Formation Java 11

Fonctions du JDK

Sommaire

- Notion d'interface fonctionnelle
- Le package java.util.function

Interface Fonctionnelle

Les expressions lambda ne sont applicables que sur des interfaces dites “fonctionnelles”.

Il s’agit d’interface avec **une seule méthode abstraite**.

Les interfaces existantes du JDK qui n’ont qu’une seule méthode abstraite peut-être vue comme des interfaces fonctionnelles

```
public interface Runnable {

    public abstract void run();

}
```

Interface Fonctionnelle

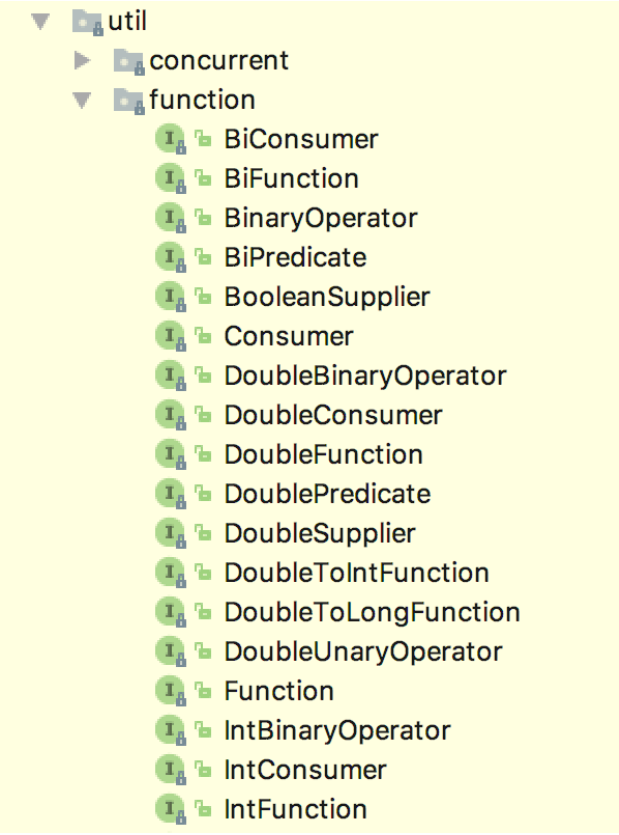
L'annotation **@FunctionalInterface** permet de vérifier à la compilation qu'une interface est bien fonctionnelle au sens Java 8 (elle ne contient qu'une seule méthode abstraite).

```
@FunctionalInterface
public interface Runnable {

    public abstract void run();

}
```

package java.util.function

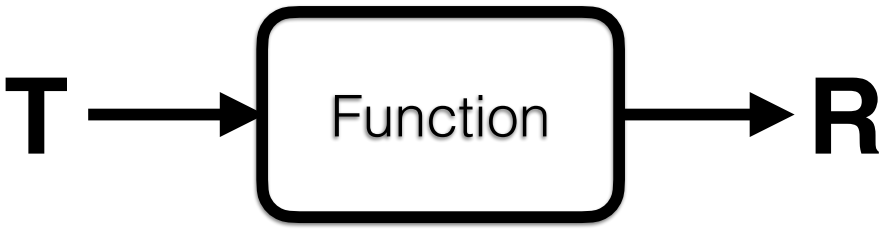
Java 8 fournit des interfaces fonctionnelles usuelles dans le package **java.util.function**.

- Function<T,R>
- BiFunction<T,U,R>
- Consumer<T>
- Supplier<T>
- …



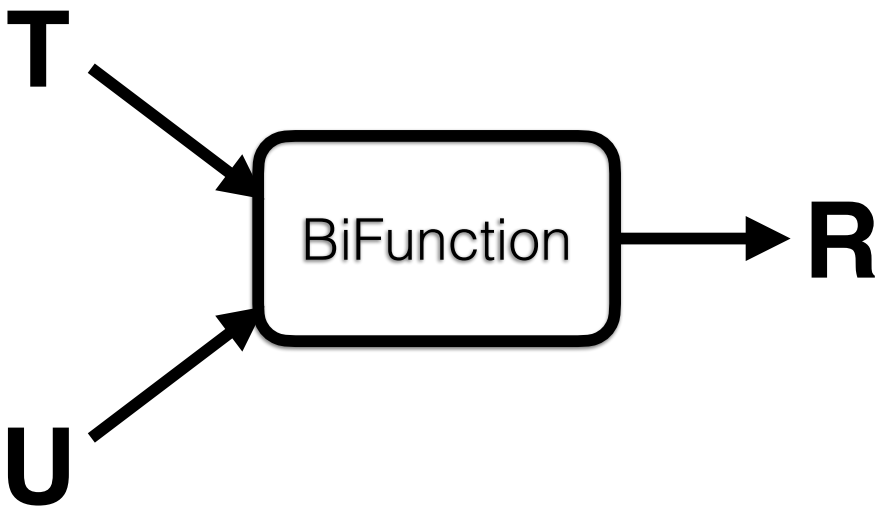
Function<T,R>

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


    R apply(T t);

}
```



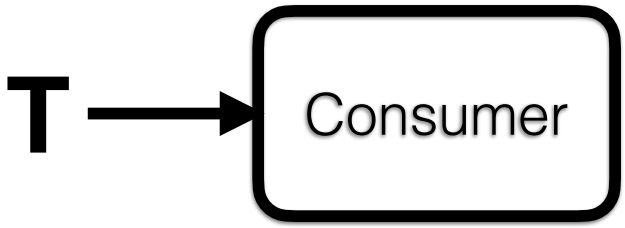
BiFunction<T,U,R>

```java
@FunctionalInterface
public interface BiFunction<T, U, R> {

    R apply(T t, U u);
```



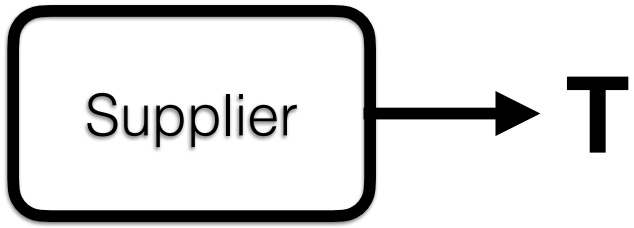
Consumer<T>

```java
@FunctionalInterface
public interface Consumer<T> {

    void accept(T t);
```



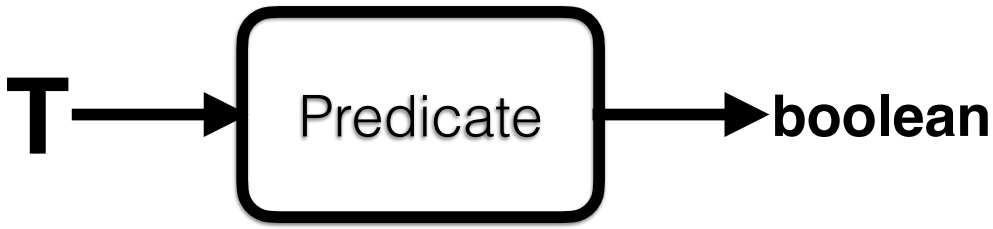
Supplier<T>

```
@FunctionalInterface
public interface Supplier<T> {

    T get();
}
```



Predicate<T>

```java
@FunctionalInterface
public interface Predicate<T> {

    boolean test(T t);
```



A quoi servent ces interfaces ?

- Possibilité de déclarer une lambda comme paramètre d'une méthode ou comme variable.

- Possibilité de chainer l'exécution d'expression lambda via les méthodes : **compose** & **andThen** fournies par certaines interfaces.

A quoi servent ces interfaces ?

```
public String maMethode(BiFunction<Integer,Integer, Integer> b) {

}


BiFunction<Integer, Integer, Integer> addition = (a,b) -> a+b;
Function<Integer, Integer> carre = (b) -> b*b;

maMethode(addition.andThen(carre));
```

Travaux Pratiques