Programación de Servicios y Procesos

Emulador de Mario Kart

Carrera de coches con Interfaz Gráfica en JavaFX

Autor:

Santi Martínez

November 18, 2025

ÍNDICE

1 Introducción

Este proyecto consiste en un emulador simplificado de una carrera de **Mario Kart** desarrollado en Java utilizando **JavaFX** para la interfaz gráfica y los conceptos fundamentales de **programación concurrente** con hilos (`Thread`).

El objetivo principal es simular una carrera entre cinco icónicos personajes del universo Mario Kart (Mario, Luigi, Bowser, Toad y Peach), donde cada corredor avanza a velocidades aleatorias y el orden de llegada es completamente impredecible en cada ejecución.

Corredores

Mario

Luigi

Bowser

Toad

Peach

Cada corredor es una instancia de la clase **Coche**, que extiende Thread. Al ejecutar el método **start()**, los cinco hilos corren en paralelo, compitiendo por llegar primero a la meta.

Gracias al uso de hilos y la aleatoriedad en los tiempos de sueño y velocidad, **cada carrera tiene un resultado diferente**, simulando perfectamente la emoción y la incertidumbre de una verdadera partida de Mario Kart.

2 Planteamiento

Inicialmente el proyecto era un programa de terminal que representaba mediante =, el avance de cada coche en la carrera.

Una vez realizado, se propuso el ejercicio de realizarlo mediante interfaz gráfica. Así que lo primero es plantear el programa:

- Interfaz gráfica con **JavaFX**
- Cinco **Label** que muestran en tiempo real el recorrido de cada coche
- Un **ImageView** por corredor que cambia al llegar a la meta (medalla de oro, plata, bronce, cuarta posición y último)

Oro

Plata

Bronce

4º puesto

5º puesto

- Avance visual mediante cadenas de caracteres.

La diferencia principal respecto al programa de terminal, fue el tener que garantizar que las actualizaciones de la interfaz gráfica se realizaran desde hilos secundarios, mediante la utilización de **Platform.runLater()**.

3 Tecnologías utilizadas

- **Java SE** – Lenguaje principal
- **JavaFX** – Interfaz gráfica y manejo de eventos
- **Hilos (Thread)** – Ejecución concurrente de los corredores
- **Maven/IntelliJ IDEA** – Gestión y desarrollo del proyecto
- **FXML** – Diseño declarativo de la interfaz

4 Uso del synchronized

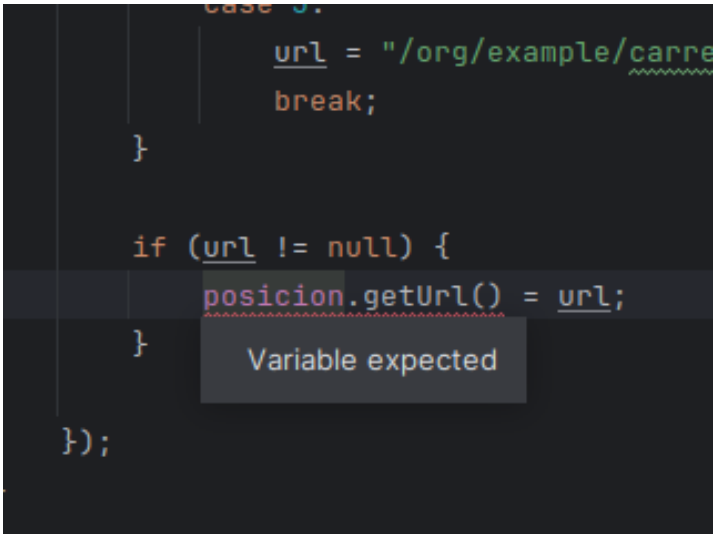
5 Problemas surgidos y soluciones

1. **Actualización de la GUI desde hilos secundarios**

   Solución: Uso obligatorio de `Platform.runLater()`

2. **Variable expected**

   El problema que más me ha llevado tiempo del programa. Estoy escribiendo esto sin todavía haber llegado a una solución, así que se va a mostrar a continuación una captura del mismo:



Solución 1: Se estaba intentando asignar un valor al resultado de un método, no a una variable; además de utilizar un getter y no un setter para añadir un valor. De ahí el primer fallo.

Solución final: No pude implementar lo que quería, que era un objeto Image a través de setUrl(), pero como ese método no existe en el objeto Image, debido a que no puede modificarse, cambié de rumbo en la idea que tenía en esta parte del programa.

3. **Return de ordenDeLlegada en clase Carrera**: El problema fue que sin razón aparente, el método ordenDeLlegada(), necesario para que devuelva ese orden y trabajar con ese dato en la UI, se devolvía de dos en dos.

```java
public class Carrera {  smz7z *

    ArrayList<String> orden = new ArrayList<>();  1 usage
    //ArrayList<Integer> velocidades = new ArrayList<>();
    int contador = 0;  2 usages

    public synchronized int ordenDeLlegada(String nombre, int velocidadMax){
        orden.add(nombre);
        //velocidades.add(velocidadMax);
        contador+=1;

        //...
        return contador;
    }
}
```

Figure 1: Devuelve: 2, 4, 6, 8, 10

Solución: Problema con mútliples llamadas al método en la clase **Coche**.

```java
// Notificar finalización
Platform.runLater(() -> {
    recorridoEnCarrera.setText(String.join( delimiter: "", carretera) + " " + this.nombre + " ¡LLEGÓ!");

    // AQUI SE SUMA 1 AL CONTADOR
    resultado.setText(String.valueOf(carrera.ordenDeLlegada(this.nombre, velocidadMaxima)));

    // AQUÍ SE SUMA OTRO AL CONTADOR
    System.out.println(carrera.ordenDeLlegada(this.nombre,velocidadMaxima));
});
```

Al realizar controles por terminal, se utilizó un System.out para comprobar qué número devolvía, sin darme cuenta que ese mismo control de errores causaba el error.

Esto provocaba la suma doble del contador, debido a que ya se había llamado con anterioridad al método para modificar el texto de otro campo.

6 Conclusión

Este proyecto ha permitido aplicar de forma práctica y divertida conceptos clave de la asignatura:

- Creación y gestión de hilos en Java
- Concurrencia y comportamiento no determinista
- Comunicación segura entre hilos y la interfaz gráfica (JavaFX)
- Diseño de aplicaciones interactivas con eventos

El resultado es una carrera visual, dinámica y adictiva que refleja fielmente la esencia de Mario Kart, demostrando que con pocos elementos se puede crear una experiencia muy entretenida.

¡Es imposible no pulsar el botón de "REINICIAR" una y otra vez para ver quién gana esta vez!

7 Bibliografía

- Oracle Docs – The Java™ Tutorials: Concurrency
  `https://docs.oracle.com/javase/tutorial/essential/concurrency/`
- OpenJFX Documentation
  `https://openjfx.io/`
- Stack Overflow – "JavaFX Platform.runLater and Task" (varias respuestas útiles)
- Imágenes de personajes y medallas extraídas del universo Mario Kart (Nintendo)