INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

Ingeniería en Sistemas Computacionales

TÓPICOS AVANZADOS DE PROGRAMACIÓN

Semestre 4°A

Cesar Ramírez Rodríguez

No. Control: S19070100

Cesar1xd1@gmail.com

Mapa conceptual: Ciclo de vida de hilos e interbloqueo

Salvador Acevedo Sandoval

26 de marzo del 2021

Mapa Conceptual – Ciclo de vida de un hilo

Ciclo de vida de Hilos

Es lo que nos indica el estatus del hilo

-New
-Runnable
-Blocked
-Waitnig
-Time waiting
-Terminated

New

Cuando un hilo es creado

Runnable

Hilo listo para ser ejecutado

Blocked

El hilo no se encuentra en espera sino bloqueado

Waiting

El hilo espera a que otro termine

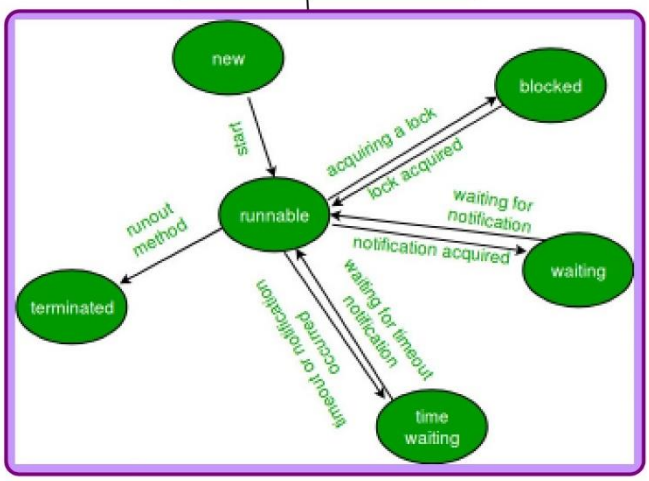
Time Waiting

El hilo esta en espera por un periodo de tiempo especifico

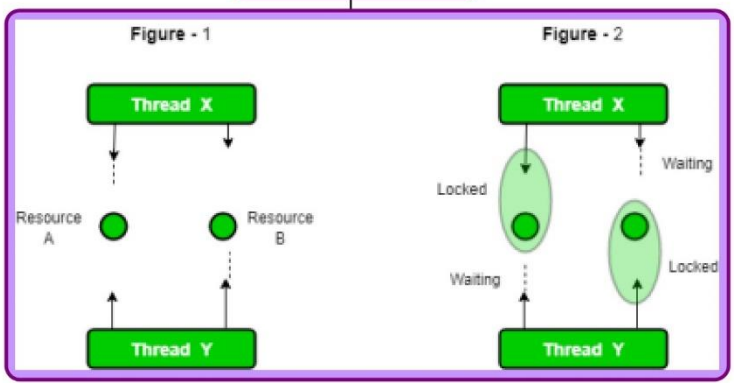
Terminated

El hilo termino su acción.

Relacion de estados



Interbloqueo



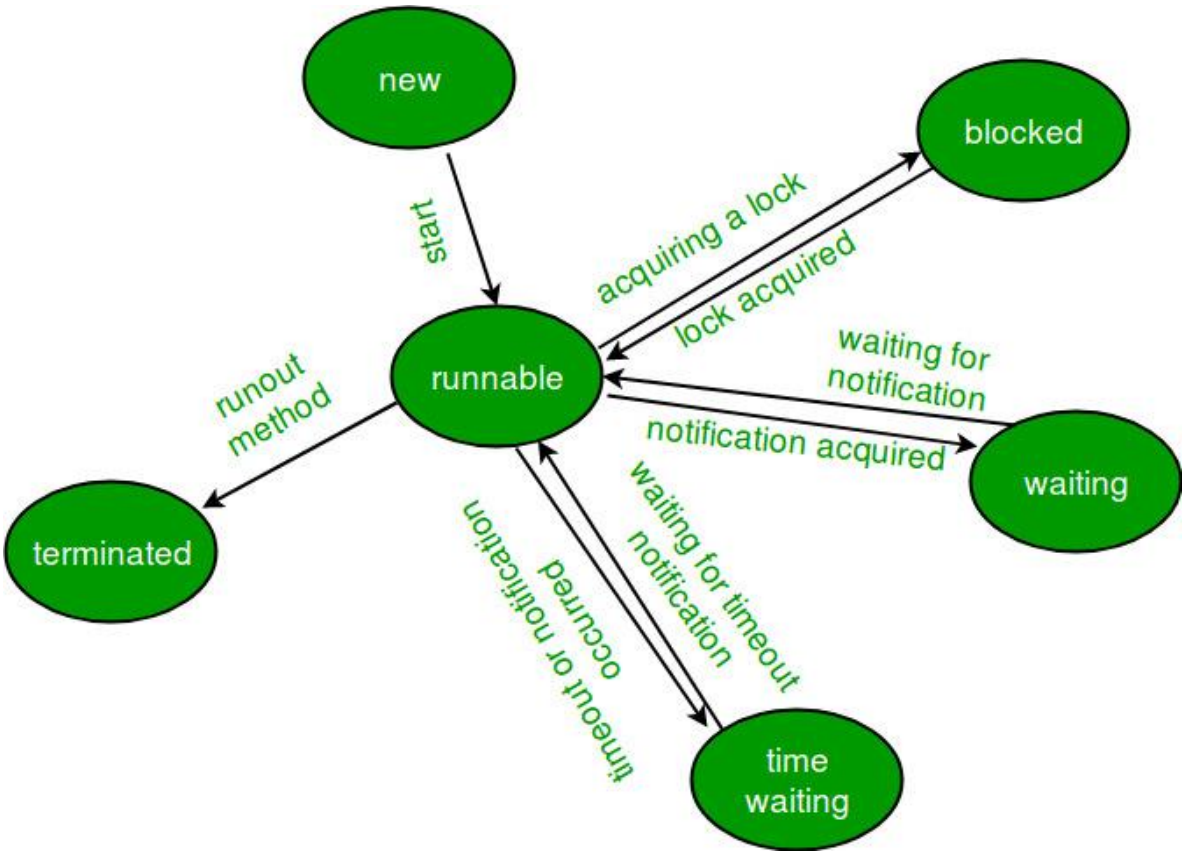
Traducción

CICLO DE VIDA Y ESTADOS DE UN HILO EN JAVA

Un thread en Java a cualquier punto del tiempo existe en cualquiera de los siguientes estados.
Un hilo se encuentra solo en uno de los estados mostrados en cualquier instante:
New

1. Runnable
2. Blocked
3. Waiting
4. Timed Waiting
5. Terminated

El diagrama visto debajo representa varios estados de un thread en cualquier instante del tiempo.



Ciclo de vida de un thread

1.- Nuevo thread: Cuando un nuevo thread es creado, está en un nuevo estado. El hilo aún no ha comenzado a ejecutarse cuando el hilo está en este estado. Cuando un hilo se encuentra en el nuevo estado, su código aún no se ha ejecutado y no ha comenzado a ejecutarse.

2.- Estado ejecutable: Un thread que está listo para ejecutarse se mueve al estado ejecutable. En este estado, un hilo podría estar ejecutándose o podría estar listo para ejecutarse en cualquier momento. Es responsabilidad del programador de subprocesos darle tiempo para que se ejecute. Un programa de subprocesos múltiples asigna una cantidad fija de tiempo a cada subproceso individual. Todos y cada uno de los subprocesos se ejecutan por un corto tiempo y luego se detiene y cede la CPU a otro subproceso, para que otros subprocesos puedan tener la oportunidad de ejecutarse. Cuando esto sucede, todos los subprocesos que están listos para ejecutarse, esperando la CPU y el subproceso que se está ejecutando actualmente, se encuentran en estado ejecutable.

3.- Estado bloqueado/espera: Cuando un thread está temporalmente inactivo, entonces está en uno de los siguientes estados:

- Bloqueado
- En espera

Por ejemplo, cuando un subproceso está esperando que se complete la E/S, se encuentra en el estado bloqueado. Es responsabilidad del programador de subprocesos reactivar y programar un subproceso bloqueado / en espera. Un subproceso en este estado no puede continuar más su ejecución hasta que se mueva al estado ejecutable. Cualquier hilo en estos estados no consume ningún ciclo de CPU. Un thread está en estado bloqueado cuando intenta acceder a una sección protegida de código que actualmente está bloqueada por algún otro subproceso. Cuando la sección protegida está desbloqueada, el programa elige uno de los subprocesos cual está bloqueado para esa sección y lo mueve al estado ejecutable. Mientras que un subproceso está en estado de espera cuando espera otro subproceso con una IV condición. Cuando se cumple esta condición, se notifica al planificador y el subproceso en espera es movido al estado ejecutable.

Si un subproceso que se está ejecutando actualmente se mueve al estado bloqueado/en espera, el programador del subproceso programa la ejecución de

otro subproceso en el estado ejecutable. Es responsabilidad del programador de subprocesos determinar qué subproceso ejecutar.

4.- Espera temporizada: un hilo se encuentra en estado de espera temporizada cuando llama a un método con un parámetro de tiempo de espera. Un hilo permanece en este estado hasta que se completa el tiempo de espera o hasta que se recibe una notificación. Por ejemplo, cuando un hilo llama a dormir o una espera condicional, se mueve a un estado de espera temporizada.

5.- Estado terminado: un subproceso termina debido a cualquiera de las siguientes razones:
-Porque existe normalmente. Esto sucede cuando el programa ha ejecutado completamente el código del hilo.

-Porque ocurrió algún evento erróneo inusual, como una falla de segmentación o una excepción no manejada.

Un hilo que se encuentra en un estado terminado ya no consume ningún ciclo de CPU.
Implementación de estados de threads en Java. En Java, para obtener el estado actual del hilo, use el método Thread.getState() para obtener el estado actual del hilo. Java proporciona la clase java.lang.Thread.State que define las constantes ENUM para el estado de un hilo, como resumen a continuación:

1.- Tipo constante: Nuevo.
Descripción: Estado de thread para un thread que aún no se ha iniciado.

2.- Tipo constante: Ejecutable.
Descripción: Estado de thread para un thread ejecutable. Un subproceso en el estado ejecutable se está ejecutando en la máquina virtual Java, pero puede estar esperando otros recursos del sistema operativo, como el procesador.

3.- Tipo constante: Bloqueado
Descripción: Estado de subproceso para un subproceso bloqueado en espera de un bloqueo de monitor. Un hilo en el estado bloqueado está esperando que un bloqueo de monitor ingrese a un bloque / método sincronizado o vuelva a ingresar a un bloque / método sincronizado después de llamar a Object.wait()

4.- Tipo constante: Esperando
Descripción: Estado de subproceso para un subproceso en espera. Un hilo está en estado de espera debido a que se llama a uno de los siguientes métodos:
- Object.wait with no timeout
- Thread.join with no timeout
- LockSupport.park

Un thread en el estado de espera está esperando para otro thread a realizar una acción en particular

5.- Tipo constante: Espera temporizada
Descripción: estado de subproceso para un subproceso en espera con un tiempo de espera especificado. Un hilo está en el estado de espera temporizado debido a que se llama a uno de los siguientes métodos con un tiempo de espera positivo especificado.

- Thread.sleep
- Object.wait with timeout
- Thread.join with timeout
- LockSupport.parkNanos
- LockSupport.parkUntil

6.- Tipo constante: Terminado Descripción: estado de thread para un thread terminado. El hilo ha completado su ejecución.
Explicación: Cuando se crea un nuevo hilo, el hilo está en el estado NUEVO. Cuando se llama al método start() en un hilo, el programador del hilo lo mueve al estado Ejecutable. Siempre que se llama al método join() en una instancia de thread, el thread actual que ejecuta esa declaración esperará a que este thread se mueva al estado Terminado. Entonces, antes de que se imprima la declaración final en la consola, el programa llama a join() en thread2 haciendo que thread1 espere mientras thread2 completa su ejecución y se mueve al estado Terminado. thread1 pasa al estado de espera porque está esperando que thread2 complete su ejecución, ya que ha llamado a join en thread2.

INTERBLOQUEO EN JAVA MULTITHREADING

La palabra clave synchronized se usa para hacer que la clase o método sea thread-seguro, cual significa que solamente un thread puede tener el bloqueo del método sincronizado y usarlo, otros threads tienen que esperar hasta que se libere el bloqueo y cualquiera de ellos adquiera ese bloqueo.
Es importante para usar si nuestro programa se ejecuta en un entorno de subprocesos múltiples donde dos o más subprocesos se ejecutan simultáneamente. Pero a veces también causa un problema que se llama Deadlock. A continuación, se muestra un ejemplo simple de la condición de interbloqueo.

Figure - 1

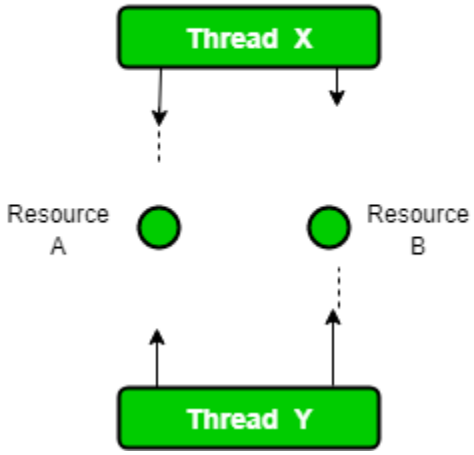
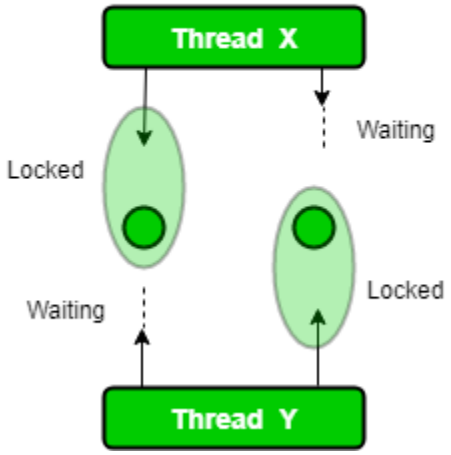


Figure - 2



No se recomienda ejecutar el programa anterior con IDE en línea. Podemos copiar el código fuente y ejecutarlo en nuestra máquina local. Podemos ver que se ejecuta por tiempo indefinido, porque los threads están en una condición de interbloqueo y no permiten que se ejecute el código. Ahora veamos paso a paso lo que está sucediendo allí.

1.- El subproceso t1 se inicia y llama al método test1 tomando el bloqueo de objeto de s1.
2.- El hilo t2 comienza y llama al método test2 tomando el bloqueo de objeto de s2.
3.- t1 imprime test 1-begin y t2 imprime test-2 begin y ambos esperan 1 segundo, de modo que ambos hilos pueden iniciarse si alguno de ellos no lo está.
4.- t1 intenta tomar el bloqueo de objeto de s2 y llamar al método test 2, pero como ya lo ha adquirido t2, espera hasta que se libere. No liberará el bloqueo de s1 hasta que obtenga el bloqueo de s2.
5.- Lo mismo ocurre con t2. Intenta tomar el bloqueo de objeto de s1 y llamar al método test1, pero ya lo ha adquirido t1, por lo que tiene que esperar hasta que t1 libere el bloqueo. t2 tampoco liberará el bloqueo de s2 hasta que se bloquee s1.
6.- Ahora, ambos subprocesos están en estado de espera, esperando que el otro libere los bloqueos. Ahora hay una carrera en torno a la condición de quién soltará el candado primero.
7.- Como ninguno de ellos está listo para liberar el bloqueo, esta es la condición de interbloqueo.
8.- Cuando ejecute este programa, parecerá que la ejecución está en pausa.

Detectar condición de bloqueo muerto II

También podemos detectar interbloqueo ejecutando este programa en cmd. Tenemos que recolectar Thread Dump. El comando para recopilar depende del tipo de sistema operativo. Si usamos Windows y Java 8, el comando es jcmd $ PID Thread.print. Podemos obtener PID ejecutando el comando jps. El volcado de hilo para el programa anterior se encuentra a continuación.
Como podemos ver, se menciona claramente que se encontró 1 punto muerto. Es posible que aparezca el mismo mensaje cuando lo pruebe en su máquina.

Evite la condición de Dead Lock

Podemos evitar la condición de bloqueo al conocer sus posibilidades. Es un proceso muy complejo y no es fácil de detectar. Pero, aun así, si lo intentamos, podemos evitarlo. Existen algunos métodos mediante los cuales podemos evitar esta condición. No podemos eliminar por completo su posibilidad, pero podemos reducirla.
Evite los bloques anidados: esta es la razón principal del bloqueo muerto. El interbloqueo ocurre principalmente cuando damos bloqueos a varios subprocesos. Evite dar bloqueo a varios subprocesos si ya le hemos dado a uno.

Evite bloqueos innecesarios: debemos bloquear solo aquellos miembros que sean necesarios.

Tener el bloqueo activado innecesariamente puede conducir a un bloqueo sin salida.

Usando unión de subprocesos: La condición de bloqueo muerto aparece cuando un subproceso está esperando que otro termine. Si se produce esta condición, podemos utilizar Thread.join con el tiempo máximo que crea que llevará la ejecución.
-Si los subprocesos están esperando que finalicen los demás, entonces la condición se conoce como interbloqueo.
-La condición de interbloqueo es una condición compleja que ocurre solo en el caso de múltiples subprocesos.
-La condición de interbloqueo puede romper nuestro código en tiempo de ejecución y puede destruir la lógica empresarial.
-Debemos evitar esta condición tanto como podamos.

Referencias

GeeksforGeeks. (2019, 8 mayo). *Lifecycle and States of a Thread in Java*. https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/

GeeksforGeeks. (2020, 24 octubre). *Deadlock in Java Multithreading*. https://www.geeksforgeeks.org/deadlock-in-java-multithreading/