Ejercicio 11.11 Implemente el manejo de eventos del menú mediante clases internas, tal como lo discutimos aquí, en su propia versión del visor de imágenes.

```
        // Recording menu options
        JMenuItem recItem = new JMenuItem( text: "Comenzar a grabar");
        recItem.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_0, SHORTCUT_MASK));
        grabarArchivo ga = new grabarArchivo();
        recItem.addActionListener(ga);
        recMenu.add(recItem);
    }

    private class abrirArchivo implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
            openFile();
            System.out.println("Opening da file");
        }
    }
    private class grabarArchivo implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) { System.out.println("RECORDING"); }
    }
    private class salir implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
            System.out.println("Shutting down noises");
            quit();
        }
    }
}
```

Ejercicio 11.12 Abra el proyecto *visor-de-imagen-0-3* y examínelo: pruebe y lea su código. No se preocupe si no comprende todo el código porque algunas de las características nuevas son temas de esta sección. ¿Qué observa sobre el uso de las clases internas para permitir que el VisorDeImagen escuche y maneje los eventos?

Existen varias clases internas dentro de la clase ImageViewer, y estas són:

-El constructor de la clase

-openFile: Esta clase se ejecuta al hacer click sobre el botón de abrir archivo en el menú. Simplemente muestra por la consola un mensaje de “open file”.

-quit: Esta clase se ejecuta al hacer click en quit desde el menú. Tiene en su interior una función que finaliza la ejecución del programa actual.

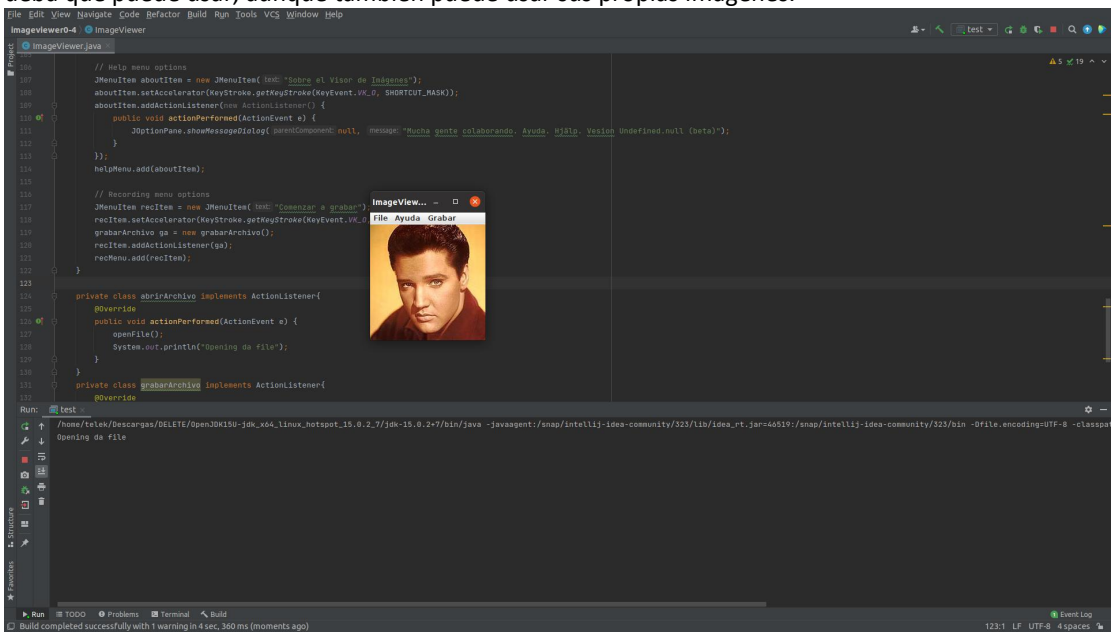
-makeFrame: Esta clase construye el panel que podemos visualizar al ejecutar el programa.

-makeMenuBar: Esta clase crea la barra de menu que se añade después al panel en makeFrame.

Ejercicio 11.13 Habrá notado que ahora, al activar el elemento *Salir* del menú, el programa finaliza. Examine cómo lo hace. Busque la documentación de la biblioteca relacionada con todas las clases y métodos involucrados.

La función salir utiliza la línea de código `System.exit(0)`. Esta función utiliza la clase System de java, y su función exit termina la ejecución de la máquina virtual de Java. El parámetro que se le pasa (0 en este caso) indica si el programa ha finalizado correctamente, o por el contrario se trata de un error. En el caso de 0, indica que ha finalizado satisfactoriamente, cualquier otro valor indica que se trata de un error.

Ejercicio 11.14 Abra y pruebe el proyecto *visor-de-imagen-0-4*. La carpeta de los proyectos de este capítulo incluye también una carpeta con imágenes. En este lugar puede encontrar algunas imágenes de prueba que puede usar, aunque también puede usar sus propias imágenes.



Ejercicio 11.15 ¿Qué ocurre cuando abre una imagen y luego cambia el tamaño de la ventana? ¿Qué ocurre si primero cambia el tamaño de la ventana y luego abre una imagen?

Si se abre una imagen y luego se cambia el tamaño de la ventana, la imagen se queda anclada a la esquina superior izquierda, y el resto de la ventana se rellena con el color de fondo de la ventana, gris por defecto.

Si se cambia el tamaño de la ventana y luego se abre una imagen no pasa nada, pues al abrir la imagen la ventana se redimensiona al tamaño de la imagen.

Ejercicio 11.16 Continuando con su última versión del proyecto, use el fragmento de código que se muestra arriba para agregar las dos etiquetas. Pruébelo. ¿Qué observa?

Lo que pasa es que se crea una ventana más pequeña en la que sólo se muestran la barra de menú y la última etiqueta añadida, y la ventana se ajusta al tamaño de ambas.

Ejercicio 11.17 Observe la IGU del proyecto calculadora que usamos en el Capítulo 6 (Figura 6.7 en pagina 188). ¿Qué tipo de contenedores y de gestores de disposición cree que se usaron? Después de responder por escrito, abra el proyecto *calculadora-gui* y controle su respuesta leyendo el código.

Creo que para el teclado se utilizó una gridLayout, la cabecera simplemente es un JTextField, y el conjunto es un BorderLayout o CardLayout.

Leyendo el código, el teclado es un GridLayout, el conjunto un Border Layout, y la cabecera es un JTextField.

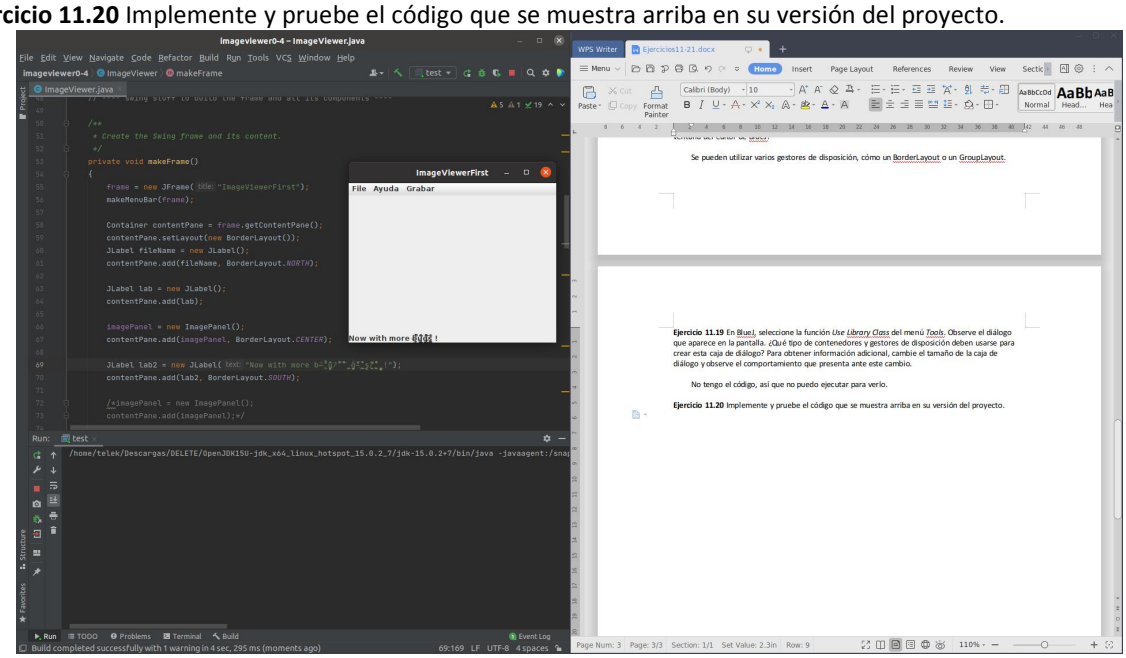
Ejercicio 11.18 ¿Qué tipo de gestores de disposición habría que usar para crear el esquema de la ventana del editor de BlueJ?

Se pueden utilizar varios gestores de disposición, cómo un BorderLayout o un GroupLayout.

Ejercicio 11.19 En BlueJ, seleccione la función *Use Library Class* del menú *Tools*. Observe el diálogo que aparece en la pantalla. ¿Qué tipo de contenedores y gestores de disposición deben usarse para crear esta caja de diálogo? Para obtener información adicional, cambie el tamaño de la caja de diálogo y observe el comportamiento que presenta ante este cambio.

No tengo el código, así que no puedo ejecutar para verlo.

Ejercicio 11.20 Implemente y pruebe el código que se muestra arriba en su versión del proyecto.



Ejercicio 11.21 Experimente con otros gestores de disposición. Pruebe en su proyecto todos los gestores de disposición mencionados anteriormente y también pruebe si se comportan como se espera.

FlowLayout: Comportamiento esperado. Al reducir el tamaño el texto se dispone debajo de la imagen.

GridLayout: No esperaba el espacio en blanco a la izquierda. Por lo demás, todo era lo que esperaba.

BoxLayout: El resultado es el que esperaba.