4. Actividad: Sincronización total de hilos

Objectivos: Comprender cómo se pueden intercalar 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 y dificultades 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.

4.1. 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.
5. Sabemos usar variables simples compartidos.
6. Sabemos usar las posibilidades de Java para sincronizar hilos de forma simple.

4.2. Un juego: ping-pong-múltiple

Queremos hacer un programa que simule un juego de ping-pong (pero con mucho más jugadoras que pasan la pelota entre ellas en un orden preestablecido). Para ello hemos de definir una clase llamada `Ping` que simulará una jugadora con su método `run()` que será ejecutado en su hilo. Es necesario que cada hilo saque un mensaje indicando qué jugadora juega.

Para realizar esta actividad:

1. Experimenta con el material presentado en clases de teoría.
2. Usa—parecido al juego real—un objeto que haga de *pelota*, de forma que cada jugadora sepa cuando tiene que jugar, que será cuando la pelota esté *en su posesión*.

   Averigua para ello cómo se puede usar un objeto en Java para sincronizar varios hilos.
3. Usa el hilo principal con el papel de *árbitro*, es decir, es el hilo que inicia y termina el partido.
4. Implementa diferentes criterios de parada, por ejemplo: número de jugadas, tiempo de jugadas, interrupción... El programa debe terminar todos los hilos participantes de forma explícita. Como siempre, el hilo principal termina el programa con el último mensaje.
5. Utiliza en tu solución los métodos `wait()`, `notify()` y `notifyAll()` de Java. ¿Observas diferencias entre `notify()` y `notifyAll()`?

6. Queremos que el árbitro decida quien será el hilo que comience a jugar.

7. Argumenta de forma semi-formal que tu programa es correcto y garantiza la semántica del juego. ¿Cómo puedes comprobar automáticamente que la salida del programa es la deseada?

8. Mide el tiempo de ejecución—con un criterio de terminación razonable—de tu partido variando el número de jugadoras de unas pocas a unas miles. ¿Qué resultado esperas para un número constante de *pasar-pelota* independientemente del número de jugadores? ¿Qué dicen tus mediciones? ¿Cuál es el método más eficiente?

9. Si se interrumpe un hilo que participa en el juego ¿qué pasa con los demás? ¿Puedes sugerir un remedio?

10. ¿Funciona tu programa concurrente con una sola jugadora también? Si no lo hace ¿qué deberías cambiar?