Java 8 en action

Denis Simon Soneira
Aurea Muñoz Hernandez

Qui sommes nous

AUREAMUNIOZ@GMAIL.COM

DENIS.SIMON@GMAIL.COM

Programmation f… partons du début

Programmation impérative

```
public int moreThanTwentyYearsOld(){

    int moreThanTwenty = 0;

    for (Person person : persons){
        if (person.age > 20){
            moreThanTwenty++;
        }
    }

    return moreThanTwenty;

}
```

Le code de la programmation impérative décrit les actions et modifications de la machine pour l'obtention des résultats.

Inspiré de la Machine de Turing.

Programmation déclarative

```
SELECT COUNT(*)
FROM PERSON p
WHERE p.age > 20
```

La programmation déclarative **décrit** le résultat qu'on veut obtenir.

Il n'y a pas de flux.

Programmation fonctionnelle

« La programmation fonctionnelle est un paradigme de la programmation qui considère le calcul en tant **qu'évaluation de fonctions mathématiques** et **rejette le changement d'état et la mutation des données**. Elle souligne l'application des fonctions, contrairement au modèle de programmation impérative qui met en avant les changements d'état »

Paul Hudak

Programmation fonctionnelle pur

```
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.

Programmation fonctionnel en JAVA 8

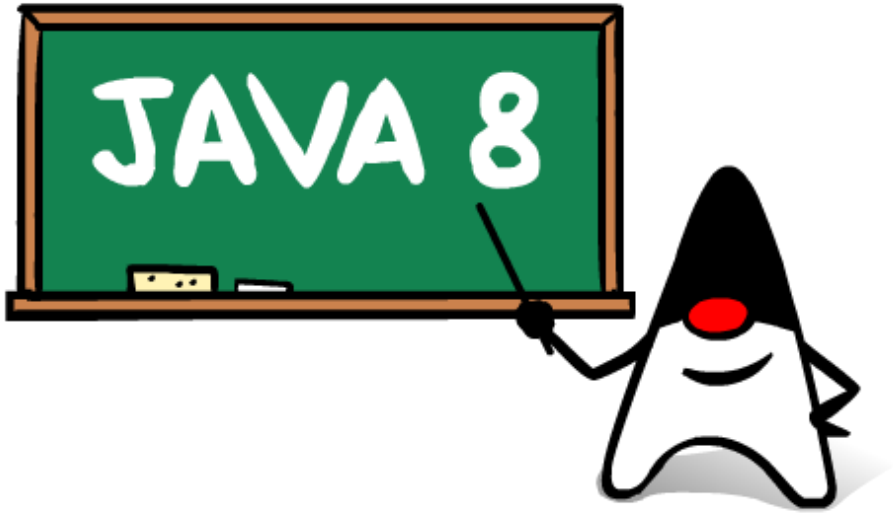
```
public int moreThanTwentyYearsOld(List<Person> persons){
    int moreThanTwenty = 0;

    for (Person person : persons){
        if (person.age > 20){
            moreThanTwenty++;
    }
  }

    return moreThanTwenty;

}
```

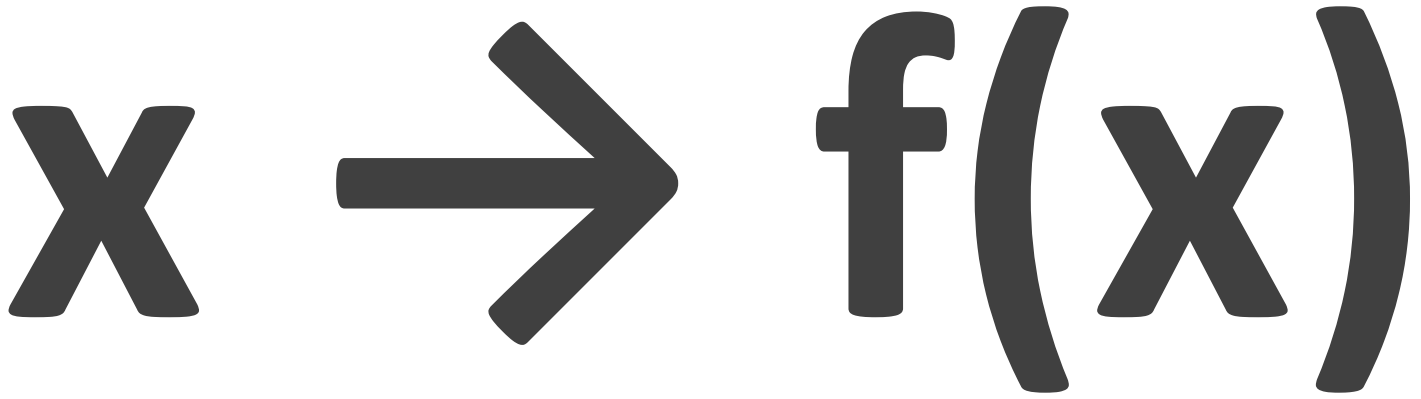
Programmation fonctionnel en JAVA 8



Programmation fonctionnel en JAVA 8

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

Expression LAMBDA



Functional Interaces

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

Une expression lambda est une instance d’une « Functional interface ».

Partons de JAVA 7

```
@FunctionalInterface
public interface Function<T, R> {
    R apply(T var1);
}
mapper = new Function<Person, Integer>(){
    @Override
    public Integer apply(Person person){
        return person.getAge();
  }
};
```

Partons de JAVA 7

```
mapper =
```

```
mapper = new Function<Person, Integer>(){

    @Override
    public Integer apply(Person person){
        return person.getAge();
  }
};
```

La signature de la méthode

```
mapper = (Person person)
```

```
mapper = new Function<Person, Integer>(){

    @Override
    public Integer apply(Person person){
        return person.getAge();
  }
};
```

L’instruction de return

```
mapper = (Person person) ->
```

```
mapper = new Function<Person, Integer>(){

    @Override
    public Integer apply(Person person){
        return person.getAge();
  }
};
```

L’instruction de return

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

```
mapper = new Function<Person, Integer>(){

    @Override
    public Integer apply(Person person){
        return person.getAge();
  }
};
```

Expression LAMBDA

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

Expression LAMBDA

```
(parameter) -> expression;
```

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

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

Référence à la méthode :

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

```
Person::getAge    // méthode statique
```

Expression LAMBDA

```
(parameter) -> {expression;}
```

```
person -> {
    String name = person.getName();
    return name + person.getAge();
}
```

Lors d'une expression avec multiples instructions, un return est OBLIGATOIRE.

Expression LAMBDA

```
(parameter1, parameter2, …)
            -> expression;
```

```
(Integer value1, Integer value2) ->
        Math.addExact(value1, value2);
```

```
(value1, value2) ->
        Math.addExact(value1, value2);
```

Référence à la méthode :

```
Math::addExact
```

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 1

Suivre les instructions sur la classe :

com.francetelecom.java8.exercise0.FunctionGenerator

Exercice 2

Récuperer de SVN le projet java8

http://www.forge.orange-labs.fr/svnroot/softcu/dojos/java8

Suivre les instructions sur la classe :

com.francetelecom.java8.exercise1.BasicCollectionOperations

Consultation de l'api:

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