Object Oriented Programming 2

Rolf Haenni & Andres Scheidegger

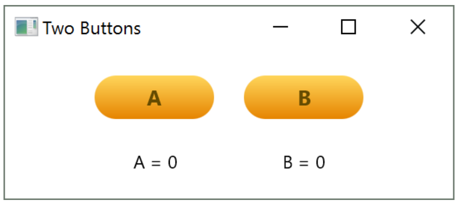
Exercises 6

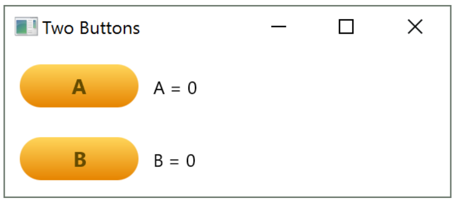
1. Installation of SceneBuilder and First Example

Install SceneBuilder on your machine. More info: https://docs.oracle.com/javase/8/scene-builder-2/work-with-java-ides/index.html.

Then create a JavaFX project for the example *ClickMeSimple* (topic06\Examples\ClickMeSimple) in your favourite IDE.

Now rework the JavaFX application with two buttons A and B. Each time button A or button B is clicked, a counter is increased and the number of clicks is displayed (separately for button A and B). Separate the logic from the layout! Provide two different UI descriptions (View.fxml)!





See also page 11 of Topic6.pdf.

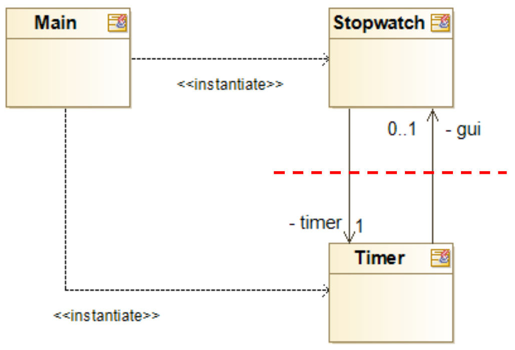
2. Observer Pattern, Stopwatch

The following three classes implement a simple stop watch (topic06\Sources\Exercises6Task2):

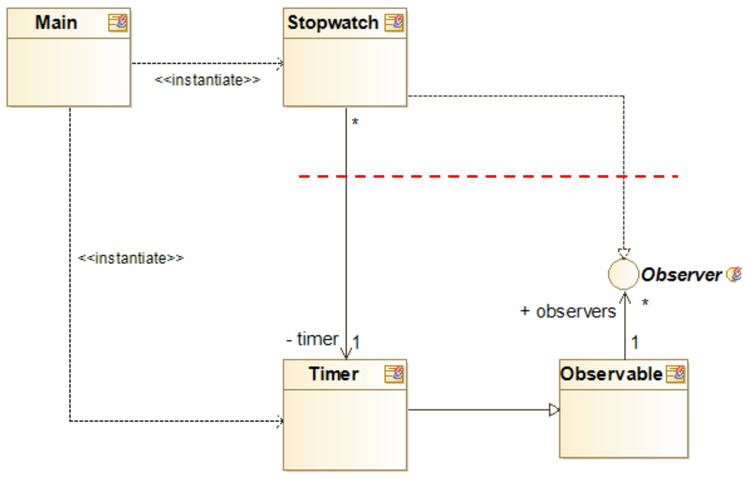
- *Main.java*
- *Timer.java*
- *Stopwatch.java*

To loosen the tight relationship between the classes *Timer* and *Stopwatch* refactor the *Timer* into an *Observable* and the *Stopwatch* into an *Observer*.

Initial relationships and dependencies:



Relationships and dependencies after refactoring:

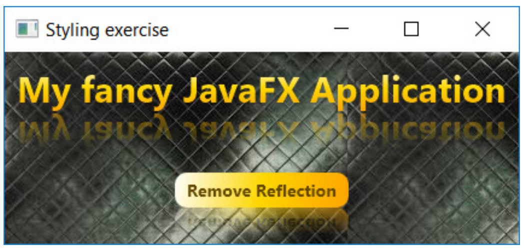


Note the resulting inversion of dependencies across the dashed red line. The model (*Timer*) does not anymore depend on the view (*Stopwatch*). The view can be replaced by whatever you wish, as long as it complies with the interface of class *Timer*. This is good object-oriented design.

To check whether you have done right, create more than one *Stopwatch* object and bind it to the same *Timer* object. For this purpose, you may either link several *Stopwatch* objects to the same stage or create a separate stage for every Stopwatch. You may even derive *Stopwatch* from *Stage*.

3. Styling, CSS and JavaFX

Implement a JavaFX application with the following UI elements:



1. A container (pane) with a background image.
2. A label with an arbitrary text. The text shall be styled as follows: Font size 20, bold, centred and a linear vertical gradient of colours (at least three colours).
3. A button, which changes its colour depending on its state (hover, pressed).

4. The button shall toggle the reflection effect of the text. To do this, use the command *label.setEffect(new Reflection());*

Use FXML (*View.fxml*) and control the layout via a CSS stylesheet.