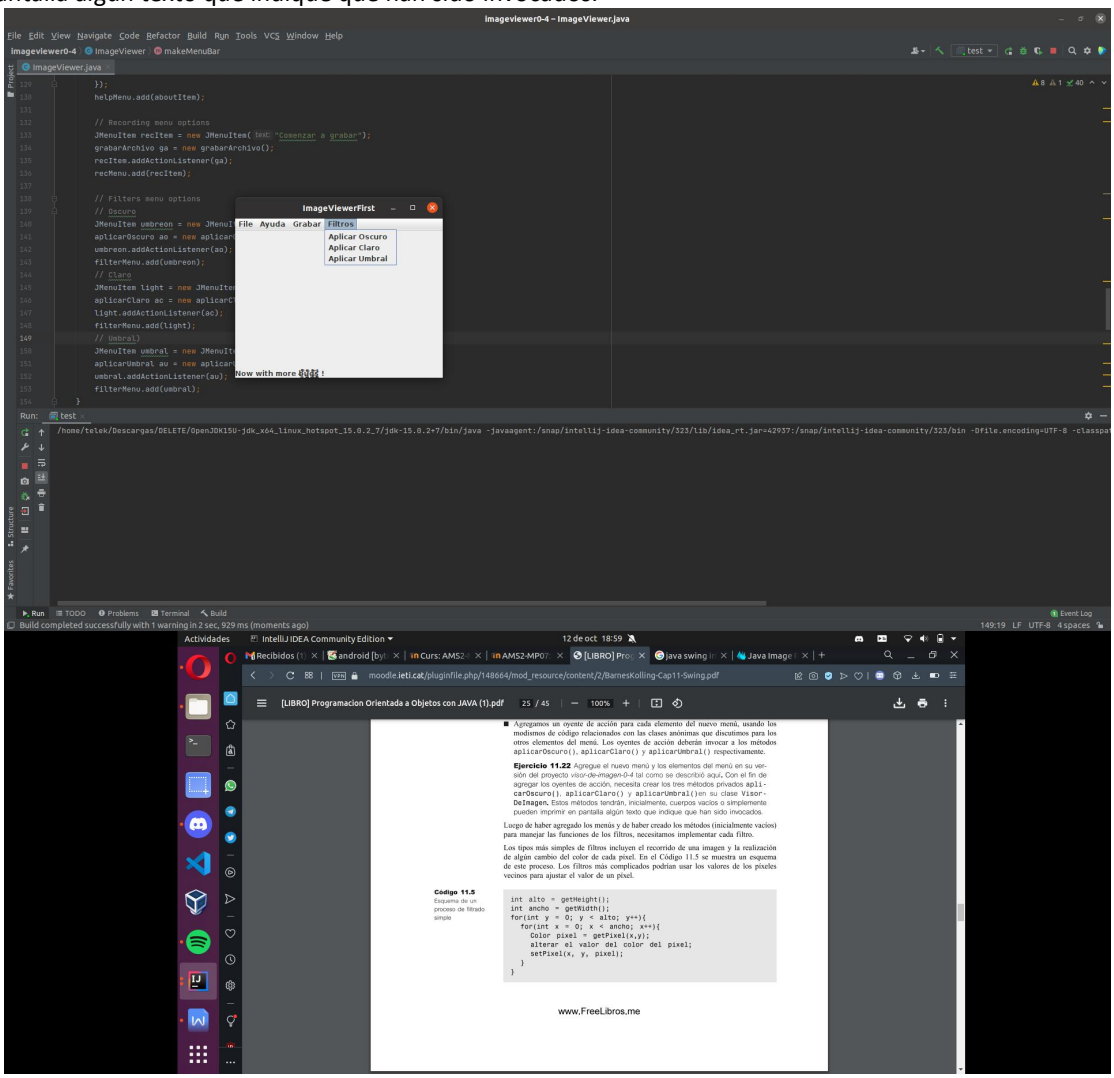
Ejercicio 11.22 Agregue el nuevo menú y los elementos del menú en su versión del proyecto *visor-de-imagen-0-4* tal como se describió aquí. Con el fin de agregar los oyentes de acción, necesita crear los tres métodos privados aplicarOscuro(), aplicarClaro() y aplicarUmbral() en su clase VisordeImagen. Estos métodos tendrán, inicialmente, cuerpos vacíos o simplemente pueden imprimir en pantalla algún texto que indique que han sido invocados.



Ejercicio 11.23 ¿Qué hace la llamada a método ventana.repaing(), que se puede ver en el método aplicarOscuro?

> La llamada a método frame.repaint() lo que hace es refrescar el contenido de la ventana, haciendo una llamada a las funciones .paint() en todos los componentes que hay en su interior.

Ejercicio 11.24 Podemos ver una llamada al método mostrarEstado que es, claramente, una llamada a un método interno. A partir del nombre podemos suponer que este método debe mostrar un mensaje de estado usando la etiqueta de estado que hemos creado anteriormente. Implemente este método en su versión del proyecto *visor-de-imagen-0-4*. (*Pista*: busque el método setText en la clase JLabel.)

```
private JFrame frame;
private ImagePanel imagePanel;

public JLabel status;

public void setStatus(String newStatus) {
    this.status.setText(newStatus);
}
```

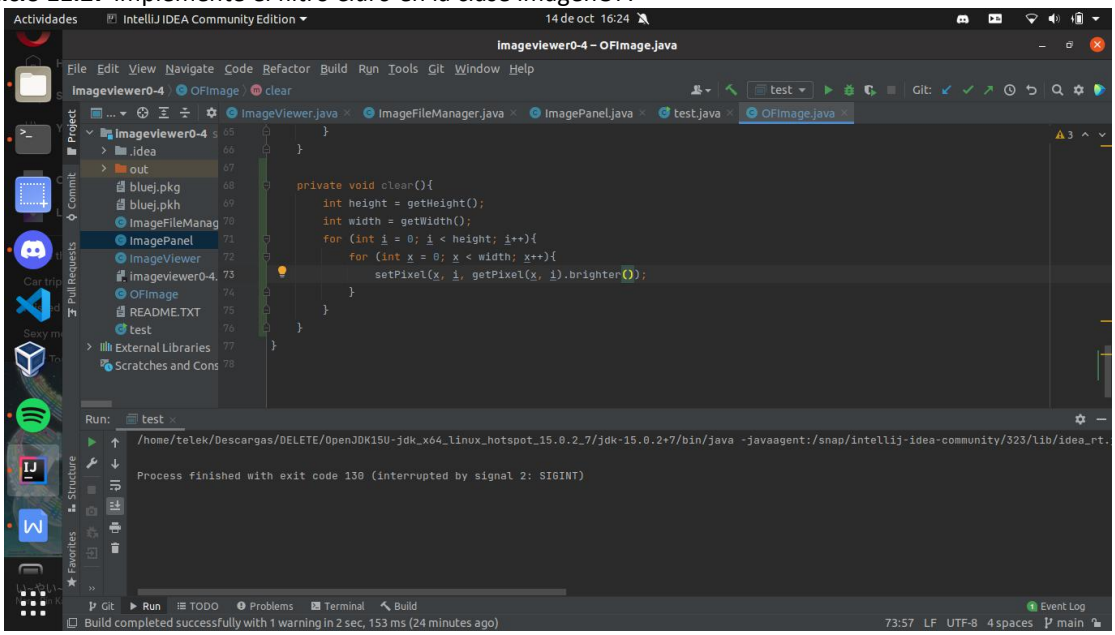
Ejercicio 11.25 ¿Qué ocurre cuando se selecciona el elemento *Oscuro* del menú si no hay ninguna imagen cargada?

Lo que pasa es que la etiqueta de estado indica que no hay ninguna imagen seleccionada.

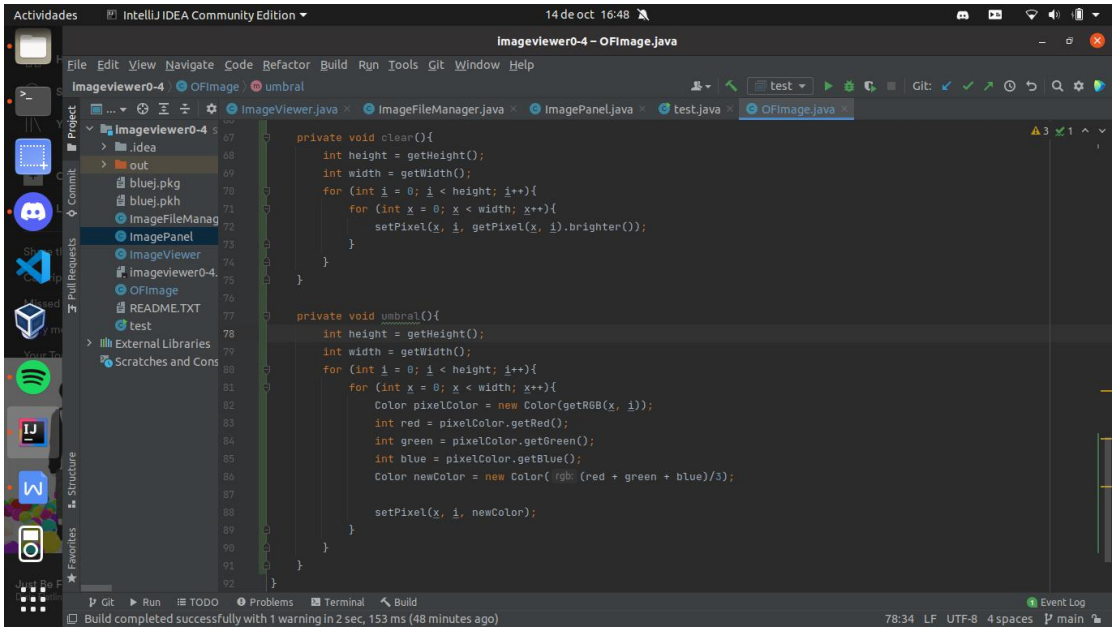
Ejercicio 11.26 Explique detalladamente cómo dunciona el método oscure de ImagenOF. (*Pista:* contiene una llamada a otro método de nombre darker. ¿A qué clase pertenece este método? Investigue.)

El método oscure de ImagenOF lo que hace es un bucle a través cada uno de los pixeles de la imagen y los va oscureciendo uno a uno con la función darker, que devuelve un nuevo color más oscuro dado un valor (el valor actual del pixel).

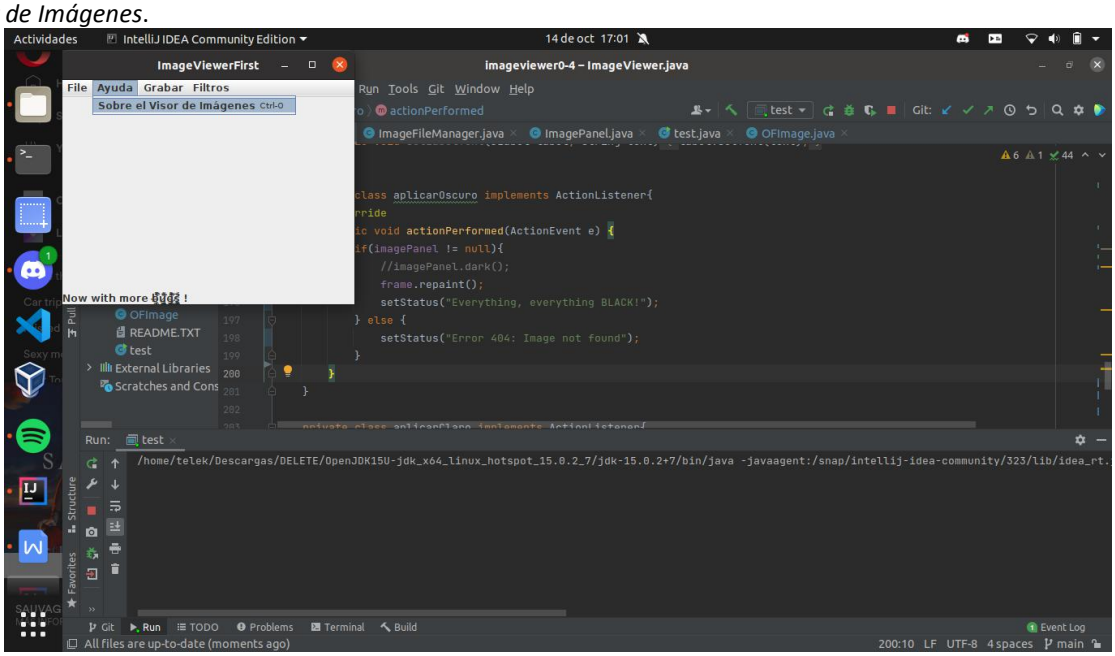
Ejercicio 11.27 Implemente el filtro *Claro* en la clase ImagenOF.



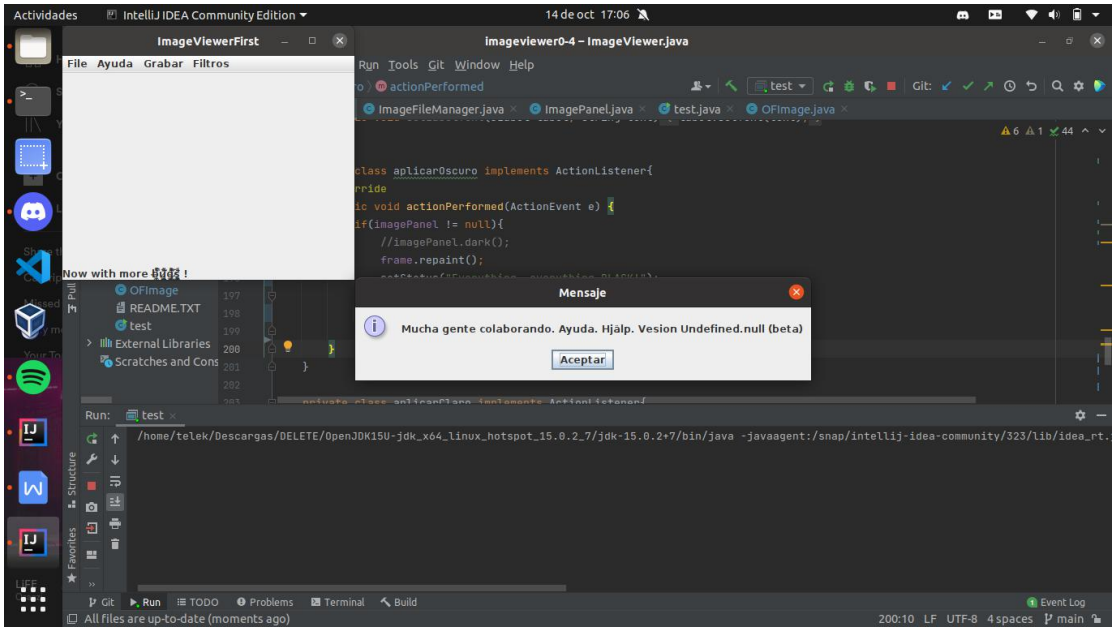
Ejercicio 11.28 Implemente el filtro *Umbral*. Para determinar el brillo de un píxel puede obtener sus valores de rojo, verde y azul y promediarlos. La clase Color define referencias estáticas que se ajustan a objetos de color negro, blanco y gris.



11.29 Agregue nuevamente un menú *Ayuda* y un elemento en este menú con la etiqueta *Acerca del visor de Imágenes*.



Ejercicio 11.30 Agregue un método con su cuerpo vacío, de nombre mostrarAcercaDe() y agregue un oyente de acción para el elemento del menú *Acerca del Visor de Imágenes...* que invoque a este método.



Ejercicio 11.31 Busque la documentación de showMessageDialog. ¿Cuántos métodos hay con este nombre? ¿Cuáles son las diferencias entre ellos? ¿Cuál podríamos usar? ¿Por qué?

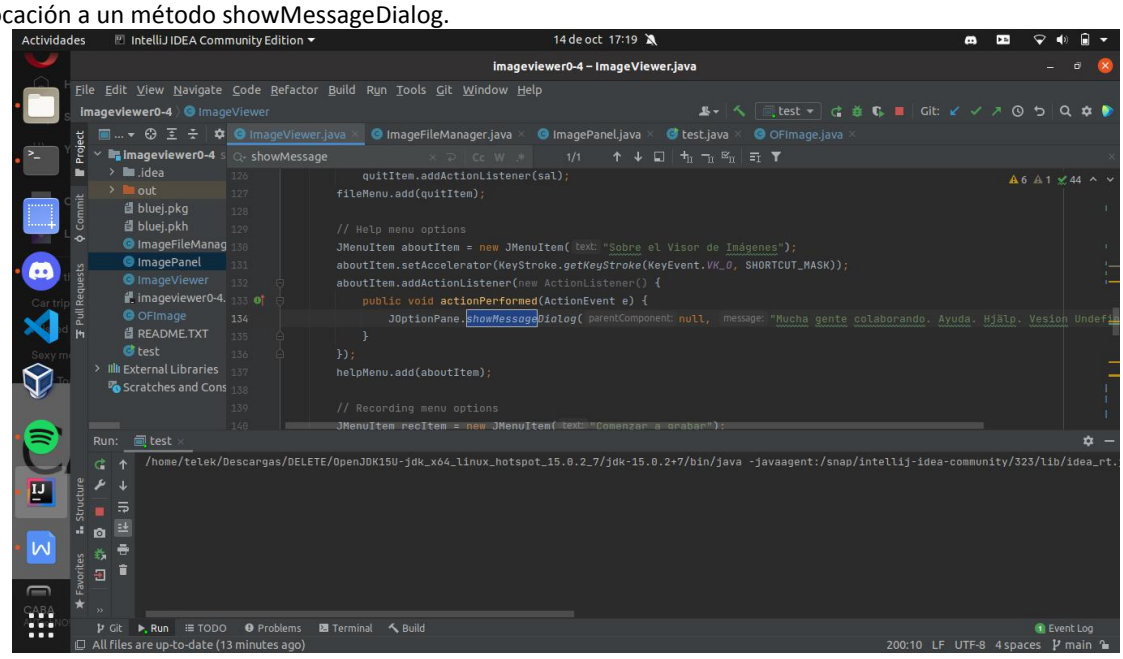
Hay tres métodos con este nombre.

showMessgeDialog(component, string) muestra una ventana emergente informativa con un botón de OK y el mensaje que se especifique en object.

showMessageDialog(component, String, String, object) muestra una ventana emergente con un botón de OK, el texto del segundo parámetro en el mensaje, el tercer parámentro será el título de la ventana y el objeto final el tipo de panel, por ejemplo JOptionPane.PLAIN_MESSAGE

showMessageDialog(component, string, string, object, Icon). Funciona igual que el anterior, pero el icono final será el simbolo que suele aparecer de información, error, etc. Se sustituye.

Ejercicio 11.32 Implemente el método mostrarAcercaDe en su clase VisorDeImagen usando una invocación a un método showMessageDialog.



Ejercicio 11.33 Los métodos showInputDialog del JOptionPane permiten solicitar al usuario el ingreso de algún dato, cuando se requiera. Por otra parte, el componente JTextField permite mostrar una zona permanente para el ingreso de texto en una IGU. Busque la documentación de esta clase. ¿Qué ingresos provocan que se notifique un ActionListener asociado con un JTextField? ¿Se puede impedir que el usuario edite el texto del campo? ¿Es posible que un oyente se notifique de cambios arbitrarios del campo de texto? (*Pista:* ¿qué uso hace un JTextField de un objeto Document?)

El actionListener detecta todos los cambios que se realizan sobre el objeto en el cual se implementa.

Se puede impedir que el usuario edite el texto.

Los oyentes se notifican de todos los cambios.