Exercises week 9

Last update 2021/10/26

Goals

The goals of this week is that you get:

- a basic understanding of *reactive programming*
- getting some experience with Java Swing (to illustrate the reactive concepts)
- handling user interfaces in Java (using RxJava and Swing).

Do this first

The exercises rely on access to an RxJava library. This can be done by including this line:

`implementation 'io.reactivex.rxjava2:rxjava:2.2.21'` [1]

in the `build.gradle` file for you project. The `build.gradle` file is in the `Exercises/code/app` directory.

In some of the exercises you will work with a simple Java Swing based user interface:
https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html.

If you are not familiar with Swing, you may find an introduction here:
https://www.javatpoint.com/java-swing.

Exercise 9.1 In the file `ExercisesCode/.../Stopwatch.java` you find a complete Java version of the stopwatch example used in the lecture and material for this week.

Mandatory

1. Revise the stopwatch, so it can measure 1/10 th of a second.
2. Make a version (Stopwatch2) that has two independent stopwatches, each with their own buttons and display.
3. Make a version (StopwatchN) that have n independent stopwatches, each with their own buttons and display. Choose n, so one row of stopwatches fit on your screen.

Exercise 9.2 This exercise makes sure that you have a working version of RxJava and is able to use it to run a few simple examples.

Mandatory

1. Make sure you can run the simple examples in steps 6 and 7 from:
   https://www.tutorialspoint.com/rxjava/rxjava_environment_setup.htm.
   Make sure that you get the same result as in the tutorial.
2. Run the example from:
   https://www.tutorialspoint.com/rxjava/rxjava_single_observable.htm.
   Make sure that you get the same result as in the tutorial.
3. Run the example:
   https://www.tutorialspoint.com/rxjava/rxjava_from_scheduler.htm
   Write down your own explanation of what happens in this example.

[1] you may use other versions than ..2.21, but the first digit of the version has to be 2

Exercise 9.3 In this example you should use the RxJava concepts to make some versions of a stopwatch. In the file `ExercisesCode/.../StopwatchRx.java` you will find (most of) the code for a RxJava based version of the stopwatch.

Mandatory

1. Replace the line //TO-DO in `ExercisesCode/.../StopwatchRx.java` with code that uses the Rx classes (`display` and `timer`) to make a working version of `StopWatchRx`.

Challenging

2. Revise the code from the first step of this exercise so that all buttons are made into observables. (Hint: You may use `ExercisesSolution/.../rxButton.java` as an inspiration.)

Exercise 9.4 In this exercise you should make an RxJava based solution of (part of) exercise 5.2 from week 5.

Mandatory

1. Make an observable `Observable<String> readWords` that can read the file `english-words.txt` file. It should override: `public void subscribe(ObservableEmitter<String> s)` so that each `s.onNext` provides the next line from `english-words.txt`.
2. Make an observer `Observer<String> display= new Observer<String>()` that will print the word emitted from `Observable<String> readWords` i.e. one string every time `onNext` is called.
3. Write a Java program that prints the first 100 word from `english-words.txt` using the the observable `readWords` and the observer `display`.
4. Write a RxJava program to find and print all words that have at least 22 letters.

Challenging

5. Write a Java Rxprogram to find all palindromes and print them (use the `isPalindrome`) method from Exercise 5.2.