EJERCICIO GUIADO. JAVA: PROGRAMACIÓN MDI CONTINUACIÓN

Ventanas Internas

Tal como se explicó en la hoja guiada anterior, una aplicación MDI contiene un panel del tipo JDesktopPane, dentro del cual se depositan objetos del tipo JInternalFrame. Los objetos JInternalFrame son ventanas internas.

El programador debe añadir a su proyecto una clase heredada de JInternalFrame. Esta clase será la ventana interna. El programador diseña el aspecto de esta ventana y añade los métodos que considere necesarios para el manejo de dichas ventanas internas.

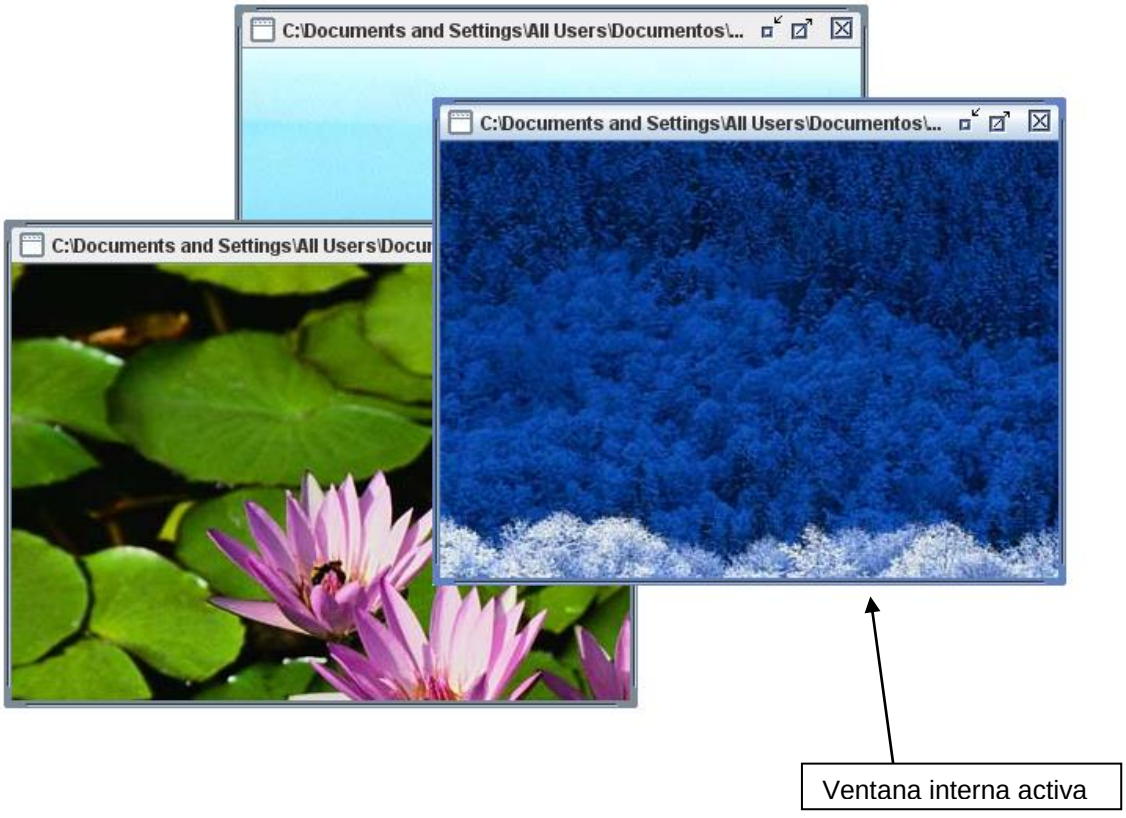
En la hoja anterior se creó un pequeño proyecto “Visor de Fotos” donde hicimos todo esto. Este programa es capaz de abrir varias fotos en sus correspondientes ventanas internas.

En esta hoja guiada veremos como podemos actuar sobre las distintas ventanas internas que han sido abiertas en el JDesktopPane, y para ello usaremos de nuevo el proyecto “Visor de Fotos”.

EJERCICIO GUIADO 1

En este ejercicio guiado inicial, programaremos la opción *Cerrar* del proyecto “Visor de Fotos”. Esta opción debe ser capaz de cerrar la ventana interna que esté activa en ese momento.

Se sabe cual es la ventana interna activa porque aparece por encima de las demás, y porque tiene su barra de título de color azul, mientras que las demás aparecen en gris. La forma de seleccionar una ventana interna es simplemente hacer clic sobre ella.



1. Abre el proyecto *VisorFotos* que realizó en la hoja anterior.

2. Accede a la ventana principal del proyecto, haciendo doble clic sobre la clase *ventanaprincipal*.

3. Accede al evento *actionPerformed* de la opción del menú *Cerrar* y programa lo siguiente:

```
    private void menuArchivoCerrarActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        ventanainterna vactiva = (ventanainterna) panelInterno.getSelectedFrame();
        if (vactiva!=null) {
            vactiva.dispose();
        }
    }
```

4. Ejecuta el programa y comprueba el funcionamiento de la opción *Cerrar*. Se recomienda que abra varias imágenes en su programa y luego seleccione una de ellas. Active la opción *Cerrar* y observe como la ventana interna se cierra.

5. El funcionamiento del código que acaba de programar es el siguiente:

   El panel interno *panelInterno* es un objeto del tipo *JDesktopPane*. Estos paneles son paneles especiales preparados para contener ventanas internas (*JInternalFrame*) y poseen algunos métodos muy útiles para manipular las ventanas internas que contienen.

   Uno de los métodos que más usaremos será *getSelectedFrame*. Este método devuelve la ventana interna seleccionada ahora mismo, o null si no hay ninguna seleccionada.

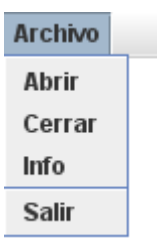
   En el código anterior, observarás que creamos un objeto *vactiva* del tipo *ventanainterna*, y dentro de él metemos la ventana interna seleccionada en este momento, ejecutando el método *getSelectedFrame*:

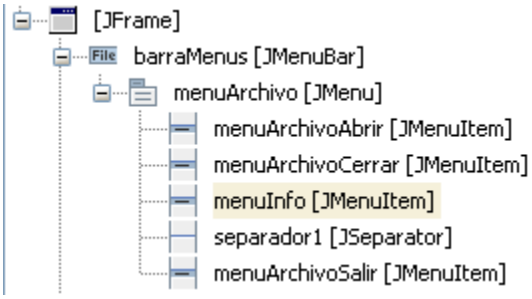
   ```
   ventanainterna vactiva = (ventanainterna) panelInterno.getSelectedFrame();
   ```

   Ahora ya podemos trabajar con *vactiva* sabiendo que se refiere a la ventana activa. Lo que se hace a continuación es cerrar la ventana activa *vactiva* usando el método *dispose* típico de los objetos de ventana.

   Esto se hace, claro está suponiendo que haya alguna ventana activa, por eso se comprueba que *getSelectedFrame* no haya devuelto *null*, porque en ese caso es que no hay ventana interna activada y por tanto no se puede cerrar.

6. Este código es muy común a la hora de trabajar en aplicaciones MDI. Primero se comprueba cual es la ventana activa y luego se actúa sobre ella. Veamos ahora otro ejemplo. Añade al menú la opción *Info*:





7. Accede al evento *actionPerformed* de esta nueva opción y programa lo siguiente:

```
    private void menuInfoActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        ventanainterna vactiva = (ventanainterna) panelInterno.getSelectedFrame();
        if (vactiva!=null) {
            String titulo = vactiva.getTitle(); //devuelve el texto de la barra
                                                 //de titulo de la ventana
            JOptionPane.showMessageDialog(null,"Camino de la imagen:\n"+titulo);
        } else {
            JOptionPane.showMessageDialog(null,"No hay ninguna imagen seleccionada");
        }

    }
```

8. Este código muestra el camino del fichero de la imagen que está seleccionada en este momento. Observa que el proceso es el mismo.

   Primero se extrae la ventana activa, usando el método *getSelectedFrame*, almacenándola en una variable llamada *vactiva*.

   ```
   ventanainterna vactiva = (ventanainterna) panelInterno.getSelectedFrame();
   ```

   Si esta variable es *null*, entonces es que no hay ninguna ventana activa.

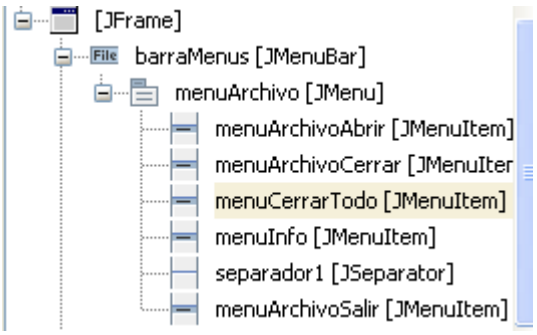
   Si es distinta de *null*, se trabaja con ella. En el ejemplo anterior extraemos el texto de la barra de título de la ventana y lo mostramos en un *JOptionPane*. (Recuerda que hemos programado la apertura de las ventanas internas de forma que en la barra de título aparezca el camino de la imagen)

9. Ejecuta el programa y comprueba el funcionamiento de la opción del menú *Info*.

10. Se ha visto que a través del método *getSelectedFrame* propio de los JDesktopPane se puede acceder a la ventana activa. Pero, ¿cómo puedo acceder a otra ventana interna aunque no esté activa?

    Para hacer esto, la clase JDesktopPane posee un método llamado *getAllFrames* la cual devuelve un vector conteniendo todas las ventanas internas que hay actualmente en el JDesktopPane.

11. Para practicar con el método indicado en el punto anterior, añade al menú *Archivo* una nueva opción llamada *Cerrar Todo*:



12. Accede al evento *actionPerformed* de la nueva opción y programa lo siguiente:

```
    private void menuCerrarTodoActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        int i;
        JInternalFrame v[] = panelInterno.getAllFrames();

        for (i=0;i<v.length;i++) {
            v[i].dispose();
        }

    }
```

13. Analicemos el código anterior.

    Lo primero que tienes que observar es el uso de *getAllFrames*. Este método devuelve un vector conteniendo todas las ventanas internas actuales. En nuestro código, almacenamos estas ventanas en un vector llamado v.

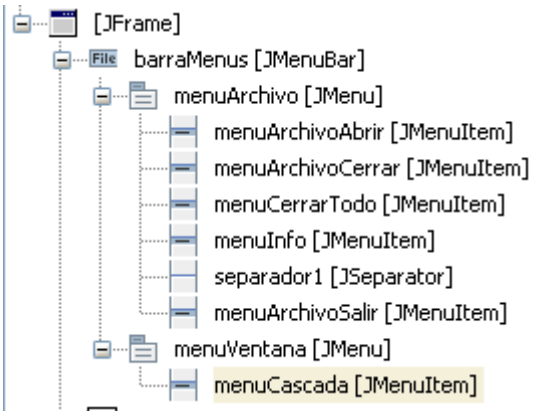
    ```
    JInternalFrame v[] = panelInterno.getAllFrames();
    ```

    A continuación, recorremos todo el vector y vamos cerrando cada ventana almacenada en el vector.

    Como puedes observar, este código hace que se cierren todas las ventanas de imagen abiertas.

14. Ejecuta el programa y comprueba el funcionamiento de la opción *Cerrar Todo*.

15. Este código es un ejemplo de actuación sobre todas las ventanas internas abiertas. A continuación usaremos esta misma idea para organizar las ventanas dentro del panel interno. Añade las siguientes opciones al menú:



16. La opción *Cascada* organizará las ventanas internas abiertas en cascada. Para ello entra en el evento *actionPerformed* de la opción y programa lo siguiente:

```
    private void menuCascadaActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:
        int i;
        int x,y; //posición de la ventana

        JInternalFrame v[] = panelInterno.getAllFrames();

        x=0;
        y=0;
        for (i=v.length-1;i>=0;i--) {
            v[i].setSize(600,400);
            v[i].setLocation(x,y);
            x=x+30;
            y=y+30;
        }
    }
```

17. Analicemos el código.

    Observa de nuevo el uso de *getAllFrames* para almacenar todas las ventanas internas abiertas en un vector v:

    ```
    JInternalFrame v[] = panelInterno.getAllFrames();
    ```

    A continuación se recorre el vector y se asigna a cada ventana interna (cada elemento del vector) un tamaño con el método *setSize* y una posición con el método *setLocation*. Las posiciones de cada ventana van variando para situar cada ventana una encima de la otra.

    El vector se recorre al revés, desde la última ventana a la primera para mejorar la visualización de dichas ventanas.

18. Ejecuta el programa y prueba el funcionamiento de la opción *Cascada*. Se recomienda que primero abra varias ventanas y luego pulse esta opción.

CONCLUSIÓN

En la programación MDI siempre actúan dos elementos:

- Un panel interno JDesktopPane, el cual contendrá las ventanas internas.

- Una clase del tipo JInternalFrame, la cual definirá el diseño de las ventanas internas.

El panel interno posee diversos métodos que permiten acceder a las ventanas actualmente abiertas dentro de él. Entre estos métodos podemos destacar:

- getSelectedFrame, que devuelve la ventana interna activa en el momento actual.

- getAllFrames, que devuelve un vector de JDesktopPane, conteniendo todas las ventanas internas abiertas en el panel interno.

Gracias a estos dos métodos podemos acceder a las ventanas internas y actuar sobre ellas.