INSTA

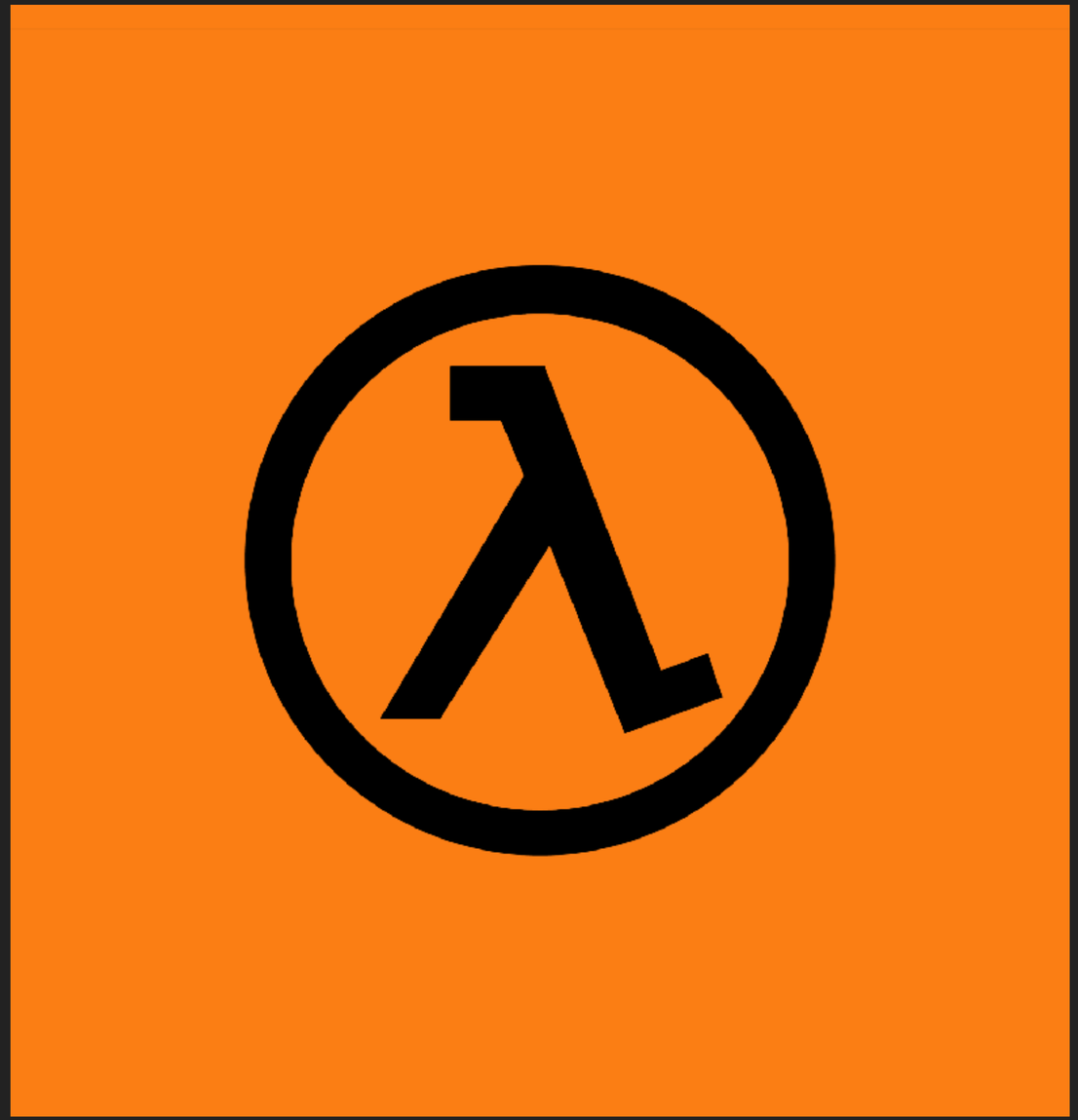
JAVA 8 – λ / STREAM

INTERVENANT

- Jonathan Fitoussi @JnFitoussi
  - Software Ingeneer @LesFurets.com
  - Lead Dev @Graficonception
  - Master Degree @UPMC
  - Java, Python, Javascript, PHP
  - Github https://github.com/jfitoussi

JAVA 8 – Λ (LES LAMBDA)

- Principale nouveauté Java8
- Heritage language fonctionnel
- Anciennement instance d'une classe anonyme



```java
List<Integer> numbers = Arrays.asList(10, 1, 1000, 100);

Collections.sort(numbers, new Comparator<Integer>() {
    @Override
    public int compare(Integer o1, Integer o2) {
        return o1.compareTo(o2);
    }
});

Collections.sort(numbers, (o1, o2) -> o1.compareTo(o2));
```

FUNCTIONAL INTERFACES

- Annotation @FunctionalInterface pour s'assurer qu'une interface ne déclare qu'une seule méthode abstraite
- De nombreuses @FunctionalInterface sont disponibles par défaut.
- D'anciennes interfaces de l'API ont été migrées, Comparable et Runnable par exemple.

```
@FunctionalInterface
public interface Function<T, R> {

    /**
     * Applies this function to the given argument.
     *
     * @param t the function argument
     * @return the function result
     */
    R apply(T t);
```

FUNCTIONAL EXAMPLES

- Une Function prend un argument et retourne un résultat.

```
Function<Integer, String> toString = n -> String.valueOf(n);
Function<String, Integer> toInteger = s -> Integer.valueOf(s);

assert "4".equals(toString.apply(4));
assert toInteger.apply("4") == 4;
```

- BiFunction un spécialisation d'une Function qui prend deux arguments et retourne un résultat.

```
BiFunction<Integer, String, String> concat = (Integer i, String s) -> s + ": " + i
```

FUNCTIONAL EXAMPLES

- UnaryOperator : Negation
- BinaryOperator : Concatenation
- Predicate : isEmpty, isTrim
- BiPredicate
- Supplier
- Consumer

RESUMÉ

- s'il n'y a pas de retour, on utilise un Consumer,
- s'il faut retourner un booléen, on utilise un Predicate,
- s'il faut produire un numérique primitif (int, double, long), on utilise un (Type)ToIntFunction, (Type)ToDoubleFunction, (Type)ToLongFunction,
- s'il faut retourner une valeur sans prendre d'argument, c'est un Supplier,

RESUMÉ

- si le seul argument de la fonction est un int, double, long on utilise un Int(something), Double(something), Long(Something),
- si la fonction prend deux arguments, c'est une Bi(something),
- si la fonction prend deux arguments de même type, c'est un BinaryOperator,
- si une fonction retourne une valeur de même type que son unique argument, c'est un UnaryOperator,
- si une fonction prend en argument un type primitif et un autre type sans retourner de valeur, c'est un Obj(Int|Double|Long)Consumer,
- sinon c'est une Function.

METHODE REFERENCE

- Java 8 introduit le mot clef :: pour extraire des références de méthodes. C'est utile pour remplacer des expressions lambda qui se contentent de faire appel à une méthode qui existe déjà.
- Function<Integer, String> toString = n -> String.valueOf(n);
- Function<Integer, String> toString = String::valueOf;