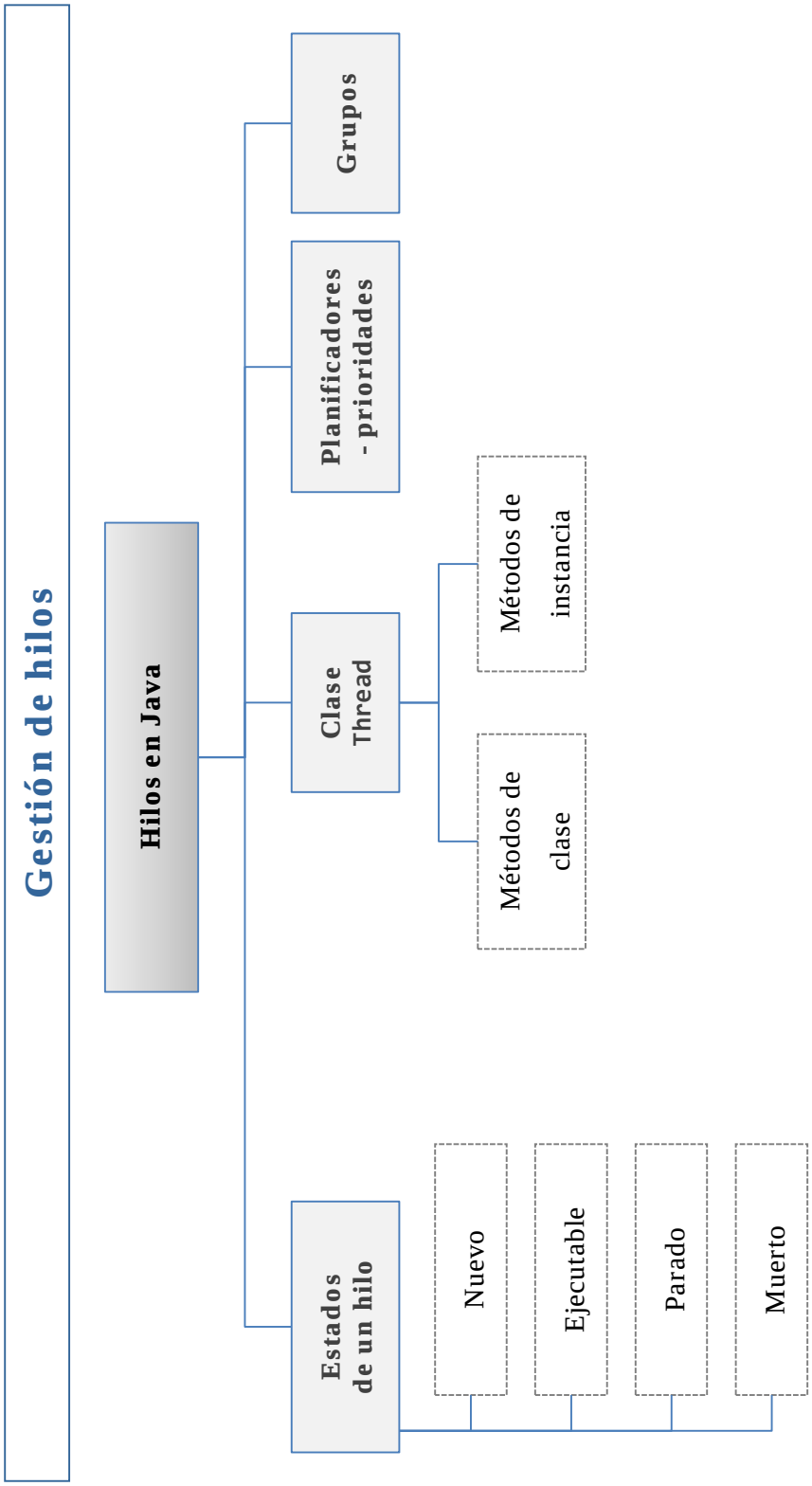
Gestión de hilos

TEMA 8

Esquema



Ideas clave

8.1. ¿Cómo estudiar este tema?

Para estudiar este tema lee el **capítulo 3 (páginas 43-56)** del siguiente libro, disponible en la Biblioteca Virtual de UNIR:

Espinosa, A. R., Argente, E. & Muñoz, F. D. (2013). *Concurrencia y sistemas distribuidos*. Valencia: Editorial de la Universidad Politécnica de Valencia.

Además, deberás leer el **capítulo 6 (páginas 200-209)** del siguiente libro, disponible en el aula virtual en virtud del artículo 32.4 de la Ley de Propiedad Intelectual:

Sznajdleder, P. A. (2013). *Java a Fondo: estudio del lenguaje y desarrollo de aplicaciones* (2ª ed.). Buenos Aires: Alfaomega Grupo Editor.

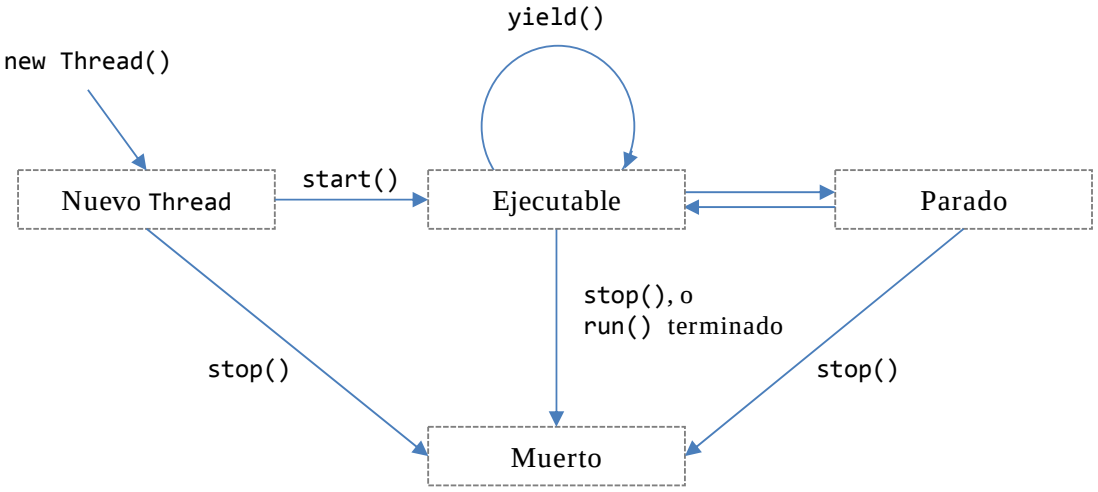
Para comprobar si has comprendido los conceptos puedes realizar el test de autoevaluación del tema.

En este tema vamos a aprender cómo se realiza la gestión de hilos en Java:

- Estados de un hilo.
- Métodos de los hilos en Java.
- Planificación de hilos.
- Creación de grupos de hilos.
- Demonios.

8.2. Ciclo de vida de un hilo

En el siguiente diagrama podemos ver los estados en los que un hilo puede estar durante su vida.



- **Estado «NuevoThread»**: hilo creado e inicializado que no ha invocado al método `start()`.

- **Estado ejecutable**: hilo que ha invocado al método `start()`. Todos los hilos que han invocado al método `start()` se encuentran en estado ejecutable, solo el hilo que posee la CPU se está ejecutando (compartición de recursos del sistema).

- **Estado parado**: hilo listo para ser usado y capaz de volver al estado «Ejecutable» en cualquier momento. Las causas por las que un hilo pasa a este estado son:

  - Llamada al método `suspend()`
  - Llamada al método `sleep( )`
  - Bloqueo por una operación de E/S
  - Llamada al método `wait( )` del hilo

- **Estado muerto**: finalización del hilo. Un hilo puede entrar en este estado por dos causas:

  - Final del método `run( )`
  - Llamada al método `stop()`

8.3. La clase `Thread`

La clase `Thread` encapsula el control necesario para la gestión de hilos.

Métodos de clase

A continuación se detalla algunos de los métodos estáticos de la clase `Thread`:

- `currentThread()`: devuelve el objeto `thread` que se está ejecutando actualmente.
- `yield()`:hace que el hilo que se está ejecutando pase a estado ejecutable, dando la oportunidad de ejecutarse a otros hilos con su misma prioridad.
- `sleep( long )`: duerme al hilo en ejecución durante los milisegundos indicados en el argumento.

Métodos de instancia

A continuación se detallan algunos de los métodos más interesantes de la clase `Thread`:

- `start()`: inicia la ejecución de un hilo. A continuación, invocará al método `run()` del hilo.
- `run()`: es llamado por el metódo `start` y contiene el código a ejecutar por el hilo. Es el único método del interfaz `Runnable`.
- `stop()`: provoca que el hilo se detenga de manera inmediata.
- `suspend()`: suspende la ejecución de un hilo, la cual se reanuda con el método `resume()`.
- `resume()`: revive un hilo suspendido.
- `setName( String )`: rermite establecer el nombre del hilo.
- `getName()`: devuelve el nombre del hilo.
- `isAlive()`: este método devuelve true si el hilo está vivo (se ha invocado `start()` para el hilo, pero no se ha invocado a `stop()`).

8.4. Planificación de hilos

La **planificación** es apropiativa por prioridades. Solo hay apropiación por parte de hilos de mayor prioridad.

Hay diez niveles de prioridad (1-10), cada uno de los cuales viene definido por una constante predefinida:

- `MIN_PRIORITY = 1`
- `NORM_PRIORITY = 5`
- `MAX_PRIORITY = 10`

Por defecto, todos los hilos se crean con prioridad normal (5). Cuando hay dos hilos de igual prioridad decide el sistema.

Métodos que se pueden utilizar para la gestión de prioridades:

- `int getPriority( )`: obtiene la prioridad del hilo.
- `void setPriority(int)`: establece la prioridad el hilo.
- `bool yield()`: permite ceder el control a otros hilos.

8.5. Grupo de hilos

Todo hilo pertenece a un grupo, por defecto es al grupo `main`. La clase `ThreadGroup` permite crear y manipular grupos de hilos.

Constructores de la clase `ThreadGroup`:

```
ThreadGroup(String nombreGrupo)
```

```
ThreadGroup(ThreadGroup gPadre, String nombreGrupo)
```

Constructores de la clase `Thread`
en los que se indica el grupo al que pertenece el hilo:

```
Thread(ThreadGroup grupo, Runnable objRun)
```

```
Thread(ThreadGroup grupo, Runnable objRun,String nombre)
```

Métodos de la clase `Thread`:

- `ThreadGroup getThreadGroup()`: obtiene el grupo al que pertenece un hilo.

8.6. Hilos de tipo demonio

Hilos de tipo demonio

Son hilos especiales y no se tienen en cuenta en finalización de aplicaciones.

Para decir que un hilo es de tipo `daemon` utilizamos el método de instancia

```
setDaemon(true).
```

Para saber si un hilo es de tipo `daemon` utilizamos el método de instancia

```
isDaemon( )
```

Lo + recomendado

No dejes de leer...

Clase `Thread`

Información detallada sobre la clase `Thread`.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
https://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html

Estados de un hilo

Artículo en el que se explica los estados de los hilos.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://jrlq.blogspot.com.es/2013/04/concurrencia-en-java-parte-1.html

+ Información

A fondo

Prioridades de los hilos

Artículo en el que se explica las prioridades de los hilos en Java.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://jrlq.blogspot.com.es/2013/05/concurrencia-en-java-parte-2.html

`Thread`

Artículo en el que se explican los métodos de la clase `Thread` y se muestran ejemplos.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.taringa.net/posts/info/16552529/Hilos-2-Threads-en-java.html

Test

1. La planificación de hilos en Java:

A. Tiene ocho niveles de prioridad.

B. Permite apropiación por parte de un hilo de menor prioridad.

C. Es apropiativa por prioridades.

D. Por defecto los hilos se crean con prioridad 4.

2. El método `yield()`:

A. Detiene un hilo de manera inmediata.

B. Cede el control a otro hilo.

C. Constituye el cuerpo del hilo.

D. Revive un hilo.

3. Un hilo en estado `Nuevo`:

A. Está inicializado y ha llamado al método `run()`.

B. Esta inicializado y ha llamado al método `start()`.

C. Está listo para ser usado.

D. Está inicializado pero no ha llamado al método `start()`.

4. Un hilo puede finalizarse:

A. Al llegar al final del método `run()`.

B. Al llegar al final del método `start()`.

C. Al llamar al método `sleep()`.

D. Al llamar al método `resume()`.

5. Para obtener la prioridad de un hilo se utiliza el método:

A. `isAlive()`

B. `getpriority()`

C. `yield()`

D. `setpriority()`

6. Señale la afirmación correcta sobre grupos de hilos:

A. Cuando se crea un hilo no podemos indicar a qué grupo pertenece.

B. Al crear un grupo de hilos no podemos asignarle un nombre.

C. Un hilo puede no pertenecer a ningún grupo de hilos.

D. Se crean mediante la clase `ThreadGroup`.

7. Para indicar que un hilo es de tipo de demonio se utiliza el método:

A. `isDaemon()`

B. `setDaemon()`

C. `getDaemon()`

D. `Daemon()`

8. Indica cuál de las siguientes afirmaciones es cierta:

A. Un hilo ejecutable no puede pasar a estado dormido o parado.

B. Un hilo pasa a estado muerto cuando se invoca al método `suspend()`.

C. Un hilo no es ejecutable si está muerto.

D. Un hilo en estado nuevo solo puede pasar a estado ejecutable.

9. Indica cuál de las siguientes afirmaciones es falsa:

A. Los hilos pueden ceder el control a hilos con menor prioridad con el método `yiedl()`.

B. Un hilo de más baja prioridad no puede desalojar a hilos de más alta prioridad.

C. Java es un lenguaje multihilado.

D. La aparición de hilos de mayor prioridad puede posponer la ejecución de hilos de menor prioridad.

10. Indica cuál de las siguientes afirmaciones es cierta:

A. Un hilo de más baja prioridad puede desalojar a hilos de más alta prioridad.

B. En Java un hilo ejecutable de más alta prioridad desaloja a los hilos de más baja prioridad.

C. Los hilos pueden ceder el control a hilos con menor prioridad con el método `yiedl()`.

D. La prioridad de un hilo no puede cambiarse.

