IES de Teis
Avda. de Galicia, 101
36216 – Vigo

Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

PRÁCTICA 4: INTERFAZ JAVAFX y CSS

Introducción:

- Iniciar al alumnado en el desarrollo y manejo de interfaces gráficas con JavaFX
- Iniciar al alumnado en el manejo de la herramienta visual Scene Builder
- Consolidar la estructura y manejo de una aplicación JavaFX (Stage, Scene, Panel, Node)

Introducción:

JavaFX y estructura de interfaz

La clase **Stage** nos permite manejar las ventanas creada por la API JavaFX, en primera instancia contamos con el Stage creado automáticamente por JavaFX, para mostrarlo usaremos el método **primaryStage.show()**, también podemos configurar la ventana, cambiando, por ejemplo, el título, tamaño, posición, etc., de momento solo establecemos el título y mostramos la ventanas, tendremos una vacía pues aun no agregamos elementos a la misma.

Un **Stage** representa nuestra ventana, esta ventana puede contener solo un **Scene**, este es un contenedor para los elementos que conforman la GUI, puede contener uno o varios elementos organizados de manera jerárquica en forma de árbol, a esto se le llama **Scene Graph**.

Los elementos que contiene un **Scene** deben extender la clase base **Node**, estos son organizados de manera jerárquica en el **scene graph**, los elementos incluidos en el scene graph son llamados **nodes**, al elemento ubicado en el tope de la jerarquía se le llamado "**root node**", este puede contener otros nodes hijos y estos a su vez pueden contener más nodes.

La clase **javafx.scene.layout.Pane** proporciona las bases para las clases que normalmente son utilizadas como contenedores, por ejemplo: **VBox**, **GridPane**, **StackPane**, otros, estos contenedores nos facilitan la organización de los nodes que agregamos a la GUI, normalmente elegimos a uno de ellos para utilizarlo como root node.

Para establecer el **root** del **scene** podemos usar el constructor o el método **setRoot(Parent)**, no necesariamente debe ser un contenedor, podemos usar cualquier clase que herede de la clase javafx.scene.Parent por ejemplo un Button o un Group.

Controles JavaFX

1. Radio Button

Los botones de radio crean una serie de elementos donde solo se puede seleccionar un elemento. Los **RadioButtons** son un ToggleButton especializado. Cuando se presiona y suelta un RadioButton, se envía un ActionEvent

Solo se puede seleccionar un **RadioButton** cuando se coloca en un **ToggleGroup**. Por defecto, un RadioButton no está en un ToggleGroup.

Llamar a **ToggleGroup.getSelectedToggle()** le devolverá el **RadioButton** que ha sido seleccionado.

Método setUserData

En JavaFX todas las clases que hereden de la clase Node, poseen los métodos **setUserData(Object)** y **getUserData()** que, en palabras de la documentación oficial, permiten guardar cualquier objeto, efectivamente anclandolo al nodo, para poder recuperarlo en un momento posterior.

Se puede utilizar el método setUserData para guardar datos personalizados en un control. Por ejemplo si queremos que al seleccionar un radioButton nos devuelva un valor que tenemos asociado.

Ejemplo:

Utilizaremos este método para guardar la instancia de la clase usuario. Como el stage es el mismo para ambas ventanas, es un excelente candidato para guardar la información ahí.

Normalmente, cuando se selecciona uno de los RadioButtons en un ToggleGroup la aplicación realiza una acción. A continuación se muestra un ejemplo que imprime los datos de usuario delRadioButton seleccionado que se ha configurado con **setUserData(Object)**

```
radioButton1.setUserData("impresionante")
radioButton2.setUserData("ok");
radioButton3.setUserData("util");
ToggleGroup group = new ToggleGroup();
group.selectedToggleProperty().addListener((obserableValue, old_toggle, new_toggle) -> {
   if (group.getSelectedToggle() != null) {
      System.out.println("Tu piensas que .... es  " +
group.getSelectedToggle().getUserData().toString());
   }
})
```

Imágenes con JavaFX

Desde el código Java el archivo de la imagen se debe cargar en un objeto Image usando una llamada al método **getClass().getResourceAsStream()** indicando como parámetro la ruta al archivo dentro de la carpeta resources como se muestra en este ejemplo de código:

```
Image img = new Image(getClass().getResourceAsStream("/images/playerShip3_orange.png"));
```

Dicho objeto Image se debe añadir a algún elemento visual de JavaFX, como un ImageView.

```
ImageView imgView = new ImageView(img);
```

Por último, para que la imagen aparezca en pantalla debe añadirse dicho componente a algún contenedor empleado en la escena del proyecto. En este ejemplo se ha creado el panel principal paneRoot para la escena y se añade la imagen a él con este código:

```
Pane paneRoot = new Pane();
var scene = new Scene(paneRoot, 640, 480);
paneRoot.getChildren().add(imgView);
```

Agregar imagen de fondo con JavaFX

1. Agregar una imagen de fondo usando CSS

CSS es una abreviatura de Cascading Style Sheets y se utiliza para diseñar páginas web. Además, CSS también se puede utilizar para diseñar aplicaciones JavaFX. Usaremos las siguientes reglas

CSS para configurar y diseñar la imagen de fondo. Puede agregar más reglas según sus necesidades.

-fx-background-image: url('image-url');

-fx-background-repeat: no-repeat;

-fx-background-size: 500 500;

-fx-background-position: center center;

Podemos usar reglas CSS en línea con la ayuda del método **setStyle()** en el nodo raíz. CSS en línea es excelente si solo queremos agregar algunas reglas. El código JavaFX completo se muestra a continuación.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.stage.Stage;
public class Main extends Application
{
   @Override
   public void start(Stage primaryStage)
   {
      StackPane root = new StackPane();
      Scene scene = new Scene(root, 650, 650);
        root.setStyle("-fx-background-image: url('https://imagen.png'); -fx-background-repeat: no-repeat; -fx-background-size: 500 500; -fx-background-position: center center;");
      primaryStage.setScene(scene);
      primaryStage.show();
   }
   public static void main(String[] args)
   {
      launch(args);
   }
}
```

CSS en línea puede volverse un poco engorroso y difícil de entender si tenemos muchas reglas. En su lugar, podemos crear un archivo CSS separado y agregar estas reglas a ese archivo. El contenido del archivo CSS se muestra a continuación.

2. Agregar una imagen de fondo usando BackgroundImage en Java

JavaFX proporciona una clase BackgroundImage, una opción conveniente para usar si no queremos agregar CSS. El constructor de esta clase toma un objeto de clase Image y otras propiedades de la imagen de fondo. La firma del constructor de esta clase se muestra a continuación.

BackgroundImage(Image img, BackgroundRepeat repeatXAxis, BackgroundRepeat repeatYAxis, BackgroundPosition pos, BackgroundSize size)

Usaremos la posición y el tamaño predeterminados, y la imagen no debe repetirse. Necesitamos usar el objeto BackgroundImage para crear una instancia de clase Background. Finalmente, pode-

mos usar setBackground() en el nodo raíz para establecer la imagen en el fondo. El código completo para esto se muestra a continuación.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.layout.*;
import javafx.stage.Stage;
public class Main extends Application {
   @Override
   public void start(Stage primaryStage)
   {
      StackPane root = new StackPane();
      Scene scene = new Scene(root, 650, 650);
      Image img = new Image("https://imagen.png");
      BackgroundImage bImg = new BackgroundImage(img,
                                 BackgroundRepeat.NO_REPEAT,
                                 BackgroundRepeat.NO_REPEAT,
                                 BackgroundPosition.DEFAULT,
                                 BackgroundSize.DEFAULT);
      Background bGround = new Background(bImg);
      root.setBackground(bGround);
      primaryStage.setScene(scene);
      primaryStage.show();
   }
   public static void main(String[] args) {
      launch(args);
   }
}
```

Cuadros de diálogos con JavaFX

- **TextInputDialog:** permite al usuario solicitar que introduzca una cadena de texto

```
TextInputDialog dialog = new TextInputDialog("42");
dialog.setHeaderText("Input your favourite int.");
dialog.setTitle("Favourite number?");
dialog.setContentText("Your favourite int: ");
Optional<String> result = dialog.showAndWait();
String s = result.map(r -> {
   try {
      Integer n = Integer.valueOf(r);
      return MessageFormat.format("Nice! I like {0} too!", n);
   } catch (NumberFormatException ex) {
      return MessageFormat.format("Unfortunately \"{0}\" is not a int!", r);
   }
}).orElse("You really don't want to tell me, huh?");
System.out.println(s);
```

- **ChoiceDialog:** Permite al usuario elegir un elemento de una lista de opciones.

```
List<String> options = new ArrayList<>();
options.add("42");
options.add("9");
options.add("467829");
options.add("Other");
ChoiceDialog<String> dialog = new ChoiceDialog<>(options.get(0), options);
dialog.setHeaderText("Choose your favourite number.");
dialog.setTitle("Favourite number?");
dialog.setContentText("Your favourite number:");
Optional<String> choice = dialog.showAndWait();
String s = choice.map(c -> "Other".equals(c) ?
          "Unfortunately your favourite number is not available!"
          : "Nice! I like " + c + " too!")
        .orElse("You really don't want to tell me, huh?");
System.out.println(s);
```

- **Alert: E**s una ventana emergente simple que muestra un conjunto de botones y obtiene un resultado según el botón que el usuario hizo clic:

  Ejemplo

  Esto permite al usuario decidir si realmente quiere cerrar la etapa primaria:

```
@Override
public void start(Stage primaryStage) {
   Scene scene = new Scene(new Group(), 100, 100);
   primaryStage.setOnCloseRequest(evt -> {
      // allow user to decide between yes and no
        Alert alert = new Alert(Alert.AlertType.CONFIRMATION, "Do you really want to
close
this application?", ButtonType.YES, ButtonType.NO);
      // clicking X also means no
      ButtonType result = alert.showAndWait().orElse(ButtonType.NO);
      if (ButtonType.NO.equals(result)) {
         // consume event i.e. ignore close request
         evt.consume();
      }
   });
   primaryStage.setScene(scene);
   primaryStage.show();
}
```

- **Dialogs** (Existen diferentes tipos de cuadros de diálogo, en función del tipo de información que se puede mostrar)
  - **Información**

```
Dialogs.create()
      .owner(stage)
      .title("Information Dialog")
      .masthead("Look, an Information Dialog")
      .message("I have a great message for you!")
      .showInformation();
```

  - **Error**

```
Dialogs.create()
      .owner(stage)
      .title("Warning Dialog")
      .masthead("Look, a Warning Dialog")
      .message("Careful with the next step!")
```

```
        .showWarning();
```

- **Confirmación**

```
Action response = Dialogs.create()
        .owner(stage)
        .title("Confirm Dialog")
        .masthead("Look, a Confirm Dialog")
        .message("Do you want to continue?")
        .showConfirm();

if (response == Dialog.Actions.YES) {
   // ... user chose YES
} else {
   // ... user chose NO, CANCEL, or closed the dialog
}
```

- **Excepción**

```
Dialogs.create()
        .owner(stage)
        .title("Exception Dialog")
        .masthead("Look, an Exception Dialog")
        .message("Ooops, there was an exception!")
        .showException(new FileNotFoundException("Could not find file blabla.txt"));
```

Secuencia y desarrollo:

Ejercicio: Partiendo del proyecto ***Ventana*** incluido en la carpeta de esta práctica, debes:

1. Revisa el código del ejemplo para entender el funcionamiento de la aplicación.
2. Diseña una interfaz que cumpla los siguientes requisitos:

   Debemos diseñar una interfaz para gestiona la matriculación de un alumno/a en un ciclo de informática.

   Los datos a introducir del alumnado es

   - ***Nombre***
   - ***Apellidos***
   - ***NIF***
   - ***Código Postal***
   - ***Correo electrónico***

   A continuación debe marcar los módulos en los que se va a matricular, siempre y cuando no supere el total de créditos permitidos (debes crear una constante en el que aparezca ese valor)

   El funcionamiento es el siguiente: A medida que se marque un módulo se va calculando el número total de horas que puede estar matriculado. Si supera el total, aparece un mensaje de error con una alerta (Alert)

   Añadir un botón que al pulsar muestre el alumno con sus módulos, horas de cada módulo y un avatar (su foto) en una nueva ventana.