Entrega 4 - Sincronización de hilos

1. a) java.util.concurrent.locks

Tras la eliminación de los bloques **synchronized** el cambio del tipo de dato del contador a **AtomicInteger** no es suficiente para mantener el contador sincronizado. Para conseguir esto se utiliza la interfaz **Lock** implementada en la clase **ReentrantLock**.

Se crea un objeto de tipo ReentrantLock en el Thread y aplicamos justo antes del incremento la siguiente orden: "ReentratLock.lock()" bloqueando el acceso a esa parte de código hasta que sea desbloqueada. Esta orden está embebida dentro de un bloque try-catch-finally donde se utiliza finally como parte para hacer la operación de desbloqueo: "ReentrarLock.unlock()". Obteniendo como resultado la siguiente salida:

Hi, i'm Hilo 0 the acum value is 0 and I increment to 2
Hi, i'm Hilo 0 the acum value is 2 and I increment to 4
Hi, i'm Hilo 0 the acum value is 4 and I increment to 6
Hi, i'm Hilo 0 the acum value is 6 and I increment to 8
Hi, i'm Hilo 0 the acum value is 8 and I increment to 10
Hi, i'm Hilo 0 the acum value is 10 and I increment to 12
Hi, i'm Hilo 0 the acum value is 12 and I increment to 14
Hi, i'm Hilo 0 the acum value is 14 and I increment to 16
Hi, i'm Hilo 0 the acum value is 16 and I increment to 18

Realización de pruebas

Para aplicar las pruebas se ha eliminado todo lo que puede hacer variable el tiempo. Para ello eliminamos todas las esperas y fijamos el número de ejecuciones, pues un numero aleatorio haría que los datos no fuesen significativos. Además, eliminamos todas las salidas por consola excepto la final, que mostrará el tiempo de ejecución.

Una vez hecho esto hemos realizado pruebas con un número creciente de incrementos tanto para el código realizado con **Lock** como el realizado con **Synchronized**.

Los resultados obtenidos no seguían una progresión, ya que se veían afectados por el entorno de ejecución. Para que esto afectase lo mínimo posible se realizaron 3 veces cada una de las pruebas (Figura 1 y 2) y se usó la media de estas pruebas para comparar los 2 métodos de sincronizado (Figura 3).

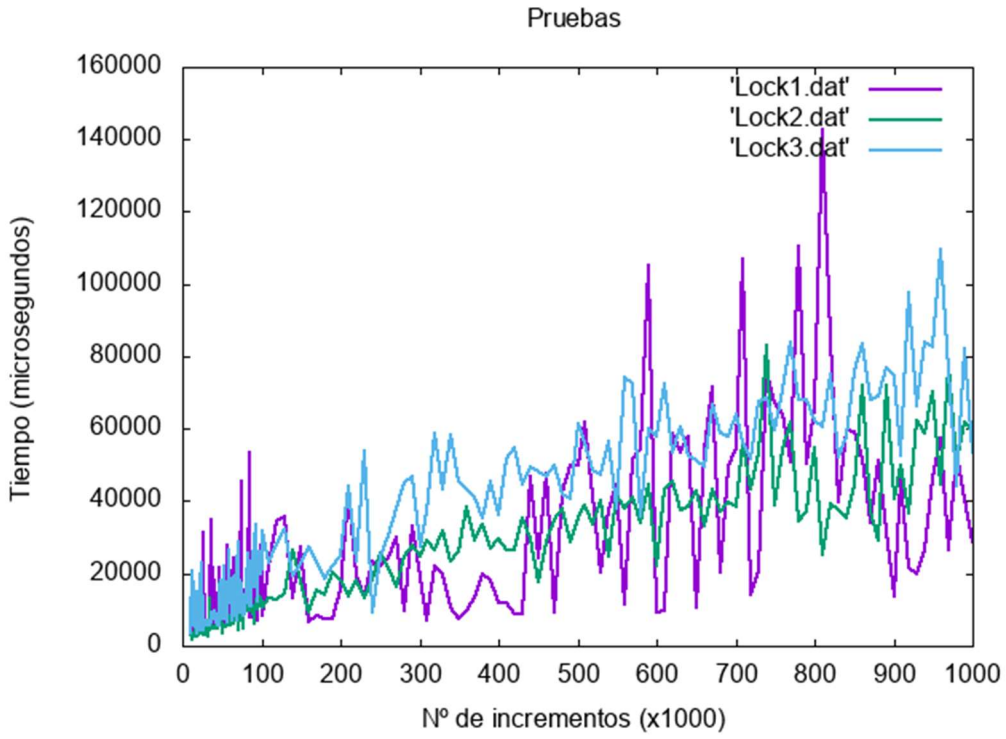

FIGURA 1

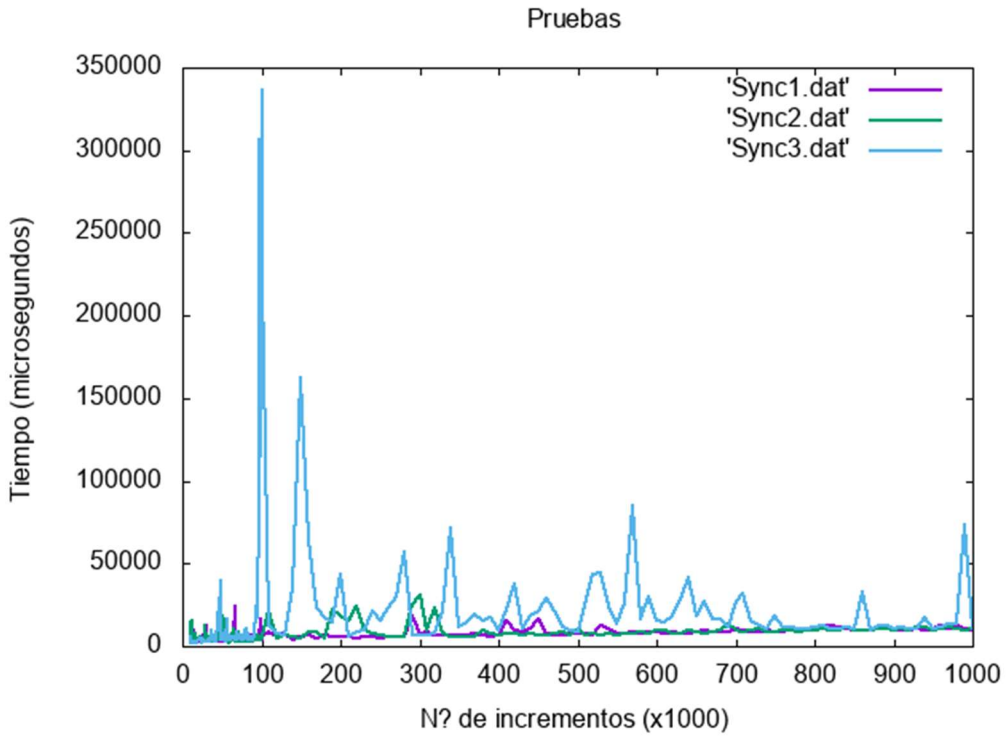

FIGURA 2

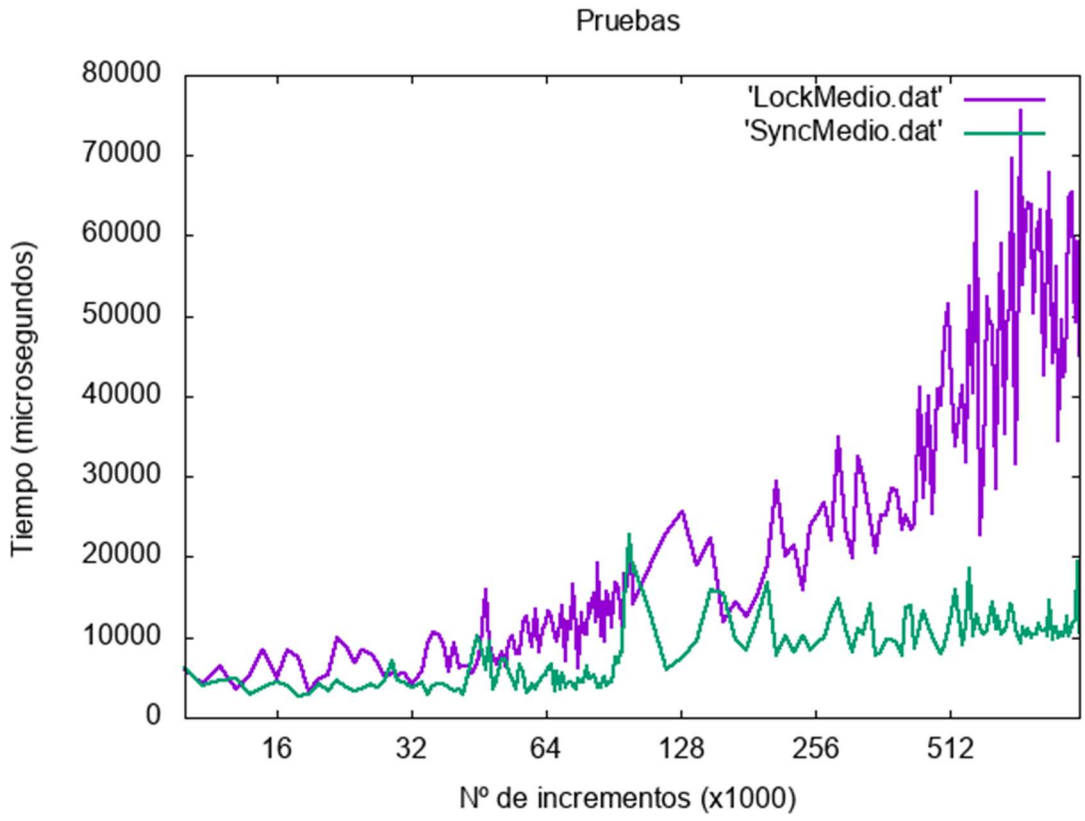

FIGURA 3

Los resultados obtenidos reflejan que para las pruebas realizadas la efectividad del método sincronizado bajo **Synchronized** es bastante más efectiva que el empleo de **Lock** sobre todo cuanto más largo sea el trabajo a ejecutar.