Práctica 1

Autores:

Víctor José Malvárez Filgueira

Martín Molina Álvarez

1. Introducción a los hilos java

1.1 Examina las dos formas que provee Java para crear un hilo: la clase Thread y la interfaz Runnable

(Sin respuesta)

1.2 Utiliza ambas formas para crear un programa que cree y ejecute un hilo que imprima en pantalla un mensaje como Hello world, I'm a java thread.

¿Hay alguna diferencia de funcionamiento entre ambas formas? ¿A nivel de diseño, cuál te parece preferible, y por qué?

No hay diferencias observables de funcionamiento. A nivel de diseño nos parece preferible runnable porque permite herencia de otras clases, mientras que en Thread estas obligado a heredar de esta clase (java no permite herencia multiple) a pesar de que la sintaxis es mas simple.

1.3 Modifica tu programa para que el hilo tarde aproximadamente un segundo en mostrar el mensaje. ¿Qué método usarás para ello?

Usamos el método estático `sleep` de la clase `Thread`.

1.4 Cuando arrancas un hilo desde el programa principal, ¿cuántos hilos hay activos? Saca una lista de los hilos activos por pantalla.

Hay 5 hilos activos: ID: 4, Name: Signal Dispatcher ID: 2, Name: Reference Handler ID: 1, Name: main ID: 8, Name: Thread-0 ID: 3, Name: Finalizer

2. Creación de múltiples hilos

2.1 Escribe un programa en Java que mediante parámetros de línea de commando reciba cuantos hilos se debe crear. El programa creará y ejecutará el número de hilos indicado. Cada hilo debe imprimir en pantalla un mensaje como Hello, I'm thread number X , y después de un segundo, un mensaje como Bye, this was thread number X , siendo X el

valor de un contador que se va incrementando con el número de hilos creados.

2.2 Queremos ahora medir el tiempo que tardan en ejecutarse todos los hilos que se crean. Copia y modifica el programa anterior para que muestre este tiempo. Asegúrate de quitar cualquier código que implemente esperas en el hilo, que pudieras haber añadido anteriormente, para medir realmente sólo el tiempo de ejecución de los hilos.

2.3 ¿Cómo debemos hacer para asegurarnos de que la medida de tiempo sea fiable? Es decir, que realmente se mide el tiempo desde que se arranca el primer hilo hasta que todos los hilos hayan finalizado. ¿Qué errores de medida de tiempo pueden ocurrir, si no nos aseguramos de que todos los hilos han acabado?

Utilizando em metodo join en cada hilo. Que se ejecute el codigo del main antes de que termine el ultimo hilo.

2.4 Copia y modifica el programa para que cada hilo mida él mismo el tiempo que tarda en ejecutarse y guarde el valor en un atributo accesible públicamente. Una vez finalizados todos los hilos, haz que el programa principal sume todos estos valores y muestre en pantalla el resultado. ¿Este valor será mayor o menor que el tiempo global que ya estabas midiendo? ¿Por qué? ¿Te sirve esta suma para algo? ¿Qué harías tú con estos valores para tener algo interesante? Compara tus mediciones con la salida del comando time en Linux que te ofrece tiempo real, tiempo del user y tiempo del sistema.

El tiempo total es mayor. Este tiempo es similar al que se tardaría si no se utilizase un sistema concurrente y cada hilo se ejecutase en secuencial.

Esta suma no es fiable. Desde que se ejecuta el hilo (empieza a contar el tiempo) hasta que de destruye (se para el contador de tiempo) la CPU no está siendo dedicada en su totalidad a dicho hilo, sino que el sistema operativo aplica una estrategia de reparto en la que se le puede quitar y ceder varias veces (dependiendo del número de instrucciones) pero sin embargo el contador de tiempo considera que se la ha estado dedicando integramente.

El tiempo real coincide con el tiempo de ejecucción de hilos global. El tiempo de usuario es mayor que el tiempo del sistema debido a que el tiempo del usuario es el tiempo que está "en el procesador" ejecutándose y el tiempo de sistema es el tiempo que está "en el

kernel" ejecutándose.

2.5 Trata de modificar el programa para poder distinguir entre tiempo de creación de hilos, tiempo de ejecución de los hilos y tiempo de sincronzación final de los hilos*. Si tan pronto como se crea cada hilo ya se intenta ejecutar, ¿se puede distinguir entre estos tiempos? ¿Qué consecuencias tiene esto a la hora de diseñar y evaluar soluciones concurrentes a un problema?

No.

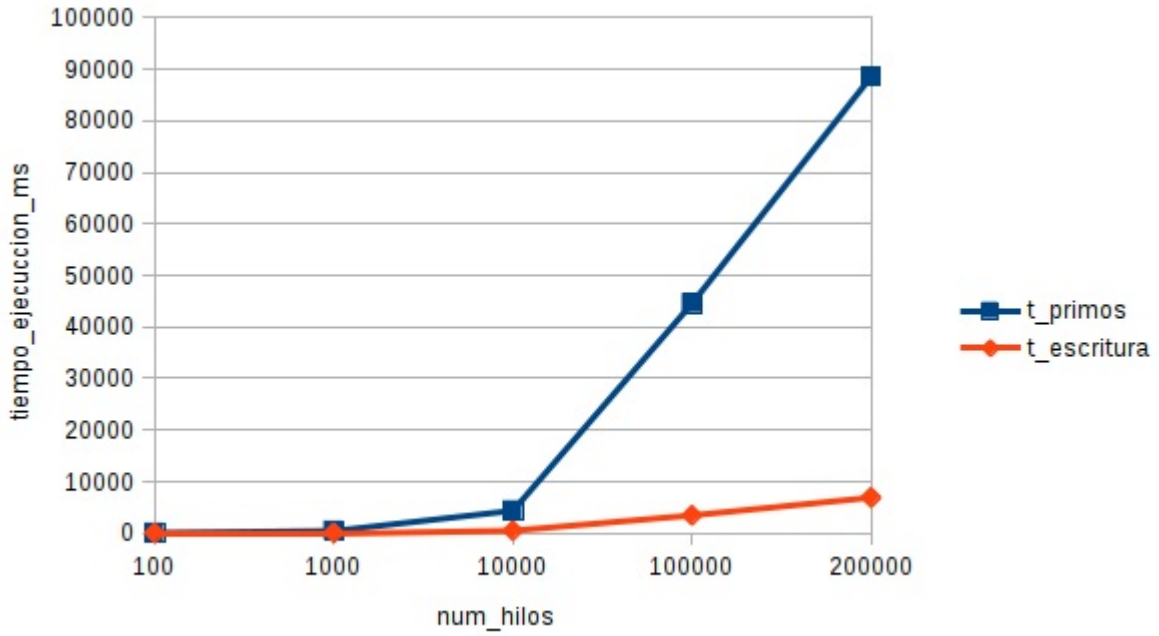
Si se crea el hilo y se intenta ejecutar no se puede distinguir el tiempo de creación con el de ejecucción de manera precisa.

2.6 Copia y modifica el programa para que los hilos o bien muestren algo por pantalla o bien que hagan por ejemplo unas operaciones matemáticas suficientemente complejas. Distingue el modo de ejecución mediante un parámetro en línea de comando. La ejecución del programa por prueba debe durar unos segundos para obtener mediciones interpretables. Pruébalo con un número creciente de hilos para conseguir finalmente una función de tiempo de ejecución en relación al número de hilos trabajadores (y modo de ejecución). ¿Cambia mucho el tiempo de ejecución respecto a cuándo se mostraba algo en pantalla? ¿Por qué crees que ocurre esto? ¡Interpreta detenidamente los gráficos que obtienes!

¡Es interesante que antes de realizar las mediciones intentes predecir los resultados por lo menos cualitativamente y averiguar si tu predicción coincida con los resultados medidos! ¡Es interesante realizar las mediciones en diferentes entornos, respecto a hardware, sistema operativo, y máquina virtual de Java!

El tiempo al realizar una operación matemática compleja es superior al tiempo de escritura en pantalla por la propia naturaleza del algoritmo matemático (generar números aleatorios en 1 y 1 millón hasta encontrar uno que sea primo).

A medida que se añaden más hilos (x10 en cada paso) el tiempo total es lineal (se ve multiplicado por 10) salvo entre 1000 y 1000. (ver anexo 2.6.1 y 2.6.2).



Anexo 2.6.2.

3. Control básico de múltiples hilos

3.1 Estudia en detalle la utilidad del método `interrupt()` de la clase Thread. ¿Qué limitaciones tiene? ¿Cómo debe ser utilizado? Aunque existen otros métodos en la clase `Thread` como `stop()` o `suspend()` ¿Por qué crees que el método `interrupt()` es el método recomendado para detener un hilo?

Qué es el programador el responsable de actuar ante una interrupción.

En el método run (o métodos que sean invocados a partir de este) se debe comprobar si el thread está interrumpido con el método `isInterrupted` de los threads o con el método estático `interrupted` de la clase `Thread`.

Sí, es preferible utilizar `interrupt` porque permite al programador establecer la forma en la que debe terminar un hilo. El método `suspend` no se debe utilizar para finalizar un hilo porque no libera la memoria del thread ya que permite que este sea renaudado con el método `resume`.

3.2 ¿Qué diferencia hay entre los métodos `interrupted()` y `isInterrupted()`? Elabora un programa en Java que demuestre claramente cómo funcionan los dos y en qué se diferencian.

El método `interrupted()` es un método estático de la clase `Thread` que permite comprobar si el thread actual ha sido interrumpido. Cada vez que se invoca el flag que

contiene si el thread está interrumpido o no se resetea, es decir, limpia las interrupciones.

El método `isInterrupted()` es un método de instancia de `Thread` que permite comprobar si ese thread ha sido interrumpido. Mantiene el flag de interrupciones sin modificarlo.

3.3 Utiliza uno de estos dos métodos (interrupted() o isInterrupted()) para elaborar un programa que lance un hilo que debe:

- mostrar un mensaje de inicio;
- luego debe dejar pasar unos 10 segundos, imprimiendo un mensaje aproximadamente cada segundo,
- y luego mostrar otro mensaje de fin, y terminar.

El usuario debe poder enviar una señal de interrupción al hilo. Si esta señal llega antes de que hayan transcurrido 5 segundos, el hilo debe mostrar un mensaje de que ha recibido la señal, y volver a ponerse a esperar, hasta terminar los 10 segundos (aproximadamente). Si llega una vez pasados 5 segundos, debe mostrar un mensaje de que ha sido interrumpido y acabar de inmediato.

+