



Java 8 en action (2)

Denis Simon Soneira
Aurea Muñoz Hernandez



Qui sommes nous

AUREAMUNIOZ@GMAIL.COM



DENIS.SIMON@GMAIL.COM



RECAPITULATIF JOUR 1

PROGRAMMATION FONCTIONELLE ET EXPRESSIONS LAMBDA

A solid orange horizontal bar at the bottom of the slide.

Programmation fonctionnelle

```
def sum(xs: List[int]): int = xs match
{
    case Nil => 0
    case head ::
        tail => head + sum(tail)
}
```

1. Pas de mutation
2. La récursion est omniprésente
3. Description du résultat.
4. La fonction est un élément à part entier du langage.

Functional Interfaces

```
@FunctionalInterface  
  
public interface Function<T, R> {  
    R apply(T var1);  
}
```

Expression LAMBDA

`(parameter) -> expression;`

`(Person person) -> person.getAge();`

`person -> person.getAge();`

Référence à la méthode :

`person::getAge // méthode d'instance`

`Person::getAge // méthode statique`


TYPES DE REFERENCE METHODS

Kind	Example
Reference to a static method	<code>ContainingClass::staticMethodName</code>
Reference to an instance method of a particular object	<code>containingObject::instanceMethodName</code>
Reference to an instance method of an arbitrary object of a particular type	<code>ContainingType::methodName</code>
Reference to a constructor	<code>ClassName::new</code>

<https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html>

JAVA 8

```
public long moreThanTwentyYearsOld(List<Person> persons) {  
    return persons.stream()  
        .filter(p -> p.getAge() > 20)  
        .count();  
}
```



Elle sort d'où cette
méthode?

LES STREAMS

UNE NOUVELLE FAÇON DE GÉRER LES COLLECTIONS

A solid orange horizontal bar spanning the width of the slide at the bottom.

L'interface Collection

```
Stream<Person> stream = persons.stream();
```

```
public interface Collection<E> extends Iterable<E> {  
    Stream<E> stream();  
}
```



⇒ Toutes les implémentations DOIVENT REIMPLEMENTER la méthode STREAM.

« Default Methods »

```
Stream<Person> stream = persons.stream();
```

```
public interface Collection<E> extends Iterable<E>{  
    default Stream<E> stream(){  
        return StreamSupport.stream(spliterator(), false);  
    }  
}
```

⇒ Les « Default Methods » peuvent être écrites sur les interfaces.

Et si..

```
interface B : default a() {...}  
interface C : default a() {...}  
class A implements B, C
```

Dans une méthode de la classe A on appel `this.a()` ?

1) `B.a()` GAGNE

2) `C.a()` GAGNE

3) Erreur de compilation



Et si..

```
interface B : default a() {...}  
interface C extends B: default a() {...}  
class A implements B :
```

Dans une méthode de la classe A on appel `this.a()` ?

1) `B.a()` GAGNE

2) `C.a()` GAGNE



3) Erreur de compilation

Et si..

```
interface C : default a() {...}  
interface B implements C : default a() {...}  
class A implements B : default a() {...}
```

Dans une méthode de la classe A on appel `this.a()` ?

1) `A.a()` GAGNE



2) `B.a()` GAGNE

3) `C.a()` GAGNE

4) Erreur de compilation

JAVA 8

```
public long moreThanTwentyYearsOld(List<Person> persons) {  
    return persons.stream()  
        .filter(p -> p.getAge() > 20)  
        .count();  
}
```


Stream

C'est une séquence d'éléments qui supporte opérations séquentiels et parallèles.

Différences

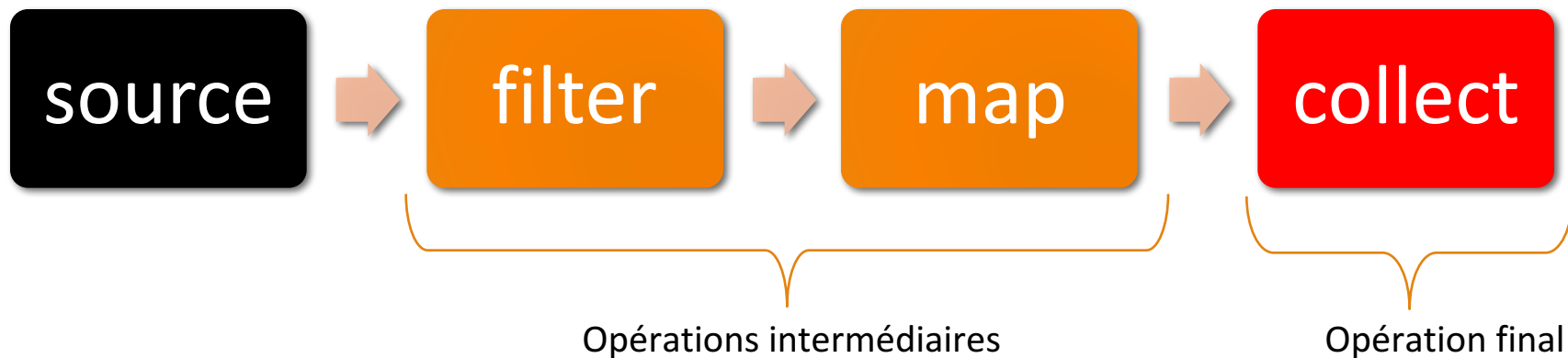
COLLECTIONS

1. Stockage des données
2. Modification de données.

STREAMS

1. Transfère de données.
2. Transformation de données.
3. Evaluation Lazy.
4. N'est pas borné
5. N'est pas réutilisable.

Stream Pipeline



Example

```
persons.stream() // Création d'un stream
```

```
    // Operations intermediaires  
    .filter(Person::isMarried) // filtrage  
    .mapToInt(Person::getAge)  
    .sorted()  
    .limit(10)
```


```
    // Operation terminal  
    .average() ;
```

Optional (Meilleur alternative que NULL)

Optional<Car>



`.isPresent() = true`

`.getValue() =` 

Optional<Car>



`.isPresent() = false`

`.getValue() = NoSuchElementException`

`.orElse(` `) =` 

Prérequis pour les exercices :

1. JDK 8 <http://www.oracle.com/technetwork/java/javase/downloads>
2. GIT <https://git-scm.com/downloads>
3. Maven 3 <https://maven.apache.org>
4. IDE JAVA (Eclipse, IntelliJ, ...)
5. Clone le repository GIT <https://github.com/2nis6mon/dojo-java8.git>

Exercice 3, 4 et 5

Suivre les instructions sur les classes avec des TODO

Consultation de l'api:

<https://docs.oracle.com/javase/8/docs/api/>