Actividad 1.5: Carrera Interfaz

Álvaro Del Valle Fernández

November 20, 2025

1 Introducción

En este documento mostrare el funcionamiento y la estructura de la actividad carrera interfaz.

2 Requisitos Funcionales

- **RF-1:** Gestionar los coches como hilos independientes
- **RF-2:** Que ningun coche se quede parado
- **RF-3:** Calcular distancia total de forma adecuada
- **RF-4:** Determiar el orden de llegada correctmaente
- **RF-5:** Uso de joins para esperar a que terminen

3 Requisitos No Funcionales

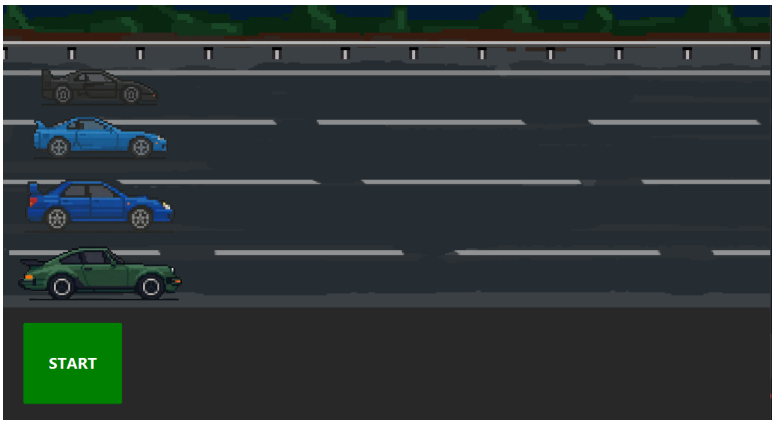
- **RNF-1:** Evitar condiciones de carrera al asignar posiciones
- **RNF-2:** Que el diseño sea minimamente interesante
- **RNF-3:** Los coches no deben de salirse de la interfaz

4 Historias de Usuario

- **HU-1:** Como usuario, quiero ver una interfaz interesante
- **HU-2:** Como usuario, quiero ver visualmente la posicion de los coches
- **HU-3:** Como usuario, quiero ver los resultados exactos

5 Arquitectura

Elementos principales: Coche, Carrera y MainController. Los coches son threads que compiten, clase Carrera gestiona las posiciones, se que no era necesario, el MainController maneja la interfaz.



6 Clases Principales

6.1 Clase MainController

Interfaz gráfica y la coordinación, se centra en mover los elementos en actualizandolos correctamente.

```
package com.example.cochesjoin;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.util.Duration;
import java.util.ArrayList;
import java.util.List;

import javafx.scene.image.ImageView;
import javafx.scene.text.Text;

public class MainController {
    @FXML
    private ImageView barrera;
```

```
    @FXML
    private Button buttonStart;

    @FXML
    private ImageView ferrariF40;
    @FXML
    private ImageView toyotaSupra;
    @FXML
    private ImageView subaruImpreza;
    @FXML
    private ImageView porscheTurbo;
    @FXML
    private Text textResultados;

    private List<Coche> coches = new ArrayList<>();

    private boolean carreraActiva = false;


    @FXML
    private void initialize() {
        Carrera.setController(this);
        /*
        cocheVista.put("Ferrari F40", ferrariF40);
        cocheVista.put("Toyota Supra MK4", toyotaSupra);
        cocheVista.put("Subaru Impreza Sti", subaruImpreza);
        cocheVista.put("Porsche 911 Turbo", porscheTurbo);*/
        buttonStart.setOnAction(event -> {

            final int distancia = 0;
            Coche c1 = new Coche("Ferrari F40", 180, distancia, this);
            Coche c2 = new Coche("Toyota Supra MK4", 180, distancia, this);
            Coche c3 = new Coche("Subaru Impreza Sti", 180, distancia, this);
            Coche c4 = new Coche("Porsche 911 Turbo", 180, distancia, this);
//guardar
            coches.clear();
            coches.add(c1);
            coches.add(c2);
            coches.add(c3);
            coches.add(c4);

            System.out.println("Ha empezado la carrera!");
            carreraActiva = true;
            animacionStart();
```

```
            c1.start();
            c2.start();
            c3.start();
            c4.start();

            //Mientras que los 4 coches no terminaron
//HILO PARA MOSTRAR DATOS
            Thread monitorThread = new Thread(() -> {
                try {
                    c1.join();
                    c2.join();
                    c3.join();
                    c4.join();
                    System.out.println("Fin!");
                    carreraActiva = false;
                  //  resultadoFin();
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            });
            monitorThread.start();

//ACTUALIZAR HILO
            Thread actualizadorThread = new Thread(() -> {
                while (!fin()) {
                    try {
                        Thread.sleep(100);
                        actualizarTodo();
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }}


                });
            actualizadorThread.start();

        });}
    //fin indice
        //-------------------------------------------------------------
        public void actualizarPosicionCoche(String nombreCoche, double distancia
            ImageView imagenCoche = obtenerImageViewPorNombre(nombreCoche);
            if (imagenCoche != null) {
                //Sin getPosicionInicial explota
                double nuevaPosX = getPosicionInicial(nombreCoche) + (distanciaR
                imagenCoche.setLayoutX(nuevaPosX);
            }
```

```
        }
         private ImageView obtenerImageViewPorNombre(String nombreCoche) {
            switch (nombreCoche) {
                case "Ferrari F40": return ferrariF40;
                case "Toyota Supra MK4": return toyotaSupra;
                case "Subaru Impreza Sti": return subaruImpreza;
                case "Porsche 911 Turbo": return porscheTurbo;
                default: return null;
            }
        }

        private double getPosicionInicial(String nombreCoche) {
            switch (nombreCoche) {
                case "Ferrari F40": return 64.0;
                case "Toyota Supra MK4": return 48.0;
                case "Subaru Impreza Sti": return 37.0;
                case "Porsche 911 Turbo": return 26.0;
                default: return 50.0;
            }
        }

        private void actualizarTodo() {
            for (Coche coche : coches) {
                if (coche.isAlive() || coche.getDistanciaRecorrida() > 0) {
                    actualizarPosicionCoche(coche.nombre, coche.getDistanciaReco
                }
            }
        }

        private boolean fin() {
            for (Coche coche : coches) {
                if (coche.isAlive()) {
                    return false;
                }}
            return true;
        }
//-----------------------------------------------------------------------

    public void agregarResultado(String mensaje) {
        if (textResultados != null) {
            Platform.runLater(() -> {
                String textoActual = textResultados.getText();
                if (textoActual == null || textoActual.isEmpty()) {
                    textResultados.setText(mensaje);
                } else {
                    textResultados.setText(textoActual + "\n" + mensaje);
```

```
                }
            });
        }
    }


    private void animacionStart(){

        fondoStart();
    };
    //FONDO=========
    private void fondoStart(){

        double barreraComienzo = barrera.getLayoutX();
        Timeline timeline = new Timeline(
                new KeyFrame(Duration.millis(50), event->{
                    if (carreraActiva) {
                    barrera.setLayoutX(barrera.getLayoutX()-30);
                    if(barrera.getLayoutX()<= (barreraComienzo -120)){
                        barrera.setLayoutX(barreraComienzo);
                    }}
                })
        );
        timeline.setCycleCount(Timeline.INDEFINITE); //LOOP
        timeline.play();}
};
```

6.2 Clase Coche

Thread que representa los coches y calcula la velocidad de tramo.

```
package com.example.cochesjoin;

public class Coche extends Thread{
    public String nombre;
    int distanciaRecorrida;
    int velocidadMaxima;
    public int posicion;
    private final MainController controller;

    public static boolean fin = false;
    private static final int distanciaCircuito=19;


    public Coche(String nombre, int velocidadMaxima,int distanciaRecorrida,Main
        this.nombre=nombre;
```

```
        this.velocidadMaxima=velocidadMaxima;
        this.distanciaRecorrida=distanciaRecorrida;
        this.posicion=0;
        this.controller=controller;
    }

//Recorrer y determinar resultados cada minuto.
    @Override
    public void run(){
        while(!fin){
        //Calculo velocidad media
            int velocidadMedia = (int) (Math.random() * (velocidadMaxima -50));
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }


            //Calcular distancia recorrida
            double distanciaTramo=(velocidadMedia / 100.0 );
            distanciaRecorrida += (int) distanciaTramo;
            controller.actualizarPosicionCoche(nombre, distanciaRecorrida);
            if (distanciaRecorrida >= distanciaCircuito && !fin) {
                Carrera.ganadores(this);
                break;
            }
        }
    }

    public double getDistanciaRecorrida() {
        return distanciaRecorrida;
    }
}
```

6.3 Clase Carrera

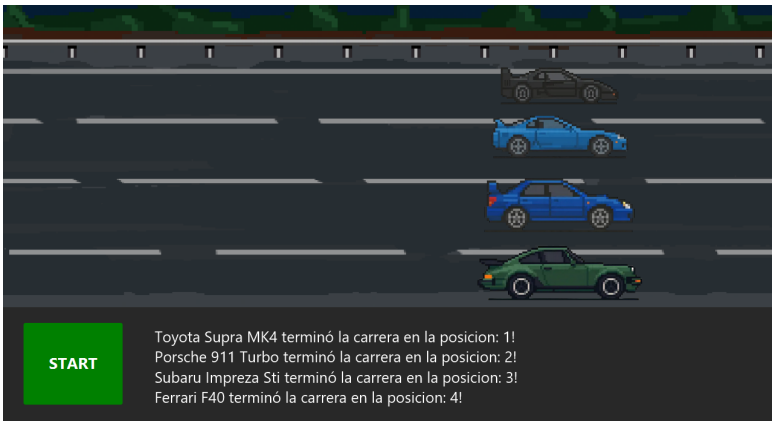
La usé para aclararme en la primera practica y gestionar los puestos de forma separada.

```
package com.example.cochesjoin;

public class Carrera {
    public static int siguientePosicion = 0;
    private static MainController controller;
    public static void setController(MainController controller) {
```

```
        Carrera.controller = controller;
    }

public static synchronized void ganadores(Coche coche) {
    siguientePosicion++;
    coche.posicion=siguientePosicion;
    controller.agregarResultado(mensajeFinal);
    if(!coche.fin){
        coche.posicion=siguientePosicion;
        if(!Coche.fin){
        if(siguientePosicion>=4){
            System.out.println("Carrera terminada!");
            coche.fin=true;
        }}
    }
}
}
```

7 Interfaz Grafica

La interfaz se compone de una barra con el boton de Start, un area para el texto y luego el elemento principal, una carretera extilo pixelart que dibuje rapidamente para darle forma al proyecto.
Los coches son assets que encontre online, los cuales tienen un estilo pixelart y dimensiones similares, perfecto para este proyecto.



8 Tecnologías Usadas

Este proyecto fue realizado con Java, JavaFX y FXML. Animaciones con Timeline de JavaFX.