# cdi_actividad_3

Autores:

- Victor Malvárez Filgueira
- Martín Molina Álvarez

## 3.1 Un simple contador concurrente

## 2. Elabora una clase Contador que tiene un método Incrementar(n) que ejecuta un bucle de n iteraciones que incrementa una variable interna cada cierto tiempo, y al acabar devuelve el valor actual de dicha variable. Elabora un programa que lanza varios hilos que comparten tal objeto contador. Cada hilo, cada cierto tiempo, debe llamar al método Incrementar(n) con n adecuado e visualizar el valor devuelto por ese método. ¿Podemos garantizar que cuando un hilo ejecuta el método Incrementar() , no interferirá con otro hilo? ¿Cómo podemos solucionarlo (hay varias posibilidades)?

No.Usando Synchronized o usando un join entre los threads, en este caso no seria concurrente.

## 3.2 Introducción a la sincronización de hilos

### 4. ¿Cuál es el resultado? ¿Cuántos hilos pueden estar simultáneamente ejecutando el método EnterAndWait()?

Los hilos se inician en orden pero algunos de ellos pueden no finalizar en el mismo orden en el que se inicializaron.

### 5. Modifica el código para que solo un hilo pueda estar ejecutando el método EnterAndWait() en cualquier instante. Pruébalo. ¿Qué supone respecto del grado de concurrencia del código este cambio?

El código no es concurrente porque solo 1 hilo puede estar llamando al EnterAndWait() de la clase A y por lo tanto cada llamada se ejecuta en secuencial.

### 6. ¿Tendría el mismo efecto el hacer que sólo un hilo pueda ejecutar el

**método run() de la clase B? ¿Por qué o por qué no?**

No, porque existen varios objetos de la clase B. Cada objeto es accedido por un único hilo, por lo tanto no se comparte y no hace falta sincronizarlo. Poner el `synchronized` en el `run()` es lo mismo que no ponerlo a efectos prácticos en este ejemplo.