# Clean Code

Gregorio Marciano

```
class 💩💩💩💩
{
  func 💩💩💩(😎: Int, 🐆: Int) -> Int
  {
    return 😎 + 🐆
  }
}

var 🐔 = 3
var 😓 = 🐔 + 2

var 💩 = 💩💩💩💩()
println(💩.💩💩💩(🐔, 🐆:😓))
```

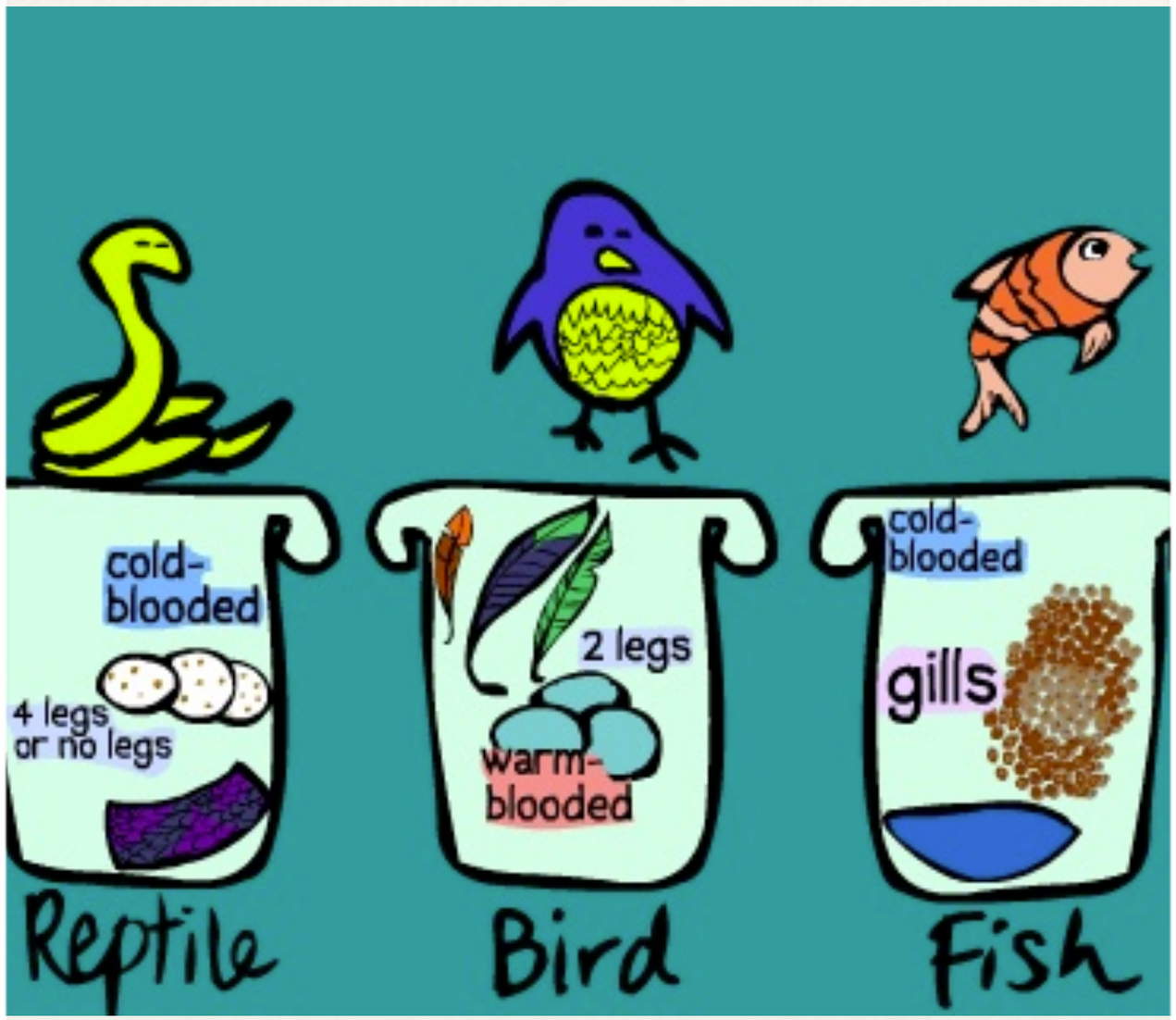
# ¿Clean code?

- Clean code is simple and direct.
- Clean code can be read, and enhanced by a developer other than its original author.
- Runs all the tests
- Contains no duplication

# Meaningful Names

Use Intention-Revealing Names
Avoid Disinformation.
Use Searchable Names
Make Meaningful Distinctions
Use Pronounceable Names
Don’t Be Cute
Use Solution Domain Names
Avoid Encodings *
  intAge
  s_Username

# Class




- Classes and objects should have noun or noun phrase names like Customer, Account, and AddressParser.
- A class name should not be a verb.
- Classes Should Be Small
- SRP: class or module should have one, and only one, reason to change.

# Class

- High cohesion
- Maintaining Cohesion Results in Many Small Classes
- Organizing for Change
- Isolating from Change

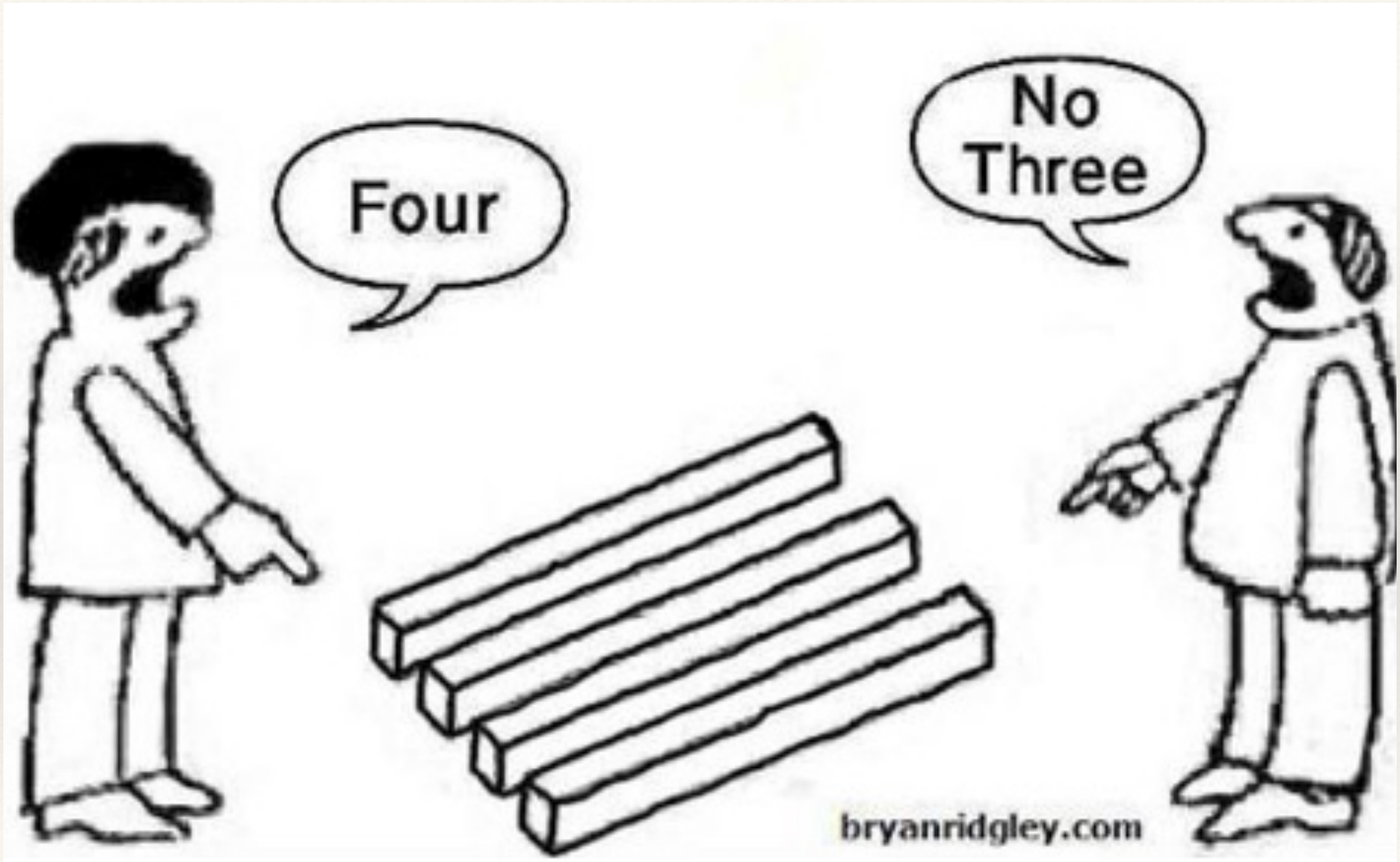
# Function

- The first rule of functions is that they should be small. The second rule of functions is that they should be smaller than that.
- Lines should not be 150 characters long. Functions should not be 100 lines long. Functions should hardly ever be 20 lines long.
- Small vertical separation

# Functions

- Do One Thing
- One Level of Abstraction per Function
- Reading Code from Top to Bottom
- Methods should have verb or verb phrase names
- Pick One Word per Concept.
- Don’t Add Gratuitous Context
- Try/Catch

# Function Arguments



- The ideal number of arguments for a function is zero (niladic). Next comes one (monadic), followed closely by two (dyadic). Three arguments (triadic) should be avoided where possible. More than three (polyadic) requires very special justification—and then shouldn't be used anyway.

- Flag Arguments
- Have No Side Effects
- Output Arguments
- Prefer Exceptions to Returning Error Codes

# How do I write Clean Code?

La presentación está basada en el libro

# Clean Code: A Handbook of Agile Software Craftsmanship

*Robert C. Martin*