Ejercicio Paleta de Colores.

Funcionalidad

La aplicación consiste en un simple selector de color mediante código hexadecimal RGB. Mediante unos controles representados por "sliders" podemos definir el valor hexadecimal que queremos representar en el círculo que usamos como muestra. Y a través de los botones de añadir y eliminar podemos crear o borrar registros de los colores que queramos conservar o no.

Descripción

El ejercicio consiste en el desarrollo de una aplicacion con interfaz grafica sencilla, con varios componentes sliders que nos permitiran movernos por la escala RGB de color y mostrar un resultado de la combinación de estos códigos de color.

Desarrollo

El proyecto completo se compone de tres clases java y el correspondiente archivo fxml que da forma a la interfaz.

Clase principal App.java

```
package es.ideas;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;
import java.io.IOException;
public class App extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        Scene escena;
        escena = new
Scene(FXMLLoader.load(App.class.getResource("view/PaletaColoresFXML.fxml")));
        stage.setScene(escena);
        stage.setTitle("Paleta de colores");
        stage.setResizable(false);
        stage.show();
    }
    public static void main(String[] args) {
        launch();
    }
}
```

Esta es la clase que contiene nuestro método main y en la que cargaremos nuestro Stage. Con los métodos 'setter' definimos la escena, el título, indicamos si nuestra escena podrá variar su tamaño y con el método .show(); mostramos nuestra escena. En main simplemente ejecutamos el método launch(); para lanzar nuestra escena con el arranque del programa.

Clase controladora, PaletaColoresController.java

```
package es.ideas.controller;

import es.ideas.model.Rgb;

import java.net.URL;
import java.util.ResourceBundle;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.Slider;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.text.Text;

/**
 *
 * @author Abel & Narciso
 */
public class PaletaColoresController implements Initializable {

    @FXML
    private Slider sliderRojo;
    @FXML
    private Slider sliderVerde;
    @FXML
    private Slider sliderAzul;
    @FXML
    private Circle circulo;
    @FXML
    private Label textoRojo;
    @FXML
    private Label textoVerde;
    @FXML
    private Label textoAzul;
    @FXML
    private Text hexColor;
    @FXML
    private Button boton;
    @FXML
    private ListView<Rgb> lista;
    private Rgb datos = new Rgb();
    private ObservableList<Rgb> listaRgb = datos.getRgb();
    @FXML
    private Button boton2;

    @Override
    public void initialize(URL url, ResourceBundle rb) {
```

```
        cambiaColor();

        //El botón "Añadir" se desactiva cuando un elemento de la lista está
seleccionado.

boton.disableProperty().bind(lista.getSelectionModel().selectedItemProperty().isNotNull(


        //El botón "Eliminar" se desactivar cuando no hay un elemento seleccionado.

boton2.disableProperty().bind(lista.getSelectionModel().selectedItemProperty().isNull())


    }

    private void cambiaColor() {
        //Se añaden los elementos del ObservableList
        lista.setItems(listaRgb);

        //Se añaden ChangeListener a los sliders
        sliderRojo.valueProperty().addListener((obs, oldVal, newVal) -> {
            //Se le da color al circulo mediante el método creaColor() que devuelve un
color
            circulo.setFill(creaColor());
            //El textoRojo informa del valor del color en todo momento
            textoRojo.setText("Rojo: " + (int) sliderRojo.getValue());
            //Actualiza código Hex
            codigoHex();
        });

        sliderVerde.valueProperty().addListener((obs, oldVal, newVal) -> {
            circulo.setFill(creaColor());
            textoVerde.setText("Verde: " + (int) sliderVerde.getValue());
            codigoHex();
        });

        sliderAzul.valueProperty().addListener((obs, oldVal, newVal) -> {
            circulo.setFill(creaColor());
            textoAzul.setText("Azul: " + (int) sliderAzul.getValue());
            codigoHex();
        });
    }

    private Color creaColor() {
        //Crea objeto de la clase color, pasandole los valores enteros de los sliders.
        int rojo = (int) sliderRojo.getValue();
        int verde = (int) sliderVerde.getValue();
        int azul = (int) sliderAzul.getValue();

        Color color = Color.rgb(rojo, verde, azul);

        return color;
```

```
    }

    private void codigoHex() {
        //se crea el código Hex pasandole los valores de los sliders. Para ello
utilizamos String.format
        hexColor.setText(String.format("#%02X%02X%02X",
                        (int) sliderRojo.getValue(),
                        (int) sliderVerde.getValue(),
                        (int) sliderAzul.getValue()));
    }

    @FXML
    private void accionNuevo(ActionEvent event) {
        Rgb rgb = new Rgb((int) sliderRojo.getValue(),
                (int) sliderVerde.getValue(),
                (int) sliderAzul.getValue(),
                hexColor.getText());

        int contador = 0;

        //Si la lista está vacía, se añade el elemento
        if(listaRgb.isEmpty()) {
            listaRgb.add(rgb);
            contador++;
        } else {
            //Sino está vacía, se recorre la lista en busca de un elemento igual al
que voy a añadir
            //Si el elemento ya está añadido, se suma 1 al contador. Para ello comparo
los códigos HEX.
            for (int i = 0; i < listaRgb.size(); i++) {
                if(listaRgb.get(i).getHex().equals(rgb.getHex())) {
                    contador++;
                }
            }
        }
        //Si el contador sigue en 0, el elemento es nuevo y se puede añadir.
        //Si no es nuevo, no se añade porque es repetido.
        if(contador == 0) listaRgb.add(rgb);
    }

    @FXML
    private void accionEliminar(ActionEvent event) {
        //Elimina de la lista el elemento seleccionado
        listaRgb.remove(lista.getSelectionModel().getSelectedItem());
    }

}
```

En esta clase controladora, definimos la funcionalidad de todos los elementos de nuestra aplicación, declarados con su correspondiente etiqueta `@FXML`. Los métodos que podemos encontrar en esta clase son:

- **initialize():** En este método definimos como actúan los botones bajo ciertas condiciones. El botón añadir se desactiva cuando hay un elemento de la lista seleccionado para evitar así la redundancia de datos en los registros de colores y el botón "eliminar" se desactiva cuando no hay ningún elemento seleccionado, ya que no habría nada que eliminar.
- **camciacolor():** Aqui añadimos los elementos de nuestro ObservableList `lista.setItems(listaRgb)` , a continuación usamos una funcion lambda para dar el color que hemos seleccionado en los sliders al círculo de color que lo representa y se modifíca el texto que representa el código RGB.
- **creacolor():** Éste método crea un objeto de la clase Color a través de la información que proporcionan los sliders.
- **codigoHex():** Creamos un código hexadecimal mediante String.format para pasar los valores de los sliders a código hexadecimal.
- **accionNuevo():** Éste método es el encargado de añadir nuestro color a la lista que tenemos para guardarlos. Esto se hace bajo ciertas condiciones: si la lista está vacía se añade el elemento, si no lo está se buscan coincidencias y se controlan mediante un contador, si está a cero indica que no hay ningún registro igual y se añade a la lista, si no, querrá decir que el registro ya existe y por lo tanto no se añadirá para evitar la redundancia de datos.
- **accionEliminar():** Es el encargado de eliminar elementos de la lista una vez los tenemos seleccionados.

Clase Rgb.java

```
package es.ideas.model;

import javafx.beans.property.IntegerProperty;
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

/**
 *
 * @author Abel & Narciso
 */
public class Rgb {
    private ObservableList<Rgb> listaRGB = FXCollections.observableArrayList();
    private final IntegerProperty red = new SimpleIntegerProperty(this, "red", 0);
    private final IntegerProperty green = new SimpleIntegerProperty(this, "green", 0);
    private final IntegerProperty blue = new SimpleIntegerProperty(this, "blue", 0);
    private final StringProperty hex = new SimpleStringProperty(this, "hex", "");

    public Rgb() {

    }

    public Rgb(int red, int green, int blue, String hex) {
        this.red.set(red);
        this.green.set(green);
        this.blue.set(blue);
        this.hex.set(hex);
```

```
    }

    public int getRed() {
        return red.getValue();
    }

    public int getGreen() {
        return green.getValue();
    }

    public int getBlue() {
        return blue.getValue();
    }

    public String getHex() {
        return hex.getValue();
    }

    public IntegerProperty redProperty() {
        return red;
    }

    public IntegerProperty greenProperty() {
        return green;
    }

    public IntegerProperty blueProperty() {
        return blue;
    }

    public StringProperty hexProperty() {
        return hex;
    }

    public void setRed(int nuevoValor) {
        this.red.setValue(nuevoValor);
    }

    public void setGreen(int nuevoValor) {
        this.green.setValue(nuevoValor);
    }

    public void setBlue(int nuevoValor) {
        this.blue.setValue(nuevoValor);
    }

    public void setHex(String nuevoValor) {
        this.hex.setValue(nuevoValor);
    }

    @Override
    public String toString() {
```

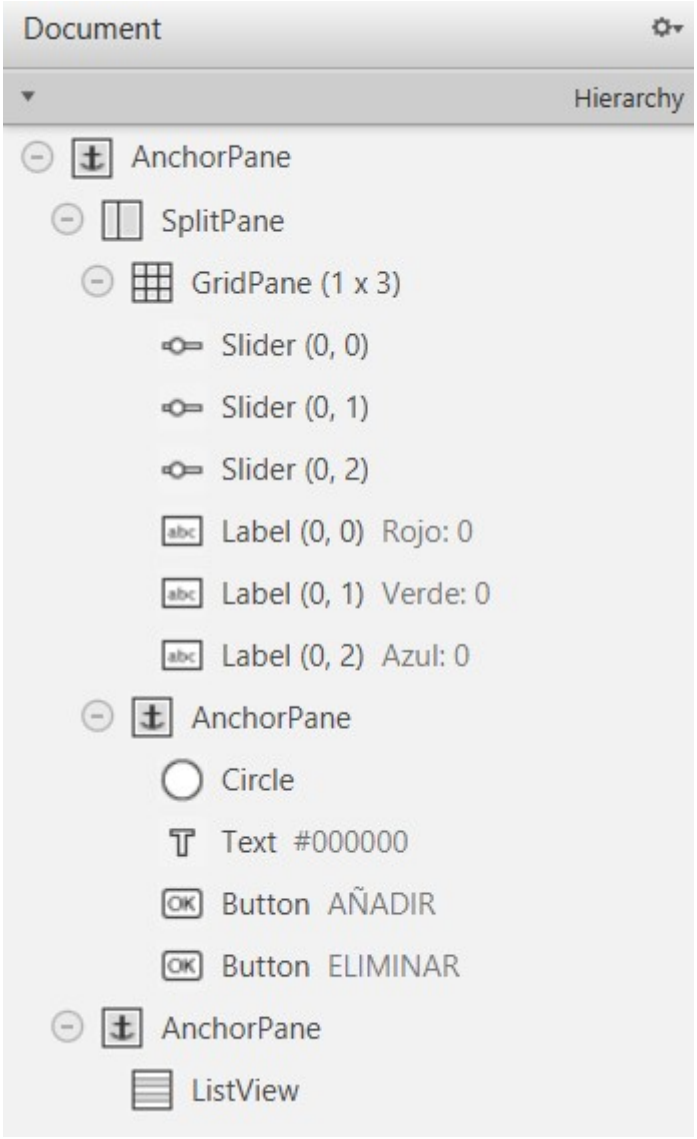
```
        //Se sobreescribe el método toString() para que escriba en la lista lo que
queramos.
        return "RGB: " + red.get() + " " + green.get() + " " + blue.get() + " - HEX: "
+ hex.get();
    }

    public ObservableList<Rgb> getRgb() {
        return listaRGB;
    }
}
```

Esta clase es muy simple, definimos nuestro modelo RGB con sus correspondientes métodos getter, setter, constructor y toString. Ademas del ObservableList y su getter para poder utilizar sus valores durante la ejecución.

Archivo primary.fxml

Este archivo es el encargado de definir nuestra interfaz gráfica. En él se puede observar la estructura de arbol que se ha utilizado en el diseño.




```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.Slider?>
<?import javafx.scene.control.SplitPane?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.ColumnConstraints?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.RowConstraints?>
<?import javafx.scene.shape.Circle?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>
```

```
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="514.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="es.ideas.controller.PaletaColoresController">
   <children>
      <SplitPane dividerPositions="0.5" prefHeight="400.0" prefWidth="600.0">
        <items>
            <GridPane prefHeight="398.0" prefWidth="300.0">
              <columnConstraints>
                <ColumnConstraints hgrow="SOMETIMES" maxWidth="370.0" minWidth="10.0"
prefWidth="268.0" />
              </columnConstraints>
              <rowConstraints>
                <RowConstraints maxHeight="138.0" minHeight="0.0" prefHeight="111.0"
vgrow="SOMETIMES" />
                <RowConstraints maxHeight="317.0" minHeight="0.0" prefHeight="99.0"
vgrow="SOMETIMES" />
                <RowConstraints maxHeight="305.0" minHeight="10.0" prefHeight="96.0"
vgrow="SOMETIMES" />
              </rowConstraints>
               <children>
                  <Slider fx:id="sliderRojo" blockIncrement="1.0" max="255.0"
minorTickCount="4" showTickLabels="true" showTickMarks="true">
                     <GridPane.margin>
                        <Insets left="5.0" right="5.0" />
                     </GridPane.margin>
                  </Slider>
                  <Slider fx:id="sliderVerde" blockIncrement="1.0" max="255.0"
minorTickCount="4" showTickLabels="true" showTickMarks="true" GridPane.rowIndex="1">
                     <GridPane.margin>
                        <Insets left="5.0" right="5.0" />
                     </GridPane.margin>
                  </Slider>
                  <Slider fx:id="sliderAzul" blockIncrement="1.0" max="255.0"
minorTickCount="4" showTickLabels="true" showTickMarks="true" GridPane.rowIndex="2">
                     <GridPane.margin>
                        <Insets left="5.0" right="5.0" />
                     </GridPane.margin>
                  </Slider>
                  <Label fx:id="textoRojo" text="Rojo: 0" GridPane.halignment="RIGHT"
GridPane.valignment="TOP">
                     <GridPane.margin>
                        <Insets right="10.0" top="30.0" />
                     </GridPane.margin>
                     <font>
                        <Font name="System Bold" size="12.0" />
                     </font>
                  </Label>
                  <Label fx:id="textoVerde" text="Verde: 0"
GridPane.halignment="RIGHT" GridPane.rowIndex="1" GridPane.valignment="TOP">
                     <GridPane.margin>
                        <Insets right="10.0" top="30.0" />
```

```
                     </GridPane.margin>
                     <font>
                        <Font name="System Bold" size="12.0" />
                     </font>
                  </Label>
                  <Label fx:id="textoAzul" text="Azul: 0" GridPane.halignment="RIGHT"
GridPane.rowIndex="2" GridPane.valignment="TOP">
                     <GridPane.margin>
                        <Insets right="10.0" top="30.0" />
                     </GridPane.margin>
                     <font>
                        <Font name="System Bold" size="12.0" />
                     </font>
                  </Label>
               </children>
            </GridPane>
          <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="398.0"
prefWidth="205.0">
               <children>
                  <Circle fx:id="circulo" layoutX="148.0" layoutY="114.0"
radius="100.0" stroke="BLACK" strokeType="INSIDE" AnchorPane.topAnchor="40.0" />
                  <Text fx:id="hexColor" layoutX="95.0" layoutY="260.0"
strokeType="OUTSIDE" strokeWidth="0.0" text="#000000" textAlignment="CENTER"
AnchorPane.topAnchor="239.99609375">
                     <font>
                        <Font name="Britannic Bold" size="24.0" />
                     </font>
                  </Text>
                  <Button fx:id="boton" layoutX="101.0" layoutY="283.0"
mnemonicParsing="false" onAction="#accionNuevo" prefHeight="56.0" prefWidth="94.0"
text="AÑADIR" AnchorPane.bottomAnchor="85.0" AnchorPane.topAnchor="283.0" />
                  <Button fx:id="boton2" layoutX="101.0" layoutY="321.0"
mnemonicParsing="false" onAction="#accionEliminar" prefHeight="30.0" prefWidth="94.0"
text="ELIMINAR" />
               </children>
            </AnchorPane>
        </items>
      </SplitPane>
      <AnchorPane layoutY="400.0" prefHeight="72.0" prefWidth="600.0">
         <children>
            <ListView fx:id="lista" layoutY="-1.0" prefHeight="112.0"
prefWidth="600.0" />
         </children>
      </AnchorPane>
   </children>
</AnchorPane>
```