CDI Práctica 1.

Daniel Camba Lamas : 44474565V Román Puga Quintairos : 44496690Q

Apartado Uno.

Subapartado Dos.

De funcionamiento que *Runnable* precisa la creación de un objeto de la calse *Thread*. A nivel de diseño, pienso que un código escrito con la clase *Thread* resultará más legible y al menos ganará eficiencia en el sentido de no necesitar crear un objeto para las acciones con el hilo principal.

Subapartado Tres.

El método `sleep(long var)`.

Subapartado Cuatro.

Usando `activeCount()` el contador de hilos activos que proporciona la clase Thread, sabemos que hay dos hilos activos, el hijo y el padre, los listamos mediante el método `list()` de la clase ThreadGroup.

Apartado Dos.

Subapartado Tres.

Debemos asegurarnos de que el padre espera a que termine la ejecución de todos sus hijos, para ello almacenamos dichos hijos en un array y luego para cada elemento del array lanzamos el método `join()`, si no hacemos esto, se mostraria un tiempo no real debido a que se mostraría una vez terminasemos de lanzar todos los hijos.

Subapartado Cuatro.

En nuestro caso resulta mayor, porque el hilo muestra información por pantalla

y cualquier proceso que deba enviar información fuera del procesador tardará un tiempo mayor. Dicho valor nos permite saber el tiempo de ejecución de los hijos lanzados. Respecto al comando `time` el tiempo total medido y el tiempo total que éste devuelve son aproximados, siendo el devuelto por `time` mayor, debido a factores de la máquina virtual de java, activación de procesos, etc.

Subapartado Cinco.

Sí, hallando la diferencia entre el instante anterior y posterior de lanzar el método `start()`.

Hay que tener en cuenta que cuando lanzas un hilo con `start()` además de lanzar el hilo, se inicia directamente el método `run()`.

Subapartado Seis.

La variación de tiempo es notable, siendo mayor cuando se muestra algo por pantalla, ya que al hacer esto el proceso debe enviar información fuera del procesador, llenar buses de datos, etc.

Apartado Tres.

Subapartado Uno.

El método `interrupt()` sólo interrumpe el hilo, cuando se recoja dicha excepción en los hilos.

Se usa `interrupt()` frente al resto, porque así puedes evaluar distintos factores, los cuales una llamada `stop()` no evaluar.

Subapartado Dos.

`interrupted()` devuelve un booleano y limpia el estado de interrupción.

`isInterrupted()` también devuleve un booleano pero no limpia el estado de interrupción.