

# **Carrera de Coches**

Programación de Servicios y Procesos

**Adrián Condines Celada**

Aula Estudio

2º Ciclo Superior - Desarrollo de Aplicaciones Multiplataforma

Curso 2025 - 2026

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Stack Tecnológico</b>	<b>2</b>
2.1. Java . . . . .	2
2.2. JavaFX . . . . .	2
2.3. CSS . . . . .	3
2.4. Visual Studio Code . . . . .	3
<b>3. Requisitos Funcionales</b>	<b>3</b>
<b>4. Requisitos No Funcionales</b>	<b>4</b>
<b>5. Casos de Uso</b>	<b>4</b>
<b>6. Historias de Usuario</b>	<b>4</b>
<b>7. Estructura del Proyecto</b>	<b>5</b>
<b>8. Clases Principales</b>	<b>6</b>
8.1. Clase App . . . . .	6
8.1.1. Descripción . . . . .	6
8.1.2. Atributos . . . . .	6
8.1.3. Métodos . . . . .	6
8.2. Clase Coche . . . . .	6
8.2.1. Descripción . . . . .	6
8.2.2. Atributos . . . . .	6
8.2.3. Métodos . . . . .	7
8.3. Clase Carrera . . . . .	7
8.3.1. Descripción . . . . .	7
8.3.2. Atributos . . . . .	7
8.3.3. Métodos . . . . .	8
<b>9. Interfaz</b>	<b>8</b>

## 1. Introducción

Esta es una aplicación Java que simula una competición entre seis vehículos mediante programación multihilo e interfaz gráfica JavaFX.

La aplicación permite observar en tiempo real el progreso de cada coche mediante barras de progreso y una clasificación dinámica.

## 2. Stack Tecnológico

### 2.1. Java



El lenguaje de programación utilizado para desarrollar la lógica de la aplicación es Java, debido a su robustez y capacidad para manejar múltiples hilos de ejecución.

### 2.2. JavaFX



JavaFX se ha utilizado como tecnología para desarrollar la interfaz gráfica de la aplicación, proporcionando una manera visual de ver el resultado del programa.

## 2.3. CSS



CSS se ha utilizado para dar estilo a la interfaz gráfica desarrollada con JavaFX, mejorando la experiencia del usuario.

## 2.4. Visual Studio Code



Como IDE se ha utilizado Visual Studio Code, que gracias a sus extensiones permite un desarrollo eficiente en Java y JavaFX.

# 3. Requisitos Funcionales

La aplicación cumple con los siguientes requisitos funcionales.

**RF-1:** Inciación de la carrera al pulsar un botón.

**RF-2:** Simulación del movimiento de cada coche de manera aleatoria hasta que llegan a la meta.

**RF-3:** Barras de progreso que simulan la distancia recorrida por cada coche.

**RF-4:** Mostrar la clasificación final de los coches por orden de llegada.

**RF-5:** Posibilidad de reiniciar la carrera para una nueva simulación.

## 4. Requisitos No Funcionales

La aplicación cumple con los siguientes requisitos no funcionales.

**RNF-1:** Manejo correcto de hilos para evitar bloqueos.

**RNF-2:** Compatibilidad con todos los sistemas operativos, ya que está desarrollada en Java.

**RNF-3:** Interfaz sencilla e intuitiva para el usuario.

## 5. Casos de Uso

A continuación se describen los casos de uso principales de la aplicación:

Casos de Uso	Descripción
Iniciar aplicación	El usuario inicia la aplicación para simular la carrera entre los seis vehículos.

## 6. Historias de Usuario

A continuación se describen las historias de usuario principales de la aplicación:

ID	Como un...	Quiero...	Para...
HU-1	Usuario	Iniciar la aplicación	Simular el funcionamiento de una carrera entre seis coches.

## 7. Estructura del Proyecto

La estructura del proyecto está organizada de la siguiente manera:

```
CarreraDeCochesInterfaz/  
|-- DOCUMENTACIÓN/  
|   |-- images/  
|       |-- css_icon.png  
|       |-- interfaz_1.png  
|       |-- interfaz_2.png  
|       |-- interfaz_3.png  
|       |-- java_icon.png  
|       |-- javafx_icon.png  
|       |-- vscode_icon.png  
|   |-- DOCUMENTACIÓN CARRERA DE COCHES - ADRIANO.tex  
|   |-- DOCUMENTACIÓN CARRERA DE COCHES - ADRIANO.pdf  
|-- src/main  
|   |-- java/  
|       |-- com/akadoble/carreradecochesinterfaz/  
|           |-- App.java  
|           |-- Coche.java  
|           |-- Carrera.java  
|   |-- resources/  
|       |-- carrera.fxml  
|       |-- styles.css
```

## 8. Clases Principales

### 8.1. Clase App

#### 8.1.1. Descripción

La clase **App** es la clase principal que inicia la aplicación JavaFX.

#### 8.1.2. Atributos

No cuenta con atributos propios, pero maneja referencias a la interfaz gráfica.

#### 8.1.3. Métodos

Los métodos de la clase **App** son los siguientes:

- **start(Stage stage)**: Método que configura y muestra la ventana principal de la aplicación.
- **main(String[] args)**: Método principal que lanza la aplicación JavaFX.

### 8.2. Clase Coche

#### 8.2.1. Descripción

La clase **Coche** representa a cada uno de los vehículos que participan en la carrera. Cada coche es un hilo independiente que simula su movimiento hasta llegar a la meta.

#### 8.2.2. Atributos

Los atributos de la clase **Coche** son los siguientes:

- **String nombre**: Nombre del coche.
- **int distanciaRecorrida**: Distancia que el coche ha recorrido hasta ese momento.
- **int distanciaTotal**: Distancia total que debe recorrer el coche para llegar a la meta.
- **int velocidadMaxima**: Velocidad máxima que puede alcanzar el coche.
- **Carrera carrera**: Referencia a la carrera en la que participa el coche.

### 8.2.3. Métodos

Los métodos de la clase **Coche** son los siguientes:

- **run()**: Método principal del hilo que simula el movimiento del coche.
- **getNombre()**: Devuelve el nombre del coche.
- **getDistanciaRecorrida()**: Devuelve la distancia recorrida por el coche.
- **getDistanciaTotal()**: Devuelve la distancia que debe de recorrer el coche para finalizar la carrera

## 8.3. Clase Carrera

### 8.3.1. Descripción

La clase **Carrera** gestiona la lógica de la carrera, incluyendo la creación de coches, el seguimiento de su progreso y la actualización de la interfaz gráfica.

### 8.3.2. Atributos

Los atributos de la clase **Carrera** son los siguientes:

- `Map<String, ProgressBar>` **barras**: Representa cada una de las barras de progreso asociadas a cada uno de los coches.
- `Map<String, Label>` **etiquetasDistancia**: Muestra la distancia recorrida por cada coche en metros.
- `List<String>` **clasificacion**: Lista que almacena el orden de llegada de los coches a la meta.
- `int` **DISTANCIA\_TOTAL**: Distancia total que deben recorrer los coches para llegar a la meta.



### 8.3.3. Métodos

Los métodos de la clase **Carrera** son los siguientes:

- **onStartButton()**: Crea los objetos de la clase Coche y los inicia como hilos independientes para comenzar la carrera.
- **añadirCarta()**: Añade una carta por cada coche en la interfaz gráfica.
- **actualizarProgreso()**: Actualiza las barras de progreso y las etiquetas de distancia en la interfaz gráfica.
- **registrarLlegada()**: Registra la llegada de un coche a la meta y actualiza la clasificación.
- **getClasificacion()**: Devuelve la lista con la clasificación final de los coches.

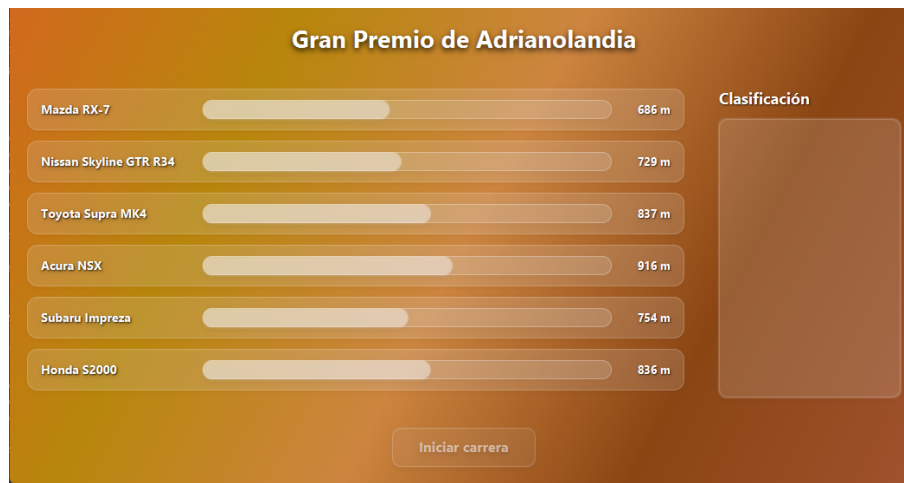
## 9. Interfaz

La interfaz gráfica de la aplicación consta de tres secciones:

- **Sección Superior**: Se compone del título de la aplicación.
- **Sección Central**: Se compone de una sección izquierda que al principio está vacía pero posteriormente se añaden los coches en ella, y una sección derecha en la que hay un panel que muestra la clasificación de los coches conforme van llegando a la meta.
- **Sección Inferior**: Incluye el botón para iniciar la simulación.



Una vez iniciada la carrera, la interfaz muestra el progreso de cada coche mediante barras de progreso y etiquetas que indican la distancia recorrida.



Al finalizar la carrera, la interfaz muestra la clasificación final de los coches en el panel derecho y la carrera puede ser iniciada de nuevo pulsando nuevamente sobre el botón.

