Actividad 1_5: Carrera Interfaz

Programación de Servicios

Curso DAM (Desarrollo Multiplataforma)

Autor: John Arenales

20 de noviembre de 2025

Índice general

Capítulo 1

Introducción

En esta documentación explico cómo pasé un proyecto que originalmente funcionaba por terminal a una versión con una interfaz gráfica usando JavaFX. Además, detallo cómo se adaptaron los hilos para que funcionaran correctamente en el nuevo entorno gráfico y cómo se incorporó el uso del método `synchronized` para evitar problemas de concurrencia.

La idea general del proyecto es simular una carrera entre varios coches. Antes, en la versión de consola, los coches simplemente imprimían en pantalla su avance. Ahora, cada coche está representado por una imagen que se mueve sobre la ventana y se muestra un podio final en una ventana emergente.

Capítulo 2

Desarrollo de la Interfaz

Para pasar el proyecto de terminal a interfaz gráfica utilicé JavaFX. El diseño se organizó con un archivo `main.fxml`, donde se colocaron cinco `ImageView` que representan los coches, un botón para iniciar la carrera y un texto informativo.

Cada coche tiene asociada una imagen en la carpeta `imgs`. El controlador principal, `HelloController`, se encarga de recibir los elementos del FXML y manejar los eventos, como el clic del botón que inicia la carrera.

El movimiento de los coches se implementa actualizando la propiedad `translateX` de cada `ImageView`. Para poder modificar la interfaz desde un hilo, es obligatorio usar `Platform.runLater()`, que asegura que los cambios se hagan en el hilo adecuado de JavaFX.

También se creó un segundo archivo FXML, `ganador-view.fxml`, para mostrar una ventana emergente con el podio. Esta ventana se abre cuando todos los coches han terminado la carrera.

Capítulo 3

Adaptación de los Hilos

En la versión antigua del proyecto, cada coche era un hilo que avanzaba y mostraba su progreso por consola. Para la versión gráfica, los hilos siguen existiendo, pero sus acciones ahora deben afectar a la interfaz.

Por eso, cada instancia de la clase `Coche` recibe su `ImageView`. Dentro del método `run()`, el hilo calcula un avance aleatorio, lo suma a su posición y luego usa `Platform.runLater()` para que su imagen se mueva en pantalla.

Además, cada coche tiene un límite de meta (800 píxeles). Cuando llega a ese punto, registra su marca en un podio compartido entre todos.

Capítulo 4

Uso del Método `synchronized`

Como todos los hilos intentan modificar la misma lista llamada `podio`, es necesario evitar que dos coches intenten escribir al mismo tiempo. Para eso se implementó un método estático y sincronizado:

```
private static synchronized void agregarAlPodio(String marca) {
    ...
}
```

El uso de `synchronized` garantiza que sólo un hilo puede entrar en este método a la vez. Gracias a esto, el orden en el que los coches llegan se registra correctamente.

Cuando la lista alcanza el número total de participantes, significa que todos los coches llegaron, y se llama al método del controlador que muestra la ventana con los resultados finales.

Capítulo 5

Conclusión

La migración de una aplicación de consola a una interfaz gráfica con JavaFX permite visualizar de forma mucho más clara el comportamiento del programa. El uso adecuado de hilos, junto con `Platform.runLater()` y el método `synchronized`, asegura que tanto la lógica como la interfaz funcionen correctamente sin errores de concurrencia.

Este proyecto me permitió practicar manejo de hilos, sincronización, FXML, controladores y ventanas emergentes, todo dentro del entorno de JavaFX, convirtiendo una simulación sencilla en una aplicación más completa e interactiva.