|  |  |  |
| --- | --- | --- |
| Desarrollo de Interfaces | | TAREA 1 UD1 |
|  | | |
|  | | |
|  |  | |
| alumno cesur 25/26  Alejandro Muñoz de la Sierra | PROFESOR  Manuel Gómez Lora | |

introduccion

En este primer acercamiento práctico a la materia de Desarrollo de Interfaces, nos sumergimos en la biblioteca OpenJFX para dar vida a una pequeña aplicación gráfica. ¿El objetivo? Principalmente, familiarizarnos con la creación de interfaces en JavaFX y con la manipulación de figuras geométricas básicas, esas que llamamos "shapes".

El desafío consistía en dar forma a una escena que albergara dos grupos bien diferenciados de elementos. Por un lado, debíamos tener tres líneas con colores, grosores y ubicaciones distintas. Por otro lado, un conjunto compuesto por un círculo, un cuadrado y un pentágono.

Además, se nos solicitó aplicar transformaciones, como la rotación, y presentar un ejemplo completamente funcional del uso del API de OpenJFX. En mi opinión, un buen punto de partida para entender las posibilidades de esta herramienta.

# 01

Preparando el terreno: El entorno de desarrollo

1.1 Instalación de JDK y Eclipse

Para empezar, optamos por instalar la versión JDK 23.0.1 junto con Eclipse IDE for Java Developers 2024-09 (4.33.0). Afortunadamente, ambas herramientas son totalmente compatibles, lo que nos permitió compilar y ejecutar proyectos que utilizan la arquitectura modular de JavaFX sin mayores complicaciones.

1.2 Instalación del JavaFX SDK

Desde la página oficial de OpenJFX (https://openjfx.io), procedimos a descargar JavaFX SDK 23.0.1 y lo descomprimimos en la ruta local:

C:\javafx-sdk-23.0.1

1.3 Creación del proyecto en Eclipse

Dentro de Eclipse, dimos vida a un nuevo Java Project, bautizado como Caso1\_UX\_Muñoz\_de\_la\_Sierra\_Alejandro. Activamos la opción "Create module-info.java file" para trabajar bajo el sistema modular de Java.

Seguidamente, generamos una User Library llamada JavaFX23, a la que añadimos todas las librerías del directorio /lib de JavaFX, y la asociamos al Modulepath del proyecto.

Dentro del paquete "application", creamos la clase principal "EjemploShapes", que contendría el código de nuestra escena.

1.4 Configuración de ejecución

En las opciones de Run Configurations, añadimos los argumentos VM necesarios para que Eclipse reconociera correctamente las librerías de JavaFX:

--module-path "C:\javafx-sdk-23.0.1\lib" --add-modules javafx.controls,javafx.fxml

Con esto, el entorno quedó listo para la acción, y la aplicación pudo ejecutarse sin problemas. (En este punto, una captura de pantalla mostrando el JDK instalado, la estructura del proyecto en Eclipse y la configuración de ejecución sería de gran ayuda).

# 02

Entendiendo los conceptos básicos

Package: Es, básicamente, una forma de ordenar las clases Java. En nuestro caso, el paquete "application" alberga todas las clases del proyecto.

Module-info.java: Define los módulos que la aplicación necesita (como javafx.controls) y los paquetes que expone.

Classpath vs Modulepath: El classpath era el método utilizado en proyectos más antiguos, mientras que el modulepath pertenece al sistema modular moderno de Java, que ofrece una gestión de dependencias más limpia y organizada.

Scene y Stage: En JavaFX, el Stage representa la ventana principal, mientras que la Scene actúa como el contenedor de los nodos gráficos (formas, botones, textos, etc.).

# 03

Desarrollo del código

(Aquí podríamos incluir el código comentado paso a paso, explicando cómo creamos las figuras, cómo asignamos colores y grosores, y cómo aplicamos rotaciones o transformaciones).

# 04

Ejecución del programa

Al ejecutar el programa, se despliega una ventana que muestra las figuras solicitadas:

Tres líneas con colores y grosores diferenciados.

Un círculo, un cuadrado y un pentágono, todos con una sutil rotación.

Un fondo de color violeta que aporta cohesión visual a la escena.

Con esto, demostramos que el API de OpenJFX funciona correctamente y que las clases gráficas principales (Line, Circle, Rectangle, Polygon) se manejan sin inconvenientes. Además, queda patente el uso de transformaciones mediante la clase Rotate. (En esta sección, una captura de la ventana con las figuras sería ideal.)

# 05

conclusiones

Este caso práctico nos ha brindado la oportunidad de afianzar los fundamentos de JavaFX y comprender cómo configurar un entorno modular dentro de Eclipse. Hemos aprendido a estructurar un proyecto con paquetes y módulos, a crear y manipular figuras geométricas, y a aplicar transformaciones. Creo que ha sido un ejercicio muy útil para entender las bases de esta tecnología. Hemos explorado la agrupación de elementos, algo que resulta bastante útil, y cómo interpretar esos avisos de Eclipse sobre accesibilidad, esos que a veces nos sacan de quicio. Vaya, que hemos cubierto un buen terreno.

En definitiva, diría que esta práctica ha sido crucial para establecer los fundamentos del desarrollo de interfaces gráficas interactivas con JavaFX. Personalmente, creo que nos deja listos para los desafíos que nos esperan en los próximos ejercicios, proyectos que, intuyo, serán más complejos. ¡Y eso es emocionante!

# 06

referencias

<https://docs.oracle.com/en/java/javase/23/>

<https://openjfx.io/>

<https://gist.github.com/imshuhao/348a31c81ac655b141ff6fdb8f2043a6>

<https://jdk.java.net/javafx23/>

<https://openjfx.io/javadoc/23/>

<https://openjfx.io/javadoc/25/javafx.graphics/javafx/scene/package-summary.html>