Tema: III.
Programación dirigida por eventos
Prácticas

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

lsi Department of Software and Computing Systems

Contenido

1. Requisitos
2. Ejercicio 1
3. Ejercicio 2
4. Ejercicio 3
5. Ejercicio 4 (I)
6. Ejercicio 4 (II)
7. Ejercicio 4 (III)
8. Objetivos...
9. Entrega...

Requisitos

- Crea un directorio llamado "`hada-p2`".
- Dentro de él crea el archivo que contendrá todo el código fuente pedido: "`hada-p2.vala`".
- Puedes crear tu propio programa principal de pruebas en un archivo llamado "`main.vala`". **No debes entregarlo**.
- Puedes compilar el código así: "`valac hada-p2.vala main.vala`", esto generará un ejecutable llamado "`hada-p2`".

Ejercicio 1

Crea en un archivo llamado "`hada-p2.vala`" una clase que represente una aplicación, p.e.;

```
1   class Application : GLib.Object {

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

7     public void run () { } // Pone en marcha la aplicacion
      public void quit () { }// Termina la aplicacion
9
      private string m_name;
11  }
```

Esta clase pertenecerá al espacio de nombres '`Hada`'.
En el archivo "`main.vala`" escribe el código de la función **main** [1] que cree un objeto de esa clase, le envie el mensaje '`run`' y luego '`quit`'.

[1] fíjate que no debe ser un método de la clase.

Ejercicio 2

A partir de aquí y hasta el final de la práctica, todas las señales, cuando corresponda, se conectarán en el constructor de la clase correspondiente.

- Añade a la clase **Application** dos señales: **void on_init()** y **void on_end()**.
- Estas señales deberán emitirse al inicio y al final de la aplicación respectivamente.
- Crea una función independiente (no un método de una clase) llamada: **void al_inicio()**.
- Esta función imprimirá por pantalla el texto: "`"\nComenzamos...\n"`".
- Conéctala a la señal **on_init**. Comprueba que se ejecuta cuando comienza la aplicación y se emite la señal correspondiente.

Ejercicio 3

- Crea una función independiente (no un método de una clase) llamada: **void al_final()**.
- Esta función imprimirá por pantalla el texto: "`"\nAcabamos...\n"`".
- Conéctala a la señal **on_end**. Comprueba que se ejecuta cuando termina la aplicación y se emite la señal correspondiente.
- Crea una función independiente (no un método de una clase) llamada: **void al_final2()**.
- Esta función imprimirá por pantalla el texto: "`"\nAcabamos de verdad...\n"`".
- Conéctala a la señal **on_end**. Comprueba ahora que, además de ejecutarse la función `al_final`, también se ejecuta `al_final2` al terminar la aplicación y emitir la señal correspondiente.

Ejercicio 4 (I)

- También en el fichero "`hada-p2.vala`" y añadida al ejemplo anterior crea una clase 'pila de enteros' (el nombre de la clase será '`Stack`') también perteneciente al espacio de nombres '`Hada`'.
- Esta clase tendrá un único constructor a partir de un 'string' que representará el nombre de la pila.
- El número máximo de elementos que tendrá será '10'.
- Dispondrá del método '`public void push (int)`' para apilar elementos y del método '`public int pop ()`' para desapilar elementos.
- Podrá emitir dos señales: '`void stack_underflow ();`' y '`void stack_overflow ();`'.

Ejercicio 4 (II)

El programa principal de prueba para esta clase estará en el fichero "`main.vala`". Renombra la función "`main`" anterior que tendrás en él a "`main_aplicacion`". La nueva función main llevará a cabo las siguientes acciones para comprobar que todo lo anterior funciona como se indica:

1. Apilará elementos hasta que se emita la señal `stack_overflow`.
2. Desapilará elementos hasta que se emita la señal `stack_underflow`, después devolverá la constante '`kERROR`'[2] que reflejará el error.

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

Ejercicio 4 (III)

3. Crea las funciones independientes (no métodos de una clase): **void on_overflow(...)** y **void on_underflow(...)**. Conéctalas a las señales correspondientes.
4. Estas funciones imprimirán por pantalla el texto: “"Stack overflow%s index =%d\n"” y “"Stack underflow%s index =%d\n"” respectivamente, donde los parámetros ‘%s’ y ‘%d’ de la cadena de formato representan el nombre de la pila y el índice de la misma **donde se ha intentado apilar o desapilar un elemento y ha provocado la emisión de esta señal**.

Objetivos...

El alumno sabe:

- ☐ Añadir señales a una clase.
- ☐ Conectar a una señal una función.
- ☐ Conectar a una señal un método.
- ☐ Crear una clase desde cero dotándola de señales y conectarle a estas el código que quiere que se ejecute en cada caso.

Entrega...

- La entrega consistirá en un fichero llamado "`hada-p2.tgz`" que contendrá el directorio "`hada-p2`" y no ocupará mas de 256KB.