Lab: Functional Programming

Preamble

To do the lab, first download and open the NetBeans project template provided on Campus.

1 Lambda Expressions and Method References

1.1 Benefits

Examine the `checkStrings` method in the `lambdas.Main` class. The purpose of the method is to check whether an array of strings all pass a certain test, specified as a `StringTester` object. Two tests are already implemented in the main method of the `lambdas.Main` class: one that uses an instance of a regular class as the string tester object, and a second that uses an anonymous class. (Both classes implement the `StringTester` interface.)

Complete the main method to perform two additional tests using (a) a lambda expression, (b) a method reference. Compare the four implementations of the tester object with respect to conciseness.

1.2 @FunctionalInterface

Tag the `StringTester` interface with the `@FunctionalInterface` annotation. Does it work? What exactly is the purpose of this annotation? Does it remind you of another annotation with the same purpose?

1.3 Playing with lambdas

An interface must be a functional interface in order to be assigned a lambda expression. Can you explain why? Conversely, lambda expression cannot be used without being assigned to a functional interface object, be it an attribute, a local variable or a parameter. Can you explain the two main reasons why?

Try to write your lambda expression using various syntaxes: with/without parenthesis for the parameter list, with/without parameter types, with/without a bloc for the lambda's body, etc.

Next, try to assign to the tester parameter lambda expressions that are not compatible with the `StringTester` interface with respect to parameter type, parameter count, return type, etc.

1.4 Lambdas and generics

Lambdas can be used with generic functional interfaces. In the `lambdas.Main` class, write the method `checkItems` that generalizes `checkStrings` by allowing an array of any type to be passed as the first parameter and a general `Tester` object as the second parameter.

Implement the four tests of exercise 1.1 using the new method and the new interface. Next, use them to check whether an array of `Integer` only contains even numbers.

Finally, check that a given lambda expression can be used with various functional interface targets. What makes it possible?

1.5 Functional programming: general-purpose functional interfaces

Browse the Java API and study the main general-purpose functional interfaces defined in the `java.util.function` package: `Function`, `Predicate`, `Supplier` and `Consumer`. For each functional interface, note the name of its (sole) abstract method. Also, note the name and the role of the various default methods defined in `Function` and `Predicate`.

Which of the above interfaces is similar to your `Test` interface? Rewrite your `checkItems` method using this general purpose interface.

1.6 Functional programming: internal iterators

Starting from Java 8, the `Iterable` interface defines a new (default) method: `forEach()`. This method is called an *internal* iterator as opposed to the *external* iterator `Iterator`. Can you explain the difference?

Using `forEach()`, write a piece of code that prints the even elements of an array of integers. Can you use forEach() to compute the greatest element of the array? Why?

1.7 The Comparator functional interface

Study the `Comparator` functional interface, especially the following static or default methods: `comparing`, `reverse` and `thenComparing`.

Using the above methods, write the piece of code that sorts an array of String:

- by increasing string length
- by decreasing string length
- by alphabetical order (primary criterion) then by decreasing length (secondary criterion)

2 Stream API

2.1 Stream API tour

Browse the Stream API and list (a) the intermediate operations, (b) the terminal operations. How can you tell them apart? Explain why the return types of `map` and `filter` differ.

Next, browse the various ready-to-use collectors defined in the class `java.util.stream.Collectors`.

2.2 Exercise 1.4 with streams

Implement again the `checkItems` method using the Stream API, using two methods. Do you still need the `checkItems` method?

2.3 More streams

Starting from an array of String, write the code that:

- computes the sum of the string lengths
- store in a `List` all the strings with an even length, sorted in reverse alphabetical order
- prints to standard output the length of the 3 three strings following the second string