Exercises, Observer SDJ2

A JavaFX temperature Converter with an observable Clock (*better version*)

When firing an event (in the `ObservableClock` class), the clock thread calls the method `firePropertyChange` in the `PropertyChangeSupport` object. This method makes a loop, calling `PropertyChange` for each of the listeners. In other words, the clock thread has to update all listeners - independent of their implementation.

Let us say that one or more of the listeners have a time consuming `PropertyChange` method, then the clock thread may not update the time every second, but only when it is done sleeping 1 second and has updated each of the listeners. This is of course not desirable that the time you get depends on how many other are listening and how well they have implemented the `PropertyChange` method.

The solution is to update the `ObservableClock` such that firing an event has its own thread, making the clock thread independent of number of listeners and how time consuming their `PropertyChange` method may be.

Part 1: Convince yourself that firing an event takes time

Step 1: Update class `TemperatureViewController`

- In method `PropertyChange`, after the `Platforrm.runlater` (and outside of it) make statements to sleep for 2000 milliseconds (simulating that it takes a long time to update the label)

Step 2: In the `run` method of the `ObservableClock` class insert print statement to print out the time just before or after firing the event

Step 3: Run the `main` method in class `Main` and observe the result. You should see that the clock do not any longer update every second.

Part 2: Make a clock class being somewhat independent of listeners

Step 1: In the `run` method of the `ObservableClock` class, create a thread to call the `firePropertyChange` method, e.g. as an anonymous thread like this:

```
    new Thread(()->property.firePropertyChange(...)).start();
```

Step 2: Run the `main` method in class `Main` and observe the result. You may see that the clock updates every second.