EXPLORE DESIGN PERFECTION

Functional programming in Java

Carlos Kavka

ESTECO SpA

esteco.com

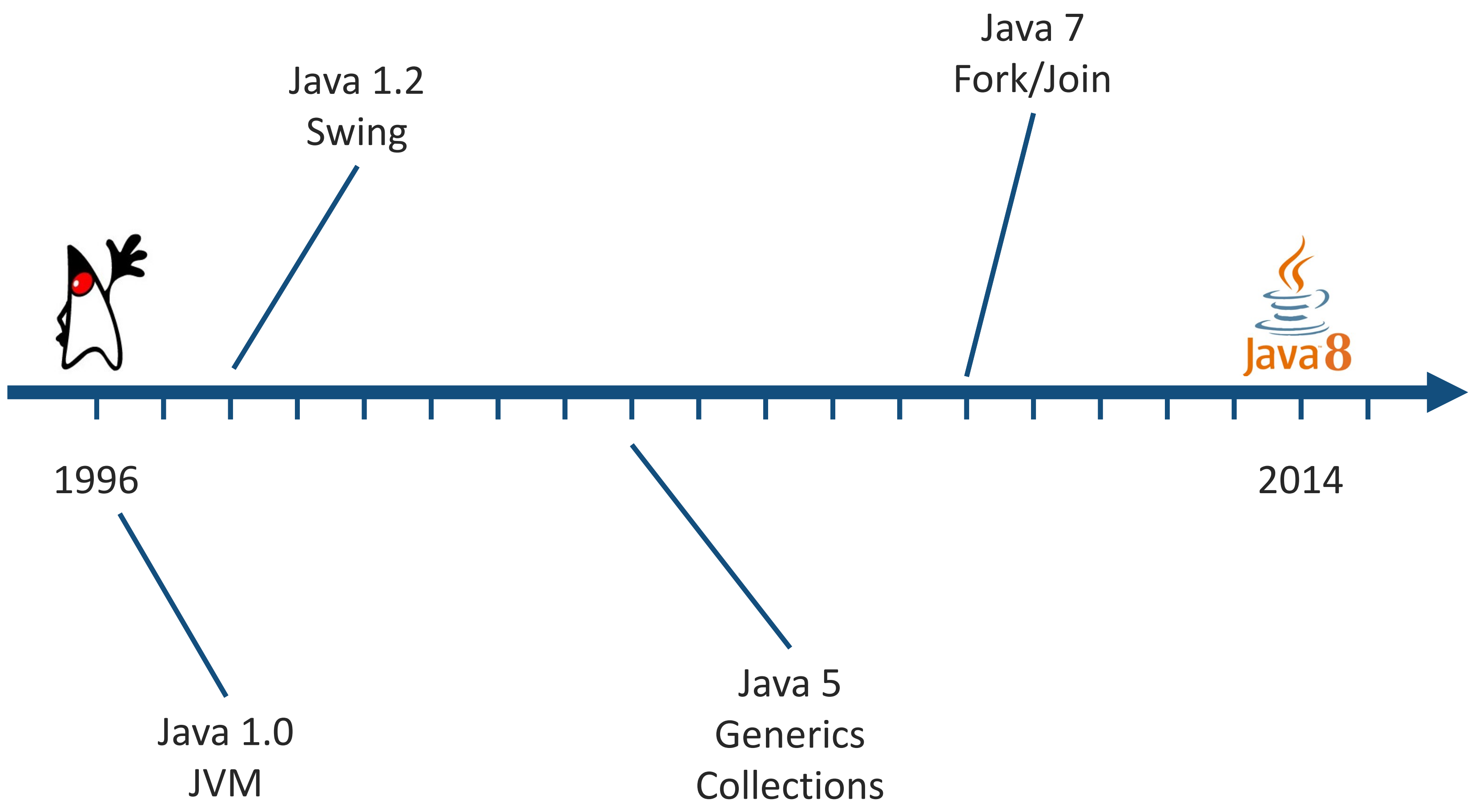
EXPLORE DESIGN PERFECTION

Functional programming in Java

Part I – Introduction

esteco.com

Java evolution



Alignment with language trends!

C#

C++

JavaScript

Ruby

Haskell

Microsoft Visual F#

Scala

Java ecosystem

Groovy

Clojure

Improvements in Java 8

Functional style of programming

Collection enhancements

Stream processing

Optional values

Lambda expressions

Method references

Default methods

Change the way of thinking!

Imperative programming

x++

style of programming modeled as a sequence of commands that modify state

Functional programming

f(g(x))

programs are expressions and transformations, modeling mathematical formulas

Change the way of thinking!

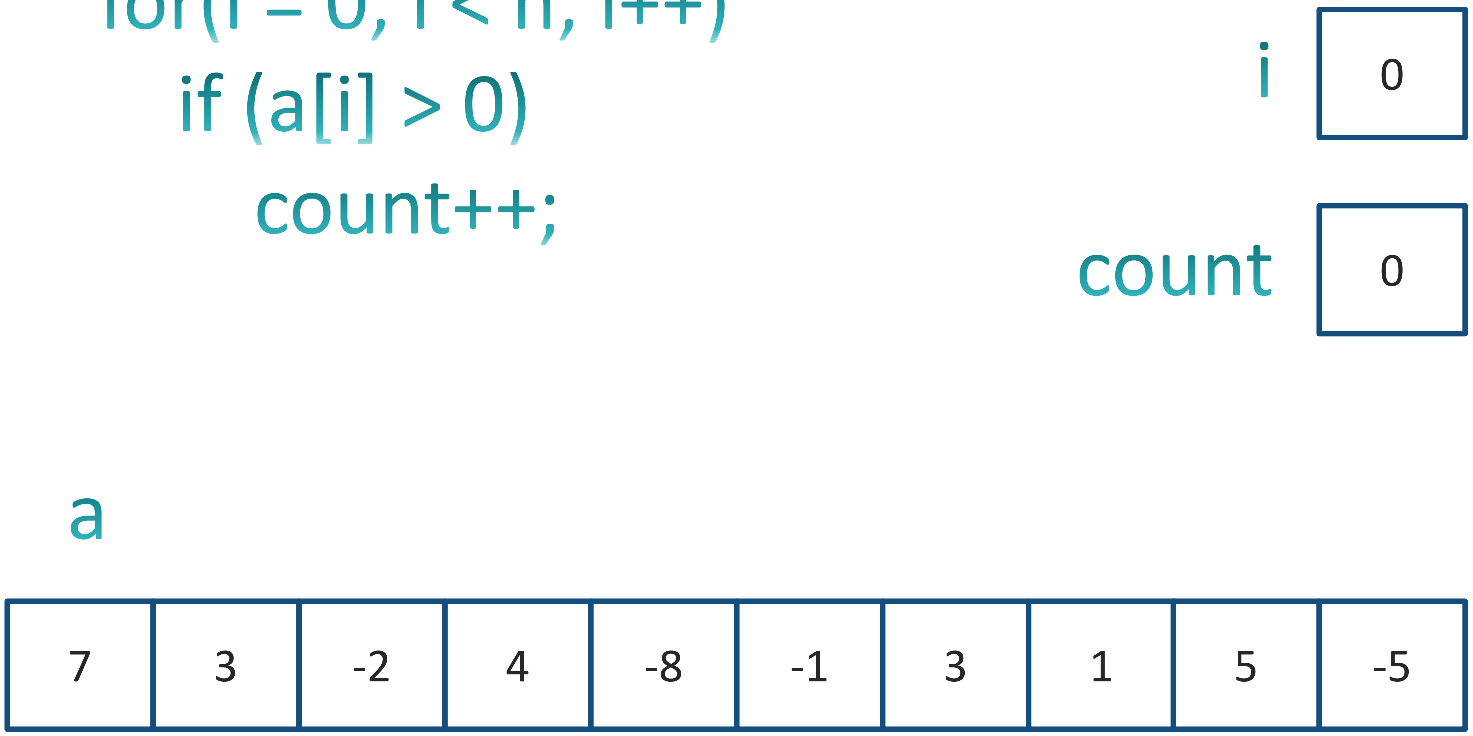
```
count = 0;
for(i = 0; i < n; i++)
    if (a[i] > 0)
        count++;
```

$/+ \circ \alpha(> \circ [id, \bar{0}] \rightarrow \bar{1} ; \bar{0})$

programming means tell —declaratively—*what* we want rather than *how* to do it.

Imperative approach

```
count = 0;
for(i = 0; i < n; i++)
    if (a[i] > 0)
        count++;
```



Functional approach

$$/+ \circ \alpha(> \circ [id, \bar{0}] \rightarrow \bar{1}\ ; \bar{0})$$

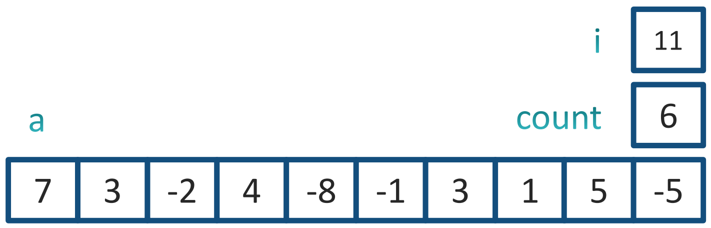
7	3	-2	4	-8	-1	3	1	5	-5

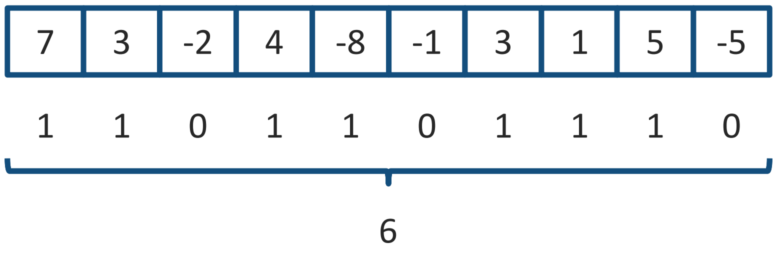
$$\underbrace{1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0}_{6}$$

What do you think of...?

```
count = 0;
for(i = 0; i < n; i++)
   if (a[i] > 0)
      count++;
```

$/+ \circ \alpha(> \circ [\mathrm{id}, \bar{0}] \rightarrow \bar{1} ; \bar{0})$





parallelism

mutable objects

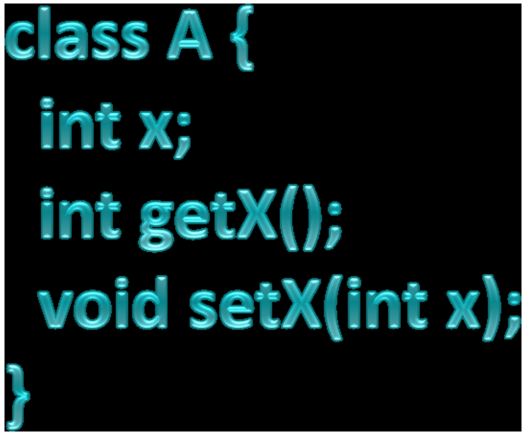
what do you think?

different approach: what vs. how

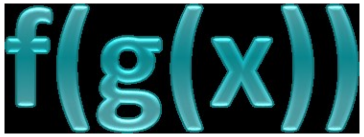
what happen if we call twice a function?

esteco.com

What about Object Oriented Programming?



```
class A {
  int x;
  int getX();
  void setX(int x);
}
```



$f(g(x))$

abstracting over
data

abstracting over
behavior

Functional programming

Is it new?

1930 - Lambda Calculus (A. Church)
1958 - Lisp (J. McCarthy)
...
1977 - FP (J. Backus)
...

What about Java 8 implementation?

- no monads
- reduced lazy evaluation
- little support for immutability

...

better than nothing!

Benefits

- Simpler, cleaner, and easier-to-read code
- Simpler maintenance
- Great for collections!
- Enhanced parallelism/concurrency for multi-core CPUs

Thank you for your attention!

ESTECO

EXPLORE DESIGN PERFECTION