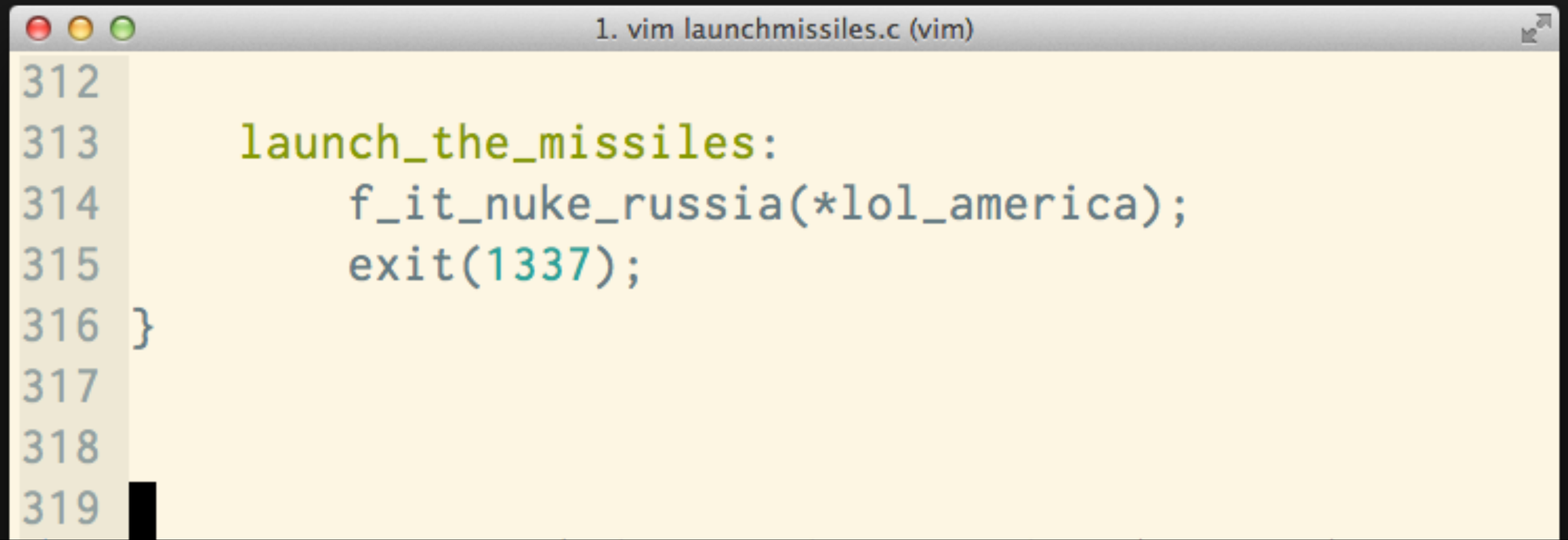
IMPERATIVE SUCKS ‘CAUSE MUTABILITY?



IMPERATIVE SUCKS ‘CAUSE MUTABILITY?



IMPERATIVE SUCKS ‘CAUSE GOTO?



WHY DOES IMPERATIVE SUCK?

Loading and unloading instructions in registers is done ***procedurally***

(yes, I'm talking about you, assembly)

OO AND FUNCTIONAL ARE BROS

Object-oriented programming is an exceptionally bad idea which could only have originated in California.

Edsger Dijkstra

The phrase ‘object-oriented’ means a lot of things. Half are obvious, and the other half are mistakes.

Paul Graham

State is the root of all evil.
In particular, functions with side effects should be avoided.

Joe Armstrong

Everything Object Oriented Programming can do can be done better in functional programming.

Doug Ransom

lol wat you mean ur code uses
a for loop in a single thread?

some idiot on IRC

Objected oriented programming and functional programming are orthogonal, not contrary to one another.

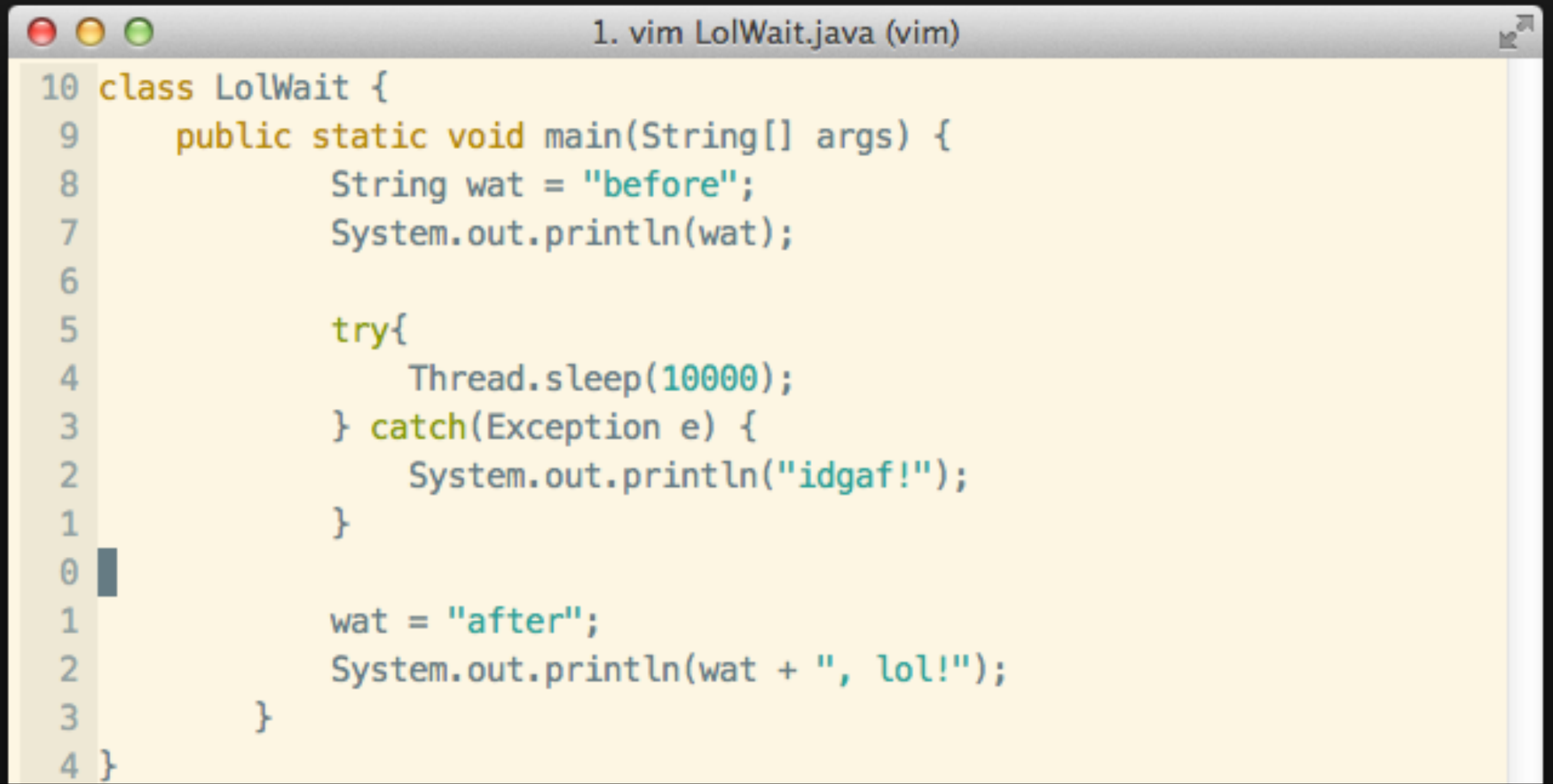
Martin Odersky

THE INERRANT FUNCTIONAL DOGMA

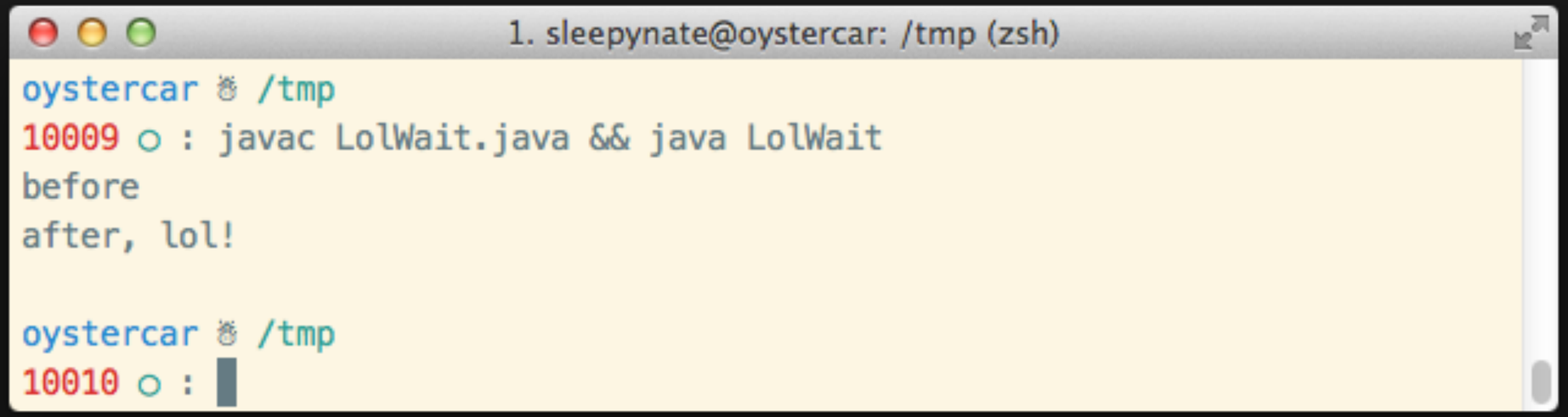
THE SACRED FUNCTIONAL VIRTUES

- Immutability is god
- First-class functions
- Composition and partial application

THERE IS NO GOD BEFORE IMMUTABILITY

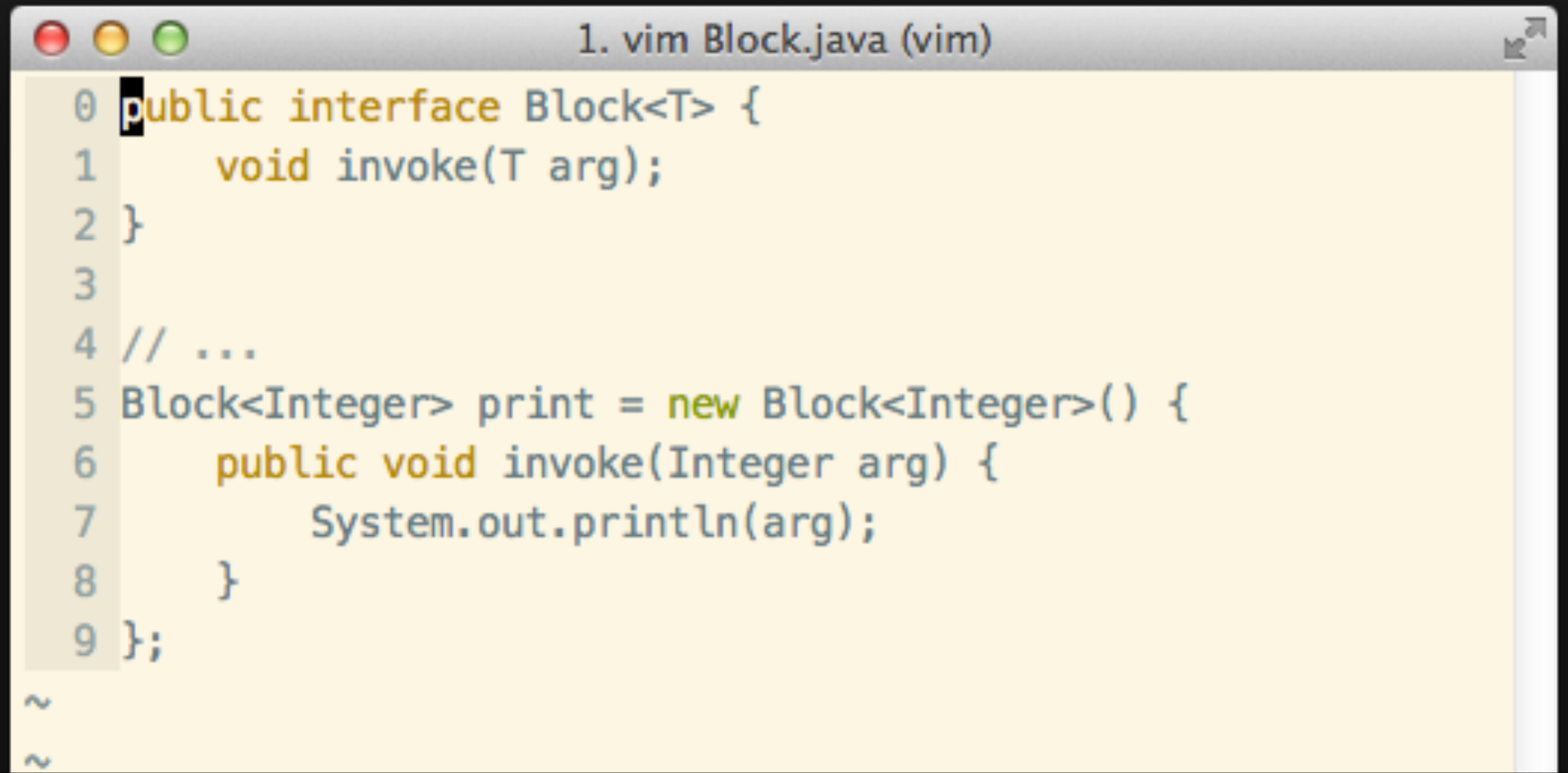


THERE IS NO GOD BEFORE IMMUTABILITY



THERE IS NO GOD BEFORE IMMUTABILITY

YOU CAN'T HAVE VERBS WITHOUT CLOSURES



TRUE POWER COMES FROM COMPOSING LOGIC

```
/**
 * Function composition.
 *
 * @param f A function to compose with another.
 * @param g A function to compose with another.
 * @return A function that is the composition of the given arguments.
 */
public static <A, B, C> F<A, C> compose(final F<B, C> f, final F<A, B> g) {
  return new F<A, C>() {
    public C f(final A a) {
      return f.f(g.f(a));
    }
  };
}
```

MY VOCABULARY SAYS YOU'RE DUMB

FUNCTIONAL S*!T-TALKING

- “A monad is just a monoid in the category of endofunctors.”
- “Applicative Functors”
- Currying
- Zygomorphic Prepromorphisms … stfu, no

THREE MONAD LAWS

Left identity: `return a >>= f ≡ f a`

Right identity: `m >>= return ≡ m`

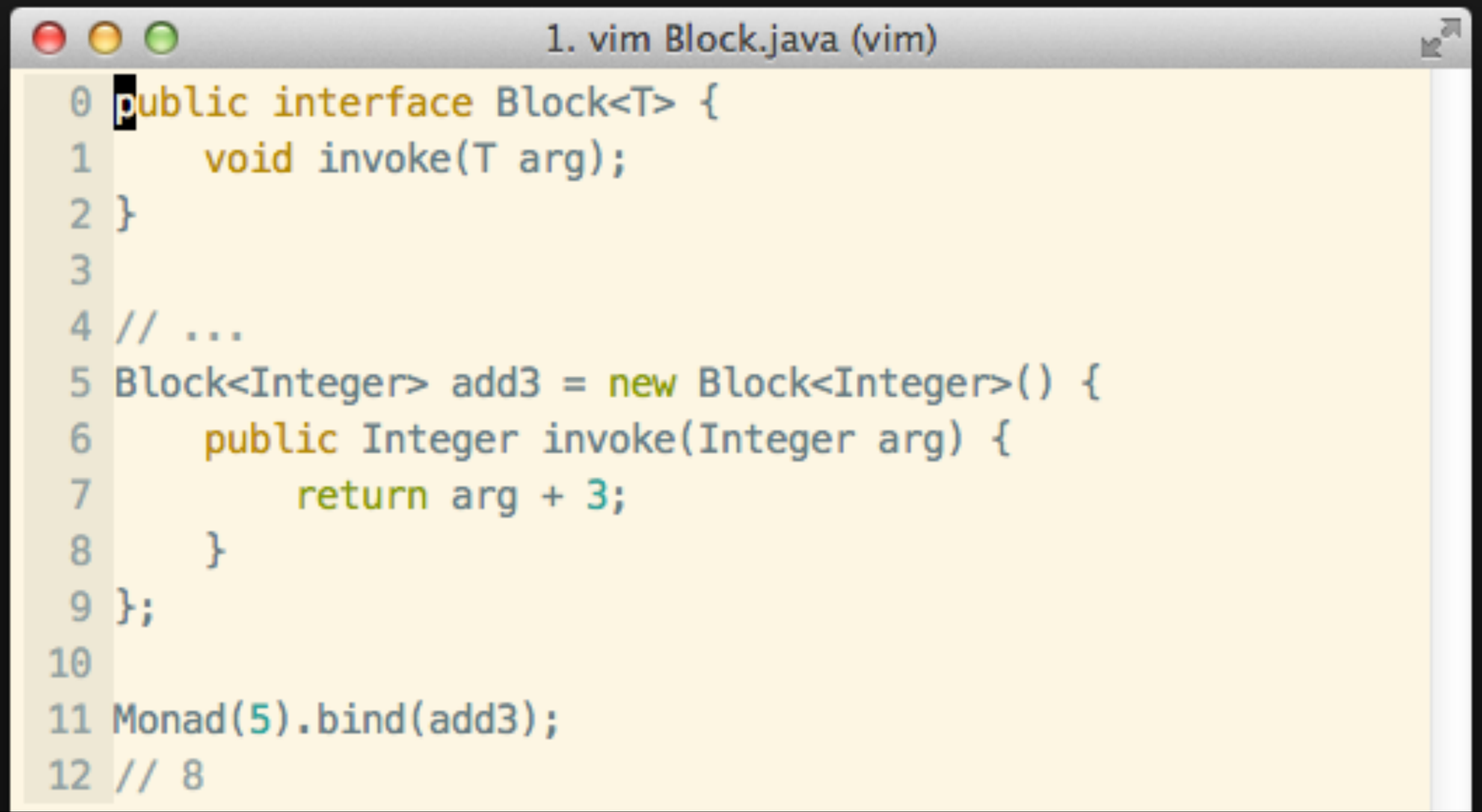
Associativity: `(m >>= f) >>= g ≡ m >>= (\x -> f x >>= g)`

THREE MONAD LAWS

- `do { f x }` is equivalent to `do { v <- return x; f v }`
- `do { m }` is equivalent to `do { v <- m; return v }`
- `do { x <- m; y <- f x; g y }` is equivalent to `do { y <- do { x <- m; f x }; g y }`

THREE MONAD LAWS

Monad(x).bind(f) is the same damn thing as f(x)



THREE MONAD LAWS

Monad(x) is the same damn thing
as Monad(x)

THREE MONAD LAWS

Monad(Monad(x).bind(f)).bind(g)

is the same damn thing as

Monad(f(x)).bind(g)

THREE MONAD LAWS

- You don’t tell noobs about Monads.
- You don’t tell noobs about Monads.
- If it’s your first Monad, you don’t write a burrito tutorial.

THE MONAD LAW

"A monad is an object that
lets you use the object inside it"

FUNCTORS

"Stuff you can map over"

APPLICATIVE FUNCTORS

"Stuff you can map over,
and get a stuff of functions back"

CURRYING

"A function that takes an argument and gives you a function that does the same thing with one less argument"

ZYGOHISTOMORPHIC PREPROMORPHISMS

"Go home haskell, you're drunk."

A CRIB SHEET

DECLARE,
DON’T ASSIGN

CALCULATE, DON’T MUTATE

PATTERN MATCH, DON’T IF-THEN-ELSE

RECUR,
DON’T LOOP

PASS FUNCTIONS, NOT STRUCTS

MAP, FILTER, AND REDUCE

DON'T BE A BULLY

Rank	Language	# Repositories Created
1	JavaScript	264131
2	Ruby	218812
3	Java	157618
4	PHP	114384
5	Python	95002
6	C++	78327
7	C	67706
8	Objective-C	36344
9	C#	32170
10	Shell	28561
11	CSS	17813
12	Perl	15412
13	CoffeeScript	11133
14	VimL	7857
15	Scala	6918
16	Go	6884
17	Prolog	5829
18	Clojure	4904
19	Haskell	4681
20	Lua	4048

Sure you can, it’s pretty much like what you do already

You