Unit: V.
User Interfaces and deferred code execution. Access to DD.BB. from desktop applications
Practical assignment.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Contenido

Requirements

- Use the same directory names, files and identifiers.
- This assignment consists of two parts: **stack** and **sql**.
- Download from the "prácticas" section of the Virtual Campus the material "`hada-p3.tgz`". When extracting it we will obtain a directory called "`hada-p3`" and inside two directories called "`stack`" and "`sql`". Each of them corresponds with one of the parts of this assignment.

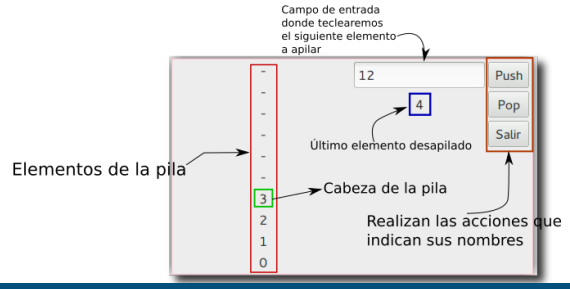
"Stack"

This part of the 3rd assignment is based on the 2nd assignment.

- Inside the directory "`stack`" create the files that will contain all the source code required: "`hada-p3-modelo.vala`", "`hada-p3-vista.vala`". The file "`model-view.vala`" is already created.
- We are going to use the class "`Hada.Stack`" created in the assignment 2, but now it will derive from the classe "`Mvc.Model`" that is in the file "`model-view.vala`". All the code of this class will be in the file "`hada-p3-modelo.vala`".
- On top of this model of the stack "`Hada.Stack`" we are going to create a `View` -the user interface- using `Gtk` with the help of `glade`. This interface will be in the file "`hada-p3.ui`".
- You can create your own test main program in a file called "`main.vala`". **You don't have to deliver it**.

"Stack"

- In the file "`hada-p3-vista.vala`" we will create a class called "`Stack_vista`", as its name indicates it will be a view for the model represented by "`Hada.Stack`". It will implement the *interface* "`Mvc.View`".
- It will have two methods[1]: "`void push_clicked (Button)`" and "`void pop_clicked (Button)`" that will be the handlers of the signals corresponding to the buttons "Push" and "Pop" from the interface.
- It will look as:



"Stack"

In order that the auto-corrector works properly with the graphical interface of your application, keep this in mind:

- The "elementos de la pila" and the "último elemento desapilado" are `widgets` of the type "Label". The names of the `widgets` "elementos de la pila" will be "`label0`", "`label1`"... "`label9`", and the name of the "último elemento desapilado" will be "`label_pop`".
- The entry field will be of the type "`Entry`" and will allow us to type the next element to push. It will be named "`entry_push`".
- The button "`Pop`" will be named "`button_pop`".
- The button "`Push`" will be named "`button_push`".
- The button "`Salir`" will be named "`button_terminar`".
- The window that contains all the elements will be named "`ventana1`".

"Stack" - working

- The main program that you use to test your interface on the class "`Hada.Stack`" will allow the user to do `Push` and `Pop` operations.
- It will reflect the stack status with the `widgets` `label0` to `label9` and the last element poped from it in `label_pop`.
- It will trigger the same signals as in the assignment 2, when necesary.
- It will end the execution when clicking on the button `salir`.

"Sql". First part

- Inside the directory "`sql`" you have the following files:
  - "`bd-vista.vala`" -*view code*-
  - "`clientecad.vala`" -*data access component code*-
  - "`clienteen.vala`" -*bussiness entity code*-
  - "`create-db.sh`" -*shell script that recreates the dd.bb. with the original data*-
  - "`Makefile`" -*compiles and creates the original dd.bb.*-
  - "`model-view.vala`" -*code of the class Mvc.Model and of the interface Mvc.View*-
  - "`vala-bd-gtk.ui`" -*user interface*-
  - "`vala-bd-gui.vala`" -*main program*-.

"Sql". First part

- This code does not compile, it is still without finishing, so the first part of the sql part consists of:
  - Doing the addings/modifications needed in order that the code compiles correctly[2].
  - Once the application compiles, which uses `gtk` and `sqlite`, it will allow you to go forward and backward from the dd.bb. test registries (`hadadb`) that will be created in the current directory when executing "`make`" or "`make crea-bd`".

    This is done calling directly to the DAC from the "view" constructor as an example so you can see an easy to understand functional application.
- You have a test main program in the file "`vala-bd-gui.vala`". **You should not deliver it**.

[2] Look at the errors we obtain when compiling

"Sql". Second part

In order to do this part, is needed that you have correctly done the first part.

- Add two buttons to the user inteface. They will be named "`w_crear`", "`w_borrar`".
- You have to connect to the signal *clicked* of the first one the method of the *BD_vista1* instance class called "`public void on_crear_clicked (Button)`", and to the second button the method "`public void on_borrar_clicked (Button)`", also belonging to the same class.

"Sql". Second part

- **"`on_crear_clicked`"** using the `DAC` for the clients, it will create a new file in the dd.bb. that will contain the `name`, `address` and `city` that the user has typed in that moment in the corresponding text entry fields of our application.
- **"`on_borrar_clicked`"** husing the `DAC` for the clients and taking into account the content that is in that moment only in thel widget "`w_nombre`", it will delete those registries of the dd.bb. which field "`nombre`" matches with the one that the user has typed in "`w_nombre`".
- Keep in mind that the DAC calls should be done from the BN layer, so in first place you need to create a BN instance from the needed data of the view, and this BN will ask the DAC to do the requested operation. Therefore, you should also create the operations needed in the BN to call the DAC operations.

Delivering

- You have to deliver a file called "`hada-p3.tgz`" that will contain the main directory and all its content.
- Its size will not be greater than 512KB.

Objectives...

The student knows:

- ☐ Creating a user interface with a graphical user interface builder.
- ☐ Loading it dynamically at execution time to show it to the user.
- ☐ Using the MVC architecture in applications with graphical user interface.
- ☐ Using a layered model for communicating with a relational database from desktop applications.
- ☐ Creating an application with a graphical user interface using the layered model and MVC, able to communicate to the relational database.