

Programación funcional con Java 8

Víctor Orozco

Nabenik

26 de noviembre de 2016

Java 8

Víctor Orozco

Intro

Java 8

Bloques

Streams

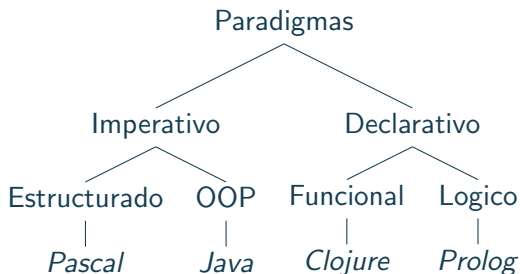
Fin



<https://www.oracle.com/java8>
<https://www.oracle.com/java8launch>

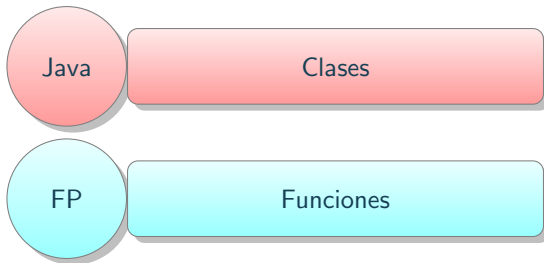
- Nashorn
- Date/Time API
- Compact Profiles
- Type Annotations
- **Default methods**
- **Streams**
- **Lambda Expressions**



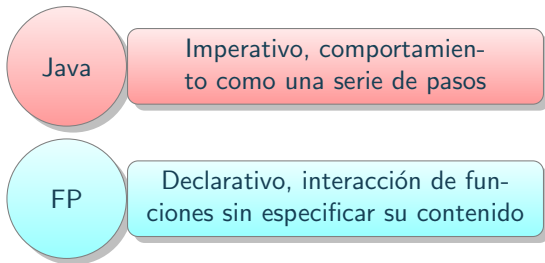


- Computación = Evaluación de funciones matemáticas (calculo de lambdas)
- NO cambios en estado
- NO mutar datos
- Declarativo → Expresiones

Java vs. Funcional (organización)



Java vs. Funcional (algoritmos)



Java vs. Funcional (Mutabilidad y estado)

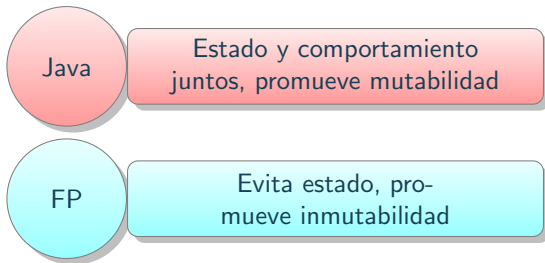
Intro

Java 8

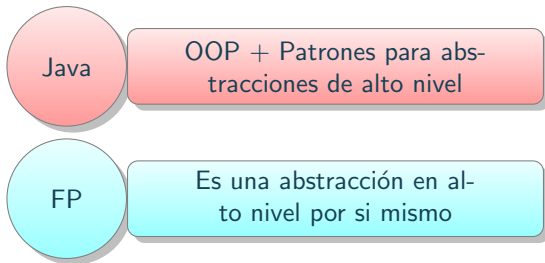
Bloques

Streams

Fin



Java vs. Funcional (Estilo)



Java vs. Funcional (Concurrencia)

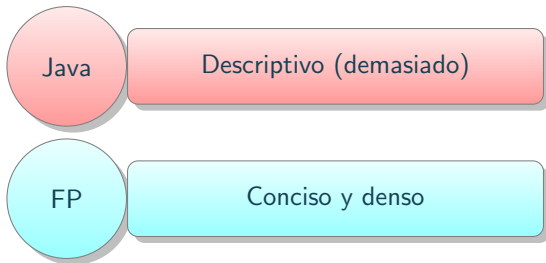
Java

Concurrencia basica con
locks y recursos compartidos

FP

Workflows paralelos sin es-
tado compartido (no locks!)

Java vs. Funcional (Código)



Un lenguaje de programación orientada a objetos con azúcares sintácticos funcionales.



¿Porque programación funcional?

- Paralelismo
- Multicore, multicpu
- Elegancia

- Java no es un lenguaje funcional puro (Clojure)
- Otras opciones JVM (Scala, Kotlin, Ceylon)
- Java soporta programación funcional a través de bibliotecas

Bloques

- Interfaces funcionales
- Referencia a funciones
- Lambdas
- Funciones predefinidas en Java 8 (`java.util.function`)
- Streams API

- Solo un método abstracto
- Interfaces ahora permiten default methods

```
@FunctionalInterface
public interface Runnable
{
    public abstract void Run();
}
```

- Permiten utilizar una función dentro de una expresión lambda
- Permiten invocar métodos existentes

- Función anónima sin asociar a un identificador
- Usadas para pasar comportamiento a funciones high-order
- Usadas para construir el resultado de una función high-order que necesita retornar una función

(parametros) → comportamiento

```
(Integer i) -> {System.out.println(i);};
```

```
i -> System.out.println(i);
```

```
i -> i*2;
```

Streams API

- Java soporta programación funcional con bibliotecas
- Más de 40 interfaces funcionales en Java 8
- Raramente se deben crear nuevas
- Streams API

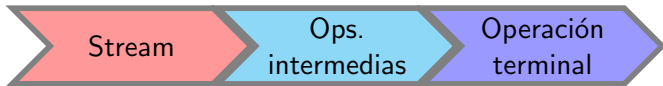

```
List<String> jedis =  
    Arrays.asList("ObiWan", "Anakin", "Yoda");  
  
jedis.forEach(s -> System.out.println(s));
```

```
List<String> jedis =  
    Arrays.asList("ObiWan", "Anakin", "Yoda");  
  
jedis.forEach(System.out::println);
```

Streams API - Operaciones con predicados

```
List<String> jedis =  
    Arrays.asList("ObiWan", "Anakin", "Yoda");  
  
Predicate<String> darthRemover =  
    s -> "Anakin".equals(s);  
  
jedis.removeIf(darthRemover);
```

- Map-Reduce
- Monads = Serie de pasos / funciones anidadas



```
winners
    .stream() //Stream
    .filter( //Intermedio +- MAP
        p -> p.getOrderId()
            .equals(winner.getOrderId())
    )
    .findFirst(); //Reduce
```

```
winners
    .stream() //Stream
    .filter( //Intermedio +- MAP
        p -> p.getOrderId()
            .equals(winner.getOrderId())
    )
    .collect(Collectors.toList()); //Reduce
```

<http://github.com/tuxtor/fpjavademo>

- Divertido
- Declarativo
- Menos código, código más legible

- Performance - invokedynamic (debatible)
- Debug
- Flexibilidad

- JDK 8 Lamdas&Streams MOOC -
<https://www.youtube.com/playlist?list=PLMod1hYilvSZL1xclvHcsV2dMiminf1>
- Functional Programming in Java: Harnessing the Power Of Java 8 Lambda Expressions - <http://www.amazon.com/Functional-Programming-Java-Harnessing-Expressions/dp/1937785467>

- jOOL
- JavaSlang
- Eclipse Collections
- Vert.x
- RxJava
- LightBend

- me@vorozco.com
- <http://vorozco.com>
- <http://github.com/tuxtor/slides>



This work is licensed under a Creative Commons
Attribution-ShareAlike 3.0 Guatemala License.