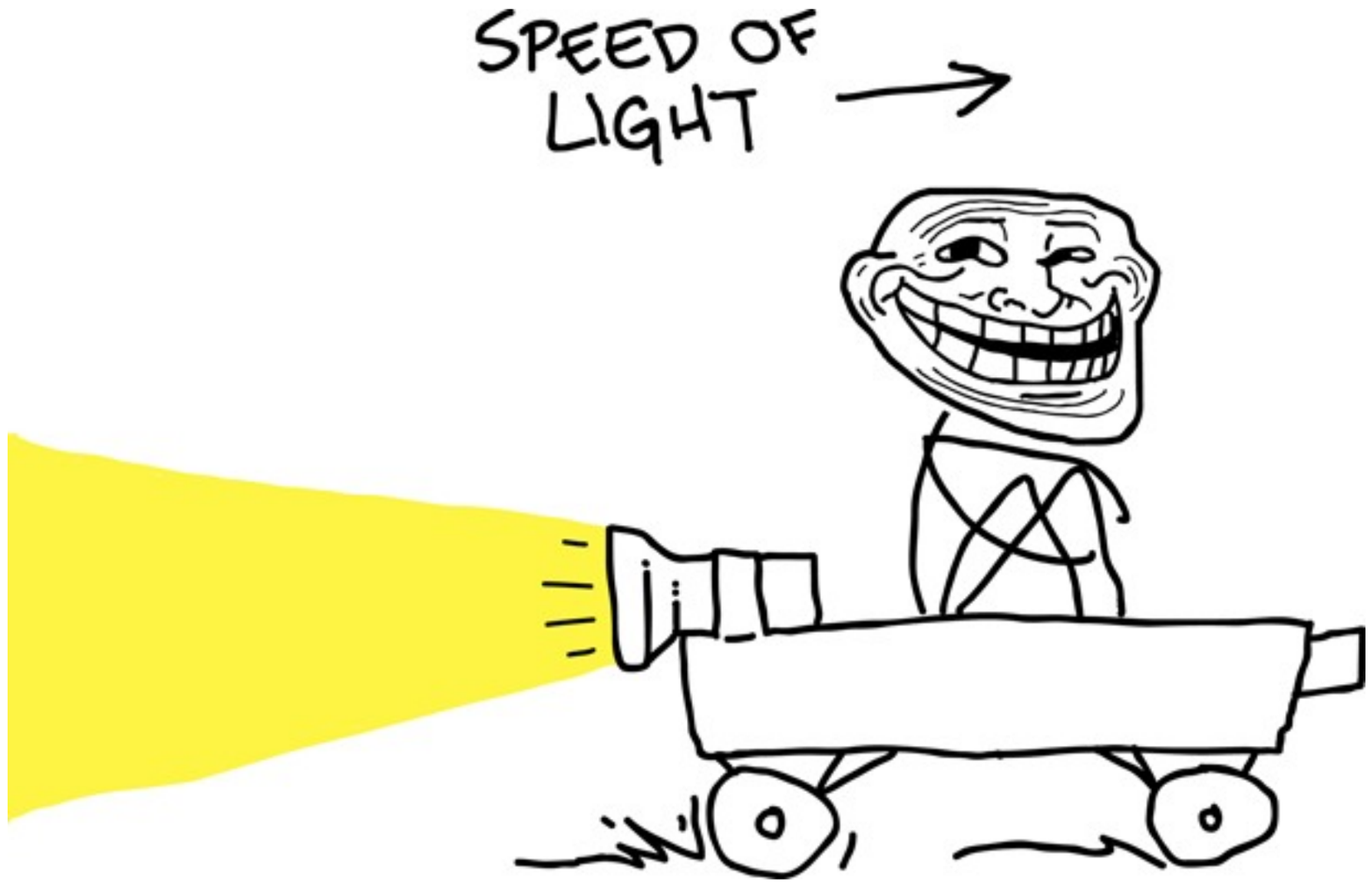
Scala ao Resgate: Testes de Software mais expressivos e com menos código na JVM

Filipe Sabella => @FilipeSabella
Lucas Torri => @lucastorri

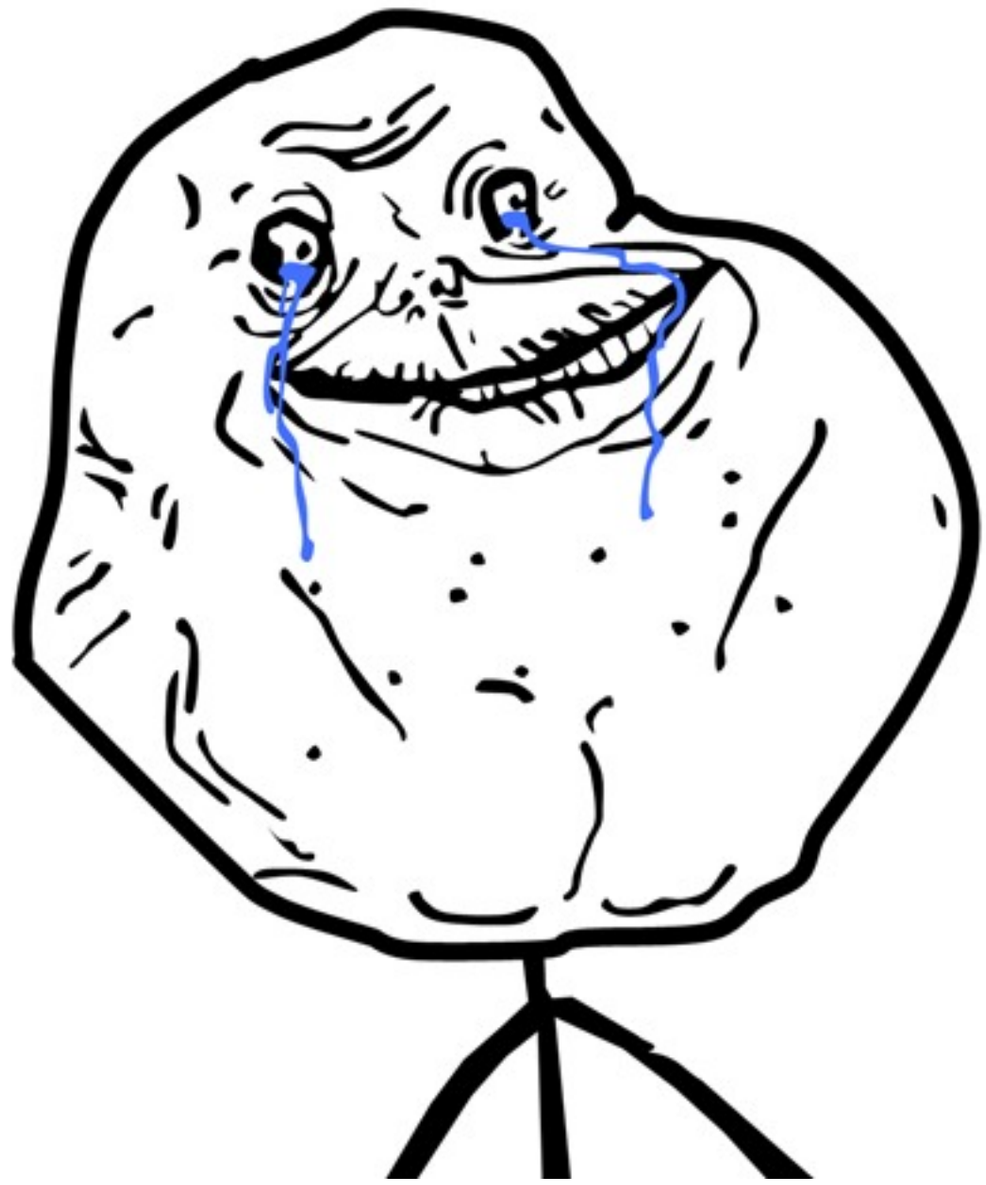
https://github.com/lucastorri/SuperMarket

Falando de Java...

Plataforma



Linguagem




JRUBY


Jython


Scala


Groovy


Clojure


Scala

```
object HelloWorld {
  def main(args: Array[String]) {
    println("Hello World")
  }
}
```

```
var someVariable = 3
someVariable = 7
```

```
val someValue = 13
someValue = 17 //error: reassignment to val
```

```
// def to define functions/methods
def isTrue(b: Boolean) = b
// implicit return type
```

```
def sayHello(msg: String = "Hi") =
    println(msg)

sayHello()
sayHello("Hello")
sayHello(msg = "Oi")
```

```
// [] instead of <>
val emptyList = List[Int]()
```

```scala
// Scala's == != Java's ==
val p1 = new Point(1, 2)
val p2 = new Point(1, 2)
p1 == p2 // true
p1 eq p2 // false
```

```
// no checked exceptions
def mayFail = throw new Exception
```

```
val mlstr =
"""
multi
line
"""
val tuple = (1, "tuple")
```

```scala
var list = List(1, 2, 3)
var sameList = 1 :: 2 :: 3 :: Nil

list(0) // 1st element

var map = Map("a" -> 1, "b" -> 2, "c" -> 3)
```

```scala
var array = Array(1, 2, 3)

var set = Set(1, 2, 3)

// and also Vector, Queue,
// Stack, LinkedList, and many more
```

```
def sayHello(g: { def hello: String }) =
  println(g.hello)

object Duck { def hello = "quack!" }
object Cat { def hello = "miau!" }

sayHello(Frog)
sayHello(Duck)
```

```scala
val xml =
<numbers>
  { for (n <- 1 to 5) yield <n>{n}</n> }
</numbers>
/**
 * <numbers>
 *   <n>1</n><n>2</n><n>3</n><n>4</n><n>5</n>
 * </numbers>
 */
```

```
val numbers = 1 to 6

numbers.foreach(println)
```

```
number.filter(e => e % 2 == 0)
numbers.filter(_ % 2 == 0)
```

```
// mixins
trait Flyer { def fly = "I'm flying" }
trait Quacker { def quack = "quack!" }

class Duck extends Flyer with Quacker
val d = new Duck
d.fly; d.quack
```

```
// single instance of a type (singleton)
object HolyGrail
var g1 = HolyGrail; var g2 = HolyGrail
g1 eq g2 // true
```

```
// generated equals, toString, hashCode, ...
case class Point(x: Int, y: Int)
```

```
class IntPlus(i: Int) {
  def weeks = i * 7
  def ago: Date = {
    val cal = Calendar.getInstance
    cal.add(Calendar.DATE, -i)
    cal.getTime
  }
}
implicit def intPlus(i: Int) = new IntPlus(i)

2.weeks.ago
// new IntPlus(new IntPlus(2).weeks).ago
// Date = Fri Jun 10 13:14:05 BRT 2011
```

```
object A {
  def methodA = 7
}
```

```
val methodA = A.methodA _
methodA() // 7
```

Testando com Scala

- ScalaCheck
- ScalaTest
- Specs
- Specs2
- Specsy

- ScalaCheck
- ScalaTest
- Specs
- Specs2
- Specsy

Demo

- Sistema de ponto de venda
  - Produtos
    - Impostos
  - Carrinho compra
  - Clientes
    - Cartão de Crédito
  - Caixa

- Sistema de ponto de venda
  - Produtos
    - Impostos
  - Carrinho compra
- Clientes
  - Cartão de Crédito
- Caixa

- A Checkout Counter
  - knows that an empty cart has price zero
  - sums the total price of the items in the cart

- A Customer
  - gets a new shopping cart from the supermarket
  - shops following a shopping list
  - gets a counter to checkout and pay
  - pays bills with credit card