Lambdas

New Austrian Coding School

Programming - memo

We can use **interfaces** to abstract our **methods**. When we do this, that interface is called a **functional interface**. We write the **implementation** of functional interfaces with **lambdas**.

Functional interfaces

The purpose of a functional interface is to **abstract** one single **function**. Functions are methods that don't **belong** to any class. If we can abstract one function with an interface, then we can also:

- **Reference** a function with a **variable**
- **Pass** a function as an **argument**
- **Return** a function from a **method**

Java 8 already provides most of the functional interfaces we will ever use. Two very relevant ones are:

Predicate
○ Receives a variable
○ Returns a boolean
	@FunctionalInterface
	public interface Predicate<T> {
	  boolean test(T var1);
	}

Function
○ Receives a variable
○ Returns a variable
	@FunctionalInterface
	public interface Function<T, R> {
	  R apply(T var1);
	}

If you are curious, here is a list of all available Java 8 functional interfaces.

Lambdas

Functional interface functions are supposed to be **short** and **not reused**. To implement those functions **efficiently** we use **lambdas**. A lambda is a **simplified** version of a method.

Method	Lambda

public String shout(String sentence) {
  return sentence.toUpperCase() + **"!"**;
}
	(e1, e2) -> { operation1; operation2; **return** result; }
	
	(sentence) -> { **return** sentence.toUpperCase() + **"!"**;}

Simplified lambda

○ If there is only **one** argument we can omit the **parenthesis** ○ If there is only **one** operation, we can omit the **curly braces** and the **return** statement	sentence -> sentence.toUpperCase() + **"!"** e -> e.toUpperCase() + **"!"**

Examples

```
List<String> names = Arrays.asList("Leon", "Iris", "David", "Laura");
Predicate<String> condition = name -> name.length() == 5;
for (String name : names) {
    if (condition.test(name)) {
        System.out.println(name);
    }
}
```

```
List<String> names = new ArrayList<>(Arrays.asList("Leon", "Iris", "David", "Laura"));

names.removeIf(e -> e.length() == 5);     ---     [Leon, Iris]     ---     Predicate

names.removeIf(e -> e.equals("Leon"));     ---     [Iris]     ---     Predicate

names.replaceAll(e -> e.toUpperCase());     ---     [IRIS]     ---     UnaryOperator

names.forEach(name -> System.out.println(name));     ---     IRIS     ---     Consumer
```