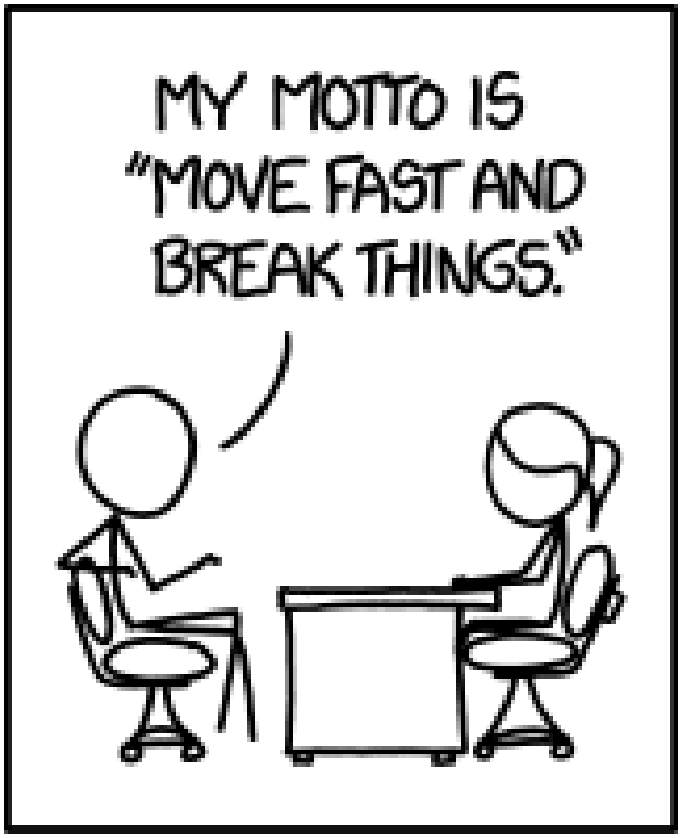
# Move Deliberately
# And Don’t Break Anything

Brian Goetz (@briangoetz)

Java Language Architect
Oracle Corp


"Move fast and
break things.
Unless you
are breaking stuff
you are not moving
fast enough."




JOBS I'VE BEEN FIRED FROM

FEDEX DRIVER
CRANE OPERATOR
SURGEON
AIR TRAFFIC CONTROLLER
PHARMACIST
MUSEUM CURATOR
WAITER
DOG WALKER
OIL TANKER CAPTAIN
VIOLINIST
MARS ROVER DRIVER
MASSAGE THERAPIST

**Creative destruction** (*schöpferische Zerstörung)*: the incessant product and process innovation mechanism by which new production units replace outdated ones.

(Joseph Schumpeter, 1942)

# MOVE FAST WITH STABLE INFRA


When you ain't got nothing
you got nothing to lose

# Programming is an *economic* activity

**Inputs**

- Programmer time
- Pizza

**Outputs**

- Working code (hopefully)
- Technical debt

# Programming is an *economic* activity

*In a profession where we carry out decade-spanning holy wars over tab widths and capitalization, it's no surprise that people get attached to their development and release habits.*

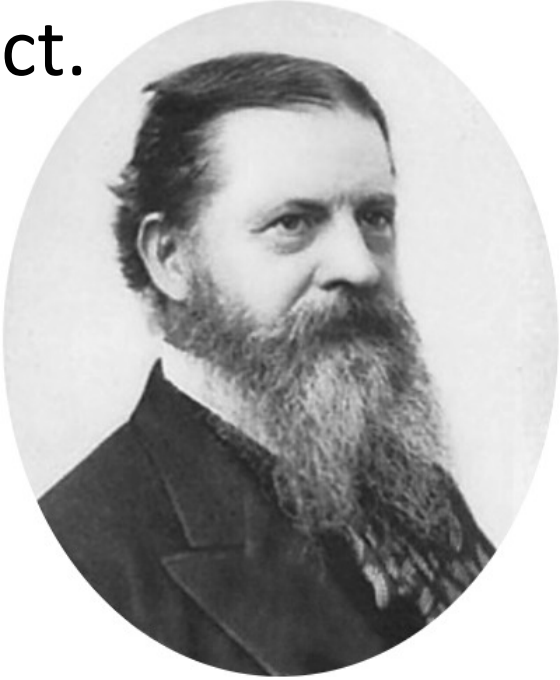
*But if shipping so much software has taught me one thing, it's to be an agnostic. Different methodologies optimize for different goals, and all of them have downsides. If you maximize for schedule predictability, you'll lose on engineer productivity …*

# Programming is an *economic* activity

***My fellow engineers, please stop asking "Is this process good or bad?" and start asking "Is it well-suited to my situation?"***

**Pragmatic maxim**: Consider what effects, that might conceivably have practical bearings, we conceive the object of our conception to have. Then, our conception of these effects is the whole of our conception of the object.

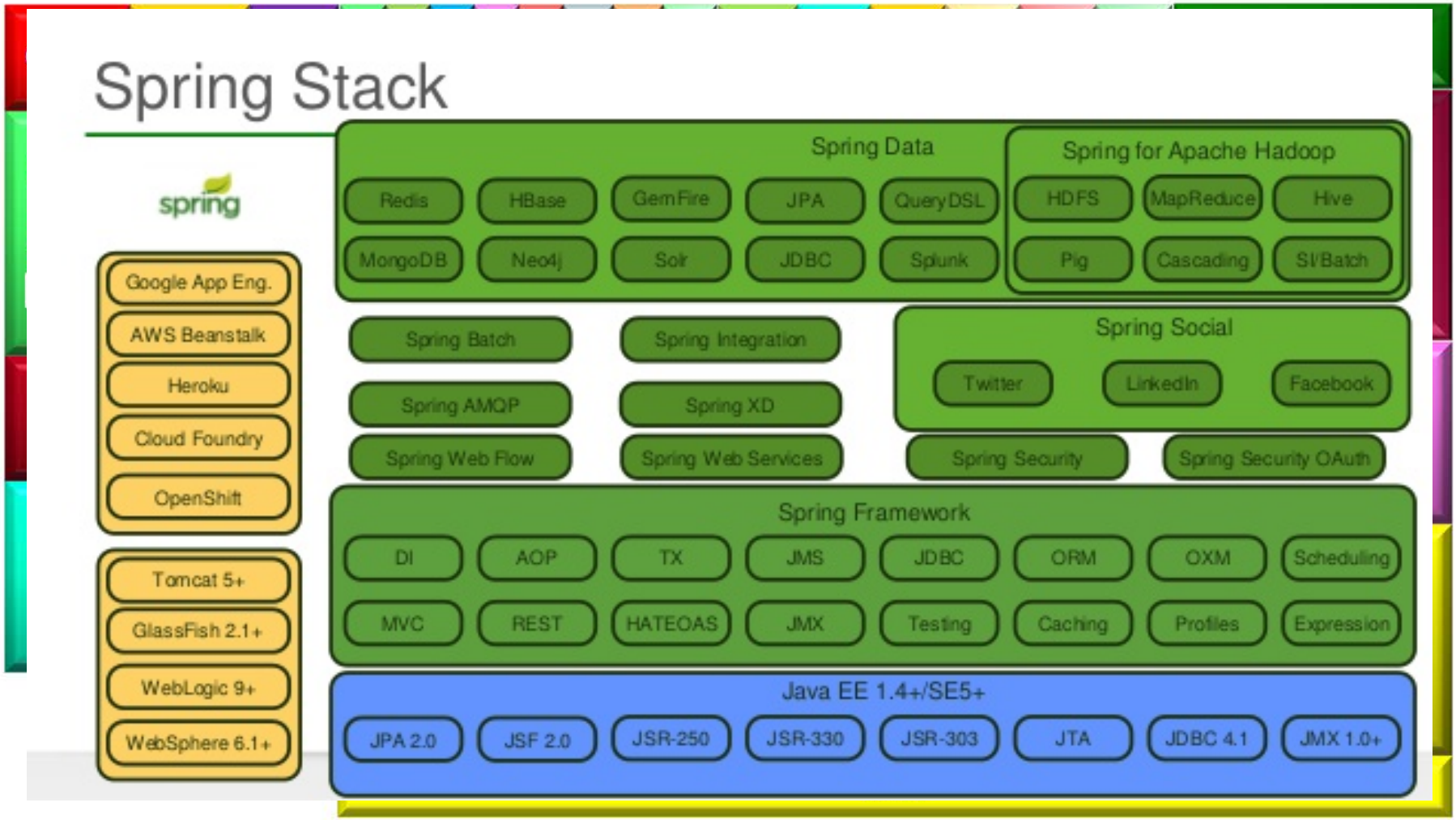
(C. S. Peirce, 1878)



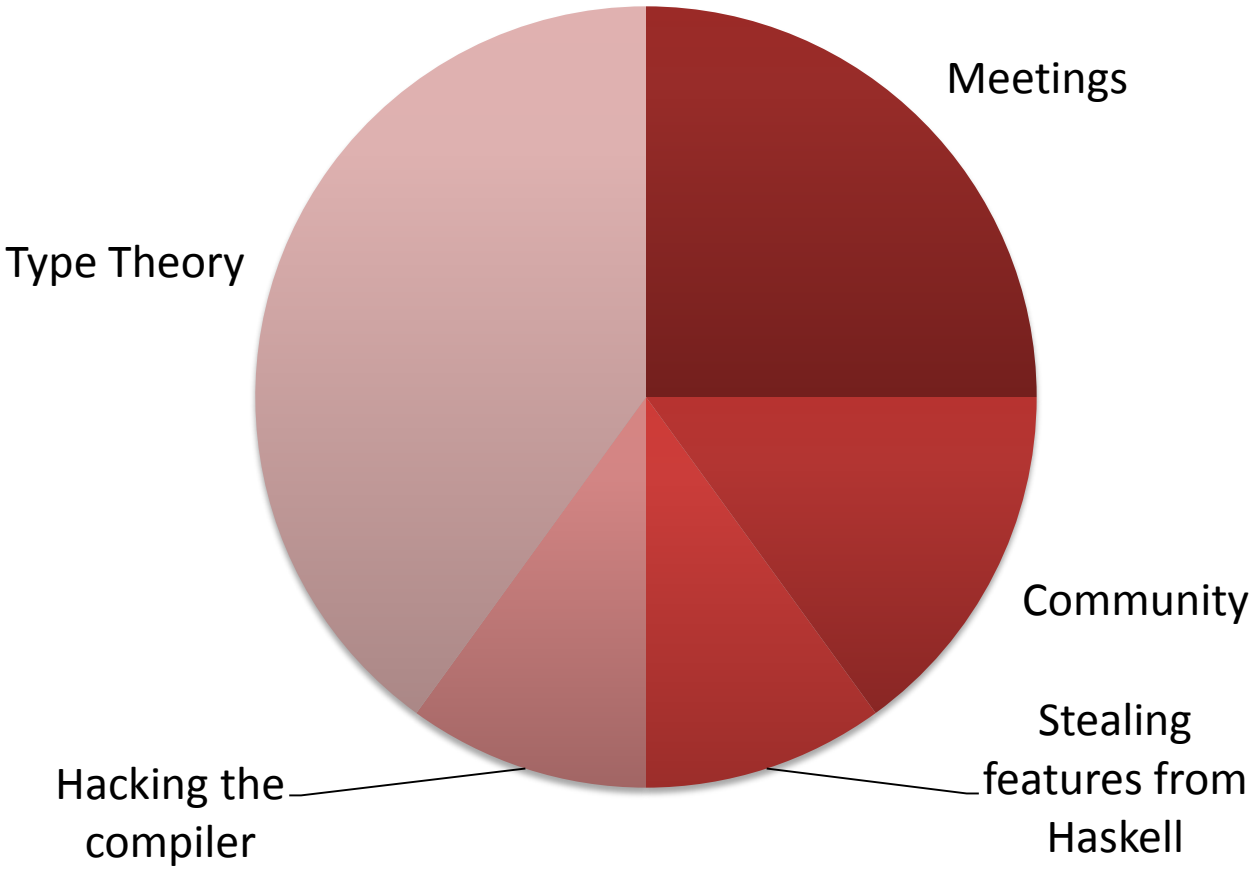
**Proto-pragmatic maxim**: There is no *good*, there is only *good for.*

Master Yoda
(OK, he didn’t really say this)

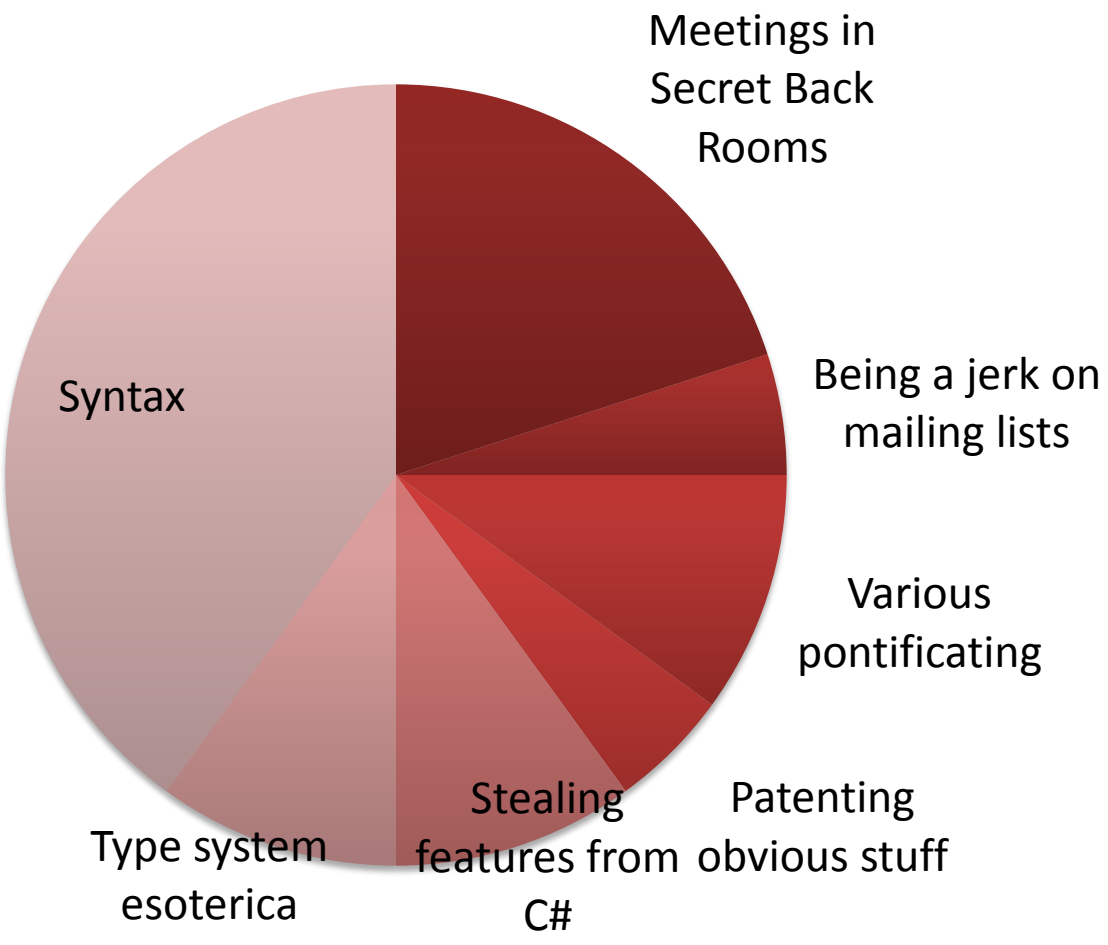
# Engineering down-stack



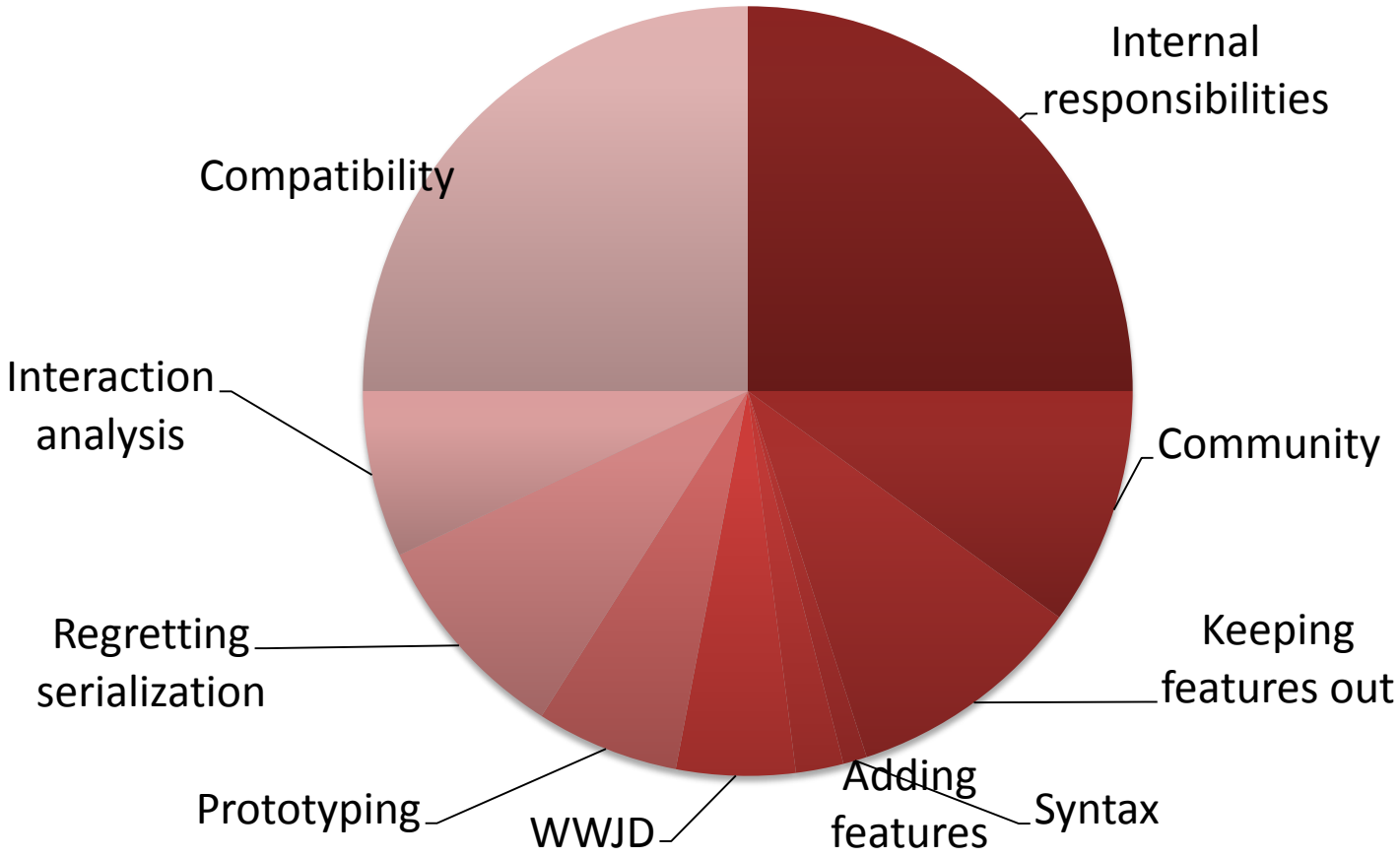
# What people think I do (academic version)



# What people think I do (naïve version)



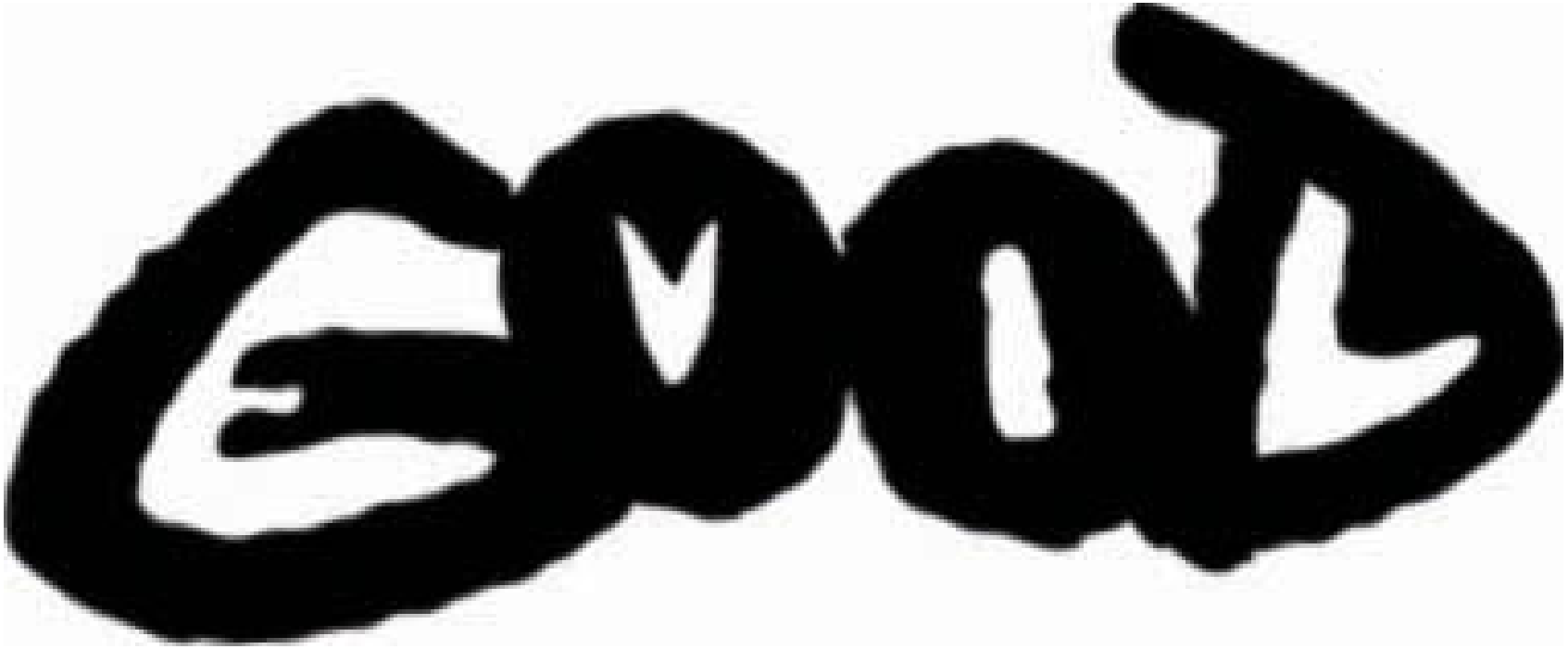
# What I think I do



# Compatibility

| | | |
|---|---|---|
| Binary Compatibility | Source Compatibility | Behavioral Compatibility |
| Performance Model Compatibility | Serialization Compatibility | Migration Compatibility |
| Representational Compatibility | Security Compatibility | Forward Compatibility |

# Seeing The Big Picture



# Suit Up

# Embrace and Extend

```
interface I {
    void m();
}

class C {
    void m() { … }
}

class D extends C implements I { }
```

```
interface I {
    default void m() { … }
}
```

# Embrace and Extend

Default method conflict resolution rules

- Rule 1 – prefer a method from a superclass over a superinterface
- Rule 2 – if I extends J, prefer a method from I over J
- Rule 3 – No rule 3!

# It Takes as Much Time As It Takes

"When you start looking at a problem and it seems really simple with all these simple solutions, you don't really understand the complexity of the problem. And your solutions are way too oversimplified, and they don't work. Then you get into the problem, and you see it's really complicated. And you come up with all these convoluted solutions. That's sort of the middle, and that's where most people stop, and the solutions tend to work for a while. But the really great person will keep on going and find the key, underlying principle of the problem. And come up with a beautiful elegant solution that works."

**Steve Jobs**

from Insanely Great: The Life and Times of Macintosh, the Computer That Changed Everything (1994) by Steven Levy.

# Don’t Punt On The Hard Cases

“When you start looking at a problem and it seems really simple with all these simple solutions, you don’t really understand the complexity of the problem. And your solutions are way too oversimplified, and they don’t work. Then you get into the problem, and you see it’s really complicated. And you come up with all these convoluted solutions. That’s sort of the middle, and that’s where most people stop, and the solutions tend to work for a while. But the really great person will keep on going and find the key, underlying principle of the problem. And come up with a beautiful elegant solution that works.”

**Steve Jobs**

from Insanely Great: The Life and Times of Macintosh, the Computer That Changed Everything (1994) by Steven Levy.

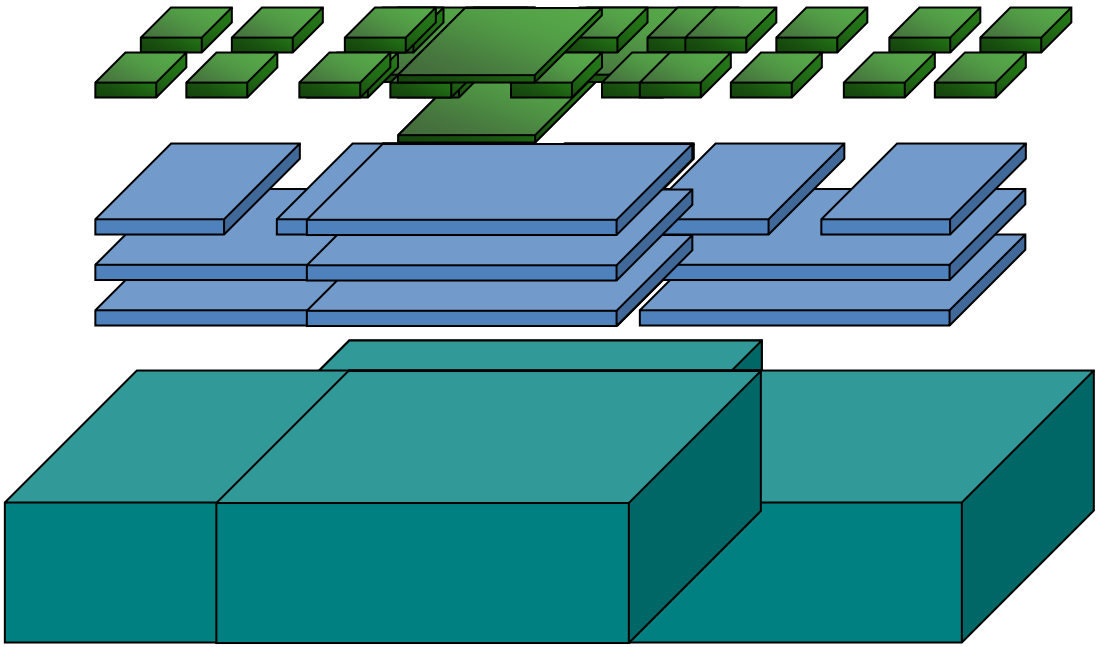
# Beware Foolish Consistency

"A foolish consistency is the hobgoblin of little minds, adored by little statesmen and philosophers and divines."

(Ralph Waldo Emerson, 1841)

# A Look Ahead

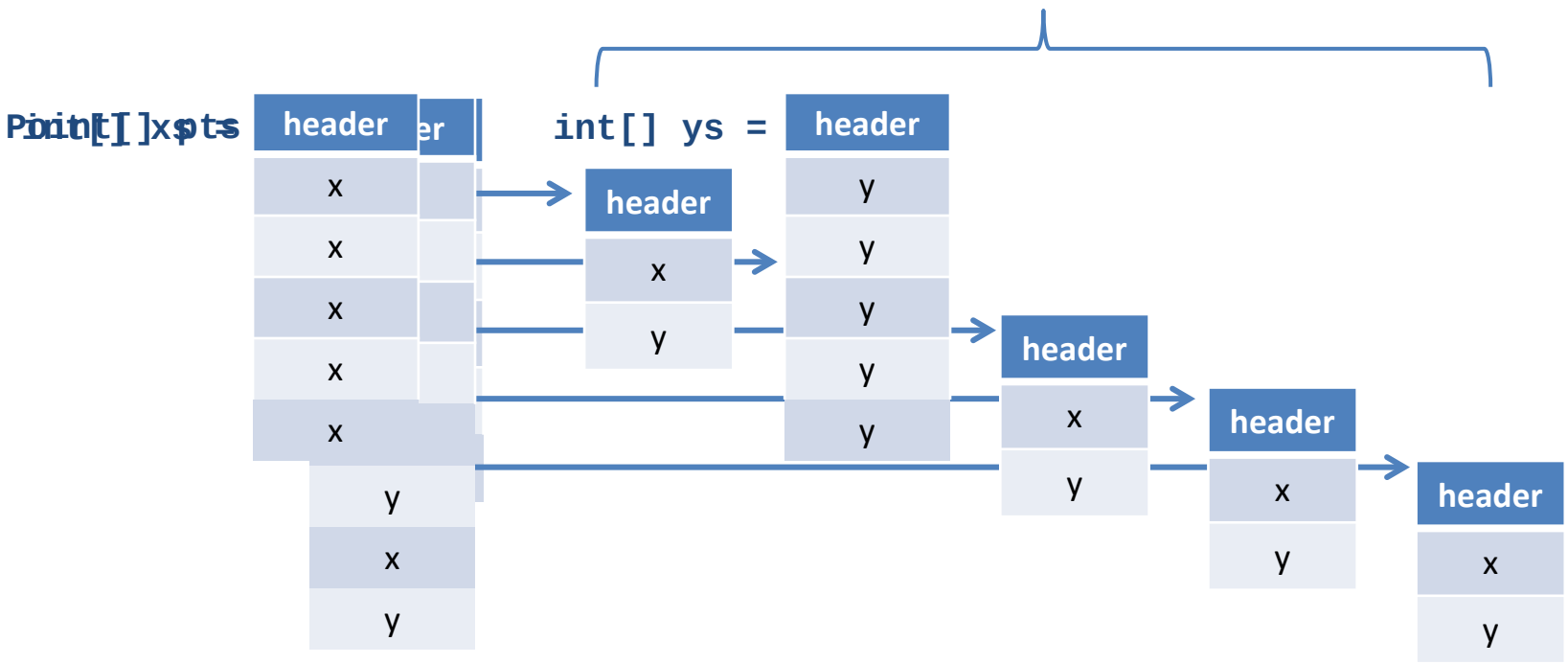
# Hardware Evolution



# Data Layout

```
final class Point {
    final int x;
    final int y;
}
```

Layout of these in memory is effectively random after GC!



# Value Types

```
value class Point {
    int x;
    int y;
}
```

```
class Rectangle {
    Point lowerLeft;
    Point upperRight;
}
```

Codes like a class, behaves like an int!

Point[] pts =

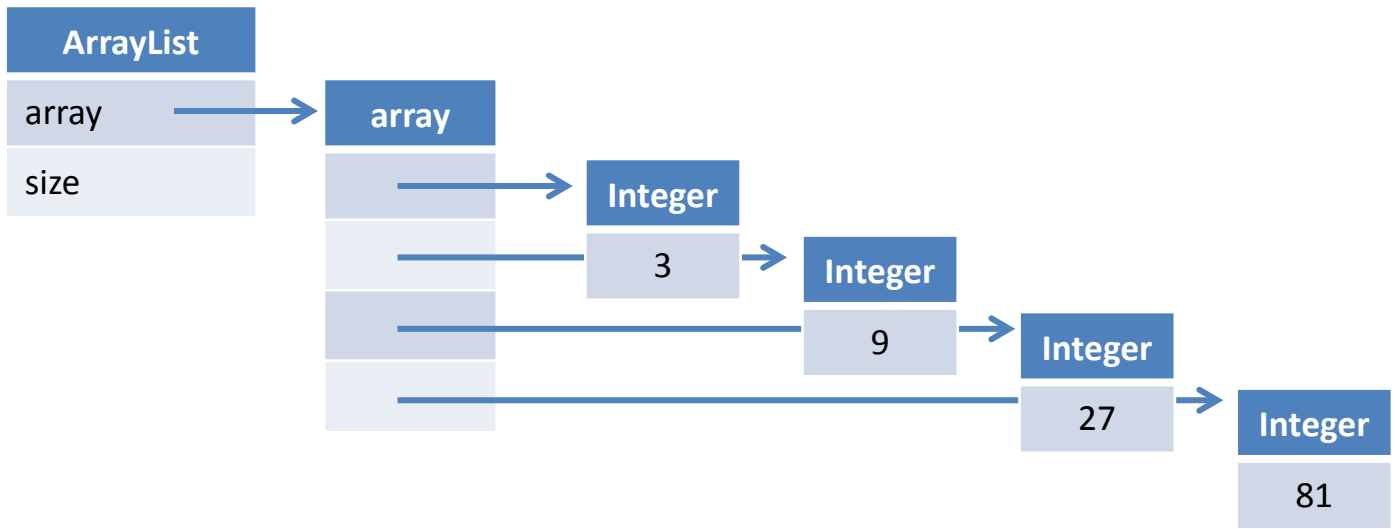
| header |
|---|
| x |
| y |
| x |
| y |
| x |
| y |
| x |
| y |

Rectangle r =

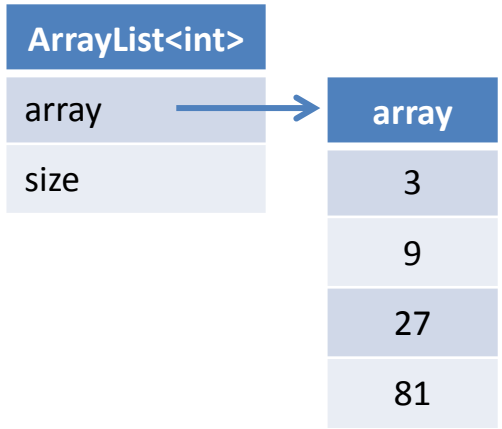
| Rectangle |
|---|
| lowerLeft.x |
| lowerLeft.y |
| upperRight.x |
| upperRight.y |

# Values and Generics

```
ArrayList<Integer>
    ints =
```




```
ArrayList<int>
    ints =
```



# Thank You

Brian Goetz (@briangoetz)

Java Language Architect
Oracle Corp