Anonymous class & lambda expr; Week 2

checkStrings(String [] strings, StringTester tester)

→ testing the strings with the StringTester object.

StringsStartsWithA ⇒ regular class that implements the StringTester interface.

### 1/ regular class

1. create regular class
2. Instantiate object of class
3. step 3 where you use it

⇒ very long. Not satisfying.

### 2/ Anonymous class

same thing but you don’t have to create an instantiate regular class.

Simpler but not satisfying yet.

### 3/ Lambdas expression

*(String* strings*)* → strings.equals(“foo”)

The type “String” is removable.

And since we only have one parameter, () are not mandatory either.

**s → s.equals(“foo”)**

* there is no return
* there is no semi-column

It only consists in calling an already existing method. When we have such lambda, we can replace the lambda with a **method reference**.

### 4/ Method reference

s → “foo”.equals(s) ⇒ “foo”.equals

### @FunctionalInterface

interface myInterface extends anotherInterface

* a functional interface has one exact method
* it operates the same way as @override.
* triggers a compiler check

### 5/ Playing with lambdas

* **FI object = p -> p + i;**

first you have to assing the object, and then call the object

compiler checks that if the method is compatible with the interface, blablabla

* **p->p+1**

other ways to write this block :

* p-> {return p+1}

1.5

java.util.function

you don’t need the Test interface

The main function interfaces are functions that predicate supplier & consummer

So we should use prettify interface instead of test.

We have different methods that help us combine functions one with another

Look at the following interfaces :

* Function
* Predicate
  + we have many methods that allows us to combine predicate with one another and other standart functions (and, or, not…)
  + test().too

1.6

InternalIterator Predicate

…

forEach( Iterable

Iterator

…)

forEach ⇒ internal

Iterator ⇒ external

2.1

fitter

map(selection.stream(),..., Arrays.stream(strings)..)

Stream<T>

…