



INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ

INGENIERÍA EN SISTEMAS COMPUTACIONALES

4to Semestre

20/03/2020

Actividad 4: Programas en Java (Concurrencia) – Mapa Conceptual

Tema 3: Programación Concurrente (MultiHilos)

Materia: Tópicos Avanzados de Programación.

Nombre del Alumno: Marín Ramírez Mario.

Número de Control: S18070186

Correo electrónico: mariomarin502t@gmail.com

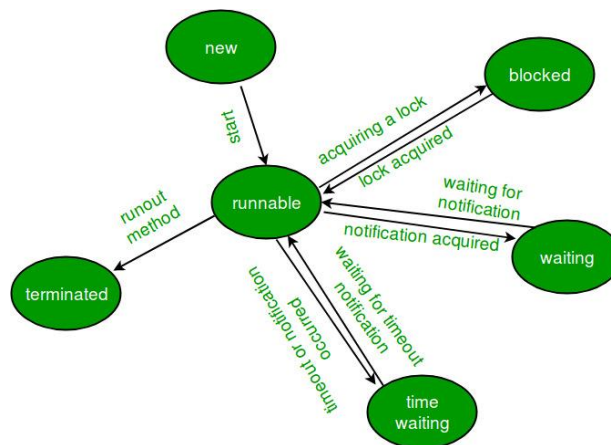
Profesor: I.S.C Acevedo Sandoval Salvador

Ciclo de vida y estados de un hilo en Java

Existe un subproceso en Java en cualquier momento en cualquiera de los siguientes estados. Un hilo se encuentra solo en uno de los estados mostrados en cualquier instante:

1. Nuevo
2. Ejecutable
3. Obstruido
4. Esperando
5. Tiempo de espera
6. Terminado

El diagrama que se muestra a continuación representa varios estados de un hilo en cualquier instante de tiempo.



Ciclo de vida de un hilo

1. **Nuevo hilo:** cuando se crea un nuevo hilo, está en el 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 subproceso que está listo para ejecutarse se mueve al estado ejecutable. En este estado, es posible que se esté ejecutando un subproceso o que esté listo para ejecutarse en cualquier momento. Es responsabilidad del planificador de subprocesos dar al subproceso, tiempo de ejecución. 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 hacen una pausa y renuncian a la CPU a otro subproceso, para que otros subprocesos tengan la oportunidad de ejecutarse. Cuando esto sucede, todos los subprocesos que están listos para ejecutarse, a la espera de la CPU y el subproceso actualmente en ejecución, se encuentran en estado ejecutable.

3. **Estado bloqueado / en espera:** cuando un hilo está temporalmente inactivo, entonces está en uno de los siguientes estados:
- Obstruido
 - Esperando

Por ejemplo, cuando un subproceso está esperando que se complete la E / S, se encuentra en el estado bloqueado. Es responsabilidad del planificador de subprocesos reactivar y programar un subproceso bloqueado / en espera. Un subproceso en este estado no puede continuar su ejecución más hasta que se mueva al estado ejecutable. Cualquier subproceso en estos estados no consume ningún ciclo de CPU.

Un subproceso está en estado bloqueado cuando intenta acceder a una sección protegida de código que está bloqueado actualmente por algún otro subproceso. Cuando la sección protegida está desbloqueada, la programación selecciona uno de los hilos que están bloqueados para esa sección y lo mueve al estado ejecutable. Mientras que un hilo está en el estado de espera cuando espera otro hilo en una condición. Cuando se cumple esta condición, se notifica al planificador y el subproceso en espera se mueve al estado ejecutable.

Si un subproceso actualmente en ejecución se mueve al estado bloqueado / en espera, el programador de subprocesos programa otro subproceso en el estado ejecutable. Es responsabilidad del planificador de subprocesos determinar qué subproceso ejecutar.

4. **Espera programada:** un subproceso se encuentra en estado de espera programada cuando llama a un método con un parámetro de tiempo de espera. Un hilo se encuentra en este estado hasta que se complete el tiempo de espera o hasta que se reciba una notificación. Por ejemplo, cuando un hilo llama a suspensión o una espera condicional, se mueve a un estado de espera temporizado.
5. **Estado terminado:** un subproceso termina debido a cualquiera de los siguientes motivos:
- Porque existe normalmente. Esto sucede cuando el código del hilo ha sido ejecutado completamente por el programa.
 - Debido a que ocurrió algún evento erróneo inusual, como una falla de segmentación o una excepción no controlada.

Un subproceso que se encuentra en un estado terminado ya no consume ningún ciclo de CPU.

Implementación de estados de subprocesos 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 subproceso, como resumen de lo que se da a continuación:

1. **Tipo constante:** nuevo

```
Declaration: public static final Thread.State NEW
```

Descripción: estado de subproceso para un subproceso que aún no se ha iniciado.

2. **Tipo constante:** runnable

```
Declaration: public static final Thread.State RUNNABLE
```

Descripción: estado de subproceso para un subproceso ejecutable. Se está ejecutando un subproceso en el estado ejecutable en la máquina virtual Java, pero puede estar esperando otros recursos del sistema operativo, como el procesador.

3. **Tipo constante:** obstruido

```
Declaration: public static final Thread.State BLOCKED
```

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

4. **Tipo constante:** esperando

```
Declaration: public static final Thread.State WAITING
```

Descripción: estado de subproceso para un subproceso en espera. Estado de subproceso para un subproceso en espera. Un subproceso está en estado de espera debido a llamar a uno de los siguientes métodos:

- Object.wait sin tiempo de espera
- Thread.join sin tiempo de espera
- LockSupport.park

5. **Tipo constante:** tiempo de espera

```
Declaration: public static final Thread.State TIMED_WAITING
```

Descripción: estado de subproceso para un subproceso en espera con un tiempo de espera especificado. Un subproceso está en el estado de espera cronometrado debido a llamar 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:** terminada

Declaration: `public static final Thread.State TERMINATED`

Descripción: estado de subproceso para un subproceso terminado. El hilo ha completado la ejecución.

Explicación: Cuando se crea un nuevo hilo, el hilo está en el estado NUEVO. Cuando se llama al método `start()` en un subproceso, el planificador de subprocesos lo mueve al estado Ejecutable. Siempre que se llame al método `join()` en una instancia de subproceso, el subproceso actual que ejecuta esa declaración esperará a que este subproceso se mueva al estado Terminado. Entonces, antes de que la declaración final se imprima en la consola, el programa llama a `join()` en `thread2` haciendo que el `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`.

Deadlock en Multithreading Java:

La palabra clave `synchronized` se usa para hacer que la clase o método sea seguro para subprocesos, lo que significa que solo un subproceso puede tener un bloqueo del método sincronizado y usarlo, otros subprocesos tienen que esperar hasta que se libere el bloqueo y cualquiera de ellos adquiere ese bloqueo.

Es importante usarlo si nuestro programa se ejecuta en un entorno multiproceso donde dos o más subprocesos se ejecutan simultáneamente. Pero a veces también causa un problema que se llama Deadlock.

Figure - 1

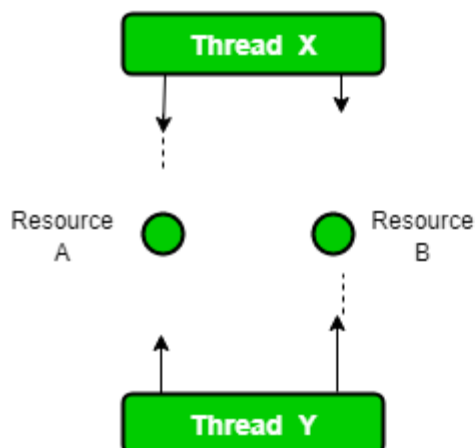
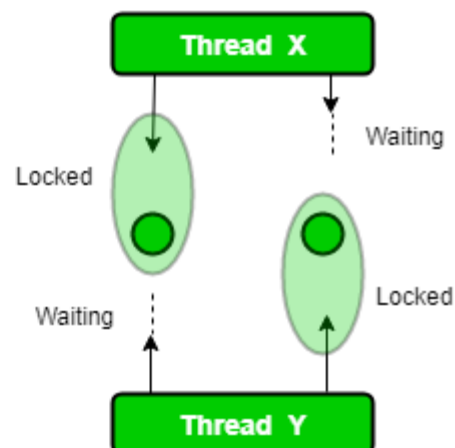


Figure - 2



Evite la condición de bloqueo muerto

Podemos evitar la condición de bloqueo muerto al conocer sus posibilidades. Es un proceso muy complejo y no es fácil de atrapar. Pero aún si lo intentamos, podemos evitar esto. Hay algunos métodos por los cuales podemos evitar esta condición. No podemos eliminar por completo su posibilidad, pero podemos reducir.

- Evite las cerraduras anidadas: esta es la razón principal de la cerradura muerta. DeadLock ocurre principalmente cuando le damos bloqueos a múltiples hilos. Evite bloquear a varios hilos si ya le hemos dado a uno.
- Evite bloqueos innecesarios: deberíamos bloquear solo aquellos miembros que sean necesarios. Tener el bloqueo innecesariamente puede conducir a un bloqueo muerto.
- Uso de unión de hilo: la condición de bloqueo muerto aparece cuando un hilo espera que otro termine. Si se produce esta condición, podemos usar Thread.join con el tiempo máximo que cree que llevará la ejecución.

Puntos importantes:

- Si los hilos esperan el uno al otro para terminar, entonces la condición se conoce como Deadlock.
- La condición de punto muerto es una condición compleja que ocurre solo en el caso de múltiples hilos.
- La condición de punto muerto puede romper nuestro código en tiempo de ejecución y puede destruir la lógica empresarial.
- Deberíamos evitar esta condición tanto como podamos.

