Instituto Tecnológico Superior de Jerez.

Jerez de García Salinas a 15 de Marzo del 2019.

Cristofer Casas Murillo.

cristofer32513@gmail.com

S17070157.

INGENIERÍA EN SISTEMAS COMPUTACIONALES.

Tópicos Avanzados de Programación.

4to. SEMESTRE.

Mapa Conceptual (Tiempo de Vida y Estado de Hilos).

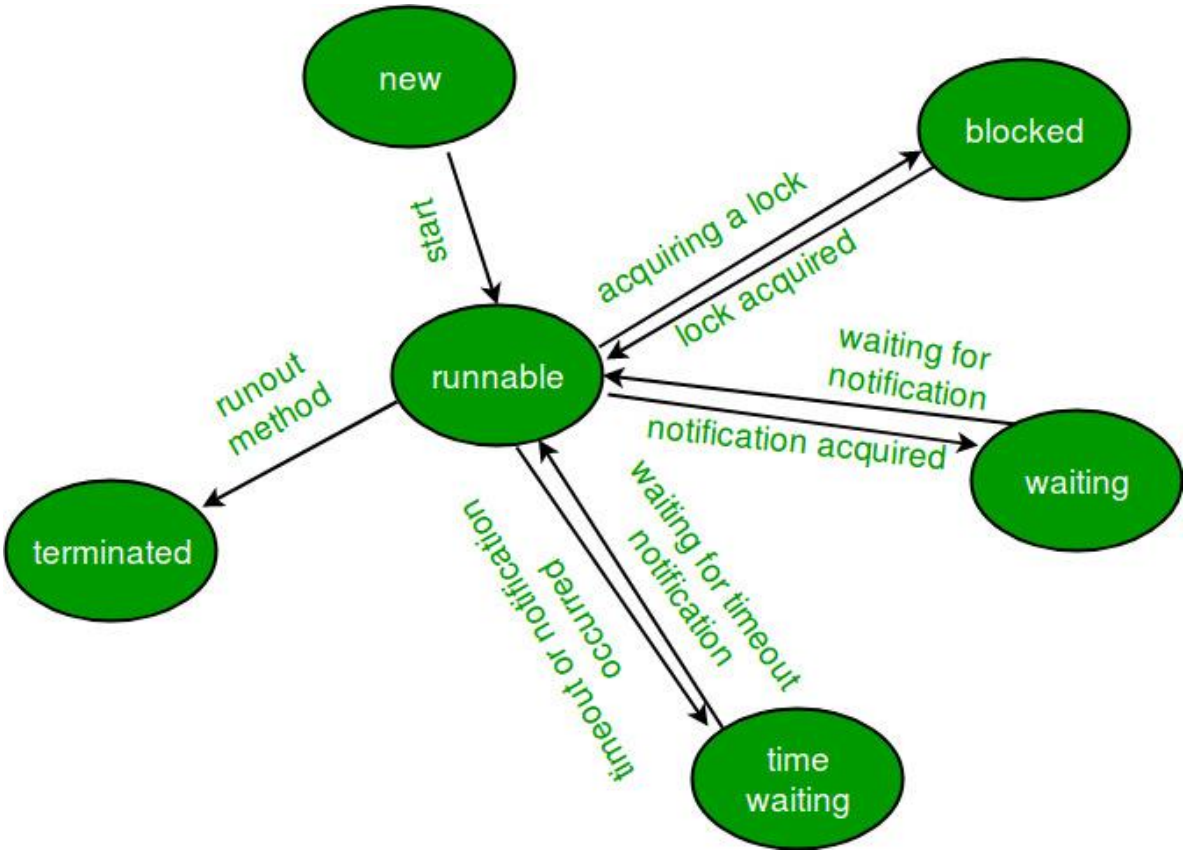
ISC. Salvador Acevedo Sandoval.

Ciclo de Vida y Estados de Un Hilo en Java.

En cualquier momento, existe un subproceso en Java al que se le asigna cualquiera de los siguientes estados. Un hilo se encuentra solo en uno de los estados mostrados en cualquier instante:

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

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



Ciclo de vida de un hilo.

1. **New Thread:** Cuando se crea un nuevo hilo, se encuentra en este estado.

   El subproceso aún no ha comenzado a ejecutarse cuando el subproceso está en este estado. Cuando un hilo se encuentra en este estado, su código aún no se ha ejecutado y por tanto no ha comenzado a ejecutarse.

2. **Runnable State:** Un hilo que está listo para ejecutarse se mueve a este estado.

   En este estado, un subproceso podría estar ejecutándose o podría estar listo para ejecutarse en cualquier momento. Es responsabilidad del programador de subprocesos dar tiempo para que se ejecute el subproceso.

   Un programa con subprocesos múltiples asigna una cantidad de tiempo fija a cada subproceso individual. Todos y cada uno de los subprocesos se ejecuta durante un breve periodo de tiempo y luego se detienen y entregan la CPU a otro subproceso, de modo que otros subprocesos puedan tener la oportunidad de ejecutarse. Cuando esto sucede, todos los subprocesos que están listos para ejecutarse, esperan al CPU y el subproceso que se esté ejecutando en el momento, se encuentran en Runnable State.

3. **Blocked / Waiting state:** Cuando un hilo está temporalmente inactivo, se encuentra en uno de los siguientes estados:

   - **Blocked state.**
   - **Waiting state.**

   Por ejemplo, cuando un hilo está esperando que se complete la E / S, se encuentra en Blocked state. Es responsabilidad del programador de hilos reactivar y programar un hilo Blocked / Waiting. Un hilo en este estado no puede continuar su ejecución hasta que se mueva a Runnable State. Cualquier hilo en estos estados no consume ningún ciclo de CPU.

   Un subproceso está en Blocked state cuando intenta acceder a una sección protegida de código que actualmente está bloqueada por otro subproceso. Cuando la sección protegida

está desbloqueada, la programación selecciona uno de los hilos que está bloqueado para esa sección y lo mueve a Runnable State.

Considerando que, un subproceso está en estado de espera cuando espera a otro subproceso en una condición. Cuando se cumple esta condición, se notifica al programador y el hilo en espera se mueve a Runnable State.

Si un subproceso que se está ejecutando actualmente se mueve Blocked / Waiting state, el programador de subprocesos programará otro subproceso en el Runnable State. Es responsabilidad del programador de hilos determinar qué hilo ejecutar.

4. **Timed Waiting:** Un hilo se encuentra en Timed Waiting cuando llama a un método con un parámetro de tiempo de espera. Un hilo se encuentra en este estado hasta que se completa el tiempo de espera o hasta que se recibe una notificación. Por ejemplo, cuando un subproceso llama a suspensión o espera condicional, se mueve al estado de espera temporizada.

5. **Terminated State:** Un subproceso termina debido a cualquiera de los siguientes motivos:

   Porque existe normalmente. Esto sucede cuando el código del hilo ha sido ejecutado por completo por el programa.

   Porque se produjo un evento erróneo inusual, como un error de segmentación o una excepción no controlada.

   Un subproceso que se encuentra en Terminated State ya no consume ningún ciclo de CPU**.**

Implementando estados de hilos en Java.

En Java, para obtener el estado actual del hilo, use el método **Thread.getState().** Java proporciona la clase **java.lang.Thread.State** que define las constantes ENUM para el estado de un hilo, como resumen de lo siguiente:

1. **Tipo constante:** New.

   Declaración: public static final Thread.State NEW

   Descripción: Estado del hilo para un hilo que aún no se ha iniciado.

2. **Tipo de constante:** Runnable.

   Declaración: public static final Thread.State RUNNABLE

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

3. **Tipo de constante:** Blocked.

   Declaración: public static final Thread.State BLOCKED

   Descripción: Estado del hilo para un hilo bloqueado esperando un bloqueo del monitor. Un subproceso en el estado Blocked está esperando a 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:** Waiting.

   Declaración: public static final Thread.State WAITING

   Descripción: Estado del hilo para un hilo en espera. Un subproceso está en estado Waiting debido a que se llama a uno de los siguientes métodos:

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

   Un hilo en el estado Waiting está esperando a que otro hilo realice una acción en particular.

5. **Tipo de constante:** Timed Waiting.

   Declaración: public static final Thread.State TIMED_WAITING

Descripción: Estado del hilo para un hilo en espera con un tiempo de espera especificado. Un subproceso está en el estado de espera programada 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 de constante:** Terminated.

   Declaración: public static final Thread.State TERMINATED

   Descripción: Estado del hilo para un hilo terminado. El hilo ha finalizado su ejecución.

Interbloqueo en multiproceso de Java.

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

Es importante utilizarlo si nuestro programa se está ejecutando 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 Deadlock.

Figure - 1

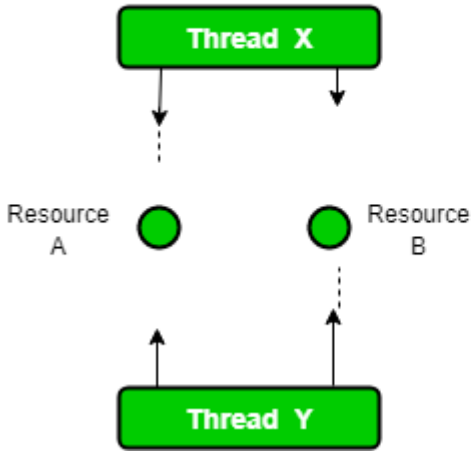
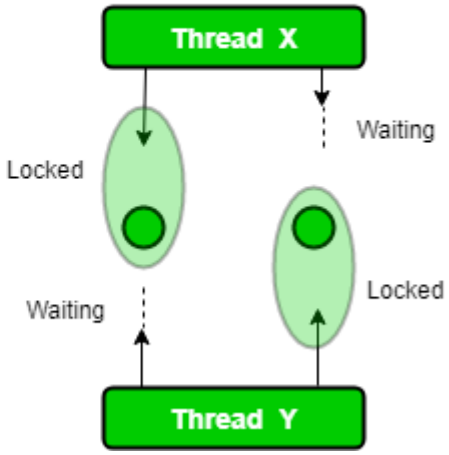


Figure - 2



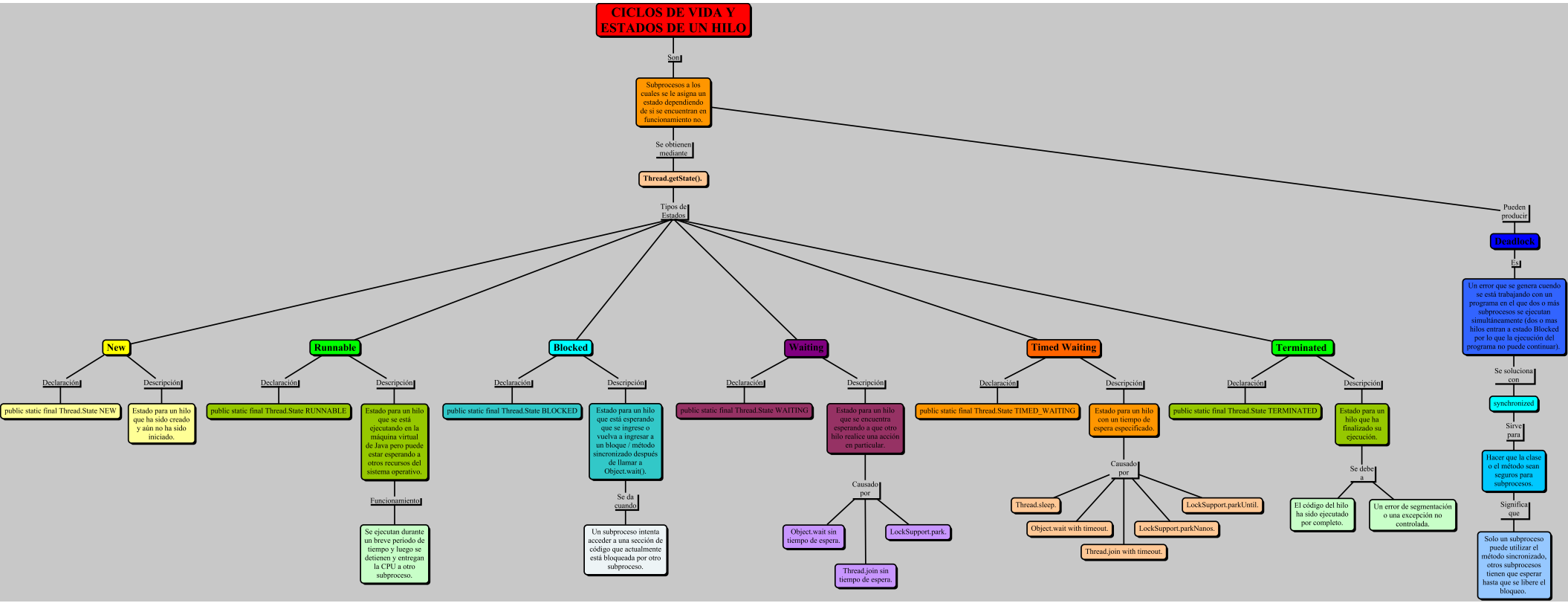
Evitar la condición de Deadlock.

Podemos evitar la condición de Deadlock conociendo sus posibilidades. Es un proceso muy complejo y no es fácil de atrapar. Pero aún si lo intentamos, podemos evitar esto. Hay algunos métodos con los cuales podemos evitar esta condición, sin embargo, no podemos eliminar por completo su posibilidad, pero podemos reducir la probabilidad de que ocurra.

- **Evite los bloqueos anidados:** Esta es la razón principal del Deadlock. DeadLock ocurre principalmente cuando damos bloqueos a múltiples hilos. Evita dar bloqueo a múltiples hilos si ya hemos dado a uno.
- **Evite bloqueos innecesarios:** Deberíamos tener bloqueo solo a los miembros que se requieren. Tener un bloqueo innecesario puede provocar un Deadlock.
- **Usando unión de hilo:** La condición de Deadlock aparece cuando un hilo está esperando a que termine otro hilo. Si se produce esta condición, podemos usar Thread.join con el tiempo máximo que crea que durará la ejecución.

Puntos importantes:

- Si los hilos están esperando el uno al otro para finalizar, la condición se conoce como Deadlock.
- La condición de Deadlock es una condición compleja que se produce solo en el caso de varios 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.


CICLOS DE VIDA Y ESTADOS DE UN HILO
Son
Subprocesos a los cuales se le asigna un estado dependiendo de si se encuentran en funcionamiento no.
Se obtienen mediante
Thread.getState().
Tipos de Estados
Pueden producir
Deadlock
Es
Un error que se genera cuendo se está trabajando con un programa en el que dos o más subprocesos se ejecutan simultáneamente (dos o mas hilos entran a estado Blocked por lo que la ejecución del programa no puede continuar).
Se soluciona con
synchronized
Sirve para
Hacer que la clase o el método sean seguros para subprocesos.
Significa que
Solo un subproceso puede utilizar el método sincronizado, otros subprocesos tienen que esperar hasta que se libere el bloqueo.
New
Declaración
public static final Thread.State NEW
Descripción
Estado para un hilo que ha sido creado y aún no ha sido iniciado.
Runnable
Declaración
public static final Thread.State RUNNABLE
Descripción
Estado para un hilo que se está ejecutando en la máquina virtual de Java pero puede estar esperando a otros recursos del sistema operativo.
Funcionamiento
Se ejecutan durante un breve periodo de tiempo y luego se detienen y entregan la CPU a otro subproceso.
Blocked
Declaración
public static final Thread.State BLOCKED
Descripción
Estado para un hilo que está esperando que se ingrese o vuelva a ingresar a un bloque / método sincronizado después de llamar a Object.wait().
Se da cuando
Un subproceso intenta acceder a una sección de código que actualmente está bloqueada por otro subproceso.
Waiting
Declaración
public static final Thread.State WAITING
Descripción
Estado para un hilo que se encuentra esperando a que otro hilo realice una acción en particular.
Causado por
Object.wait sin tiempo de espera.
LockSupport.park.
Thread.join sin tiempo de espera.
Timed Waiting
Declaración
public static final Thread.State TIMED_WAITING
Descripción
Estado para un hilo con un tiempo de espera especificado.
Causado por
Thread.sleep.
Object.wait with timeout.
Thread.join with timeout.
LockSupport.parkNanos.
LockSupport.parkUntil.
Terminated
Declaración
public static final Thread.State TERMINATED
Descripción
Estado para un hilo que ha finalizado su ejecución.
Se debe a
El código del hilo ha sido ejecutado por completo.
Un error de segmentación o una excepción no controlada.

Referencias.

- Nitiraj Singh Rathore. (NE). Lifecycle and States of a Thread in Java. 2019, de geeksforgeeks.org Sitio web: https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/

- NE. (NE). Deadlock in Java Multithreading. 2019, de geeksforgeeks.org Sitio web: https://www.geeksforgeeks.org/deadlock-in-java-multithreading/