The purpose of this assignment is to practice programming with lambda expressions.

Exercise 1 – Population

Import the source code contained in the es1 folder in your IDE. The program creates a list of random people (Students, Workers and Person) and outputs statistics about the population. To compute the statistics, anonymous inner classes and nested interfaces are used.
The goal of this exercise is to introduce utility methods that accept lambda expressions as parameters and that allow the replacement of the repeated *for loops* and the anonymous inner classes in the program.

1) Using the logic found in the *main* method, add the following utility methods:
   a) `List<Person> search(List<Person> population, EvaluateOperation op)`
      that returns a new list containing Person instances for which the operation provided as argument evaluates to true.
   b) `Map<String, List<Person>> categorizeString(List<Person> population, CategorizeOperation<String> op)`
      `Map<Integer, List<Person>> categorizeInteger(List<Person> population, CategorizeOperation<Integer> op)`
      that return a map of String/Integer as keys and lists of persons categorized by the *CategorizeOperation* as values.
2) Replace the loops in the main method with the newly created utility methods.
3) Annotate the *EvaluateOperation* and *CategorizeOperation* interfaces as *FunctionalInterface.* Replace the anonymous inner classes by introducing lambda expressions when calling *search*, *categorizeString* and *categorizeInteger*.

Exercise 2 – Text analyzer

Write a program that reads a text file and outputs the following information:
- the longest word
- the shortest word with at least 5 characters
- the word that contains the highest number of vowels
- the list of words that start with a vowel
- the list of words that start with a capital T

The program must include the following helper methods, which must be used to gather the information described above:
- `private static String find(List<String> words, BiPredicate<String, String> operation)`
- `private static List<String> findAll(List<String> words, Predicate<String> operation)`

HINTS:
- The following one-liner may be used to read the contents of a text file into "words":
  `Arrays.asList(new String(Files.readAllBytes(Paths.get("/path/to/file.txt"))).split("\\s+"))`
- The `BiPredicate<String, String> operation` is used to test 2 strings.
  E.g. when searching for the longest word, the first parameter may be the longest word known so far and the second parameter is the current word to test. If the expression returns true a new longest word is found.

Exercise 3 – Optional
Open the *es3/SortingExample* file in your IDE. The program creates a list of random integers, sorts them by calling the *list.sort(new MySortingLogic())* method and then prints the sorted list to the console.
The goal of this exercise is to implement the same sorting logic but, instead of using a new instance of MySortingLogic class when calling *list.sort()*, use:
- an anonymous inner class,
- a lambda expression,

- an instance method reference of the *MySortingLogic* class,
- a static method reference, by adding a static method to the *SortingExample* class.

Implement the alternatives in the respective methods to validate your implementation.