Diseño de Interfaces de Usuario.

Práctica 1

Caso de estudio: Reglas de oro y test de usabilidad

Brian Palmés Gómez
78593384Z

TAREAS

Tarea ULPGC vs. MIT

Teniendo en mente únicamente las reglas de oro, si ya dispusieras del Grado en Ingeniería Informática y accedieras a la web de la ULPGC buscando las ofertas de trabajo disponibles para tu perfil.

¿Encuentras la información? ¿Cómo valoras su accesibilidad y presentación?

En la web de la ULPGC se encuentra fácilmente la bolsa de trabajo en la propia página de inicio y sin necesidad de navegar por la web.
Una vez haces click sobre el texto que señala la bolsa de trabajo te lleva a otra página donde están agrupados los diferentes empleos que oferta la ULPGC.

El hecho de que se encuentre en el inicio hace que su accesibilidad sea buena ya que no requiere buscarla y su presentación es muy buena ya que no desentona con el carácter informativo de la web de la ULPGC cuya página de inicio en general muestra mucha información de forma sencilla e intuitiva pero sin sobrecargar la web de opciones que puedan dificultar al usuario la navegación en caso de querer encontrar algo que no vea a primera vista.

Reglas de Oro.

Consistencia: Totalmente de acuerdo

Usabilidad universal: Totalmente de acuerdo

Informativo: Totalmente de acuerdo

Flujo y cierre: Totalmente de acuerdo

Prevenir errores: Totalmente de acuerdo

Deshacer: Totalmente de acuerdo

Sentir el control: De acuerdo

Memoria a corto plazo: De acuerdo

La ULPGC es una universidad relativamente pequeña y joven, **¿qué ocurre si intentamos realizar la misma tarea en una universidad de gran tradición, prestigio y presupuesto como el MIT?**

Tras intentar localizar posibles ofertas de trabajo para tu perfil con ambas webs, evalúa ambas con las reglas de oro, visualizando la comparativa entre ambas con un diagrama de Kiviat.

MIT

<u>Reglas de Oro.</u>

Consistencia: Totalmente de acuerdo

Usabilidad universal: Totalmente de acuerdo

Informativo: De acuerdo

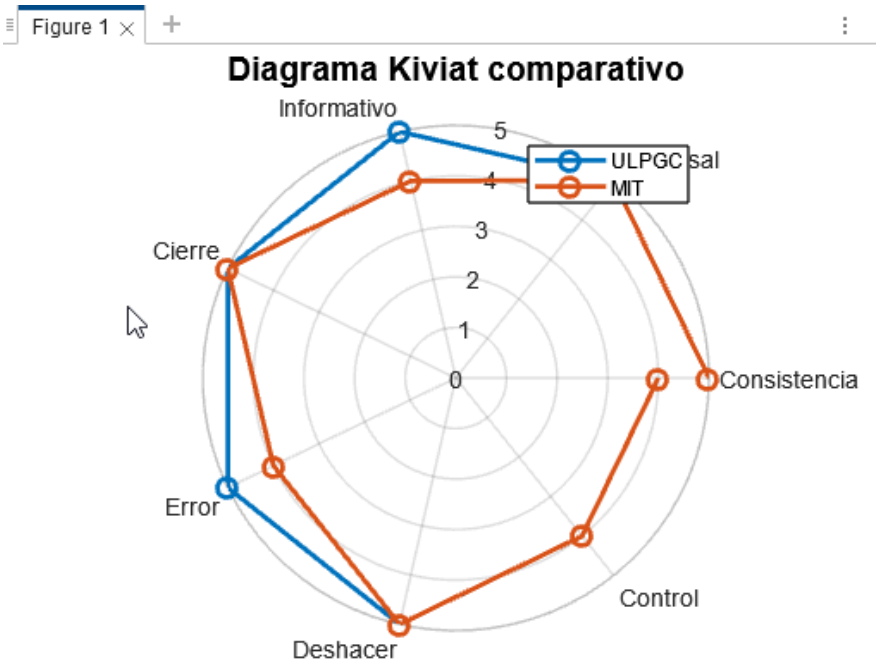
Flujo y cierre: Totalmente de acuerdo

Prevenir errores: De acuerdo

Deshacer: Totalmente de acuerdo

Sentir el control: De acuerdo

Memoria a corto plazo: De acuerdo



Tarea accedaCRIS

AccedaCRIS es el portal, de reciente creación, dedicado a divulgar y dar visibilidad a la producción científica y académica de la ULPGC. La segunda tarea a realizar consiste en hacer uso de accedaCRIS para obtener algunas respuestas, dando la libertad de realizar otras tareas en el portal de cara a una evaluación de la usabilidad del mismo.

¿Cuántos TFG se han presentado en la EII en 2019?

2852 Trabajos de Final de Grado en 2019.

¿Cuántos TFG se presentaron en la EII tutorizados por alguno de los profesores de DIU durante 2019?

Modesto Castrillón Santana tutoriza 3 TFG en 2019 .

Práctica 2

Introducción a Java Swing GUI

TAREA

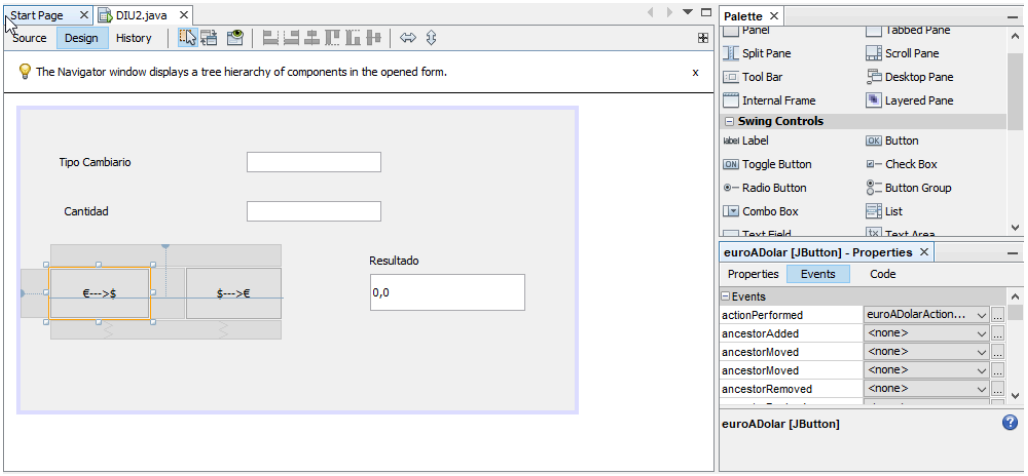
Conversor de divisas.

El objetivo de esta tarea es que el alumnado comience a programar utilizando el Java GUI Builder integrado en Netbeans para el desarrollo de interfaces de usuario en Java utilizando los componentes de Swing.

Para ello, se **debe diseñar e implementar una aplicación que permita convertir de euros a dólares estadounidenses y a la inversa, utilizando únicamente componentes de tipo botón y campos de texto.**

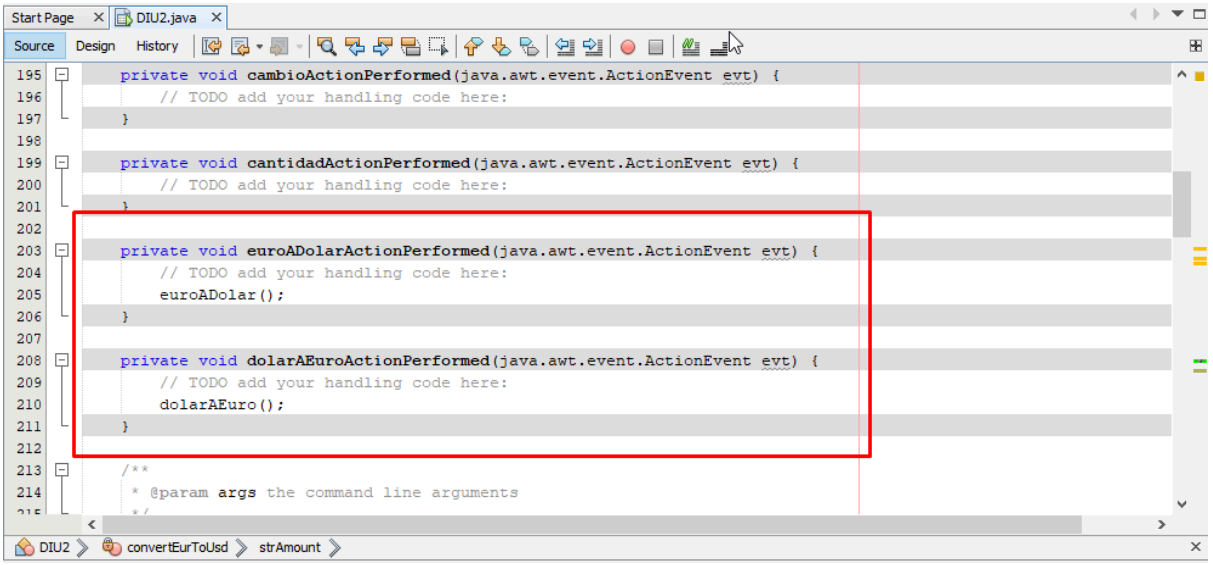
La aplicación deberá permitir al usuario:
-Introducir la equivalencia de 1 euro en dólares.
-Introducir una cantidad en euros o dólares y obtener su equivalencia en la otra divisa.
-El resultado se debe expresar con dos dígitos decimales (correspondientes a los céntimos).

Como se pide en el enunciado de esta práctica usaremos el Java GUI Builder de Java Swing
para implementar nuestro sencillo conversor de divisas usando únicamente componente tipo botón y textfield.



El builder nos permite diseñar interfaces en Java y añadir componentes a modo de constructor por bloques que consiste en arrastrar los componentes gráficos que vas a usar en tu programa a un lienzo del que java generará un código al que luego añadiremos eventos y acciones personalizadas.

Por ejemplo como se ve en la imagen el botón del diseño de euros a dólar tiene asignado un evento acctionPerformed a ese evento le asignaremos un código que se ejecutará cuando el usuario pulse el botón.



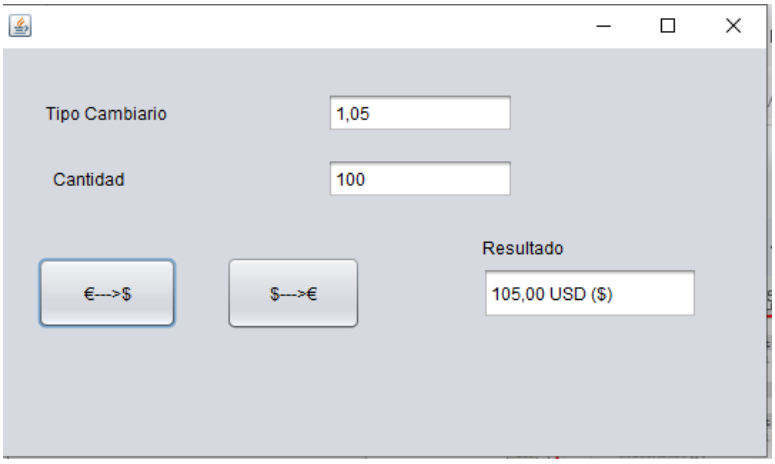

```
  //Logic functions
private void dolarAEuro() {
    //dummy
    this.resultado.setText(String.format("%,.2f", this.convertUsdToEur())+ " EUR (€)");
    this.cantidad.setText(cambioATexto());

}

private void euroADolar() {
    this.resultado.setText(String.format("%,.2f", this.convertEurToUsd()) + " USD ($)");
    this.cantidad.setText(cambioATexto());

}
```

Quedando así el diseño final de este sencillo conversor de euros a dolar y dolar a euros.



Código: https://github.com/bpalmes/DIU/tree/main/DIU1
***El código es el de la práctica dos aunque el proyecto y la carpeta se llamen DIU1**

Práctica 3.

Java Swing. Paneles

TAREA.

El objetivo de esta práctica es introducir los paneles así como los componentes de la barra deslizante y área de texto. La aplicación deberá mostrar en un área de texto los elementos de una matriz cuadrada de dimensión 10 × 10 que superen el valor seleccionado mediante la barra de deslizamiento. Es decir, cada vez que el usuario mueva el cursor de la barra deslizante,

Sólo se mostrarán en el área de texto aquellos elementos de la matriz que tengan un valor superior al seleccionado. Para su diseño se deben utilizar únicamente los componentes Swing vistos en las clases de prácticas.

A modo de resumen, la aplicación deberá permitir al usuario:

Introducir los valores máximos y mínimos que contiene la matriz de enteros.

Mostrar la matriz cuadrada en un área de texto con valores enteros generados aleatoriamente entre los valores máximo y mínimo introducidos, ambos inclusive.

Incluir una barra deslizante con marcas y etiquetas de valores que permita seleccionar dinámicamente el umbral.

De esta forma los elementos de la matriz con valor mayor que el seleccionado se mantienen visibles, y los elementos con valor menor o igual que el seleccionado se mostrará como un guión (-) en el área de texto.

Hay que indicar que:
Se debe separar en dos paneles diferentes los elementos que configuran el resultado del área de texto donde se muestra la matriz filtrada.

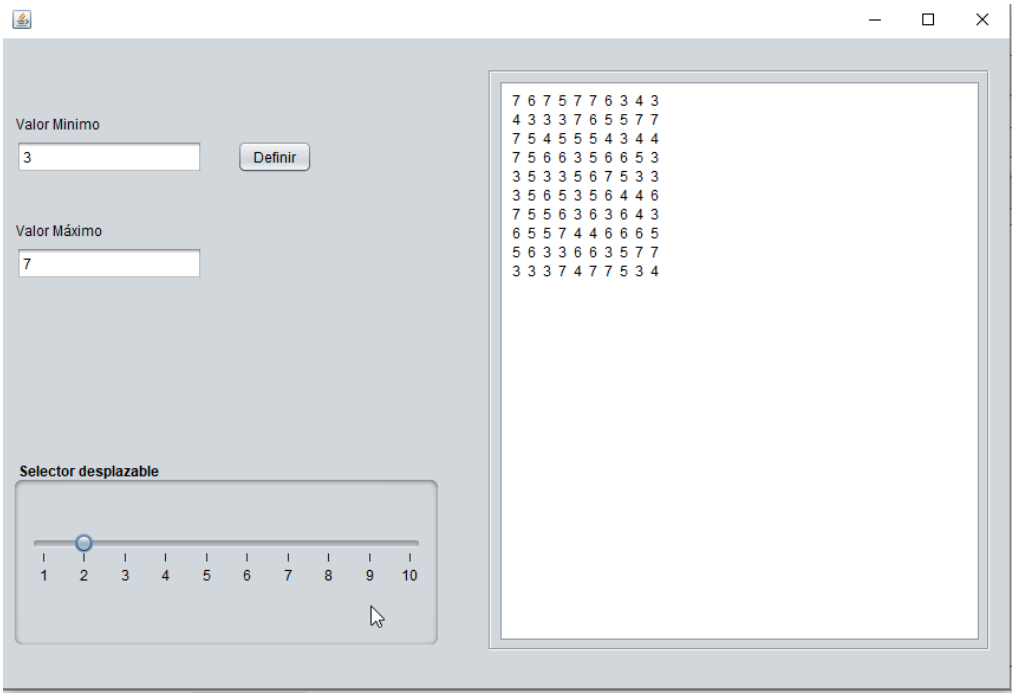
No se dispondrá de ningún botón para comenzar el procesamiento.

Cualquier cambio del valor máximo o mínimo implica inicializar y mostrar la nueva matriz.
Los cambios en el umbral usando la barra de deslizamiento solo implicarán el filtrado

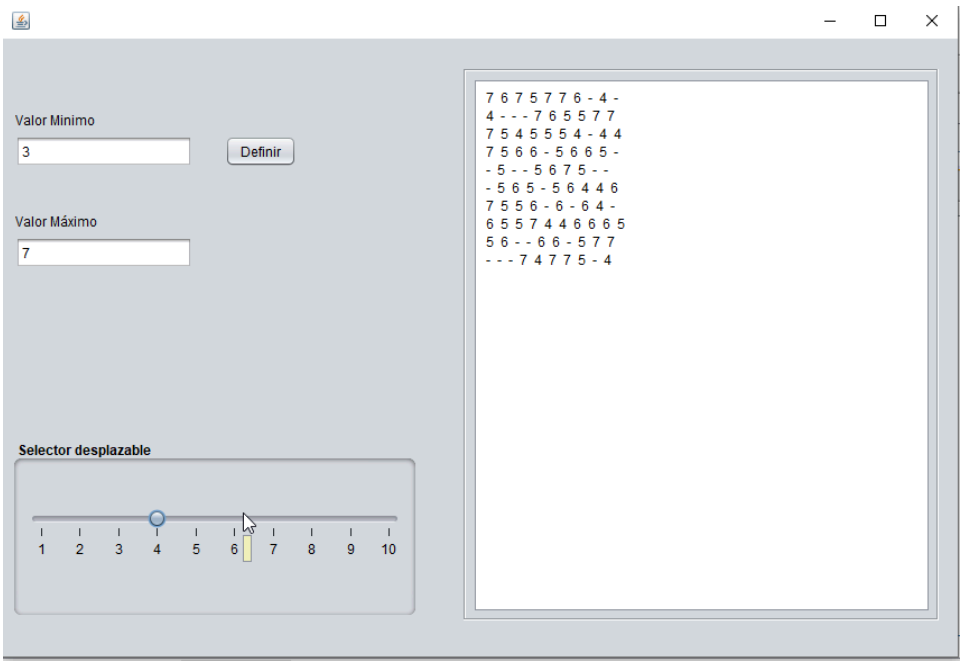
de los valores menores y la actualización del área de texto.

Se valorará en la evaluación de la práctica:
El correcto funcionamiento de la aplicación.
La disposición de los elementos de la interfaz.
La facilidad de interpretación de la funcionalidad de la aplicación.
La redacción de la memoria.

A continuación vemos la interfaz ya implementada se le introducen los valores 3 y 7 como máximos y mínimos y una vez pulsamos en definir se genera un matriz aleatoria con valores entre los máximos y mínimos definidos.



Después usamos la barra deslizante para tapar los valores como se pide en la práctica y a medida que desplazamos los valores por debajo del valor señalado por el panel deslizante irán desapareciendo en la matriz en tiempo real y sustituidos por guiones.
po real y sustituidos por guiones.



código :https://github.com/bpalmes/DIU/tree/main/DIU**3**

Práctica 4

Java Swing. Combo Box y primitivas de dibujo

TAREA

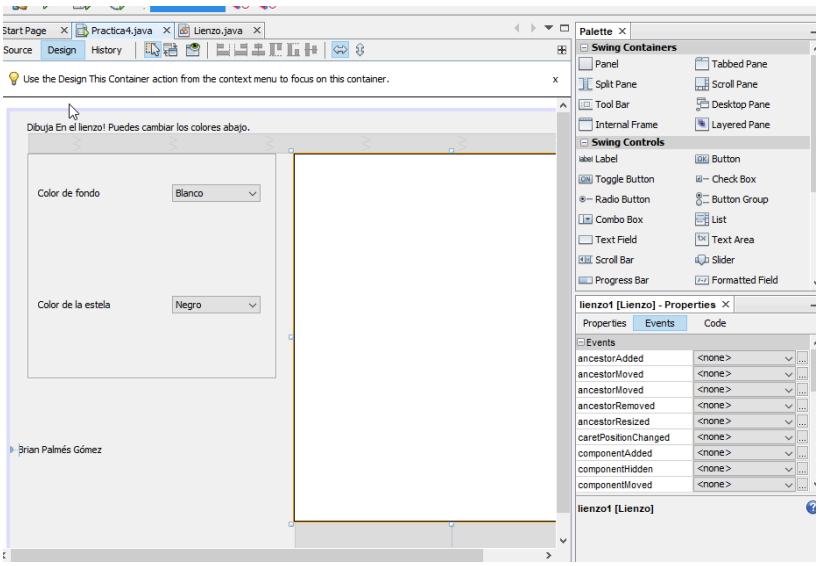
Dibujo de la estela del ratón

El objetivo de esta práctica es desarrollar una aplicación en Java que permita seleccionar al usuario el color tanto del fondo como de la estela y mostrar la estela del ratón mientras se mueve.

La misma estará compuesta por 5 círculos del color seleccionado y sobre un fondo también con el color seleccionado para el mismo. En el campus virtual de la asignatura está disponible un vídeo con una breve demostración.

Tanto para el color del fondo como de la estela, el usuario tendrá al menos tres opciones diferentes donde elegir utilizando un combo box.

Siguiendo el vídeo del profesor la tarea consiste en casi lo mismo, se crea un Jframe como el de la imagen donde se añaden los elementos y los comboBox con las diferentes opciones así como el Lienzo que será un JPanel donde se dibuja la estela del ratón.



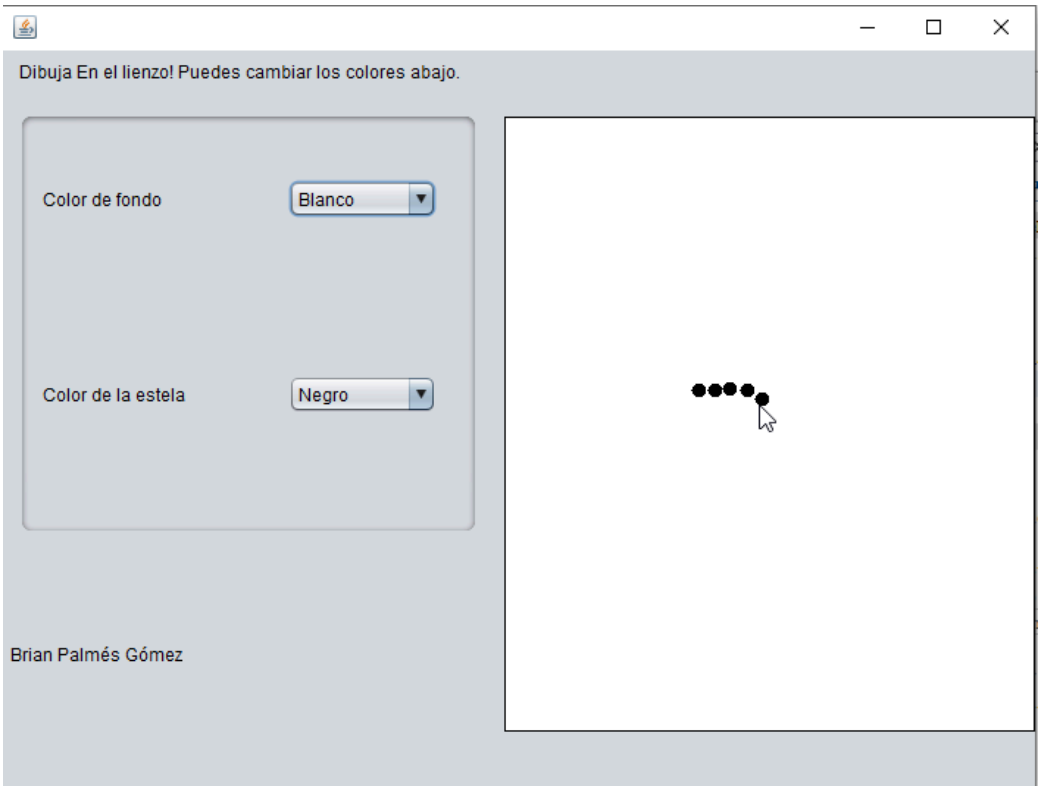
En la clase principal se recogerán los eventos y las acciones pertinentes de cada elemento tanto las diferentes elecciones de los comboBox como los eventos asociados al ratón sus coordenadas para dibujar la estela.
Mientras que en la clase lienzo que extenderá de un JPanel se encargará de dibujar la estela y el fondo según las opciones elegidas por el usuario.

```
    private void colorFondoActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        lienzo1.pintaLienzo(colorFondo.getSelectedIndex());
    }

    private void lienzo1MouseMoved(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        if(retraso < 10){
            retraso++;
        }else{
            lienzo1.coordenadas(evt.getX(), evt.getY());
            retraso = 0;
        }
    }

    private void colorEstelaActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        lienzo1.pintaPincel(colorEstela.getSelectedIndex());
    }
```

A continuación una captura del diseño final



código :https://github.com/bpalmes/DIU/tree/main/DIU6

Práctica 5

Java Swing. Check Box, Radio Button e Imágenes.

TAREA

El objetivo de esta práctica será desarrollar una aplicación que muestre una imagen a lacual se le podrá activar o desactivar la visualización de sus canales de color (rojo, verde y azul).

Para ello, se dispondrán cuatro controles tipo Check Box etiquetados como "Todos","Rojo","Verde" y "Azul".

Al principio todos los Check Boxes estarán activados y la imagen se verá normalmente. Si se desactiva alguno de los controles rojo, verde o azul, la imagen se mostrarásin esos canales de color.

Al menos uno de los canales rojo, verde o azul siempre estará activo para evitar que la imagen se vea completamente negra. Al desactivarse alguno de los CheckBoxes de color, el Check Box "Todos" también se desactivará automáticamente.

Si se activa el Check Box "Todos", el resto de los canales se activarán automáticamente. No seráposible desactivar todos los canales de color desactivando el Check Box "Todos".

Sobre esta imagen se añadirá un logotipo (una imagen PNG con transparencia) situado en alguna de las cuatro esquinas de la imagen anterior.

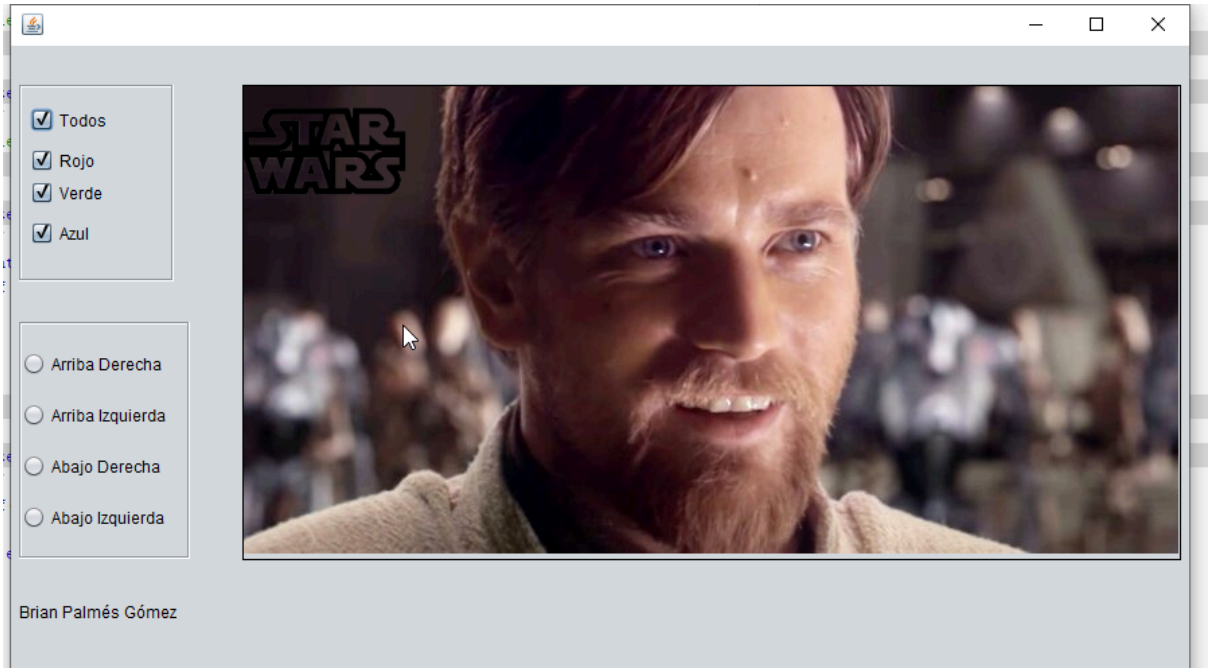
La esquina será seleccionada por uno de los cuatro Radio Button que también deben añadirse. Inicialmente, el logotipo aparecerá en la esquina superior derecha.

Por tanto la aplicación deberá permitir al usuario:

Mostrar una imagen de tamaño máximo de 1024x768.
Seleccionar qué componentes de color (rojo, verde, azul) de la imagen se mostrarán, mostrando siempre al menos una componente.
Seleccionar todas componentes con un Check Box para restaurar la visualización de la imagen original.
Mostrar un logotipo que el usuario podrá elegir en cuál de las cuatro esquinas de la imagen aparece.
Se valorará en la evaluación de la práctica:
El correcto funcionamiento de la aplicación.
La disposición de los elementos de la interfaz.
La facilidad de interpretación de la funcionalidad de la aplicación.
La redacción de la memoria

Práctica Realizada:

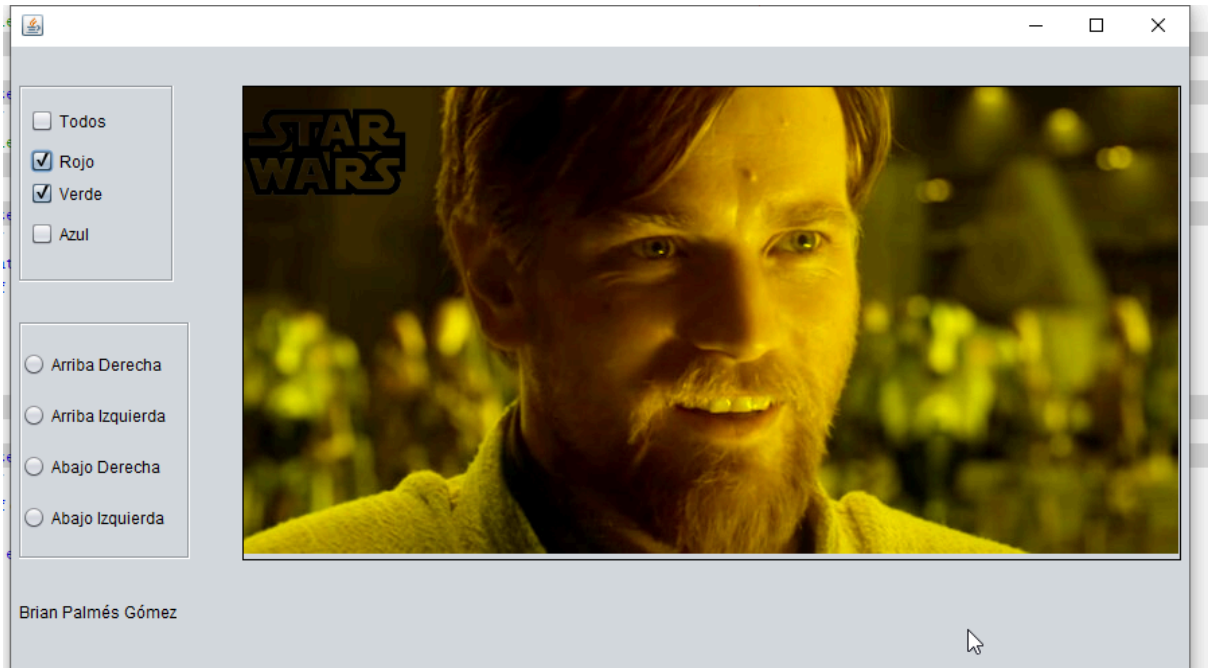
1. Captura de la práctica completa.

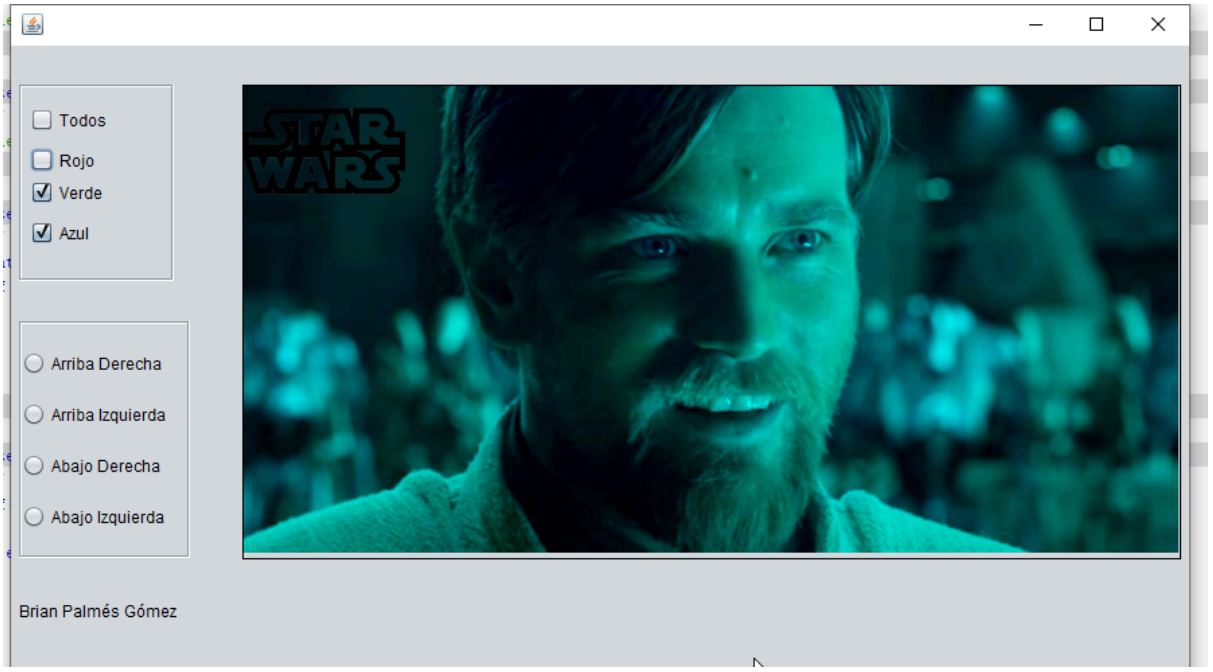


En la captura anterior vemos el diseño de la interfaz y como se pide en el enunciado podemos ver la imagen con todos los colores activados y el check box con las opciones para desactivar los colores y debajo los radiobutton con las opciones para mover el logo transparente por la foto.

2.CheckBox

En la siguientes capturas podemos ver como el checkbox va desactivando el color de según desmarcamos las opciones teniendo que tener siempre una marcada.





Para la implementación del checkbox hemos seguido las instrucciones del video de la práctica y usado el archivo utils para esta práctica proporcionado por los profesores.

Como en prácticas anteriores java nos proporciona clases para controlar los eventos de la interfaz gráfica en el caso del checkbox los eventos se gestionan con la clase ItemEvent.

Luego recogemos en una variable el estado del checkbox y cuando el estado cambia según qué checkbox haya cambiado se cambia el valor booleano.

```
private void VerdeItemStateChanged(java.awt.event.ItemEvent evt) {
    // TODO add your handling code here:
    if(!Rojo.isSelected() && !Azul.isSelected()){
        Verde.setSelected(true);
    } else {
        int state = evt.getStateChange();
        if (state != ItemEvent.SELECTED) {
            todos.setSelected(false);
        }
        lienzol.pinta(Rojo.isSelected(), Verde.isSelected(), Azul.isSelected());
    }
}
```

Por ejemplo en esta imagen vemos la gestión del evento cuando hay un cambio en el checkbox del botón verde.

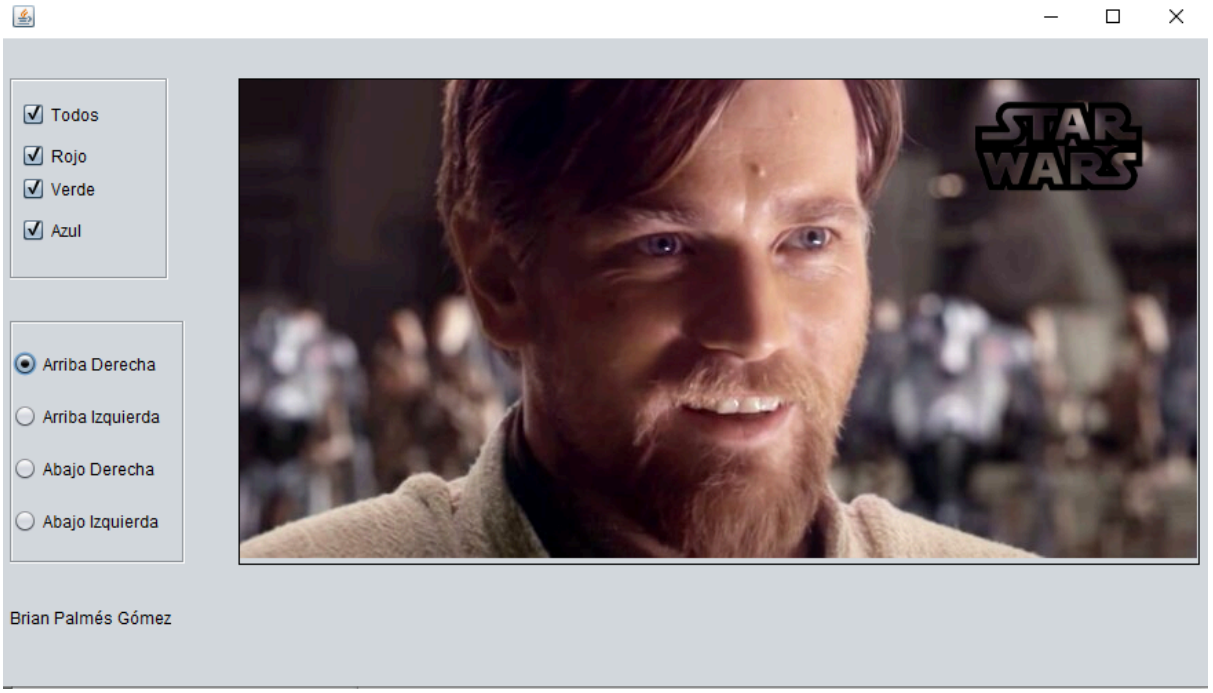
El método pinta de la clase lienzo usa las clases auxiliares para cambiar los colores y se le pasa el valor booleano de los checkbox al estar o no pulsado.

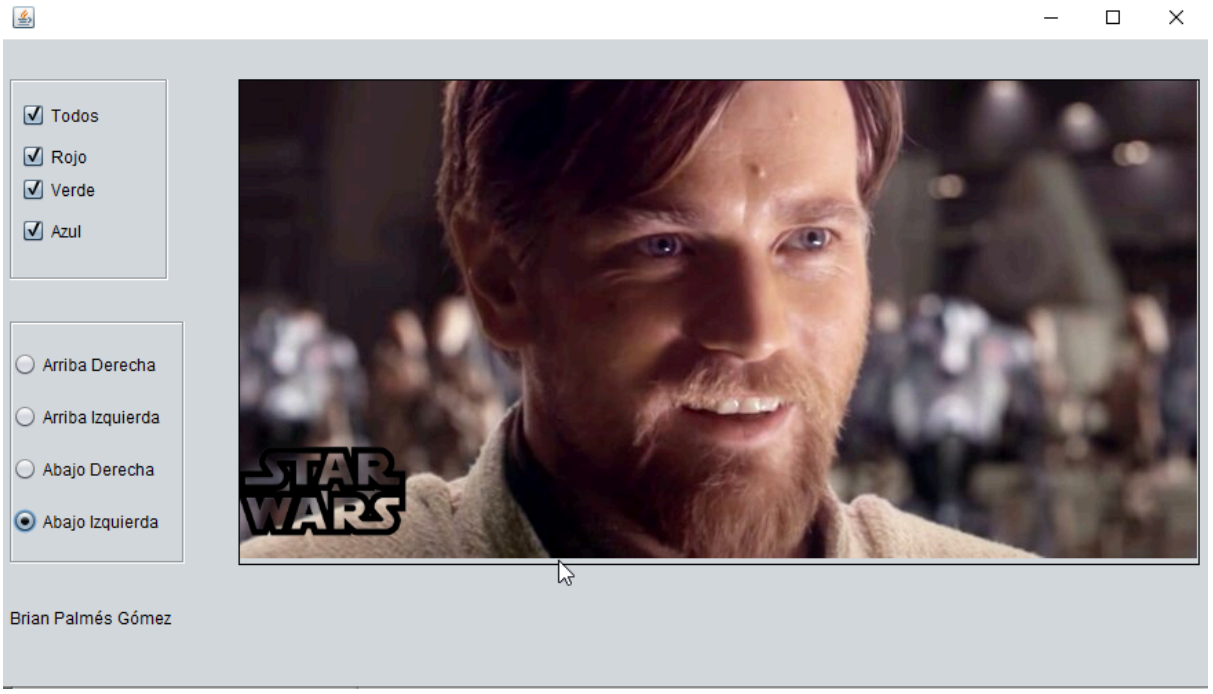
El If inicial controla primero que no puedas quitar todos los colores en este caso si los otros dos checks no están activados verde nunca se pondrá en false.

El else desactiva el checkbox “todos” si no hay alguno seleccionado.

3. Radio Button.

Por otro lado se nos pide que cuando cambies la opción en el radio button el logo se mueva a la posición elegida por la pantalla. La diferencia con el Checkbox es que solo puedes marcar una de las opciones





Para su implementación lo más relevante es el manejador de eventos que cambia las coordenadas del logo al ser seleccionado esa opción usando las clases auxiliares el método coordenadas repinta el JPanel con el logo posicionado en las nuevas coordenadas.

```
private void downDerechaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    lienzol.coordenadas(535, 250);
}
```

```
}
@Override
public void paintComponent(Graphics g){
    this.g = g;
    super.paintComponent(g);
    g.drawImage(picture, 0, 0, null);
    g.drawImage(logo, x, y, null);
}
public void coordenadas(int x, int y) {
    this.x = x;
    this.y = y;
    this.repaint();
}
```

código :https://github.com/bpalmes/DIU/tree/main/DIU5

Práctica 6

Java Swing. Barra de menú y ventanas de diálogo.

Tarea.

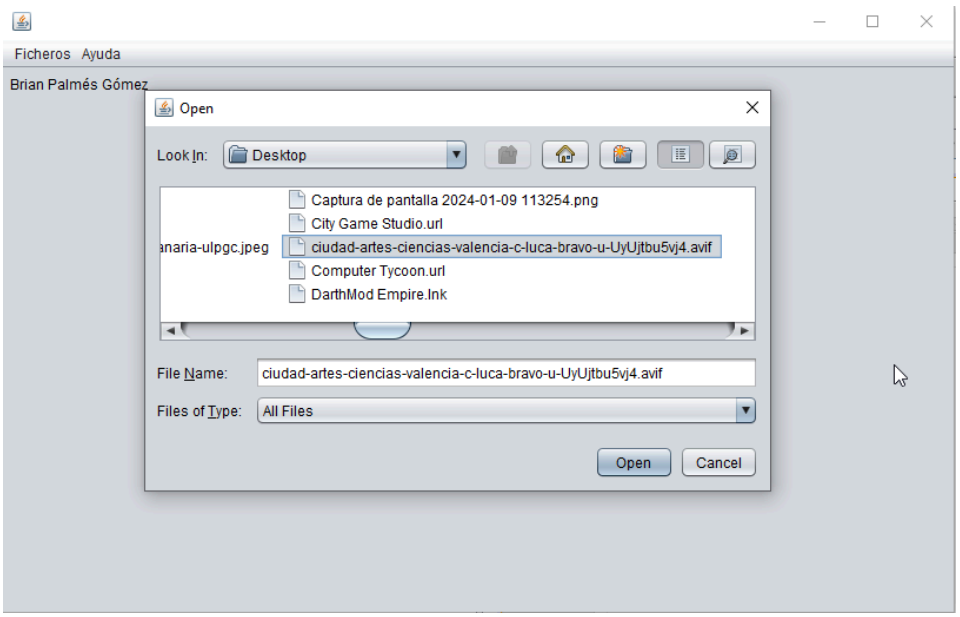
El objetivo de la práctica será la implementación de una aplicación para realizar un proceso de umbralizado sobre una imagen que el usuario podrá seleccionar de una carpeta
carpeta del equipo. La imagen debe tener una resolución inferior a 1024x768 para que pueda
visualizarse completa en la pantalla.

La aplicación desarrollada deberá tener un título en la ventana principal así como mostrar
agrupadas en diferentes menús de una barra de menú las siguientes opciones:

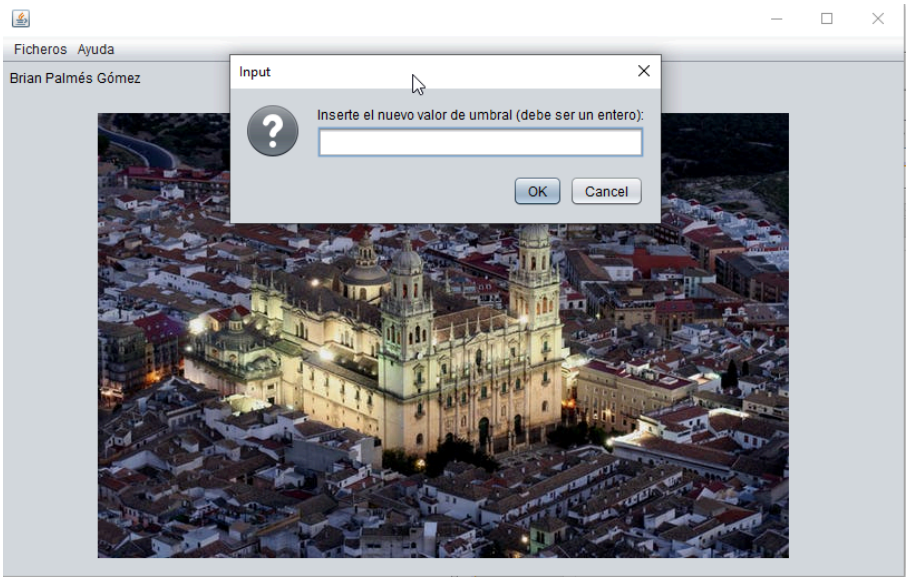
Permitir al usuario abrir una imagen de una carpeta y visualizarla.
Permitir al usuario guardar en una carpeta la imagen umbralizada.
Aplicar el proceso de umbralizado solicitando mediante un cuadro de diálogo el valor del umbral.
Mostrar la imagen umbralizada.
Disponer de un menú "Ayuda" que muestre información "Acerca de" la aplicación con
información sobre la misma.
Incluir la opción de salir del programa y el cierre de la ventana solicitando confirmación
del usuario.

Añadir para cada elemento de los menús un atajo de teclado.
Se valorará en la evaluación de la práctica:
El correcto funcionamiento de la aplicación.
La disposición de los elementos de la interfaz.

Como se pide en la tarea diseñamos una interfaz en la que accediendo al menú seleccionamos ficheros y luego abrir y se nos abre un JFileChooser con el que podemos seleccionar la foto que queremos abrir en la aplicación.



Una vez seleccionamos la foto podemos ir a editar y se nos abrirá un JDialog donde introduciremos un valor entero que será el valor de umbral.



Una vez umbralizada la imagen podemos guardarla si seleccionamos guardar en el menú de ficheros y seleccionamos guardar se nos abrirá otro JChooserFile donde podemos elegir donde queremos guardar el archivo nuevo umbralizado.

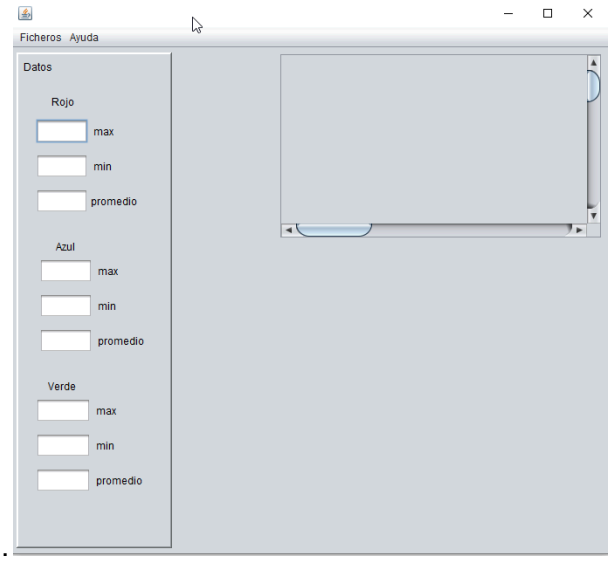
código :https://github.com/bpalmes/DIU/tree/main/DIU_6

Práctica 7
Java Swing. Panel deslizable y barras de desplazamiento.

TAREA.

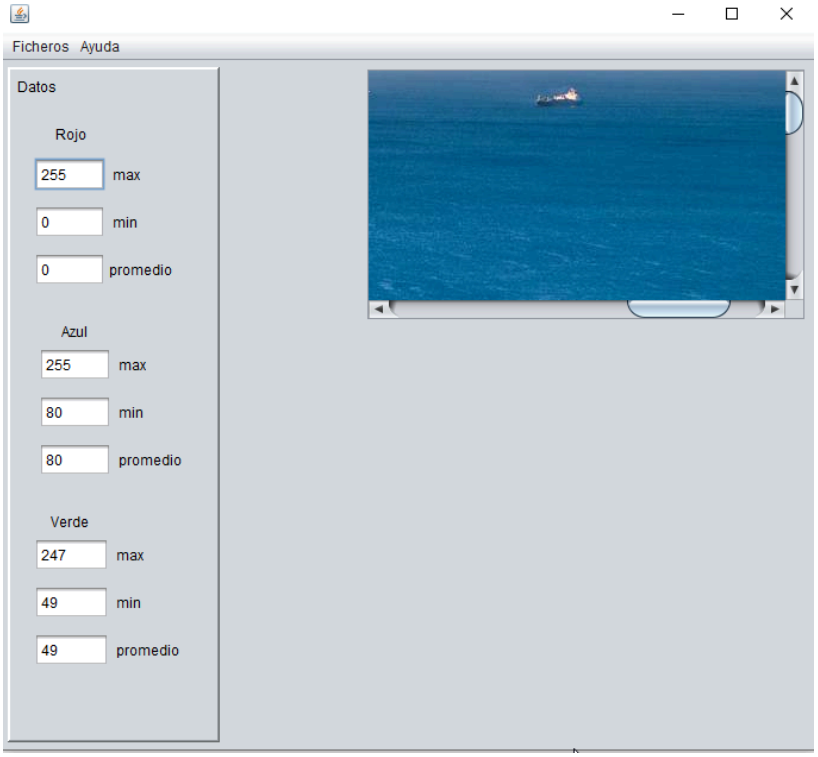
El objetivo de esta práctica es la utilización de las clases antes mencionadas además del
manejo de los eventos asociados a las mismas. Para ello y como demostrador del uso de
las mencionadas clases, el estudiante realizará una aplicación que disponga de la siguiente
funcionalidad:
Permitir al usuario elegir una imagen y mostrarla haciendo uso de un scrollpane.
Mostrar mediante campos de texto no editables el valor máximo, mínimo y promedio de las componentes roja, verde y azul de la porción de la imagen visible en el scrollpane.
Los valores anteriores se deben actualizar a medida que el usuario se desplaza por la imagen utilizando las barras de desplazamiento del
scrollpane, o si se produce un cambio de tamaño de la ventana de la aplicación.

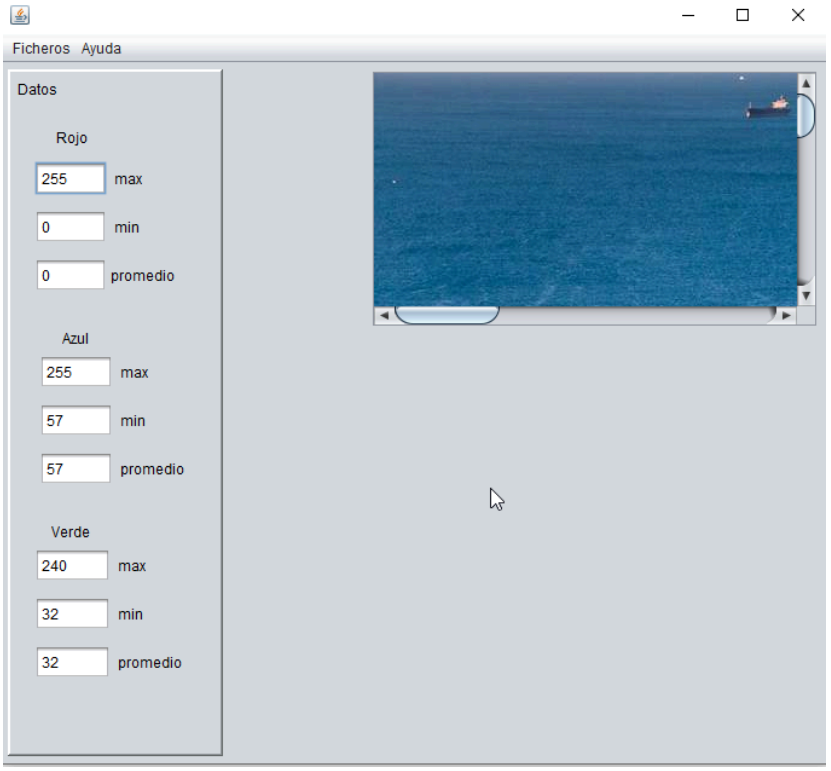
Como se pide en la tarea se diseña una interfaz que usando un menú puede abrir una imagen a elección del usuario



.

Una vez cargada la foto podemos usar el scrollpane para recorrer la imagen el recuadro se irá moviendo por la imagen se obtiene el máximo y mínimo del canal c de cada color.





Código:https://github.com/bpalmes/DIU/tree/main/DIU7

Práctica 10.

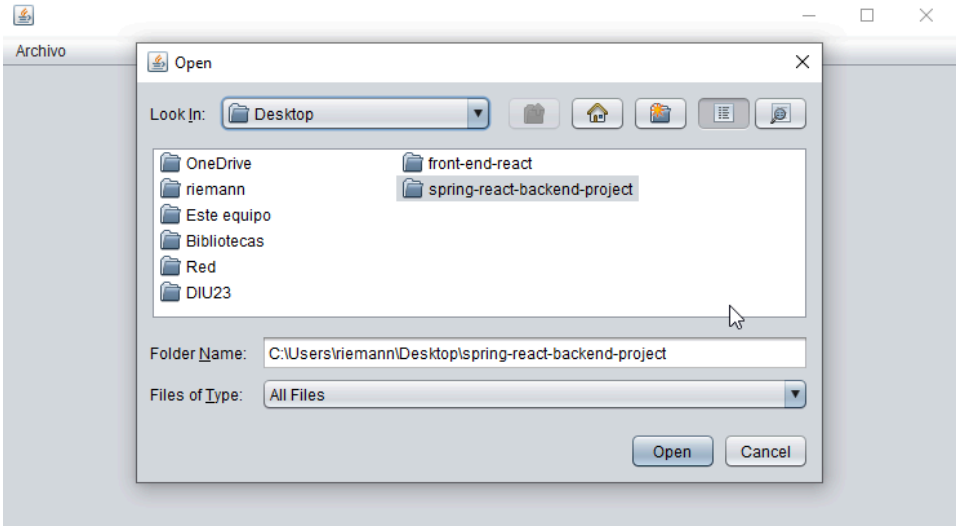
Java Swing. SwingWorker y barra de progreso.

Tarea.

El objetivo de esta práctica es implementar una aplicación que permita aplicar los conceptos y controles vistos hasta ahora en la asignatura. En ese sentido, la aplicación será un compresor de archivos para lo cual mostrará al usuario los archivos de una carpeta previamente seleccionada y de la cual, el usuario seleccionará los archivos que desee comprimir,
así como el archivo comprimido (zip) resultante.
Para generar el archivo comprimido zip utilizaremos el paquete de clases java.util.zip que proporciona Java, En la sección 10.5 se muestra un ejemplo de uso.
La funcionalidad que deberá tener la aplicación a desarrollar en esta práctica será la siguiente:
Permitir al usuario seleccionar una carpeta existente y mostrará en la aplicación los archivos que contiene.
Permitir que el usuario seleccione los archivos de la carpeta que desea añadir al archivo
comprimido.
Permitir al usuario elegir la ubicación y nombre del archivo comprimido. Por defecto, tendrá el nombre de la carpeta que contiene los archivos a comprimir.
Mientras se comprimen los archivos se mostrará con una barra de progreso el avance del proceso.
Se podrá cancelar el proceso de compresión mientras se realiza.

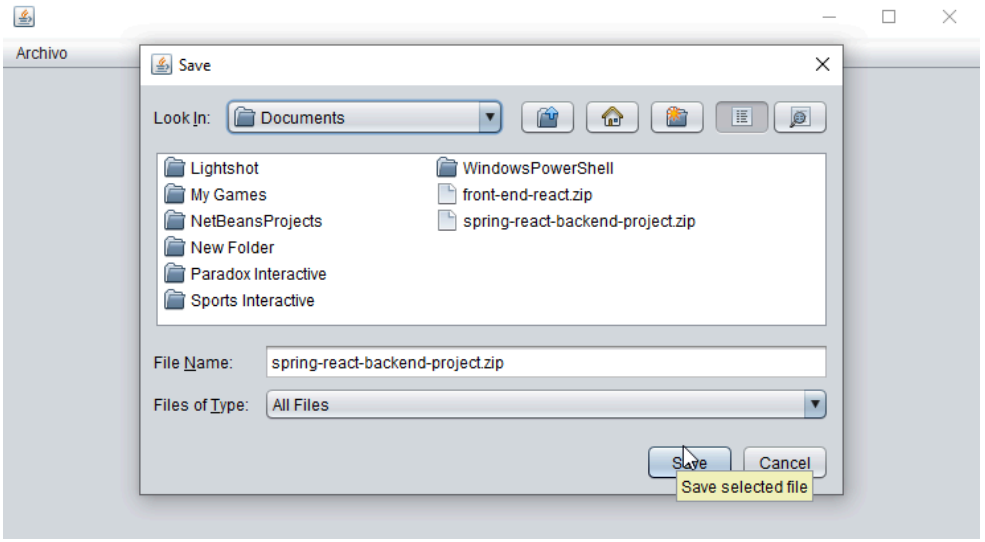
Como se pide en la tarea se diseña una sencilla interfaz con la que gestionaremos la compresión de archivos en segundo plano.

Al abrir la aplicación podremos a través del menú seleccionar un archivo, como en otras prácticas se abrirá un JChooseFile con el que podremos seleccionar solo carpetas y con el que el usuario podrá elegir qué carpeta o directorio desea comprimir.



Al elegir el archivo se abre otro JChooseFile con el que podrás elegir donde guardar el nuevo archivo comprimido.

Lamentablemente no he conseguido implementar la barra de progreso durante la compresión.



Código: https://github.com/bpalmes/DIU/tree/main/Practica10