UD2 - A5.- Simulación de una Carrera de Caballos con Hilos y Salida a Ficheros

---

1. Objetivo de Aprendizaje:

El objetivo de esta actividad es que los alumnos desarrollen una aplicación multihilo en la que cada hilo simule un caballo en una carrera. Los estudiantes aprenderán a controlar varios hilos, gestionar el orden de llegada y enviar la salida de cada carrera a un fichero de texto. Esta actividad también refuerza la capacidad de trabajar con múltiples procesos y manejar la salida hacia archivos.

---

2. Enunciado:

Desarrolla una aplicación en Java que simule una **carrera de caballos**. Cada caballo será una instancia de la clase `Caballo` y tendrá propiedades como su **nombre** y métodos para simular su avance en la carrera. El programa principal deberá:

1. Crear una clase `Caballo` con las propiedades y métodos necesarios para avanzar en una carrera.

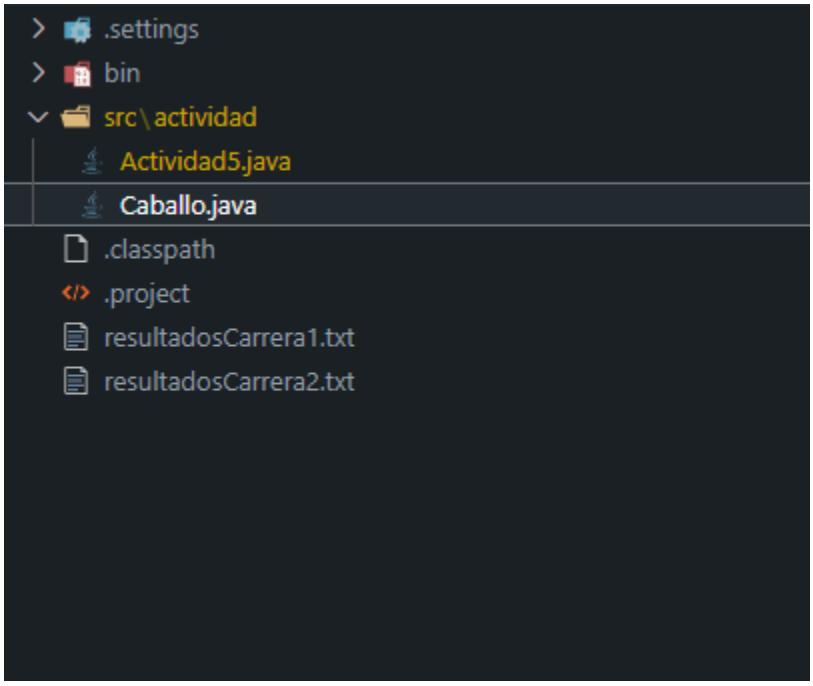
```java
package actividad;

import java.util.List;
import java.util.concurrent.CountDownLatch;

public class Caballo implements Runnable {
    private String nombre;
    private CountDownLatch latch;
    private List<String> resultados;

    public Caballo(String nombre, CountDownLatch latch, List<String> resultados) {
        this.nombre = nombre;
        this.latch = latch;
        this.resultados = resultados;
    }

    @Override
    public void run() {
        try {
            // Simula el avance del caballo en la carrera
            Thread.sleep((long) (Math.random() * 1000));
            resultados.add(nombre);
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        } finally {
            latch.countDown();
        }
    }
}

```

2. Simular la carrera lanzando varios hilos (caballos) que compitan entre sí.

```java
public class Actividad5 {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        List<Caballo> caballosCarrera1 = new ArrayList<>();
        List<Caballo> caballosCarrera2 = new ArrayList<>();
        CountDownLatch latch1 = new CountDownLatch(count:5);
        CountDownLatch latch2 = new CountDownLatch(count:5);
        List<String> resultadosCarrera1 = Collections.synchronizedList(new ArrayList<>());
        List<String> resultadosCarrera2 = Collections.synchronizedList(new ArrayList<>());

        for (int i = 1; i <= 5; i++) {
            caballosCarrera1.add(new Caballo("Caballo" + i, latch1, resultadosCarrera1));
            caballosCarrera2.add(new Caballo("Caballo" + (i + 5), latch2, resultadosCarrera2));
        }

        for (Caballo caballo : caballosCarrera1) {
            new Thread(caballo).start();
        }
        for (Caballo caballo : caballosCarrera2) {
            new Thread(caballo).start();
        }

        try {
            latch1.await();
            latch2.await();
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }

        escribirResultados(archivo:"resultadosCarrera1.txt", resultadosCarrera1);
        escribirResultados(archivo:"resultadosCarrera2.txt", resultadosCarrera2);
    }

    private static void escribirResultados(String archivo, List<String> resultados) {
        try (FileWriter writer = new FileWriter(archivo)) {
            for (String resultado : resultados) {
                writer.write(resultado + "\n");
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

3. Informar del **orden de llegada** de los caballos al final de la carrera.

4. Iniciar dos carreras simultáneamente, cada una enviando su salida (el progreso y resultados de los caballos) a un **fichero de texto** distinto.




```
1  Caballo4
2  Caballo2
3  Caballo5
4  Caballo3
5  Caballo1
6
```

resultadosCarrera2.txt

```
1  Caballo7
2  Caballo6
3  Caballo10
4  Caballo8
5  Caballo9
6
```