Módulo profesional Programación

UD11: Introducción a JavaFX

Interfaces

En primer lugar, tenemos que saber qué es una interfaz. Una interfaz es una conexión entre el usuario y el ordenador. Así, tenemos dos tipos de interfaces:

- GUI significa la interfaz gráfica de usuario (Graphical User Interface), es una interfaz en la que se utilizan el teclado, el mouse y los dispositivos de E/S para realizar acciones.
- CLI significa interfaz de línea de comandos (Command Line Interface), es una interfaz en la que el teclado se usa principalmente para escribir comandos e interactuar con el ordenador.

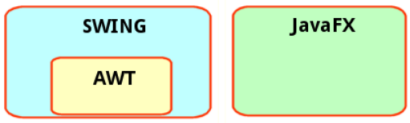
AWT

Cuando se introdujo Java, las clases GUI se agruparon en una biblioteca conocida como AbstractWindows Toolkit (AWT).

AWT está bien para el desarrollo de interfaces gráficas de usuario simples, pero no para desarrollar proyectos GUI completos. Además, AWT es propensa a errores específicos de la plataforma.

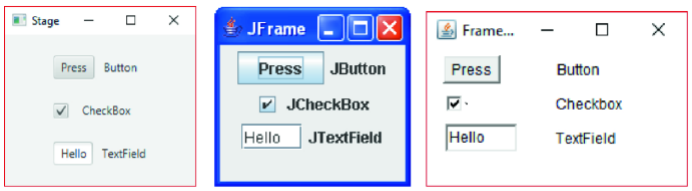
La podríamos bautizar como la "old school", la vieja escuela de las interfaces gráficas. Se centra en el sistema operativo (SO) para dibujar gráficos, por lo que cada SO o plataforma tendrá su propia GUI . Esto, significa que se verá diferente o tendrá un aspecto distinto en cada una de las plataformas. Como por ejemplo:

SWING



Swing. extiende de la librería gráfica AWT y proporciona un conjunto de componentes bastante "ligero". Y que trata de tener el mismo comportamiento independientemente del SO o plataforma en el que se ejecuta.

En la actualidad ha sido reemplazada por una plataforma GUI conocida como **JavaFX**.



(a) JavaFX (b) Swing (c) AWT (Windows)

JavaFX

JavaFX incorpora modernas GUI que permiten desarrollar aplicaciones de Internet enriquecidas. Una aplicación de Internet enriquecida (RIA) es una aplicación Web diseñada para ofrecer las mismas características y funciones normalmente asociadas a las aplicaciones de escritorio.

Una aplicación JavaFX puede ejecutarse sin problemas en un escritorio y desde un navegador web.

Además, JavaFX ofrece soporte multitáctil para dispositivos táctiles como tabletas y teléfonos inteligentes. JavaFX incorpora animaciones en 2D, 3D y reproducción de vídeo y audio, y se ejecuta como aplicación independiente o desde un navegador.

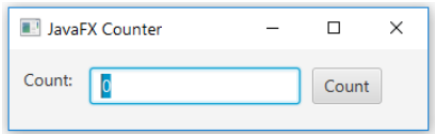
Estructura de una aplicación JavaFX

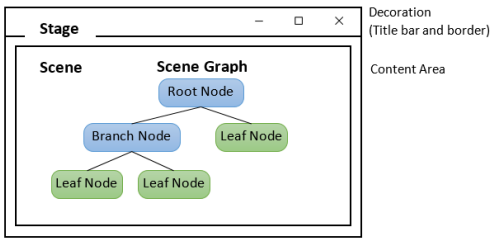
Una aplicación **GUI** desarrollada con JavaFX está compuesta por tres componentes esenciales:

1- Stage - Escenario.

2- Scene - Escena.

3- Scene Graph of Nodes – Grafo de Nodos.

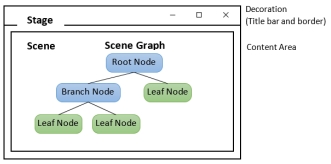




Estructura de una aplicación JavaFX

JavaFX utiliza la metáfora de un teatro para modelar la aplicación gráfica.

- Un escenario (definido por la clase javafx.stage.Stage) representa el contenedor de nivel superior, normalmente una ventana.
- Los elementos de la interfaz de usuario, como los controles, están contenidos en una escena (definida por la clase javafx.scene.Scene).
- Una aplicación puede tener más de una escena, pero sólo una de ellas puede mostrarse en el escenario en un momento dado.
- Un escenario se divide en decoración (barra de título y borde) y el área de contenido.



JavaFX

La clase Application perteneciente al paquete javafx.application es el punto de partida de cualquier aplicación desarrollada en JavaFX.

Por lo tanto para crear una aplicación con esta tecnología la clase principal debe heredar de esta clase (Application) e implementar su método abstracto start(), este método es el que permitirá inicializar la interfaz gráfica.

```
public class Main extends Application {

    @Override
    public void start(Stage primaryStage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource(name:"counterfx.fxml"));
        primaryStage.setTitle("Mi counter app");
        primaryStage.setScene(new Scene(root, 400, 300));
        primaryStage.show();
    }

    Run | Debug
    public static void main(String[] args) {
        launch(args);
    }
}
```

Esta clase posee tres métodos de suma importancia los cuales son los referentes al ciclo de vida de una aplicación JAVAFX. start() es un método abstracto, que debe ser sobreescrito. Los métodos init() y stop() tienen una implementación por defecto que no hace nada.

Un escenario primario es creado por el runtime de JavaFX, y pasado a la Aplicación como un argumento en el método start() de la Aplicación.

Ciclo de vida de una aplicación JavaFX

JavaFX lleva a cabo el ciclo de vida de una aplicación de la siguiente manera:

- Construye una instancia de Application.
- Llama al método init() de Application.
- Llama al método start (javafx.stage.Stage) de la aplicación, y pasa el escenario primario como argumento.
- Espera a que la aplicación termine (por ejemplo, cerrando todas las ventanas).
- Llama al método stop() de la aplicación.

Ciclo de vida de una aplicación JavaFX

El contenido de una escena se representa en un grafo de escena jerárquico (en forma de árbol) de nodos. Un nodo es un objeto visual de un grafo de escena. Para construir una escena, necesitamos pasar la raíz del grafo de escena y anchura y altura opcionales.

Un nodo se define mediante una clase abstracta javafx.scene.Node, que es la superclase de todos los elementos de interfaz de usuario:

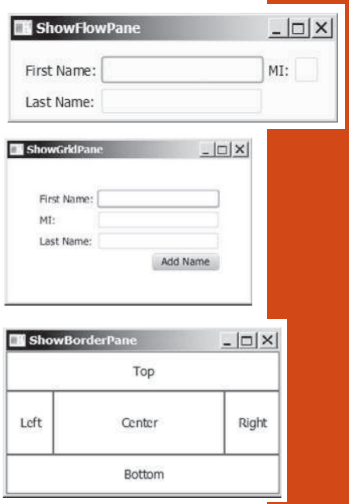
- **Controles (Componentes):** Label, TextField, Button.
- **Layout Pane (Contenedores):** StackPane, FlowPane, GridPane, BorderPane.
- **Formas geométricas:** Circle, Rectangle, Polygon, Sphere, Box.
- **Elementos multimedia:** por ejemplo, ImageView, MediaView (reproducible por reproductor multimedia)

Controles

Package	javafx.scene.control
JavaFX Class	Purpose
Label	Displays an image or read-only text. Labels are often used to identify the contents of *TextFields*.
TextField	A single-line text box for accepting user input.
Button	Command button that the user clicks to signal that an operation should be performed.
RadioButton	Toggle button that the user clicks to select one option in a group.
CheckBox	Toggle button that the user clicks to select or deselect 0, 1, or more options in a group.
ComboBox	List of options from which the user selects one item.
Slider	Displays a set of continuous values along a horizontal or vertical line. The user can select a value by moving the thumb.

Clases de disposición de controles comúnmente utilizadas

Class	*Description*
`Pane`	Base class for layout panes. It contains the `getChildren()` method for returning a list of nodes in the pane.
`StackPane`	Places the nodes on top of each other in the center of the pane.
`FlowPane`	Places the nodes row-by-row horizontally or column-by-column vertically.
`GridPane`	Places the nodes in the cells in a two-dimensional grid.
`BorderPane`	Places the nodes in the top, right, bottom, left, and center regions.
`HBox`	Places the nodes in a single row.
`VBox`	Places the nodes in a single column.



Gestionando las interacciones con el usuario

Cuando se ejecuta un programa Java GUI, el programa interactúa con el usuario, y los eventos dirigen su ejecución. Esto se denomina programación dirigida por eventos.

Un evento puede definirse como una señal al programa de que algo ha sucedido. Los eventos son desencadenados por acciones externas del usuario, como movimientos del ratón, clics y pulsaciones de teclas. El programa puede elegir responder a un evento o ignorarlo.

Gestionando las interacciones con el usuario

Tipos de eventos

User Action	*Source Object*	*Event Type Fired*
Click a button	`Button`	`ActionEvent`
Press Enter in a text field	`TextField`	`ActionEvent`
Check or uncheck	`RadioButton`	`ActionEvent`
Check or uncheck	`CheckBox`	`ActionEvent`
Select a new item	`ComboBox`	`ActionEvent`
Mouse pressed	`Node, Scene`	`MouseEvent`
Mouse released		
Mouse clicked		
Mouse entered		
Mouse exited		
Mouse moved		
Mouse dragged		
Key pressed	`Node, Scene`	`KeyEvent`
Key released		
Key typed		

Gestionando las interacciones con el usuario

Cuando un usuario interactúa con un control, el control dispara un evento.

Para que nuestra aplicación sea notificada de cuando ocurre un evento en particular proveemos un controlador de evento también llamado listener, que se ejecuta cuando ocurre el evento.

Diseño de aplicaciones JavaFX

JavaFX apoya y fomenta el patrón de diseño (MVC) Model-View-Controler.

- Modelo: gestiona los datos de la aplicación
- Vista: presenta la interfaz de usuario
- Controlador: maneja los eventos generados por el usuario y comunica los cambios al modelo, el cual actualiza su estado y comunica cualquier cambio al controlador. El controlador actualiza la vista para reflejar los cambios.