**Universidad Autónoma del Estado de México**

**Unidad Académica Profesional Tianguistenco**

**Ingeniería en Software**

**Programación paralela**

**Procesamiento paralelo en Java**

**Maestro:** Ing. Gustavo Gómez Vergara.

**Alumno:** Daniel Jiménez Díaz.

**Grupo:** S6

**Introducción**

A la hora de desarrollar un programa con un coste computacional elevado, programar de una forma óptima y eficiente no es suficiente. Los ordenadores que utilizamos actualmente, incluso muchos de los sistemas móviles, disponen de privilegiados recursos hardware que en muy pocas ocasiones son aprovechados como se merecen.

Java proporciona soporte para la implementación y la ejecución de tareas en paralelo, pero por defecto las aplicaciones desarrolladas en este lenguaje serán ejecutadas de forma secuencial.

**¿Qué es el paralelismo?**

Paralelismo es la ejecución simultánea de dos o más tareas. Se considera una propiedad del hardware, ya que requiere recursos físicos para ejecutar cada tarea simultáneamente, y su objetivo se basa en realizar una tarea en el menor tiempo posible.

El paralelismo acelera la ejecución de una tarea dividiéndola en computaciones independientes y ejecutándola sobre hardware capaz de realizar computaciones simultáneas, como por ejemplo un procesador con varios núcleos. Pero cuando ejecutamos una tarea paralelizada en múltiples ordenadores, en vez de en los múltiples “Cores” de un solo ordenador, decimos que la computación paralela es distribuida. Por ejemplo, cada búsqueda en Google se ejecuta simultáneamente en cientos de ordenadores, cada uno de los cuales busca al mismo tiempo en un subconjunto del índice del web.

**Paralelismo vs Concurrencia**

El término paralelismo suele confundirse en numerosas ocasiones con el concepto de concurrencia. Ambos se basan en la programación multitarea, pero no son lo mismo.

Mientras que el paralelismo es la ejecución simultánea de dos o más tareas, la concurrencia es la ejecución de dos o más tareas en periodos de tiempo solapados, pero no necesariamente simultáneos. La concurrencia, por tanto, es una propiedad del programa. Su computación debe dividirse en tareas independientes, cuya ejecución puede o no solaparse. Puede existir sin soporte hardware multitarea, por ejemplo, los sistemas operativos multitarea ejecutados en máquinas con un procesador de un solo núcleo asignan breves intervalos de tiempo a cada tarea, dando la ilusión de ejecución simultánea.

Frente a la computación paralela, la programación concurrente enfatiza más la interacción entre tareas, centrándose en coordinar el acceso a recursos mutables compartidos mediante algoritmos que evitan la interferencia entre tareas y la inconsistencia de los recursos.

**Procesamiento paralelo en Java**

En Java el soporte a la concurrencia (paralelismo) basado en hilos se hace en el propio lenguaje y no en base a una librería auxiliar (p.ej. PThreads, hilos win32).

El soporte de Java a los hilos se hace mediante los siguientes elementos:

1. Dotando a la clase ancestral Object de los métodos **Wait()**, **Notify()** y **NotifyAll()** que dan soporte a los eventos.
2. Asociando un lock a cada clase y a cada objeto, la obtención y la devolución del lock se hace mediante la utilización de la palabra reservada synchronized mediante la cual se definen métodos o bloques de código a los que se accede en exclusión mutua.
3. Proporcionando la clase Thread en la librería java.lang que implementa los mecanismos básicos para realizar hilos.
4. La interfaz Runnable permite la creación de hilos en clases que extienden otra clase diferente de Thread.
5. Otras funcionalidades disponibles son:
   1. El control de prioridad
   2. La posibilidad de limitar el número de hilos activos mediante la clase ThreadPool.
   3. La posibilidad de controlar grupos de hilos mediante la clase ThreadGroup.

**Creación y control de un hilo**

**Métodos de creación de un hilo.**

En Java se puede crear un hilo mediante dos mecanismos:

1. Extendiendo la clase Thread
2. Implementado la interfaz Runnable

Tanto Thread como Runnable forman parte de java.lang con lo que no requieren ser importados explícitamente.

Ambos métodos tienen en común el procedimiento básico que consiste en implementar el método run(). El método para iniciar el hilo depende de la forma como se halla creado:

1. Extendiendo la clase Thread Se llama al método start de los objetos que instancian la clase
2. Implementando Runnable Se crea un objeto Thread con el constructor que tiene como parámetro un objeto Runnable y llamando posteriormente al método start de dicho objeto.

**Control de un hilo mediante métodos de Thread.**

La clase **Thread** contiene una serie de métodos que permiten controlar el funcionamiento de los hilos:

1. ***isAlive*:** Devuelve un valor true si el hilo este activo, es decir su método run ha sido iniciado con start y todavía no ha finalizado.
2. ***join*:** Condiciona la continuación del hilo actual a la terminación de otro hilo, la duración de esta espera se puede limitar.

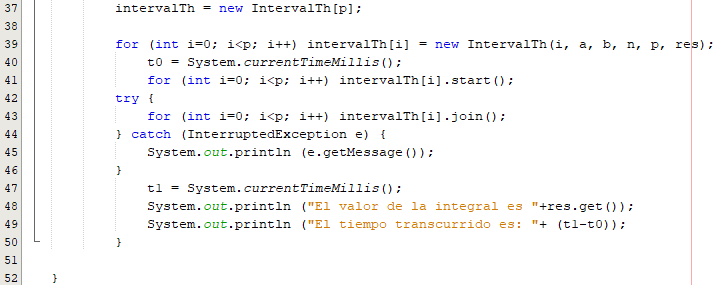
setName y getName Permiten dar un nombre a un hilo y obtener dicho nombre, respectivamente. Esto tiene interés sobre todo para la depuración. La función setName se puede evitar utilizando los constructores adecuados de Thread.

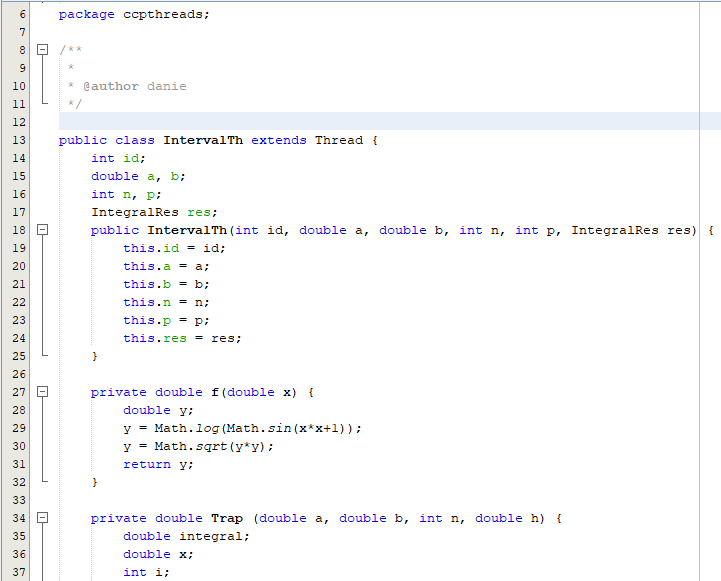
También se dispone de los siguientes métodos estáticos:

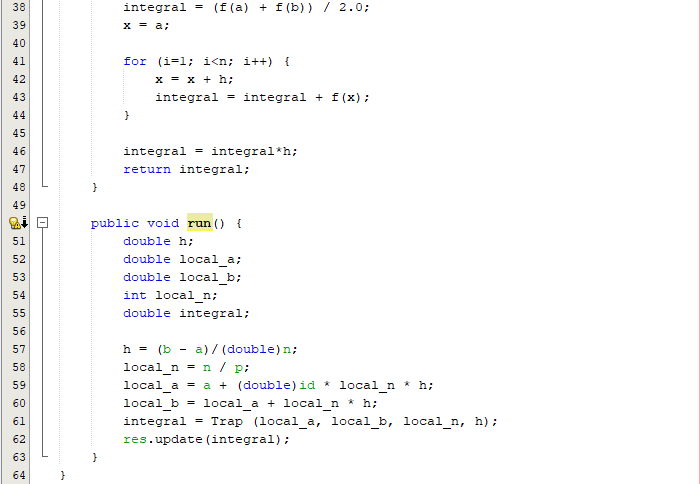
1. ***sleep***: Detiene la ejecución del hilo que hace la llamada durante el tiempo que se especifica.
2. ***currentThread***: Permite identificar al hilo que hace la llamada.
3. ***enumerate***: Devuelve una matriz con referencias a todos los hilos creados por el programa (no necesariamente activos) y su número.
4. ***activeCount***: Devuelve el número de hilos creados por el programa (no necesariamente activos).

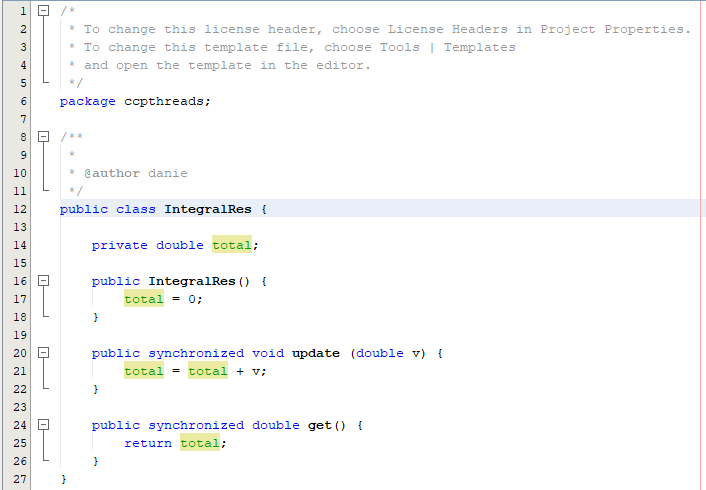
**Programa de procesamiento paralelo en java (ejemplo).**

El siguiente programa usa hilos de procesamiento para resolver una integral numérica.

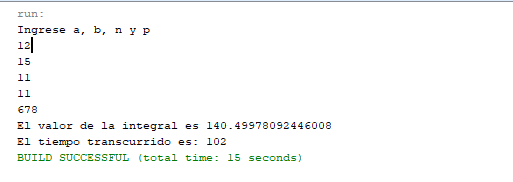
**Clase NumIntegralTh:** Es la clase principal que lanza los hilos.

**Clase IntervalTh:** Es la clase que define cada hilo.



**Clase IntegralRes:** Que define el acceso al resultado total en exclusión mutua.

**Resultados obtenidos:**

****