# EJERCICIO GUIADO. JAVA: PROGRAMACIÓN MDI

## Programación SDI y Programación MDI

Todo programa tiene una interfaz gráfica de usuario (gui) la cual permite a este manejar el programa de forma sencilla. La interfaz gráfica de usuario consta de la ventana principal, cuadros de diálogo, botones, cuadros de texto, etc…

Según el tipo de interfaz que tenga el programa, las aplicaciones se suelen dividir en dos tipos: Aplicaciones SDI y Aplicaciones MDI.

### Aplicaciones SDI (Single Document Interface)

SDI se puede traducir como interfaz de documento único. Esto quiere decir que las aplicaciones SDI solo pueden mostrar el contenido de un documento a la vez.

Un ejemplo práctico de aplicación SDI es el bloc de notas de Windows. Si en el bloc de notas quieres escribir un nuevo documento, tienes que cerrar antes el documento con el que estás trabajando, ya que este programa no admite el manipular varios escritos a la vez.

Hasta el momento, las aplicaciones de manejo de documentos que hemos realizado hasta ahora han sido de tipo SDI.

### Aplicaciones MDI (Multiple Documento Interface)

MDI se puede traducir como interfaz de múltiples documentos. Esto quiere decir que las aplicaciones MDI pueden mostrar varios documentos a la vez.

Un ejemplo práctico de aplicación MDI es el programa de retoque fotográfico Photoshop. En él, el usuario puede abrir varias fotos y trabajar con todas ellas.

Las aplicaciones MDI normalmente constan de una ventana principal, la cual, puede contener otras ventanas interiores. Cada documento que se abre aparece en una ventana interior.

En este ejercicio guiado, se explicarán las nociones básicas para crear una aplicación MDI en Java con NetBeans.

**EJERCICIO GUIADO 1**

Se pretende crear un visor de imágenes MDI, es decir, que permita la visualización de varias imágenes a la vez. El programa constará de una ventana principal con un menú. Las opciones de este menú permitirán al usuario abrir varias imágenes y cerrarlas a su gusto.

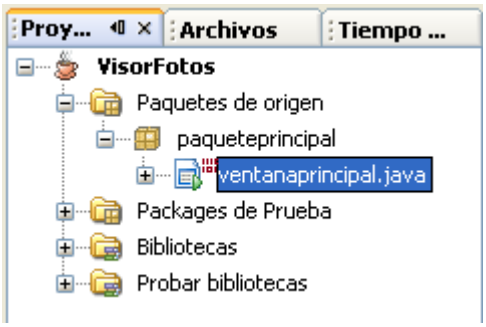
Al ser un proyecto MDI, las imágenes se abrirán en ventanas internas. Estas ventanas internas tendrán que ser diseñadas de forma adecuada.

Para crear este proyecto, tendremos que seguir tres pasos generales:

- Diseño de la ventana principal.
- Diseño de las ventanas internas.
- Programación de la ventana principal / internas.

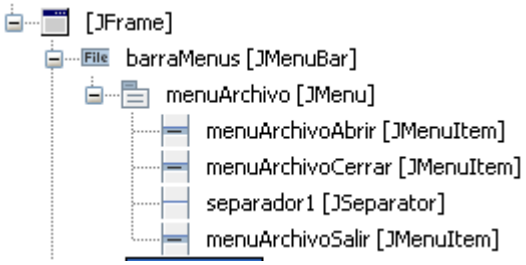
## Diseño de la ventana principal

1. Crea un nuevo proyecto. El nombre del proyecto será *VisorFotos*. Añade un paquete llamado *paqueteprincipal*. Dentro de dicho paquete añade un JFrame llamado *ventanaprincipal*.

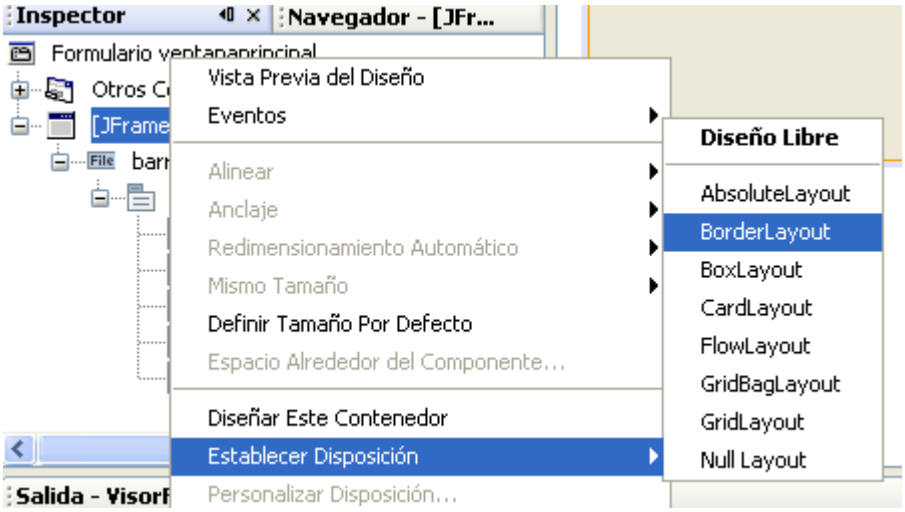



2. Añade a la ventana principal una barra de menús, con una única opción *Archivo*, que contenga a su vez las siguiente opciones:

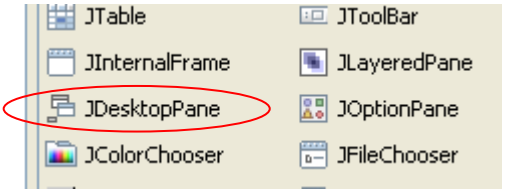

Asigna los siguientes nombres a cada elemento del menú:



3. Establece un layout de tipo *BorderLayout* al JFrame. Recuerda que este tipo de distribución divide la ventana en cinco zonas: norte, sur, este, oeste y centro.

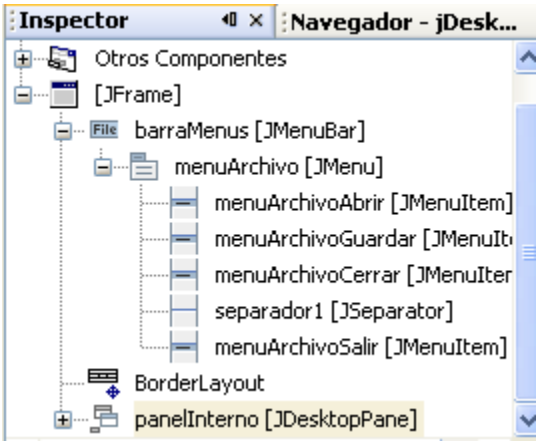



4. En la zona central de la ventana principal colocaremos un panel, pero no será un JPanel, como siempre, sino otro tipo de panel. Debes colocar un panel del tipo JDesktopPane:




Este tipo de panel es usado como contenedor de las ventanas internas de una aplicación MDI. En NetBeans este tipo de panel se muestra de color azul, para distinguirlo de los paneles normales, por eso, cuando añadas el panel al JFrame este quedará así:

5. Cámbiale el nombre al JDesktopPane y asígnale el nombre *panelInterno*:




6. Si ejecutas el programa ahora, verás que la ventana sale reducida al mínimo tamaño posible. Esto lo vamos a evitar haciendo que la ventana aparezca maximizada al ejecutarse el programa. También le asignaremos un tamaño inicial. Para ello, acude al constructor del JFrame y programa lo siguiente:

```
/** Creates new form ventanaprincipal */
public ventanaprincipal() {
    initComponents();
    ConfiguracionVentana();
}

public void ConfiguracionVentana() {
    this.setSize(800,600);
    this.setExtendedState(JFrame.MAXIMIZED_BOTH);
}
```

Llamada a un método *ConfiguracionVentana*

Programación de dicho método.

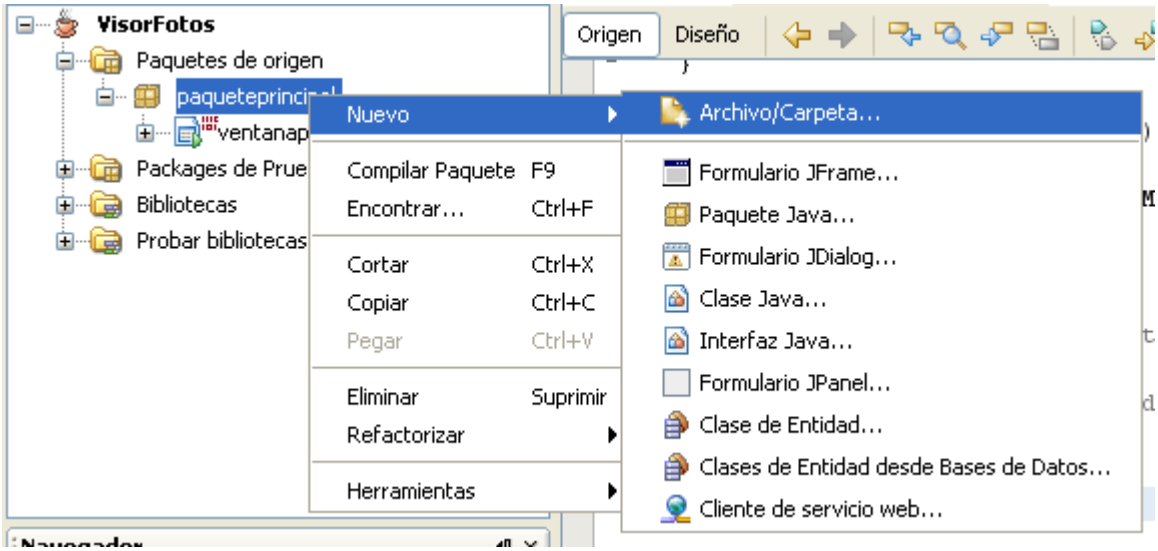
Como ves, en el constructor se llama a un método *ConfiguracionVentana* y en este método se asigna un tamaño por defecto a la ventana de 800x600 y se maximiza.

**Diseño de las ventanas internas**

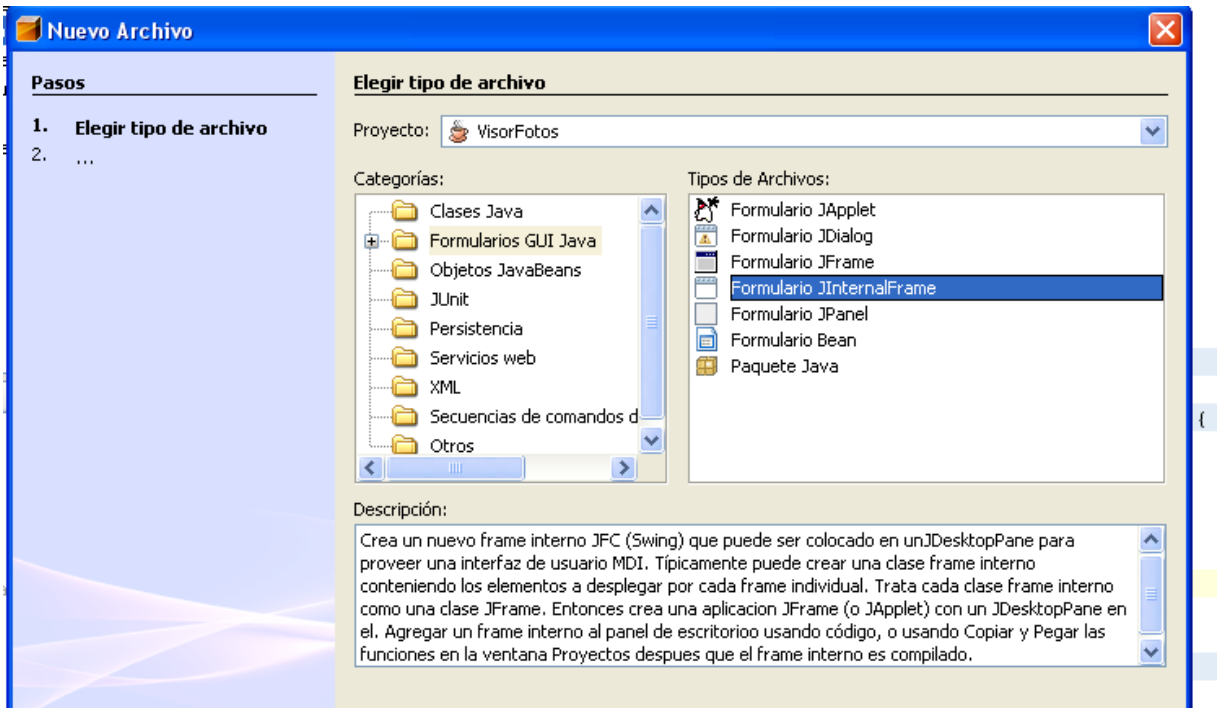
Se pretende que cuando se abra una imagen, el programa muestre una ventana interna que contenga la imagen y nada más. La barra de título de esta ventana interna contendrá el camino de la imagen.

Para diseñar la ventana interna de la aplicación, sigue los pasos que se indican a continuación:

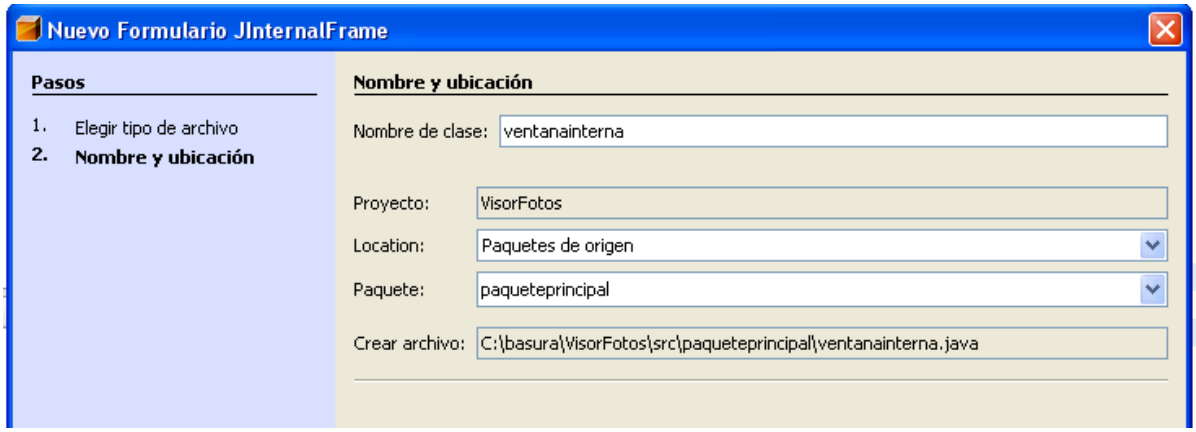
7. Una ventana interna de una aplicación MDI es un objeto de la clase JInternalFrame. Será necesario añadir esta clase al proyecto. Para ello, haz clic con el derecho sobre el *paqueteprincipal* y elige *Nuevo → Archivo/Carpeta*.



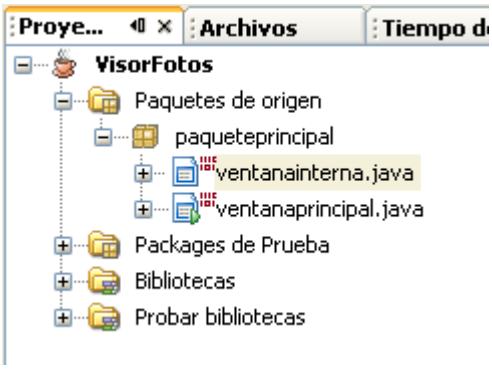

8. Luego elige *Formularios GUI Java* y dentro de esta categoría *Formulario JInternalFrame*.



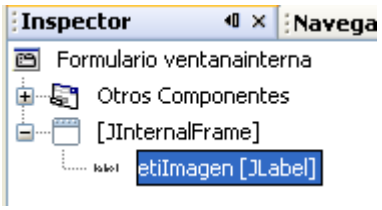
9. El nombre que le pondremos a este tipo de formulario será el de *ventanainterna*.




10. Si observas la ventana de proyecto, verás que ahora tenemos dos clases: la ventana principal, y la ventana interna. Ahora diseñaremos la ventana interna, para ello, haz doble clic sobre *ventanainterna*.




11. Como se dijo anteriormente, las ventanas internas mostrarán simplemente la imagen que se abra. Para ello, solo hace falta introducir una etiqueta (JLabel) que será la que contenga la imagen. Esta etiqueta debe ocupar toda la ventana, no tendrá ningún texto dentro, y su nombre será *etiImagen*.




La etiqueta se llamará *etiImagen*…

Y ocupará todo el *JInternalFrame*

12. En las hojas guiadas anteriores, se ha hablado de la programación orientada a objetos, y se ha comentado que los objetos poseen propiedades. Estas propiedades son básicamente variables globales internas. Para poder acceder a estas propiedades, es necesario programar métodos *set*.

    Pues bien, la etiqueta *etiImagen* de la clase *ventanainterna*, no es mas que una propiedad de la ventana interna, y para poder trabajar con ella, será necesario programar un método *set* que permita modificar la etiqueta a nuestro antojo.

    Básicamente, este método set debe ser capaz de introducir en la etiqueta una imagen. Haremos que este método reciba como parámetro el camino de la imagen a mostrar.

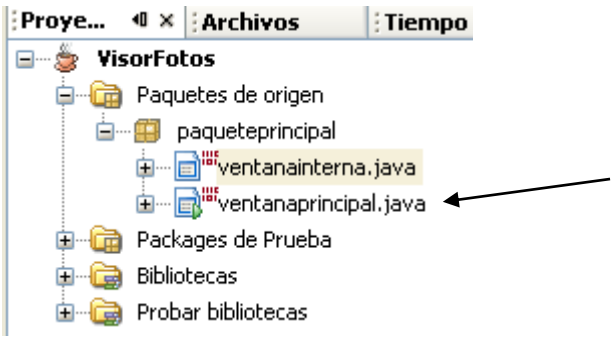
13. Así pues, entra en la zona de código de la *ventanainterna* y añade después del constructor el siguiente método:

```
public class ventanainterna extends javax.swing.JInternalFrame {

    /** Creates new form ventanainterna */
    public ventanainterna() {
        initComponents();
    }

    //Método que coloca imagen en la etiqueta etiImagen
    //Recibe como parametro el camino de la imagen
    public void setImagen(String camino) {
        ImageIcon imagen = new ImageIcon(camino);
        etiImagen.setIcon(imagen);
    }
```

Método para acceder a la etiqueta *etiImagen*

14. En la programación MDI, será habitual crear métodos para poder acceder a los distintos elementos de la ventana interna (etiquetas, cuadros de texto, etc) En otras ocasiones, tendremos que crear métodos get para obtener información de las ventanas internas. Es algo muy similar a la programación de diálogos propios que se vio en hojas anteriores.

15. Bien, con la programación de este método de acceso a la etiqueta de la ventana interna, hemos terminado con el diseño de esta ventana, ahora empezaremos a programar la ventana principal. Vuelve a ella haciendo doble clic sobre la *ventanaprincipal* en la zona de proyectos:



**Programación de la ventana principal**

Se pretende ahora programar la opción *Abrir* del menú de forma que se elija el fichero de imagen a mostrar y se muestre este en una ventana interna.

La opción *Cerrar* del menú permitirá cerrar la ventana interna activa en un momento determinado.

Sigue los pasos que se describen a continuación:

16. Ya estamos preparados para programar las opciones del menú. Empezaremos por la opción Abrir. Accede al *actionPerformed* de la opción Abrir y programa lo siguiente:

```
    private void menuArchivoAbrirActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:

        JFileChooser abrir = new JFileChooser();
        int boton = abrir.showOpenDialog(null);
        if (boton==JFileChooser.APPROVE_OPTION) {

        }

    }
```

    La opción Abrir se encargará de abrir un fichero de imagen y mostrarlo en una ventana interna del programa. Lo primero que hace esta opción es mostrar el cuadro de diálogo Abrir, que se usa para indicar el fichero que se quiere abrir.

    Se crea un objeto del tipo JFileChooser a través de la línea:

    ```
    JFileChooser abrir = new JFileChooser();
    ```

    Luego se le da la orden de que muestre el cuadro de diálogo Abrir. La variable *boton* recoge el botón pulsado por el usuario (Aceptar / Cancelar)

    ```
    int boton = abrir.showOpenDialog(null);
    ```

    Luego, a través de un if, compruebo si se ha pulsado el botón Aceptar…

    ```
    if (boton==JFileChooser.APPROVE_OPTION) {

    }
    ```

    Dentro de este if tendremos que programar la acción correspondiente a abrir la imagen y mostrarla en pantalla. Vamos a ello.

17. Programa dentro del if anterior lo siguiente. El código se comentará a continuación.

```
    private void menuArchivoAbrirActionPerformed(java.awt.event.ActionEvent evt) {
// TODO: Agrege su codigo aqui:

        JFileChooser abrir = new JFileChooser();
        int boton = abrir.showOpenDialog(null);
        if (boton==JFileChooser.APPROVE_OPTION) {
            //Creación de la ventana interna para la imagen
            ventanainterna vi = new ventanainterna();
            vi.setResizable(true);
            vi.setMaximizable(true);
            vi.setIconifiable(true);
            vi.setClosable(true);

            panelInterno.add(vi);

            //Se asigna la imagen y se muestra la ventana interna
            String camino = abrir.getSelectedFile().toString();
            vi.setImagen(camino);
            vi.setTitle(camino);
            vi.setVisible(true);
        }

    }
```

Atento a este código, porque define la creación de ventanas internas, conteniendo la imagen elegida para mostrar. Es el corazón del programa…

La primera instrucción, crea una ventana interna llamada *vi*. Como puedes observar, es la creación de un objeto *vi* de la clase *ventanainterna*.

```
ventanainterna vi = new ventanainterna();
```

Lo siguiente que se hace con el objeto *vi* creado, es definir sus características como ventana. Concretamente se decide que sea una ventana con posibilidad de cambiar de tamaño (*setResizable*), una ventana que pueda ser maximizada (*setMaximizable*), una ventana que pueda ser minimizada (*setIconifiable*) y finalmente que pueda ser cerrada (*setClosable*)

```
vi.setResizable(true);
vi.setMaximizable(true);
vi.setIconifiable(true);
vi.setClosable(true);
```

Una vez definidas dichas características de la ventana interna, esta se añade al panel interno de la ventana principal, al que le dimos el nombre *panelInterno*.

```
panelInterno.add(vi);
```

Ahora hay que introducir la imagen elegida dentro de la etiqueta de la ventana. Primero hay que recoger el camino del fichero de imagen elegido en el cuadro de diálogo *abrir*.

```
String camino = abrir.getSelectedFile().toString();
```

Ahora aprovechamos el método *setImagen* que programamos oportunamente dentro de la clase *ventanainterna* para situar dicha imagen dentro de la etiqueta.

```
vi.setImagen(camino);
```

Colocamos el camino de la imagen en la barra de título de la ventana interna:

```
vi.setTitle(camino);
```

Finalmente hacemos visible la ventana interna.

```
vi.setVisible(true);
```

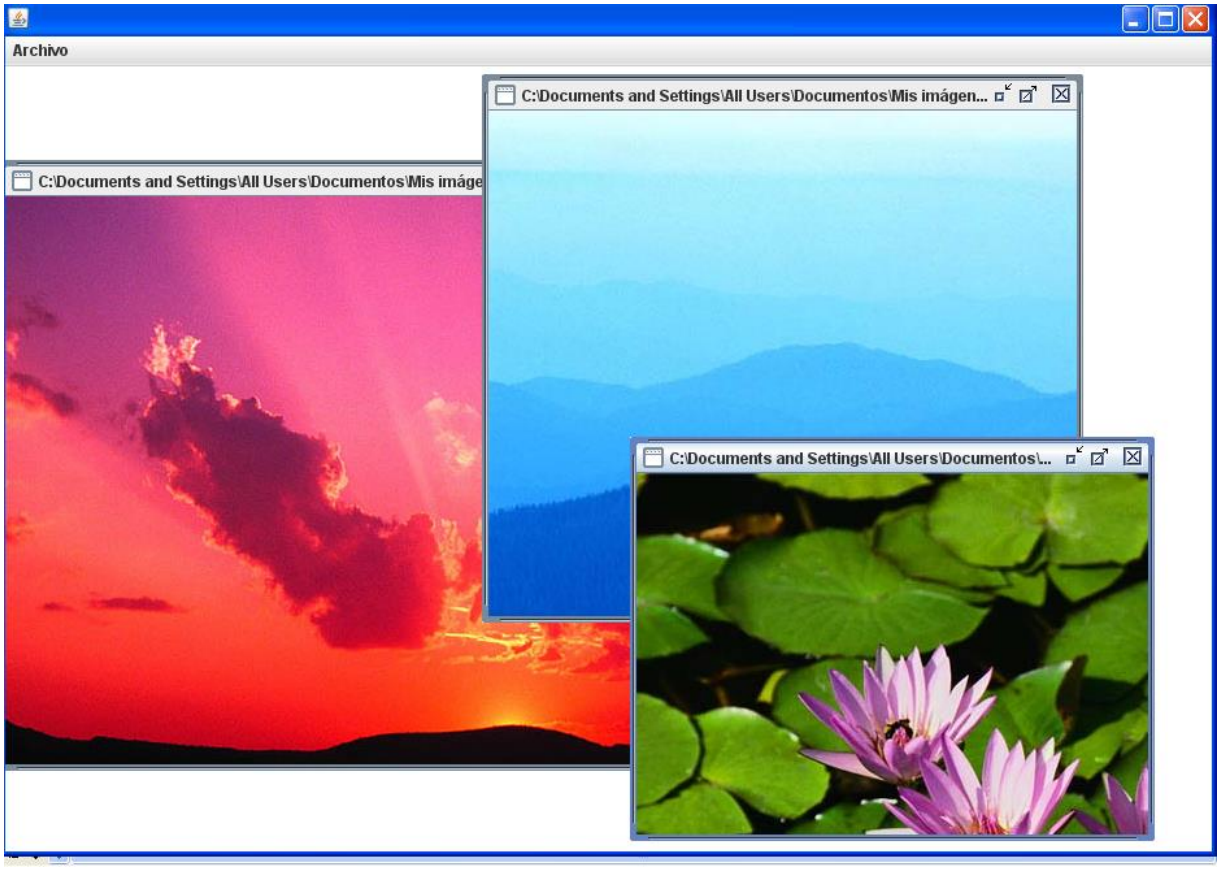
18. Ejecuta el programa y prueba a abrir varios ficheros de imagen.

    Observa la creación de las ventanas internas.

    Observa como todas tienen el mismo aspecto (una etiqueta única en la ventana conteniendo la imagen)

    Observa la posibilidad de moverlas, cambiarlas de tamaño, maximizarlas, minimizarlas y cerrarlas.

    Observa como todas las ventanas internas están encerradas dentro de los límites del panel interno de la ventana principal:




19. Este proyecto se usará en las próximas hojas. Guárdalo.

**CONCLUSIÓN**

**La programación MDI consiste en crear aplicaciones capaces de abrir varios ficheros y mostrarlos en distintas ventanas internas.**

**Una aplicación MDI cuenta con dos elementos básicos:**

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

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

**La clase JInternalFrame que añadamos, tendrá el nombre que queramos asignarle y nos servirá de plantilla para crear las ventanas internas de nuestro proyecto. Esta clase podrá tener métodos internos para acceder a los elementos de las ventanas internas.**

**Desde la ventana principal, se crearán objetos de la clase ventana interna, y se definirán opciones relativas a la posibilidad de maximizar, cerrar, minimizar, etc, dichas ventanas.**

**Desde la ventana principal se usarán los métodos programados en la ventana interna para poder manejarla con facilidad.**