Unit: III.
Event driven programming
Lab session

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Languages and computing systems
University of Alicante

Content

Requirements

- Make a directory called "`hada-p2`".
- Inside it, do a new file that will contain all the requested source code: "`hada-p2.vala`".
- You can also create your own main program (to test) in a file called "`main.vala`". **This has not to be delivered.**
- You can compile the code in this way: "`valac hada-p2.vala main.vala`", this generates an executable file called "`hada-p2`".

Exercise 1

Create in a file called "`hada-p2.vala`" a class that represents an application, i.e.;

```
1   class Application : GLib.Object {

3     public Application (string name) {
        m_name = name;
5     }

7     public void run () { } // It starts the application
      public void quit () { }// It ends the application
9
      private string m_name;
11  }
```

This class will belong to the namespace '`Hada`'.
In the file "`main.vala`" write the code of the **main** function [1] which creates an object of that class, sends the message '`run`' and next '`quit`'.

[1] note that it does not have a method of the class.

Exercise 2

From now on, until the end of the assignment, all the signals will be connected, when needed, in the constructor of the corresponding class.

- Add to the **Application** class two signals: **void on_init()** and **void on_end()**.
- These signals should be triggered on the start and on the end of the application respectively.
- Create an independent function (not a class method) called: **void al_inicio()**.
- This function will print on screen the text: "`"\nComenzamos...\n"`".
- Conect this function to the signal **on_init**. Check that it is executed when the application starts and trigger the corresponding signal.

Exercise 3

- Create an independent function (not a class method) called: **void al_final()**.
- This function will print the following text: “"\nAcabamos...\n"”.
- Connect it with the signal **on_end**. Check that it is executed when the application ends and the corresponding signal is triggered.
- Create an independent function (not a class method) called: **void al_final2()**.
- This function will print the following text: “"\nAcabamos de verdad...\n"”.
- Connect it with the signal **on_end**. Now check that, besides the funtion `al_final` is executed, the function `al_final2` is also executed when the application is finished and the corresponding signal is triggered.

lsi Department of Software and Computing Systems

Exercise 4 (I)

- Also in the file "`hada-p2.vala`" and added to the previous exercise, create a class 'pila de enteros' (the name of the class will be '`Stack`) also belonging to the namespace '`Hada`'.
- This class will have a unique constructor from a 'string' that will represent the name of the stack.
- The maximum number of elements that it will contain will be '`10`'.
- It will have the method '`public void push (int)`' for pushing elements into the stack and the method '`public int pop ()`' for popping elements from the stack.
- It will be able to trigger two signals: '`void stack_underflow ();`' and '`void stack_overflow ();`'.

Exercise 4 (II)

The main program to test this class will be in the file "`main.vala`". Rename the previous function "`main`" that you will have in it to "`main-aplicacion`". the new main function will perform the following actions:

1. When an element is pushed, it will check if the stack is full, if it is not, it will push the element, and if it is full, it will trigger the signal `stack_overflow`.
2. When an element is popped, it will check if the stack is empty, if it is not empty, it will pop the element, and if it is empty, it will trigger the signal`stack_underflow`, and then it will return the constant '`kERROR`'[2] that will reflect the error.

[2] `public const int kERROR = -100;`

Exercise 4 (III)

3. Create the corresponding independent functions (not methods of a class): **void on_overflow(...)** and **void on_underflow(...)**. Connect them to the corresponding signals.
4. These functions will print at the screen the text: "`"Stack overflow %s index = %d\n"`" and "`"Stack underflow %s index = %d\n"`" respectively, where the parameters '`%s`' and '`%d`' of the formatting string represent the name of the stack and the index of the stack where we have tried to push or pop an element and it has caused the triggering of this signal.

Objectives...

The student knows:

- ☐ Adding signals to a class.
- ☐ Connecting a signal to a function.
- ☐ Connecting a signal to a method.
- ☐ Create a class from scratch with signals and connecting them to the code that should be executed in each case.

Entrega...

- You have to deliver a file called "`hada-p2.tgz`" which will contain the directory "`hada-p2`" and its size will not be greater than 256KB.