Tema: V.
Interfaces de usuario y ejecución diferida de código. Acceso a BB.DD. desde aplicaciones de escritorio Prácticas.

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

Contenido

Requisitos

- Respeta nombres de directorios, archivos e identificadores.
- Esta práctica consta de dos apartados: **stack** y **sql**.
- Descarga de la sección de prácticas del Campus Virtual el material "`hada-p3.tgz`". Al descomprimirlo creará un directorio llamado "`hada-p3`" y dentro de él dos directorios llamados "`stack`" y "`sql`". Cada uno de ellos se corresponde con uno de los dos apartados de esta práctica.

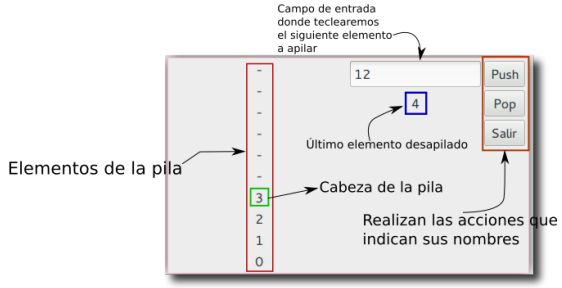
Apartado "stack"

Este apartado de la práctica 3 se basa en la práctica 2.

- Dentro del directorio "`stack`" crea los archivos que contendrán todo el código fuente pedido: "`hada-p3-modelo.vala`", "`hada-p3-vista.vala`". El fichero "`model-view.vala`" ya lo tienes creado allí.
- Vamos a usar la clase "`Hada.Stack`" creada en la práctica 2, solo que ahora derivará de la clase "`Mvc.Model`" que tienes en "`model-view.vala`". Todo el código de esta clase estará en el archivo "`hada-p3-modelo.vala`".
- Sobre este modelo de la pila "`Hada.Stack`" vamos a crear una `Vista` -el interfaz de usuario- en `Gtk` con la ayuda de `glade`. Este interfaz estará en el archivo "`hada-p3.ui`".
- Puedes crear tu propio programa principal de prueba en un archivo llamado "`main.vala`". **No debes entregarlo**.

Apartado "stack"

- En el archivo "`hada-p3-vista.vala`" crearemos una clase llamada "`Stack_vista`", como su nombre indica será una vista sobre el modelo representado por "`Hada.Stack`". Implementará el *interfaz* "`Mvc.View`".
- Tendrá dos métodos[1]: "`void push_clicked (Button)`" y "`void pop_clicked (Button)`" que serán los manejadores de las señales correspondiente de los botones "Push" y "Pop" del interfaz.
- Tendrá este aspecto:



Apartado "stack"

Para que el autocorrector funcione con el interfaz gráfico de tu aplicación ten en cuenta lo siguiente:

- Los "elementos de la pila" y el "último elemento desapilado" son `widgets` de tipo "Label". Los nombres de los `widgets` "elementos de la pila" serán "`label0`", "`label1`"... "`label9`", y el del "último elemento desapilado" será "`label_pop`".
- El campo de entrada será de tipo "`Entry`" y permitirá teclear el siguiente elemento a apilar. Se llamará "`entry_push`".
- El botón "`Pop`" se llamará "`button_pop`".
- El botón "`Push`" se llamará "`button_push`".
- El botón "`Salir`" se llamará "`button_terminar`".
- La ventana que contiene a todos se llamará "`ventana1`".

Apartado "stack" - funcionamiento

- El programa principal que construyas para probar tu interfaz sobre la clase "`Hada.Stack`" permitirá que el usuario vaya haciendo operaciones `Push` y `Pop`.
- Reflejará el estado de la pila con los `widgets` `label0` a `label9` así como el último elemento sacado de la misma en `label_pop`.
- Seguirá emitiendo las mismas señales que en la práctica-2 cuando corresponda.
- Finalizará la ejecución cuando el usuario pulse el botón `salir`.

Apartado "sql". Primera parte

- Dentro del directorio "`sql`" dispones de los siguientes ficheros:
  - "`bd-vista.vala`" *-código de la vista-*
  - "`clientecad.vala`" *-código del componente de acceso a datos-*
  - "`clienteen.vala`" *-código de la entidad de negocio-*
  - "`create-db.sh`" *-guion del shell que recrea la bb.dd. con los valores originales-*
  - "`Makefile`" *-compila y crea la bb.dd. original-*
  - "`model-view.vala`" *-código de la clase Mvc.Model y del interfaz Mvc.View-*
  - "`vala-bd-gtk.ui`" *-interfaz de usuario-*
  - "`vala-bd-gui.vala`" *-programa principal-*.

Apartado "sql". Primera parte

- Este código no compila, está aún por terminar, por lo tanto la primera parte de este apartado consiste en que:
  - Hagas los añadidos/modificaciones para que compile[2].
  - Una vez que compile la aplicación obtenida, la cual hace uso de `gtk` y `sqlite`, te permitirá moverte adelante y atrás por todos los registros de la bb.dd. de prueba (`hadadb`) que se habrá creado en el directorio actual al ejecutar "`make`" o "`make crea-bd`".

    Esto se hace llamando al CAD directamente desde el constructor de la "vista" a modo de ejemplo para que tengas una aplicación funcional facil de comprender.
- Dispones de un programa principal de prueba en el archivo llamado "`vala-bd-gui.vala`". **No debes entregarlo**.

[2] Fíjate en los errores que da al compilar para que te ayude en este apartado

Apartado "sql". Segunda parte

Para realizar esta segunda parte es indispensable que hayas conseguido resolver la primera.

- Añade dos botones al interfaz de usuario. Se llamarán "`w_crear`", "`w_borrar`".
- A la señal *clicked* del primero deberás conectarle el método de instancia de la clase *BD_vista1* llamado "`public void on_crear_clicked (Button)`", y al segundo "`public void on_borrar_clicked (Button)`", también de instancia y perteneciente a la misma clase.

Apartado "sql". Segunda parte

- **"`on_crear_clicked`"** haciendo uso del `CAD` para los clientes, creará un nuevo registro en la bb.dd. que contendrá el `nombre`, `dirección` y `ciudad` que el usuario haya tecleado en ese momento en los campos de entrada de texto correspondientes de nuestra aplicación.
- **"`on_borrar_clicked`"** haciendo uso del `CAD` para los clientes y teniendo en cuenta el contenido que haya en ese momento solamente en el widget "`w_nombre`", borrará aquellos registros de la bb.dd. cuyo campo "`nombre`" coincida exactamente con el que el usuario haya tecleado en "`w_nombre`".
- Ten en cuenta que las llamadas al CAD se deben hacer desde la EN, por tanto primero deberás crear una EN a partir de los datos necesarios de la vista, y esta EN pedirá al CAD que lleve a cabo la operación pedida. Deberás por tanto crear también las operaciones necesarias en las EN para que llamen a los CAD.

Entrega

- La entrega consistirá en un fichero llamado "`hada-p3.tgz`" que contendrá el directorio principal y todo su contenido.
- No ocupará mas de 512KB.

Objetivos...

El alumno sabe:

- ☐ Crear un interfaz de usuario con un constructor gráfico de interfaces de usuario.
- ☐ Cargarlo dinámicamente en tiempo de ejecución para mostrarlo al usuario.
- ☐ Usar la arquitectura MVC en aplicaciones con interfaz gráfico de usuario.
- ☐ Usar un modelo de capas para comunicarse con un gestor de bases de datos relacionales desde aplicaciones de escritorio.
- ☐ Crear una aplicación dotada de interfaz gráfico de usuario que usando el model de capas y MVC es capaz de comunicarse con el gestor de bases de datos relacionales.