TRES.

Daniel Camba Lamas, Román Puga Quintairos.

3.2) ¿Podemos garantizar que cuando un hilo ejecuta el método Incrementar(), no interferirácon otro hilo? ¿Cómo podemos solucionarlo?

No sin aplicar ciertos elementos de sincronización extra, como declarar la variable a modificar como **Volatile** ó declarar como **Synchronized** el método o métodos que vayan a modificar la variable a proteger de inconsistencias.

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

Todos los que lo llamen pueden ejecutarlo de manera simultanea.

3.3.5) 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 éste cambio?

Añadiendo **Synchronized** en la declaración del método de la *clase **A***. Esto supone que la ejecución de dicho método se resolverá de manera *secuencial* no *paralela*.

3.3.6) ¿Tendría el mismo efecto el hacer

que sólo un hilo pueda ejecutar el método run() de la claseB? ¿Por qué o por qué no?

Tendría el mismo efecto si el código fuera a quedar así, pero si quisieramos añadir más métodos en `run()` todos ellos sufrirían la misma suerte de ejecución *secuencial* de forma que es mejor aplicarlo sólo a los métodos concretos en los que necesitemos controlar posibles inconsistencias, no al método principal del hilo, ya que entonces perderíamos todo el potencial que la concurrencia nos aporta.