Tema 4: Hilos 1

1. Introducción.

Los programas realizan actividades o tareas, y para ello pueden seguir uno o más flujos de ejecución. Dependiendo del número de flujos de ejecución, podemos hablar de dos tipos de programas:

- **Programa de flujo único:** Es aquel que realiza las tareas una a continuación de la otra, de manera secuencial, lo que significa que cada una de ellas debe concluir por completo, antes de que pueda iniciarse la siguiente.
- **Programa de flujo múltiple:** Es aquel que coloca las tareas a realizar en diferentes flujos de ejecución, de manera que cada uno de ellos se inicia y termina por separado, pudiéndose ejecutar éstos de manera simultánea o concurrente.

La programación multihilo o consiste en desarrollar programas o aplicaciones de flujo múltiple. Cada uno de esos flujos de ejecución es un hilo.

2. Conceptos sobre hilos.

Un hilo, denominado también subproceso, es un flujo de control secuencial independiente dentro de un proceso y está asociado con una secuencia de instrucciones, un conjunto de registros y una pila.

Podemos hacer las siguientes observaciones:

- Un hilo no puede existir independientemente de un proceso.
- Un hilo no puede ejecutarse por si solo.
- Dentro de cada proceso puede haber varios hilos ejecutándose.
- Un único hilo es similar a un programa secuencial; por si mismo no nos ofrece nada nuevo.
- Es la habilidad de ejecutar varios hilos dentro de un proceso lo que ofrece algo nuevo y útil, ya que cada uno de estos hilos puede ejecutar actividades diferentes al mismo tiempo.

1. Recursos compartidos por los hilos.

Un hilo lleva asociados los siguientes elementos:

- Un identificador único.
- Un contador de programa propio.
- Un conjunto de registros.
- Una pila (variables locales).

Por otra parte, un hilo puede compartir con otros hilos del mismo proceso los siguientes recursos:

- Código.
- Datos (como variables globales).

- Otros recursos del sistema operativo, como los ficheros abiertos y las señales.

El hecho de que los hilos compartan recursos (por ejemplo, pudiendo acceder a las mismas variables) implica que sea necesario utilizar esquemas de bloqueo y sincronización, lo que puede hacer más difícil el desarrollo de los programas y así como su depuración.

2. Ventajas y uso de hilos.

Como consecuencia de compartir el espacio de memoria, los hilos aportan las siguientes ventajas sobre los procesos:

- Se consumen menos recursos en el lanzamiento y la ejecución de un hilo que en el lanzamiento y ejecución de un proceso.
- Se tarda menos tiempo en crear y terminar un hilo que un proceso.
- La conmutación entre hilos del mismo proceso es bastante más rápida que entre procesos.

Es por esas razones, por lo que a los hilos se les denomina también procesos ligeros.

Se aconseja utilizar hilos en una aplicación cuando:

- La aplicación maneja entradas de varios dispositivos de comunicación.
- La aplicación debe poder realizar diferentes tareas a la vez.
- Interesa diferenciar tareas con una prioridad variada. Por ejemplo, una prioridad alta para manejar tareas de tiempo crítico y una prioridad baja para otras tareas.
- La aplicación se va a ejecutar en un entorno multiprocesador.

3. Multihilo en Java. Librerías y clases.

1. Utilidades de concurrencia del paquete java.util.concurrent.

Concretamente estas utilidades están dentro de los siguientes paquetes:

- **java.util.concurrent:** En este paquete están definidos los siguientes elementos:
  - *Clases de sincronización:* Semaphore, CountDownLatch, CyclicBarrier y Exchanger.
  - *Interfaces para separar la lógica de la ejecución:* Executor, ExecutorService, Callable y Future.
  - *Interfaces para gestionar colas de hilos:* BlockingQueque, LinkedBlokingQueque, nArrayBlockingQueque, SynchronousQueque, PriorityBlockingQueque y DelayQueque.
- **java.util.concurrent.atomic:** Incluye un conjunto de clases para ser usadas como variables atómicas en aplicaciones multihilo y con diferentes tipos de dato, por ejemplo AtomicInteger y AtomicLong.
- **java.util.concurrent.locks:** Define una serie de clases como uso alternativo a la cláusula sinchronized. En este paquete se encuentran algunas interfaces como por ejemplo Lock, ReadWriteLock.

4. Creación de hilos.

En Java, un hilo se representa mediante una instancia de la clase java.lang.thread. Este objeto thread se emplea para iniciar, detener o cancelar la ejecución del hilo de ejecución.

Los hilos o threads se pueden implementar o definir de dos formas:

- **Extendiendo la clase thread.**
- **Mediante la interfaz Runnable.**

En ambos casos, se debe proporcionar una definición del método run(), ya que este método es el que contiene el código que ejecutará el hilo, es decir, su comportamiento.

Extender la clase thread es el procedimiento más sencillo, pero no siempre es posible. Si la clase ya hereda de alguna otra clase padre, no será posible heredar también de la clase thread (recuerda que Java no permite la herencia múltiple), por lo que habrá que recurrir al otro procedimiento.

Implementar Runnable siempre es posible, es el procedimiento más general y también el más flexible.

Para saber qué hilo se está ejecutando en un momento dado, utilizamos el método currentThread() y obtenemos su nombre invocando al método getName(), ambos de la clase thread.

1. Creación de hilos extendiendo la clase Thread.

Para definir y crear un hilo extendiendo la clase thread, haremos lo siguiente:

- Crear una nueva clase que herede de la clase thread.
- Redefinir en la nueva clase el método run() con el código asociado al hilo.
- Crear un objeto de la nueva clase thread. Éste será realmente el hilo.
- Una vez creado el hilo, para ponerlo en marcha o iniciarlo invocamos al método start() del objeto thread.

2. Creación de hilos mediante la interfaz Runnable.

Para definir y crear hilos implementando la interfaz Runnable seguiremos los siguientes pasos:

- Declarar una nueva clase que implemente a Runnable.
- Redefinir en la nueva clase el método run() con el código asociado al hilo.
- Crear un objeto de la nueva clase.
- Crear un objeto de la clase thread pasando como argumento al constructor el objeto cuya clase tiene el método run().
- Una vez creado el hilo, para ponerlo en marcha o iniciarlo invocamos al método start() del objeto thread.