

## Ejercicio 11

```

elementoAbrir=new JMenuItem( text: "Abrir");
new *
elementoAbrir.addActionListener(new AbrirActionListener() {
    new *
    public void actionPerformed(ActionEvent e)
    {
        abrirArchivo();
    }
});
elementoAbrir.setActionCommand("Añadir");
menuArchivo.add(elementoAbrir);

elementoGrabar=new JMenuItem( text: "Grabar");
elementoGrabar.addActionListener( this);
elementoGrabar.setActionCommand("Grabar");
menuArchivo.add(elementoGrabar);

elementoSair=new JMenuItem( text: "Salir");
new *
elementoSair.addActionListener(new SairActionListener() {
    new *
    public void actionPerformed(ActionEvent e)
    {
        sair();
    }
});
elementoSair.setActionCommand("Salir");
menuArchivo.add(elementoSair);

```

```

1 usage 1 inheritor new *
class AbrirActionListener implements ActionListener{
    1 override new *
    @Override
    public void actionPerformed(ActionEvent e) {
        abrirArchivo();
    }
}

1 usage 1 inheritor new *
class SairActionListener implements ActionListener{
    1 override new *
    @Override
    public void actionPerformed(ActionEvent e) {
        sair();
    }
}
}

```

## Ejercicio 12

En el proyecto 0-3 se instancia la la clase `ActionListener()` directamente a la hora de añadir un Action Listener al objeto de la interfaz. Haciendo uso de “`new ActionListener()`” dentro del “`AddActionListener(new ActionListener())`”

```

JMenuItem quitItem = new JMenuItem(text: "Quit");
quitItem.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) { quit(); }
});
fileMenu.add(quitItem);

```

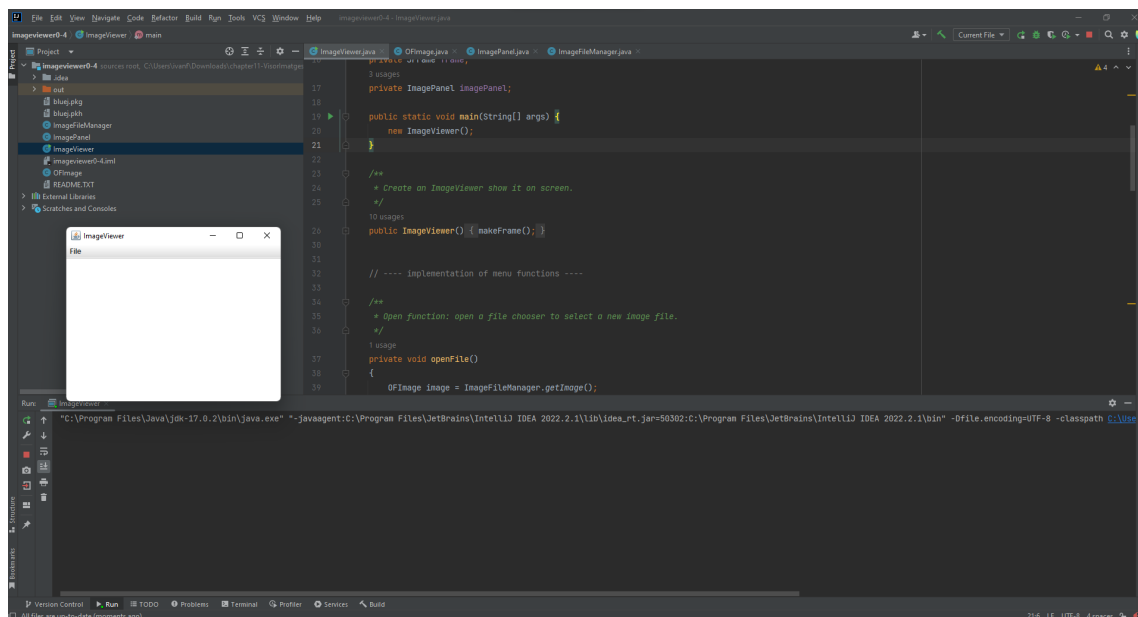
### Ejercicio 13

Hasta ahora al presionar la opción “Salir” el programa se cerraba haciendo uso del package “System”, usando la siguiente línea dentro del código.

```
System.exit( status: 0);
```

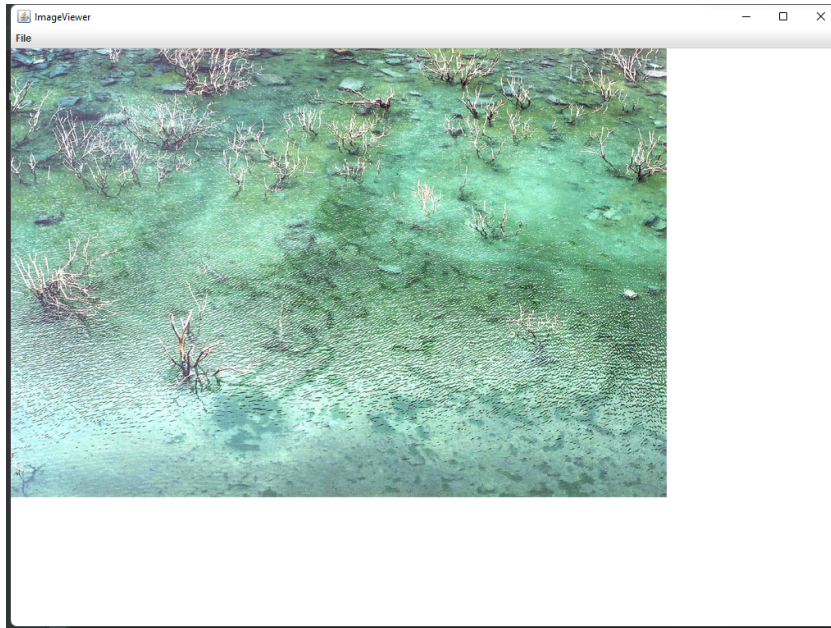
En esta línea usamos el Package System y el método “Exit()” a este método le pasaremos un int para que el programa sepa en qué estado se ha cerrado la app.

### Ejercicio 14

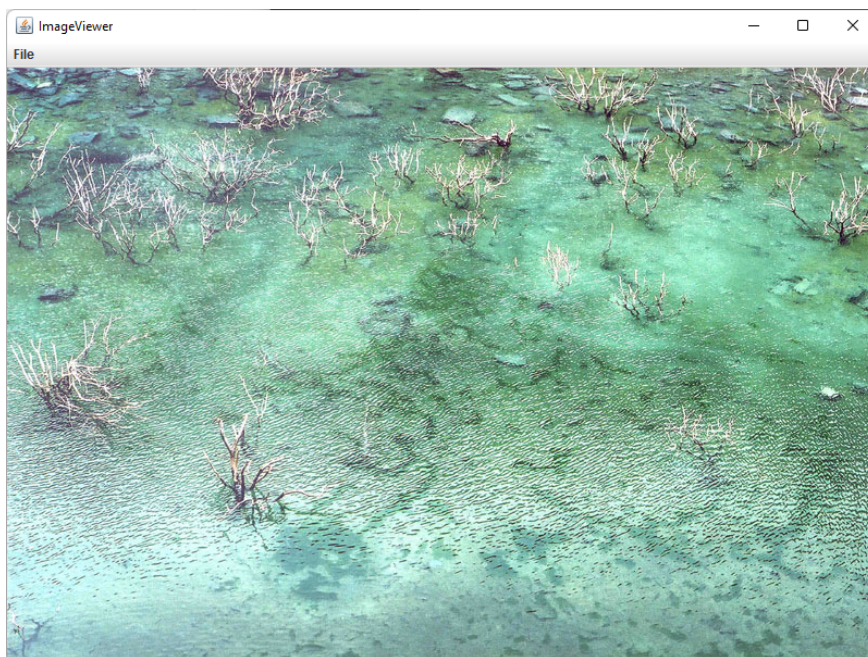


### Ejercicio 15

Cuando abro una imagen y luego expando la pantalla haciendo que ocupe mas espacio la imagen se queda estática ocupando lo que estaba ocupando a la hora de abrirla (Espacio en base a la resolución de la misma) y al lado en la zona de ventana que hemos expandido solo se verá el fondo de la aplicación.



*Pero, si Primero la expando y luego la abro el JFrame se ajustará a la resolución de la imagen.*



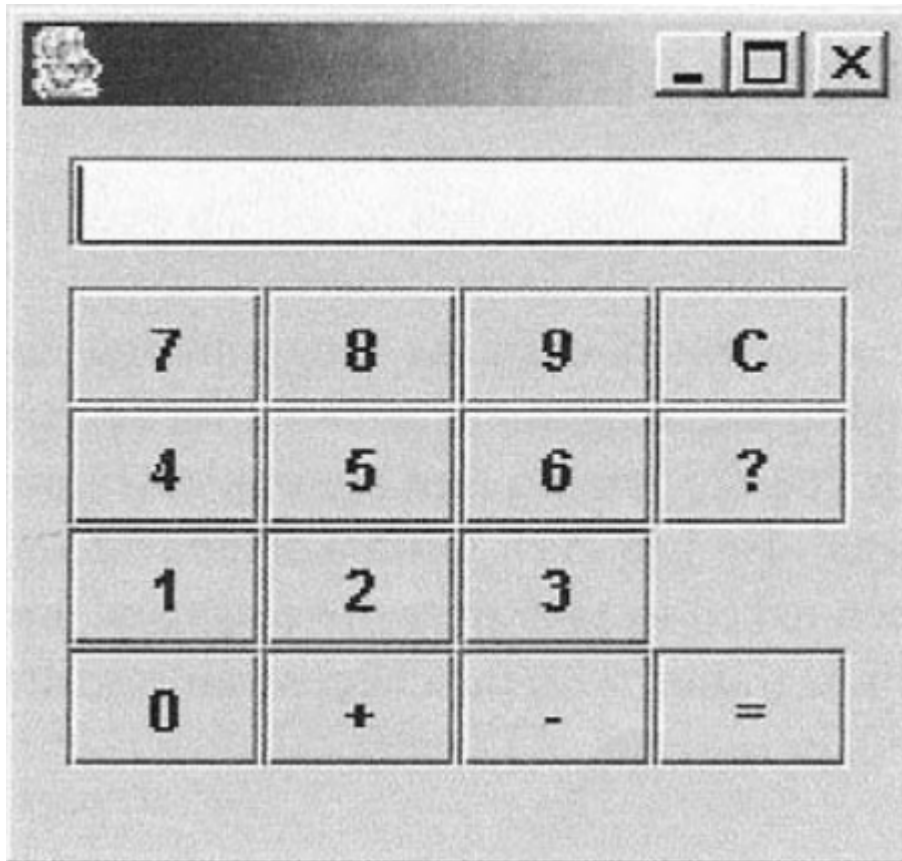
## Ejercicio 16

```
JLabel etiquetaNombreDeArchivo = new JLabel( text: "JLabel - 1");  
panelContenedor.add(etiquetaNombreDeArchivo);  
  
JLabel etiquetaEstado = new JLabel( text: "JLabel - 2");  
panelContenedor.add(etiquetaEstado);
```



*Solo se visualiza el último JLabel que hemos insertado ya que se ha sobrescrito al anterior al no decirle donde queremos que esté ubicado.*

## Ejercicio 17



Viendo la interfaz, creo que se ha realizado con un `gridlayout` de 4 por 4, ya que con este layout podemos setear un número de filas y columnas y ajustar el tamaño y relación de aspecto entre los diferentes objetos.

En este caso al ser todos los objetos botones el `gridlayout` ha ajustado todos los botones para tener el mismo tamaño y poder dar como resultado la UI de la imagen.

Además creo que han añadido otro layout que contiene el `grid layout` para poder poner el `JPanel` con el texto que nos mostrará el resultado de los número que vayamos escribiendo.

### Ejercicio 18

Un `JFrame` que contenga un `JPanel`, en este `JPanel` tendremos el `Borderlayout` con 2 `JPanels` mas, uno seteado en `NORTH` y otro en `SOUTH`, el de `NORTH` tendrá 2 `JPanel` en un `border layout`, 1 seteado en `EAST` que tendrá la posibilidad de hacer la accion de la parte derecha y un segundo seteado en `WEST` que a su vez estará con un `GRIDLAYOUT` y dentro botones.

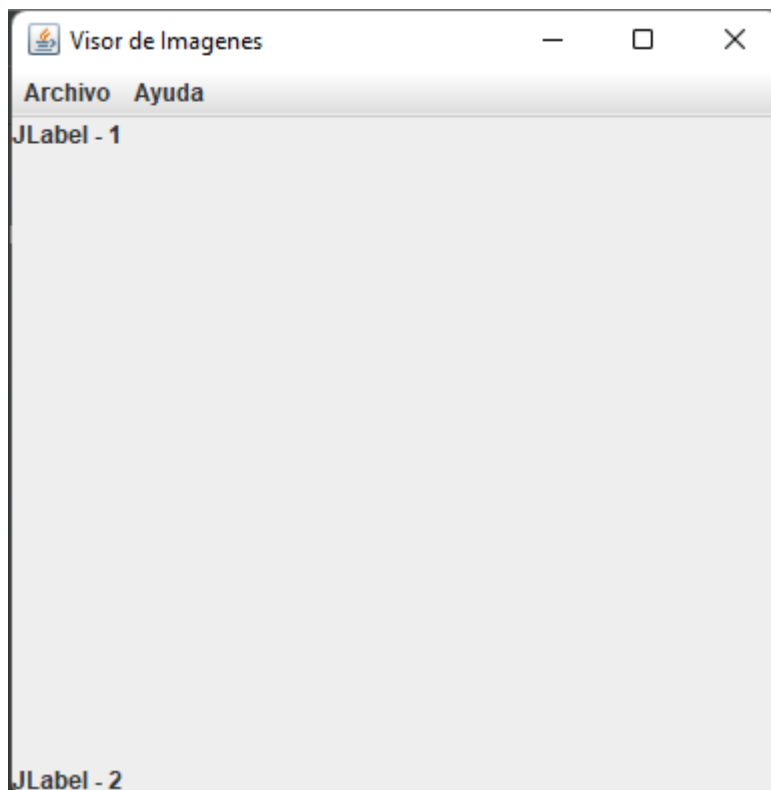
En el `JPanel` de la zona inferior tendremos anidados dos `JPanels` con un `BorderLayout` en `WEST` y `EAST` respectivamente, el `WEST` tendrá un `JPanel` anidado con un `GRIDLAYOUT` para poder tener los botones como se muestra, el `EAST` tendrá un `JPanel` pero con un `scroll vertical` y posiblemente con un `GRIDLAYOUT` para poder poner otros paneles y/o botones.

### Ejercicio 19

No se ha de hacer.

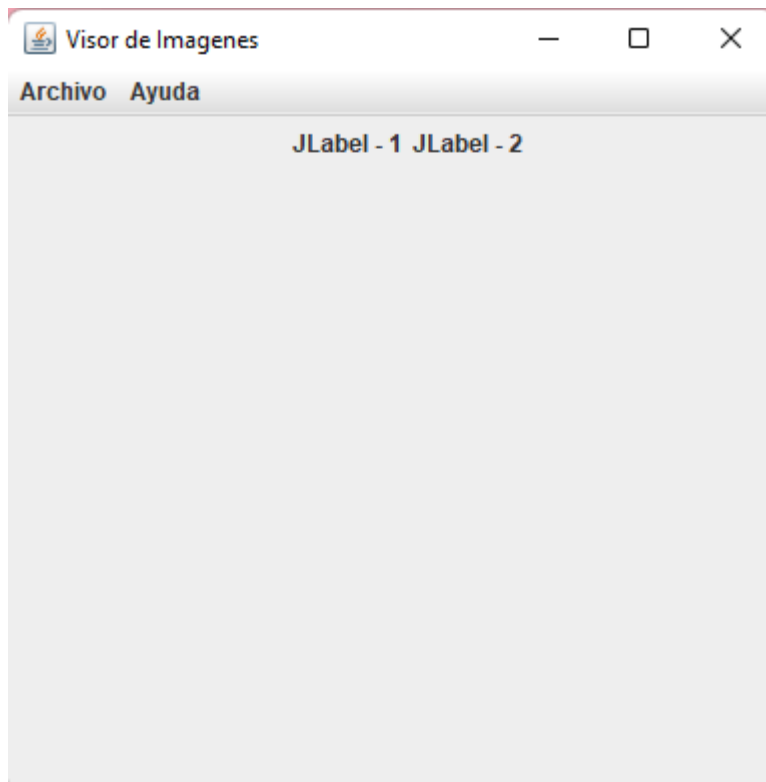
## Ejercicio 20

```
22     private void construirVentana() {  
23         JFrame ventana = new JFrame( title: "Visor de Imagenes");  
24  
25         Container panelContenedor = ventana.getContentPane();  
26         panelContenedor = ventana.getContentPane();  
27         panelContenedor.setLayout(new BorderLayout());  
28  
29         Container panelDeImagen = new JPanel();  
30         panelContenedor.add(panelDeImagen, BorderLayout.CENTER);  
31  
32         JLabel etiquetaNombreDeArchivo = new JLabel( text: "JLabel - 1");  
33         panelContenedor.add(etiquetaNombreDeArchivo, BorderLayout.NORTH);  
34  
35         JLabel etiquetaEstado = new JLabel( text: "JLabel - 2");  
36         panelContenedor.add(etiquetaEstado, BorderLayout.SOUTH);
```



FlowLayout()

```
panelContenedor.setLayout(new FlowLayout());  
Container panelDeImagen = new JPanel();  
panelContenedor.add(panelDeImagen);  
  
JLabel etiquetaNombreDeArchivo = new JLabel(text: "JLabel - 1");  
panelContenedor.add(etiquetaNombreDeArchivo);  
  
JLabel etiquetaEstado = new JLabel(text: "JLabel - 2");  
panelContenedor.add(etiquetaEstado);
```



GridLayout()

```
panelContenedor.setLayout(new GridLayout( rows: 2, cols: 2));  
Container panelDeImagen = new JPanel();  
panelContenedor.add(panelDeImagen);  
  
JLabel etiquetaNombreDeArchivo = new JLabel( text: "JLabel - 1");  
panelContenedor.add(etiquetaNombreDeArchivo);  
  
JLabel etiquetaEstado = new JLabel( text: "JLabel - 2");  
panelContenedor.add(etiquetaEstado);
```

