Instituto Tecnológico Superior de Jerez

Jerez, Zacatecas

20 de marzo de 2020

Felipe de Jesús Márquez Olague

flype360360@gmail.com

S18070182

Ingeniería en Sistemas Computacionales

Cuarto Semestre

Tópicos avanzados de programación

Mapa conceptual

M.T.I. Salvador Acevedo Sandoval

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:

Nuevo

Ejecutable

Obstruido

Esperando

Tiempo de espera

Terminado

Ciclo de vida de un hilo

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.

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.

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.

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.

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.

Punto muerto en Java Multithreading

La palabra clave sincronizada 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. A continuación se muestra un ejemplo simple de la condición de punto muerto.

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. Dead Lock 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.

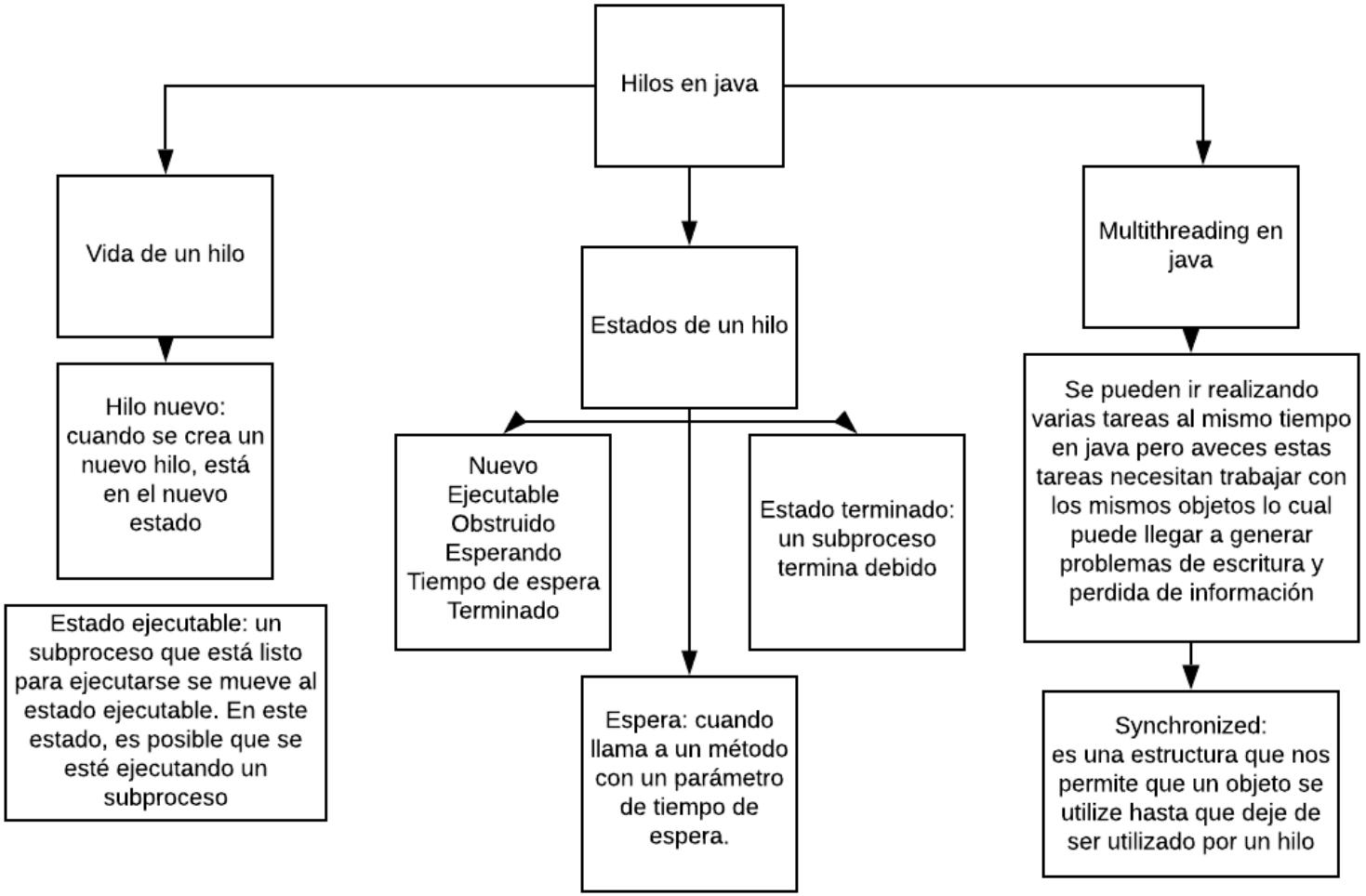
Puntos importantes :

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.


Hilos en java
Vida de un hilo
Hilo nuevo: cuando se crea un nuevo hilo, está en el nuevo estado
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
Estados de un hilo
Nuevo
Ejecutable
Obstruido
Esperando
Tiempo de espera
Terminado
Estado terminado: un subproceso termina debido
Espera: cuando llama a un método con un parámetro de tiempo de espera.
Multithreading en java
Se pueden ir realizando varias tareas al mismo tiempo en java pero aveces estas tareas necesitan trabajar con los mismos objetos lo cual puede llegar a generar problemas de escritura y perdida de información
Synchronized: es una estructura que nos permite que un objeto se utilize hasta que deje de ser utilizado por un hilo

Bibliografía

GueeksForGueeks. (20 de 03 de 2020). *GueeksForGueeks*. Obtenido de GueeksForGueeks: https://www.geeksforgeeks.org/lifecycle-and-states-of-a-thread-in-java/

GueeksForGueeks. (20 de 03 de 2020). *GueeksForGueeks*. Obtenido de GueeksForGueeks: https://www.geeksforgeeks.org/deadlock-in-java-multithreading/