

Introducción a la Programación Concurrente en Java

PROGRAMACIÓN CONCURRENTE EN JAVA – TEMA 4.1



Universidad
Rey Juan Carlos

Gestión de hilos en Java

PROGRAMACIÓN CONCURRENT EN JAVA

- **Introducción a la PC en Java**
 - **Historia de la Concurrencia en Java**
 - Finalización de hilos
 - Hilos demonios (*daemon*)
- Exclusión mutua en Java
- Sincronización condicional en Java
- Conclusiones

Historia de la Concurrency en Java

GESTIÓN DE HILOS EN JAVA

- La tecnología Java está formada por:
 - **Lenguaje de Programación Java:** Lenguaje Orientado a Objetos, con tipado estático y recolector de basura
 - **API estándar:** Conjunto de clases e interfaces disponibles en cualquier plataforma
 - **Máquina Virtual de Java (JVM):** Entorno de ejecución de Java que permite, junto con la API, que los programas escritos para esta tecnología sean portables y se puedan ejecutar en cualquier SO

Historia de la Concurrency en Java

GESTIÓN DE HILOS EN JAVA

Año	Versión Java	Concurrency
1995	Java v1.02	Hilos y monitores
1996	Java v1.1	
1998	Java 2 v1.2	Estructuras de datos concurrentes. Se desaconsejó el uso de ciertas funciones de gestión de hilos por ser inseguras (stop, pause, resume)
2004	Java 2 SE v1.5 o 5.0	Muchas mejoras en concurrency: Ejecutores, Colas de procesamiento, Temporizadores, Mejores estructuras de datos concurrentes
2006	Java SE 6	Colas de procesamiento mejoradas
2011	Java SE 7	Framework Fork/Join
2015	Java SE 8	Streams paralelos, Future, expresiones lambda

Historia de la Concurrency en Java

GESTIÓN DE HILOS EN JAVA

- Es muy importante tener en cuenta la **evolución** de las herramientas de concurrencia en Java cuando se **revise** documentación
- A lo largo del tiempo se ha ido **evolucionando** y es posible que ciertas prácticas hayan quedado **desaconsejadas**
 - Normalmente marcado por *deprecated*.
 - `Thread.stop()`, `Thread.suspend()`, `Thread.resume()`

Gestión de hilos en Java

PROGRAMACIÓN CONCURRENTE EN JAVA

- **Introducción a la PC en Java**
 - Historia de la Concurrencia en Java
 - **Finalización de hilos**
 - Hilos demonios (*daemon*)
- Exclusión mutua en Java
- Sincronización condicional en Java
- Conclusiones

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- Es muy sencillo **iniciar** la ejecución de un nuevo hilo
- En la mayoría de las ocasiones, un hilo se ejecuta hasta que ha **terminado de ejecutar** todas sus instrucciones
- Algunas veces, es necesario **detener la ejecución** de un hilo **antes** de que acabe por sí mismo

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- Esto se puede deber a:
 - **Cancelación del usuario:** A través de GUI o por red
 - **Actividades limitadas en el tiempo:** Por ejemplo algoritmos que buscan una solución y devuelven la mejor solución encontrada cuando se cumple el tiempo de búsqueda
 - **Eventos que ocurren en otros hilos de la aplicación:** Por ejemplo algoritmos implementados en varios hilos
 - **Errores en la aplicación:** Un hilo puede detectar un error que obligue a los demás hilos a finalizar

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

• Thread.stop()

- Aparentemente la mejor forma de finalizar un hilo sería disponer de un método **stop()** en la clase **Thread**
- Cuando se ejecutase el método **stop()** en el objeto de la clase **Thread** que representa el hilo, el hilo debería finalizar su ejecución
- Esta solución podría ser **peligrosa**
- La tarea que realiza el hilo que finaliza se quedaría **a medias** y los objetos que se estuvieran manipulando se quedarían en un estado **inconsistente**
- Esto produciría **errores imprevisibles** en la aplicación

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- **Thread.stop()**

- Al diseñar Java se pensó que el método **stop()** era una forma razonable de finalizar hilos
- El método stop() sigue **existiendo** en la clase **Thread**
- Desde Java 1.2 (1998) se **desaconseja** su uso (el método está **obsoleto, deprecated**) por los problemas mencionados
- No han eliminado el método en las nuevas versiones de Java por **compatibilidad** hacia atrás

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- Interrupción de un hilo
 - Para indicar a un hilo que debe finalizar su ejecución, se le envía una **interrupción**
 - Esta técnica se asemeja a las interrupciones de los procesos en los **sistemas operativos**
 - Para interrumpir un hilo, se invoca el método **interrupt()** en el objeto de la clase **Thread** que representa el hilo

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- Interrupción de un hilo
 - El hilo **no puede finalizar** su ejecución en un punto arbitrario (por los problemas indicados)
 - El hilo tiene que **preguntar periódicamente** si otro hilo le ha interrumpido
 - Si es así, debería **liberar los recursos** apropiadamente, **cerrar las conexiones** y dejar los objetos en un **estado estable**
 - Un hilo puede determinar si ha sido interrumpido invocando el método estático **Thread.interrupted()**

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

```
public static void main(String[] args) {  
    new Thread(() -> {  
        generateNumbers();  
    }).start();  
  
    Scanner teclado = new Scanner(System.in);  
    System.out.print("Pulse ENTER para finalizar...");  
    teclado.nextLine();  
  
    t.interrupt();  
  
    System.out.println("Hilo interrumpido.");  
}
```

El programa espera
a que el usuario
pulse ENTER

Se interrumpe al
proceso de generación
de números

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

```
public static void generateNumbers() {  
    try {  
        PrintWriter writer = new PrintWriter("output.txt");  
    } catch (IOException e) {  
        System.err.println("Exception using file");  
        e.printStackTrace();  
        System.exit(1);  
    }  
    while (true) {  
        BigInteger prime =  
            BigInteger.probablePrime(1024, new Random());  
        writer.println(prime.toString());  
        if (Thread.interrupted()) {  
            writer.append("Fin fichero");  
            writer.close();  
            return;  
        }  
    }  
}
```

El proceso genera números primos y los guarda en un fichero

Si se interrumpe, se cierra el fichero y finaliza la ejecución

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- Un hilo se puede bloquear a la espera de que se cumpla una condición, en un **sleep** hasta que haya pasado cierto tiempo, a la espera de datos de un **socket**...
- Es posible que **otros hilos** quieran **interrumpir** al hilo que está bloqueado
- Pero si está bloqueado no puede consultar el método **Thread.interrupted()**

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- Para solucionar este problema, la mayoría de los métodos que se bloquean a la espera de una determinada condición elevan la excepción **InterruptedException**
- Esta excepción se **eleva** cuando se **interrumpe** el hilo

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

```
public static void generateNumbers() {  
    try {  
        FileWriter writer = new FileWriter("output.txt");  
        while (true) {  
            BigInteger prime =  
                BigInteger.probablePrime(1024, new Random());  
            writer.append(prime.toString());  
            writer.append("\r\n");  
            try {  
                Thread.sleep(1000);  
            } catch (InterruptedException e) {  
                writer.append("Fin fichero");  
                writer.close();  
                return;  
            }  
        }  
    } catch (IOException e) { //catch de la apertura de fichero  
        System.err.println("Exception using file");  
        e.printStackTrace();  
        System.exit(1);  
    }  
}
```

El proceso genera números primos y los guarda en un fichero

Si esperamos 1 seg entre cada generación de un número, se desbloquea el método en cuanto se interrumpe el hilo

Finalización de hilos

GESTIÓN DE HILOS EN JAVA

- Para finalizar un hilo, el propio hilo tiene que estar preparado para ello, **cerrando** recursos y **finalizando** su ejecución
- Tiene que revisar periódicamente si le han interrumpido (**Thread.interrupted()**)
- Si está bloqueado, se eleva una excepción cuando le interrumpen (**InterruptedException**)
- Se pueden usar **ambas técnicas** conjuntamente
- Aunque es habitual usar las **interrupciones** para la **finalización**, se podría usar para otro tipo de notificación

Gestión de hilos en Java

PROGRAMACIÓN CONCURRENT EN JAVA

- **Introducción a la PC en Java**
 - Historia de la Concurrencia en Java
 - Finalización de hilos
 - **Hilos demonios (*daemon*)**
- Exclusión mutua en Java
- Sincronización condicional en Java
- Conclusiones

Hilos demonios (*daemon*)

GESTIÓN DE HILOS EN JAVA

- Un hilo se puede marcar como **demonio** (*daemon*)
- Los hilos demonios **finalizan automáticamente** cuando han finalizado todos los hilos que no son demonios de un programa
- La palabra **demonio** es el nombre que reciben los servicios en máquinas Unix

Hilos demonios (*daemon*)

GESTIÓN DE HILOS EN JAVA

- Un hilo es **demonio** si el hilo que lo crea también lo es
- Los métodos **isDaemon()** y **setDaemon(...)** permiten cambiar esta propiedad de los hilos
- **No se suelen usar mucho** estos hilos porque es bastante habitual que los hilos tengan que realizar tareas de limpieza (cierre de ficheros, conexiones, ...) antes de finalizar

Otras cuestiones sobre hilos en Java

GESTIÓN DE HILOS EN JAVA

- Por completitud, veremos otras cuestiones más accesorias sobre los hilos en Java
 - **Prioridad** de ejecución
 - **Grupos** de hilos con **ThreadGroup**
 - **Métodos auxiliares** sobre hilos

Prioridad de ejecución

OTRAS CUESTIONES SOBRE HILOS EN JAVA

- Los hilos en Java tienen un **prioridad** de ejecución
- La prioridad de un nuevo hilo es la misma que la del hilo que lo creó
- La clase **Thread** tiene los métodos:
 - **setPriority(int p)**: Establece la prioridad. Su valor tiene que estar comprendido entre **Thread.MIN_PRIORITY** (1) y **Thread.MAX_PRIORITY** (10)
 - **int getPriority()**: Devuelve la prioridad del hilo

Prioridad de ejecución

OTRAS CUESTIONES SOBRE HILOS EN JAVA

- Java se pueda ejecutar en **diferentes** sistemas operativos y arquitecturas hardware
- Por esto no se puede asegurar una **semántica concreta** para las prioridades de los hilos
- La prioridad de un hilo es un “**deseo**” del desarrollador que la JVM y el SO respetarán en la medida de sus posibilidades

Prioridad de ejecución

OTRAS CUESTIONES SOBRE HILOS EN JAVA

- Usos habituales de las prioridades

Valores	Uso
10	Gestión de crisis
7-9	Interfaz de usuario
4-6	Entrada / Salida
2-3	Tareas en segundo plano
1	Ejecutar si no hay otra cosa

Grupos de hilos con ThreadGroup

OTRAS CUESTIONES SOBRE HILOS EN JAVA

- Los objetos de la clase **Thread** se pueden agrupar en grupos representados por objetos de la clase **ThreadGroup**
- Un **ThreadGroup** puede tener dentro a otros grupos de hilos, creando una estructura de árbol
- Los **ThreadGroup** no se utilizan mucho. Se usan únicamente para:
 - Limitar la prioridad de los hilos que contienen
 - Gestionar ciertas propiedades de los hilos de forma conjunta

Métodos auxiliares sobre hilos

OTRAS CUESTIONES SOBRE HILOS EN JAVA

- La clase **Thread** proporciona algunos métodos auxiliares
 - **Thread.currentThread()**: Método estático que devuelve el objeto de la clase Thread que representa el hilo que llama al método
 - **getName()**: Devuelve el nombre del hilo
 - **isAlive()**: Indica si el hilo está en ejecución
 - **getState()**: Devuelve el estado del hilo (nuevo, ejecutando, esperando...)
 - Revisar el JavaDoc de la clase **Thread**

Ejercicio 4.1

GESTIÓN DE HILOS EN JAVA

- Se desea implementar en Java un programa con dos hilos, el hilo principal Main y un hilo de Mensajes
- El hilo principal:
 - Crea el hilo de mensajes
 - Espera a que el hilo de mensajes finalice
 - Si no lo hace, cada segundo imprime “Todavía esperando...”
 - Cuando ha pasado un tiempo máximo de 5 segundos, si el hilo de mensajes no ha terminado todavía, dice “Cansado de esperar!”, le interrumpe y espera a que termine
 - Al finalizar el hilo dice “Por fin!”

Ejercicio 4.1

GESTIÓN DE HILOS EN JAVA

- El hilo de mensajes:
 - Dice las siguientes frases cada dos segundos: "La vida es bella", "O no...", "Los pajaritos cantan", "Y molestan..."
 - Si el hilo principal le interrumpe antes de terminar, dice "Se acabó!"
- Cada hilo debe indicar su nombre cada vez que dice algo