Concurrencia

**Nombres**: Laura Catalina Preciado Ballen

Daniel Santiago Arcila Martinez

Universidad Distrital Francisco José de Caldas

# Programación Avanzada

**Fecha:** Lunes 25 de mayo

**Colombia, Bogotá D.C**

# Resumen

Dos o más procesos decimos que son concurrentes, cuando son procesados al mismo tiempo, es decir, que para ejecutar uno de ellos, no hace falta que se haya ejecutado otro.

En sistemas multiprocesador, esta ejecución simultánea podría conseguirse completamente, puesto que podremos asignarle, por ejemplo, un proceso A al procesador A y un proceso B al procesador B y cada procesador realizarán la ejecución de su proceso.

En este ejemplo tendríamos dos hilos, uno para el proceso A y otra para el proceso B; aunque podriamos tener mas o solamente uno dependiendo del requerimiento, esto gracias al sistema operativo, y estos hilos al igual que el proceso pueden estar en estado preparado, bloqueado o activo en un cierto instante de tiempo.

A menudo se nos presenta el problema de conseguir que una aplicación funcione correctamente, pero no es solo eso, sino que lo haga de la manera más rápida para satisfacer los requisitos del cliente, por eso la concurrencia es muy importante en la programación, ya que nos facilita muchos procesos.

***Palabras clave: Concurrencia, hilos, software concurrente, esto de los hilos, paralelismo*.**

# 

# ¿Qué es un la Concurrencia?

Concurrencia es la tendencia de las cosas a producirse al mismo tiempo en un sistema. La concurrencia es un fenómeno natural, por supuesto. En el mundo real, en un momento dado, suceden muchas cosas de forma simultánea. Cuando se diseña software para supervisar y controlar los sistemas del mundo real, hay que enfrentarse a esta concurrencia natural.

Si cada actividad concurrente evolucionará de manera independiente, de forma totalmente paralela, esto sería relativamente sencillo: podríamos limitarnos a crear programas separados para gestionar cada actividad. Los retos de diseñar sistemas concurrentes derivan principalmente de las interacciones que se producen entre actividades concurrentes. Cuando las actividades concurrentes interactúan, es necesario algún tipo de coordinación.

En lo que se refiere a temas de concurrencia en sistemas de software, hay dos aspectos importantes en general: poder detectar y responder a los sucesos externos que se producen en un orden aleatorio y garantizar que estos sucesos se responden en un intervalo requerido mínimo.

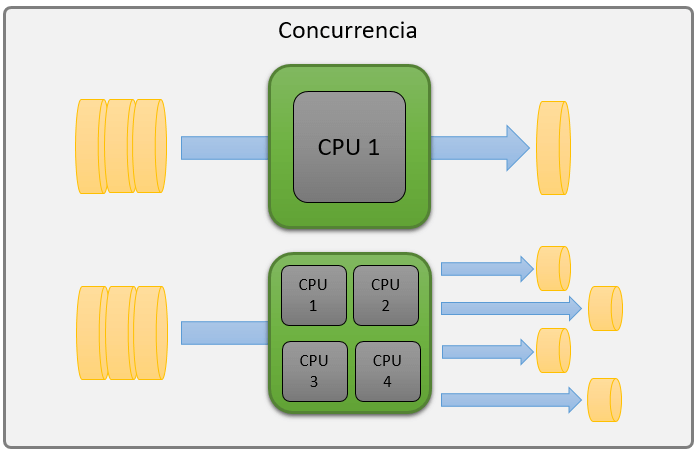
Existen dos conceptos básicos asociados a la concurrencia:

**Proceso:** Es un programa en ejecución. Tiene su propio espacio de memoria, enlaces a recursos, los procesos están aislados entre sí.

**Hilo:** Es un camino de ejecución dentro de un proceso. Cada proceso tiene al menos un hilo, llamado hilo principal. Los hilos comparten los recursos del proceso, incluida la memoria, por lo que pueden comunicarse entre sí.

# ¿Por qué nos interesa la concurrencia?

Algunas de las fuerzas conductoras de la concurrencia son externas. Es decir, que las imponen las demandas del entorno. En los sistemas del mundo real, se producen muchas cosas de manera simultánea que el software debe tratar "en tiempo real". Para ello, muchos sistemas de software en tiempo real deben ser "reactivos". Deben responder a sucesos generados externamente que pueden producirse en momentos un tanto aleatorios, en un orden algo aleatorio, o ambas cosas.

El diseño de un programa de procedimiento convencional para manejar estas situaciones es extremadamente complejo; puede ser mucho más sencillo particionar el sistema en elementos de software concurrentes para manejar cada uno de estos sucesos. La frase clave es "puede ser", ya que la complejidad también se ve afectada por el grado de interacción entre los sucesos.

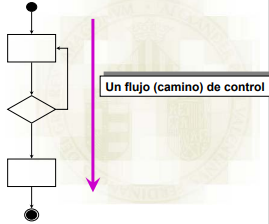
Y ya que el número de posibles rutas de ejecución en el sistema puede ser extremadamente grande, el resultado final puede ser indeterminado; esto quiere decir que el uso de recursos compartidos puede ser una fuente de Bloqueo mutuo.

# 

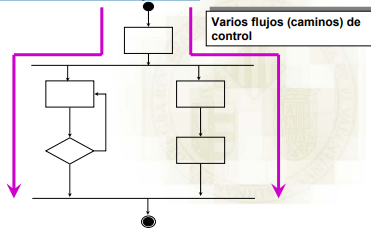
# Lenguajes Concurrentes

El concepto fundamental de la programación concurrente es la noción de Proceso.Esto exige que diversos hilos de control en un mismo programa colaboren para resolver un problema.

En un proceso tradicional, un proceso posee un único hilo de control.



En cambio un programa concurrente da lugar, a un proceso con varios hilos de ejecución.



Cabe resaltar que la concurrencia en software es un concepto lógico y no implica la existencia de paralelismo en el hardware esto significa que:

* Las operaciones hardware ocurren en paralelo si ocurren al mismo tiempo.
* Las operaciones (software) en un programa son concurrentes si pueden ejecutarse en paralelo, aunque no necesariamente deben ejecutarse así.

Y por último un lenguaje de Programación será concurrente si posee las estructuras necesarias para definir y manejar diferentes tareas (hilos de ejecución) dentro de un programa. Ejemplos: Java, Ada

# Mecanismo de Threads

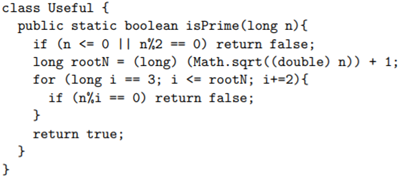
Una aplicación Java o applet puede contener secciones de código que se ejecutan simultáneamente. Una sección de código que se ejecuta independientemente se conoce con el nombre de thread o proceso ligero. Un thread comparte el espacio de direcciones con la aplicación principal o applet, y por tanto, puede acceder a los datos que son visibles a la aplicación principal o applet, así como a cualquier otro thread que se ejecuta concurrentemente.

De tal modo, debe tenerse mucho cuidado al accesar los datos compartidos, ya que un thread puede estar modificando un dato compartido mientras otro lo está leyendo.

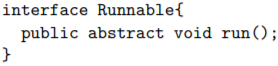
Los detalles exactos de cómo se implementa un thread dependen de la máquina donde la aplicación o applet se ejecuta. En una máquina multiprocesador puede haber una verdadera ejecución simultánea de threads, mientras que en una máquina uniprocesador la ejecución simultánea se simula mediante conmutar rápidamente entre threads individuales para dar la ilusión de una ejecución simultánea.

Por ejemplo, una aplicación en Java que prueba si un número natural es primo, y puede realizar esta actividad mientras que otros threads realizan otras actividades. El método estático **isPrime()** en la clase **Useful** prueba si un número **long** es primo o no. Sin embargo, este proceso toma bastante tiempo si el número es grande.

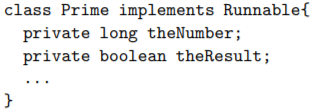
De tal modo, el código de la clase **Useful** se muestra a continuación:



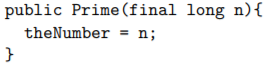
Para crear un “objeto activo”, se requiere que tal objeto implemente la interfaz **Runnable**. Esta interfaz define un solo método llamado **run()** que se sobrecarga con el método que eventualmente ejecuta un thread por separado. La interfaz **Runnable** se define como sigue:



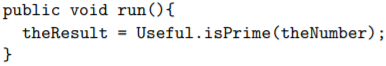
La clase **Prime** implementa la interfaz **Runnable**:

****

El constructor de la clase registra cualquier dato de entrada que requiera el thread. En este caso en particular, el n´umero que se comprueba si es primo:



El método **run()** implementa el cómputo para determinar si el número es primo. El cuerpo de este método llama al método estático **isPrime()** para realizar tal cómputo.



El método **result()** se usa para retornar el resultado del cómputo que se ha almacenado en la variable **theResult** de tipo **boolean**.



Nótese que el método **result** presenta la respuesta correcta sólo si el método **run()** ha terminado.

# 

# 

# 

# 

# 

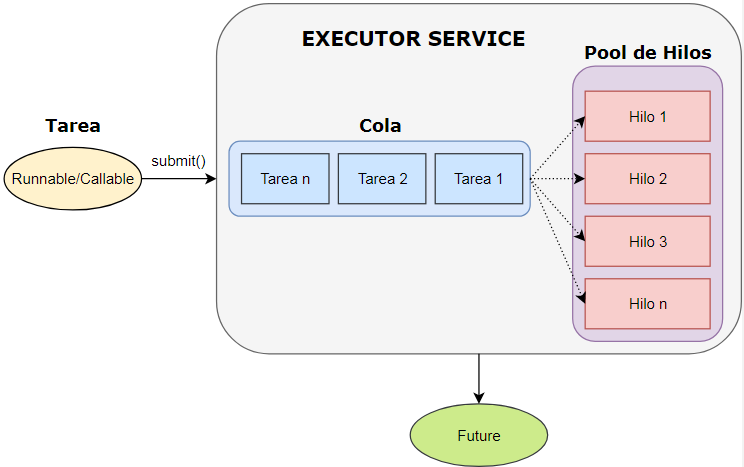
# 

# 

# Package java.util.concurrent

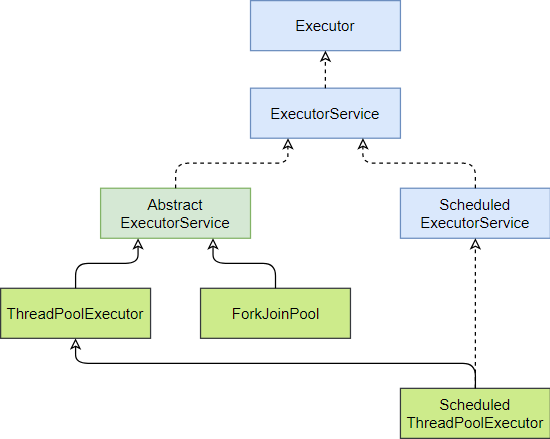
El package java.util.concurrent contiene un buen número de clases e interfaces para trabajar con concurrencia. Uno de estos es:

**Executor:** Es un framework para la ejecución de tareas de manera concurrente. Desacopla la creación y manejo de los hilos de las tareas:



Se encarga de crear un pool de hilos que llevarán a cabo las tareas. Desde fuera simplemente enviamos las tareas que se deseen realizar y éstas se añaden a una cola. El ejecutor se encarga de pasarlas a ejecución, gestionando la utilización de los hilos disponibles. Desde el lado de las tareas no hay que preocuparse de la creación de los mismos, de si están libres u ocupados, o de cerrarlos.

A modo esquemático, la API de Java ofrece la siguiente implementación del framework:



* **Executor:** Una sencilla interfaz con el método ‘**execute(Runnable)**’. Es la definición mínima del framework y se encarga de ejecutar una tarea.
* **ExecutorService:** Añade funcionalidades para el manejo del ciclo de vida de las tareas.
* **ScheduledExecutorService:** Añade funcionalidades para la programación de las tareas.

# Conclusiones

# La programación concurrente es bastante usado en la actualidad ya que gracias a este modelo podemos realizar aplicaciones tales como: Servicios WEB, sistemas multimedia, cálculo numérico, procesamientos entrada/salida. simulación de sistemas dinámicos, interacción operador/máquina (GUIs), tecnologías de componentes y código móvil.

Todo sin los problemas de la programación clásica que son:

**Modificability**:Productos muy rígido y difíciles de mantener.

**Reliability**: Presentan fallos y su depuración es muy difícil.

**Transportability**: Hay problemas para migrar entre plataforma

**Timeless**: Requieren para su ejecución más tiempo.

**Efficiency:** Sólo utilizan una parte de la capacidad de hardware.

Además de traer beneficios adicionales tales como: Facilitar el diseño orientado a objeto de las aplicaciones, ya que los objetos reales son concurrentes, hace posible compartir recursos y subsistema complejos, en sistemas monoprocesador permite optimizar el uso de los recursos, facilita la programación de tiempo real, ya que se pueden concebir como procesos cuya ejecución se planifican de acuerdo con la urgencia, permitir reducir los tiempos de ejecución sobre plataformas multiprocesadoras, facilita la realización de programas fiables por despliegue dinámico de los procesos en los procesadores.

# Bibliografías

Álvarez, D. V. (s.f.). webprogramacion. Obtenido de: <https://webprogramacion.com/1/sistemas-operativos/procesos-e-hilos.aspx>

Desconocido. (s.f.). Obtenido de: <http://informatica.uv.es/iiguia/LP/teoria/apuntes/cuatr1/tema3_1_concurrencia.pdf>

Desconocido. (31 de Enero de 2019). oracle application Express. Obtenido de: <https://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/core.base_rup/guidances/concepts/concurrency_EE2E011A.html>

Posgrado en Ciencia e Ingenier´ıa de la Computación UNAM. (s.f.). Obtenido de: <http://lya.fciencias.unam.mx/jloa/concurrencia.pdf>

Rodríguez, D. D. (29 de agosto de 2019). the softtek blog. Obtenido de: <https://blog.softtek.com/es/java-concurrency>

wikipedia. (16 de marzo de 2020). wikipedia enciclopedia libre. Obtenido de: <https://es.wikipedia.org/wiki/Concurrencia_(inform%C3%A1tica)>