Java Beans & Reflection

Exercise 1

Using `NetBeans` create a Java Bean named `TempLabel` that implements a temperature converter from Celsius to Farenheit. This Bean must extend `JLabel` redefining the `setText` method in such a way that `setText("c")` visualizes the value obtained by converting `c` into Farenheit.

To test the code write a `main` program that generates some Celsius values and prints each of them along with its corresponding Farenheit value obtained using the `setText/getText` methods of `TempLabel`.

- **Goal**: Warming up!
- **Expected output**: A working `TempLabel` bean for the next exercise.

Exercise 2

Export the `TempLabel` bean of the Exercise 1 in a `jar` file and import it inside the bean palette of `NetBeans`. In a new project, create a class that extends a `JFrame`. Add to this frame a `JText`, a `JButton` and a `TempLabel` bean. The last bean is not visible, but you can act on it using the `Navigator` window of `NetBeans`.

Complete the small application so that the user writes a temperature in the `JText` and then clicks on the button to show in the `TempBean` the corresponding Farenheit value. Inspect the code generated by `NetBeans`. Save the developed bean in a `jar` file.

- **Goal**: Using simple beans and programming their interactions for developing more complex beans.
- **Expected output**: The `jar` file containing the bean, executable with `java -jar ...`; the source files of `TempLabel` and of the frame.

Exercise 3

Write a Java program that takes as argument from the command line the name of a Java Bean class and inspects it by using the reflection API. Your program must print:

- the properties and their capabilities (read-only, read-write);
- the events it permits to subscribe.

Test your program with the bean developed in Exercise 2 and with beans from the standard library, as `AbstractButton`, `JButton`, `JFrame`, etc..

- **Goal**: Learning the Java Reflection API. Understanding the role of introspection in `NetBeans`.
- **Expected output**: A working class implementing the specification (Java source).

Exercise 4

Extend the `TempBean` of Exercise 1 making `text` a **constrained property**, to validate the provided input. The bean should register itself as `VetoableChangeListener`, and it should block any attempt to set the `text` property if the provided string does not represent a `double`, or if its value is smaller than the absolute 0 (-273,15 Celsius degrees).

Make a copy of the source code of Exercise 2, and reengineer it in order to include the new bean in place of the original `TempBean`.

- **Goal**: Working with constrained properties. Understanding the cost of replacing a bean.
- **Expected output**: The `jar` file containing the bean, executable with `java -jar ...`, and the corresponding Java sources.

Date: 2018-10-14T20:23+0200

Author: Andrea Corradini & Matteo Busi

Org version 7.9.3f with Emacs version 24