Java 8: Más funcional que nunca

Víctor Orozco

Nabenik

November 14, 2015

JAVA 2015 DAY Guatemala

Víctor Orozco

- Developer (JVM/Open Source Advocate)
- Ex-JUG Leader
- Consultor independiente (Nabenik)
- @tuxtor
- The J*

Java 8

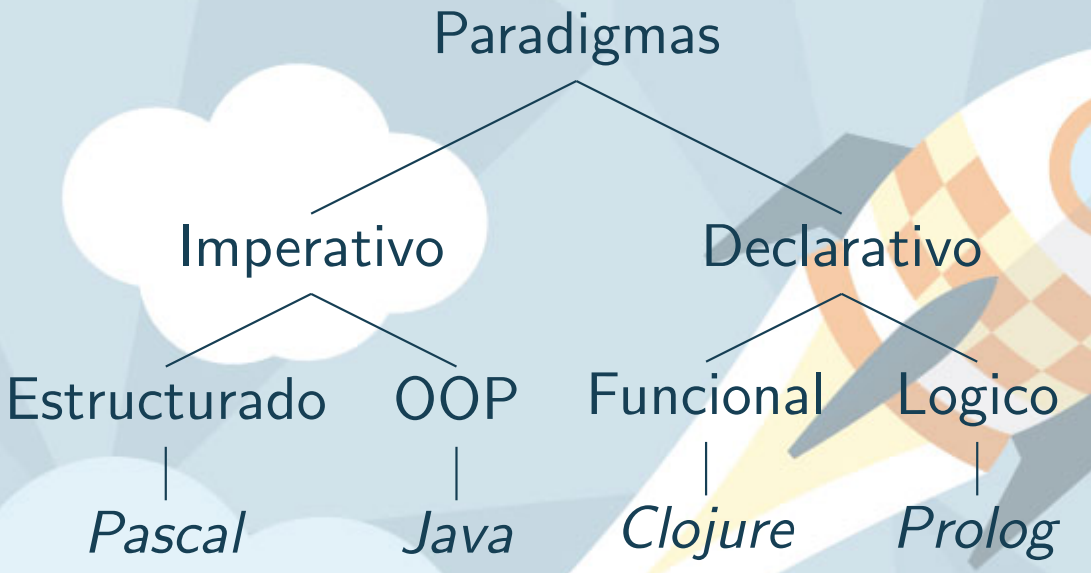
Java 8

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

Java 8

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

Paradigmas (Simplificación)



Programación funcional

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

Java vs. Funcional (organización)

Java vs. Funcional (algoritmos)

Java	Imperativo, comportamiento como una serie de pasos
FP	Declarativo, interacción de funciones sin especificar su contenido

Java vs. Funcional (Mutabilidad y estado)

Java	Estado y comportamiento juntos, promueve mutabilidad
FP	Evita estado, promueve inmutabilidad

Java vs. Funcional (Estilo)

Java	OOP + Patrones para abstracciones de alto nivel
FP	Es una abstracción en alto nivel por si mismo

Java vs. Funcional (Concurrencia)

Java	Concurrencia basica con locks y recursos compartidos
FP	Workflows paralelos sin estado compartido (no locks!)

Java vs. Funcional (Código)

Java	Descriptivo (demasiado)
FP	Conciso y denso

Java 8

Un lenguaje de programación orientada a objetos con azucares sintácticas funcionales.

¿Porque programación funcional?

- ¡Paralelismo!
- Multicore, multicpu
- Elegancia

Programación funcional en Java 8

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

Bloques

Bloques funcionales en Java 8

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

Interfaces funcionales

- Solo un método abstracto
- Interfaces ahora permiten default methods

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

Referencias a funciones

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

Expresion lambda

- 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

Expresion lambda (anatomia)

(parametros) $\rightarrow$ comportamiento

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

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

i -> i*2;
```

Streams API

Funciones predefinidas

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

Streams API - Listas

```
List<String> jedis =
        Arrays.asList("ObiWan", "Anakin", "Yoda");

jedis.forEach(s -> System.out.println(s);
```

Streams API - Listas

```
List<String> jedis =
        Arrays.asList("ObiWan", "Anakin", "Yoda");

jedis.forEach(System.out::println);
```

Streams API - Operaciones con predicados

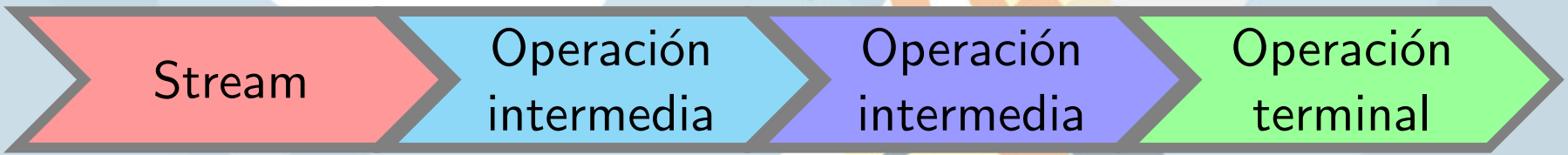
```
List<String> jedis =
        Arrays.asList("ObiWan", "Anakin", "Yoda");

Predicate<String> darthRemover =
        s -> "Anakin".equals(s);

jedis.removeIf(darthRemover);
```

Streams API

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



Streams API - Map-reduce

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

Streams API - Map-reduce

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

Ejemplo

http://github.com/tuxtor/fpjavademo

Programación funcional - Bueno

- Divertido
- Declarativo
- Menos código, código más legible

Programación funcional - Malo

- Performance - invokedinamic (debatible)
- Debug
- Flexibilidad

Lecturas recomendadas

- JDK 8 Lamdas & Streams MOOC - https://www.youtube.com/playlist?list=PLMod1hYiIvSZL1xclvHcsV
- Functional Programming in Java: Harnessing the Power Of Java 8 Lambda Expressions - http://www.amazon.com/Functional-Programming-Java-Harnessing-Expressions/dp/1937785467

Gracias

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

JAVA DAY 2015 Guatemala