3. Actividad: Sincronización de hilos

Objectivos: Comprender cómo se pueden comunicar varios hilos de ejecución compartiendo variables comunes de manera segura usando sincronización. Saber definir tipos de datos simples que sean seguros en entornos concurrentes usando sincronización. Conocer las principales operaciones sobre hilos relacionadas con la sincronización.

Metodología: Esta práctica se realizará en horas de prácticas presenciales y en horas no presenciales. El tiempo de dedicación estimada es de 2 semanas.

3.1. Repaso de las prácticas anteriores

1. Sabemos como lanzar cierto número de hilos basado en parámetros de la línea de comando.
2. Sabemos sincronizar los hilos al final del programa/hilo principal.
3. Sabemos medir el tiempo tanto de la parte concurrente del programa como de la parte secuencial del programa.
4. Sabemos distribuir una carga de trabajo a varios hilos. Los hilos realizan su trabajo de forma cuasi-paralelo e independiente y se sincronizan cuando todas las tareas hayan terminado.

3.2. Un simple contador concurrente

1. 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)?

3.3. Introducción a la sincronización de hilos

1. Implementa una clase `A` que contenga un solo método `EnterAndWait()`. Este método debe hacer lo siguente, en este orden:

   - imprimir un mensaje indicando cual es el hilo que está comenzando a ejecutarlo;
   - luego se detendrá durante unos segundos;
   - y vuelve a imprimir otro mensaje indicando el hilo que está acabando de ejecutar el método.

2. Realiza una clase `B` que implemente `Runnable`, que reciba como parámetro un objeto de la clase `A`, y en el método `run()` simplemente llame al método `EnterAndWait()` de ese objeto.

3. Una clase con un método `main()`, en el que se crea un objeto de la clase `A`, y varios objetos de la clase `B` a los que se les pasa a todos como parámetro el mismo objeto de la clase `A`. Es decir, todos los objetos de la clase `B` compartirán el mismo objeto de la clase `A`. Después se crearán y ejecutarán el mismo número de hilos que de objetos de tipo `B` tengamos, pasándoles como parámetros los objetos de la clase `B`, de forma que cada método `run()` de cada objeto de clase `B` se ejecute en un hilo diferente.

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

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?

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?