

**Java - elements of functional programming ( I )**

**Working environment setup**

1. Download and unzip tab06 source code
   1. Download tab06.zip from the course site (moodle)
   2. Unzip it (you get tab06 directory)
   3. Move tab06 to programming-in-java directory, i.e.,

programming-in-java tab00

...

tab06 <--

gradte

...

1. [ IntettiJ ] Add tab06 module to the programming-in-java project
   1. In the *Project* window click settings.gradte file to open it
   2. Modify its content to the following form:

**rootProject.name = 'programming-in-java' include 'lab00'**

**...**

**include 'lab05' include 'lab06'**

* 1. Save the file
  2. Click Load Gradte Changes (a small box in the top right corner)

1. **Functional interfaces, lambda expressions and method references**

Analyse the source code in package tst06\_01

**Exercises**

* 1. Explain the following concepts: *functional interface*, *lambda expression*, and *method reference*
  2. Write and test anonymous functions (lambda expressions) corresponding to:





* 1. Write and test anonymous functions corresponding to:

**sqrt, abs, log, id**



1. Given:

**@FunctionalInterface interface FunIf<T, R> {**

**R apply(T t);**

**}**

complete the following code using lambda expressions:

**FunIf<String, Integer> f1 = ; FunIf<Integer, String> f2 = ; FunIf<Double, Double> f3 = ; FunIf<Integer, Boolean> f4 = ; FunIf<Boolean, Integer > f5 = ; FunIf<Boolean, Boolean > f6 = ;**

1. Repeat the previous exercise using method references instead of lambda expressions (*note*: you should probably implement these methods first)
2. **Standard functional interfaces**

Analyse the source code in package tst06\_02

**Exercises**

* 1. Familiarize yourself with the functional interfaces available in java.util.function

package

* 1. For each of the standard functional interfaces white at least one example that demonstrates its use, i.e.:

**BiConsumer<String, String> bc = (s1, s2) -> System.out.println(s1 + " " + s2);**

**BiFunction... = (..., ...) -> ...**

**BinaryOperator...**

**...**

* 1. Explain the rationale behind the primitive type specialisations of the standard generic functional interfaces (e.g., BooteanSupptier , DoubteConsumer )

1. **Higher-order functions**

Analyse the source code in package tst06\_03

**Exercises**

* 1. Using sumOfWith , without defining any new functions, calculate 
  2. Write and test function

**DoubleUnaryOperator expApproxUpTo(int n) {**



that returns the n-th order

expApproxUpTo(n) =

Maclaurin polynomial of the function , i.e.

.

1. Write and test function

**DoubleUnaryOperator dfr(DoubleUnaryOperator f, double h) {**

**//...**

**}**

that returns for a given function , the approximation of its first derivative calculated as (finite-difference): .

Check the approximation errors corresponding to different values of h

1. (*optional*) Write and test function

**DoubleUnaryOperator d2f(DoubleUnaryOperator f, double h) {**

**//...**

**}**

that returns for a given function , the approximation of its second derivative ; use different finite-difference schemes

1. Analyse and test the following method:

**private static <T, R> List<R> applyAll(List<Function<T, R>> fs, T x0) { List<R> ys = new ArrayList<>();**

**for (var f : fs) { ys.add(f.apply(x0));**

**}**

**return Collections.unmodifiableList(ys);**

**}**

**4) Function composition**

Analyse the source code in package lst06\_04

**Exercises**

1. Using Function.compose create

for the following pairs of and :

**//...**

**}**

1. Repeat the previous exercise using Function.andThen
2. (*optional*) Write a function/method that composes a given list of functions
3. **Dealing with optional data**

Analyse the source code in package tst06\_05

**Exercises**

* 1. Familiarize yourself with class Optional
  2. Describe pros and cons of the following approaches to represent a "no-valid-result" of a function/method:

throwing an exception returning null

using Optional

* 1. Write three variants of a method that returns the tail of a given list (see headOf\_v1 ,

headOf\_v2 , headOf\_v3 in tst01\_05 )

* 1. Review the code of proj1 and identify the methods that could have Optionat as the return type

1. **Push the commits to the remote repository**