



INSTITUTO TECNOLÓGICO SUPERIOR DE JEREZ



Libramiento Fresnillo-Tepetongo, Fracc. Los Cardos, 99863 Jerez de  
García Salinas, Zac. Fecha de entrega: 01 de marzo del 2024

Alumno: Emmanuel Rogelio Robles Dorado

Numero de control: 22070034

Correo electrónico:

[emmanuel.rd@tecjerez.edu.mx](mailto:emmanuel.rd@tecjerez.edu.mx)

Carrera: Ingeniería en sistemas computacionales Materia:

Fundamentos de bases de datos Semestre: 4to

Actividad: Investigación de AI

Docente: ISC Salvador Acevedo Sandoval

## Ciclo de vida y estados de un hilo en java

En cualquier momento de la ejecución, un hilo en java existe sólo en uno de los siguientes estados:

New State / nuevo  
Runnable State / ejecutable  
Blocked State / bloqueado  
Waiting State / esperando  
Timed Waiting State / esperando por tiempo  
Terminated State / terminado

Ciclo de vida de un hilo

Hay varios estados que un hilo puede adoptar durante su ciclo de vida como los mencionados a continuación:

**Nuevo hilo:** cuando se crea un hilo, éste se encuentra en el estado nuevo, pues todavía no ha comenzado su ejecución.

**Ejecutable:** un hilo está listo para correr en cualquier instante o puede que ya se esté ejecutando se mueve al estado Ejecutable, el manejador de hilos debe asignar tiempo al hilo para ejecutarse.

Un programa multihilo asigna tiempo fijo a cada hilo por separado, los cuales corren por poco tiempo antes de ceder el CPU a otro hilo. Cuando esto pasa tanto el hilo en ejecución, como todos los hilos listos para correr y esperando al CPU se mueven al estado Ejecutable

**Bloqueado:** el estado del hilo será bloqueado cuando intenta obtener un lock ocupado por otro hilo, es decir, trata de trabajar con datos que ya fueron reclamados antes por otro hilo.

**En espera:** el hilo entrará en espera cuando llame al metodo wait() o join(), y solo será Ejecutable de nuevo cuando otro hilo le notifique o el hilo al que espera termine.

**En espera por tiempo:** un hilo yace esperando por tiempo cuando llama a un método con un parámetro de tiempo. El hilo esperará hasta que se complete su tiempo de espera o se le notifique. Sucede para métodos como sleep() y wait condicional.

**Terminado:** El estado de un hilo se vuelve Terminado debido a alguna de las siguientes razones:

Su proceso concluyó normalmente.

Sufrió un error durante su ejecución, como un error de segmentación o una excepción.

Implementando estados de hilo en java

En java, el método Thread.getState() permite obtener el estado del hilo actual. Java provee la clase Thread.State del paquete java.lang, la cual define constantes enum para los estados de un hilo

1. Nuevo

Estado de hilo para un hilo que aún no se ha ejecutado

`public static final Thread.State NEW`

## 2. Ejecutable:

Estado de hilo para un hilo ejecutable. Un hilo ejecutable se ejecuta en la máquina virtual de java pero puede que esté esperando recursos del SO, como un procesador

A thread in the runnable state is executing in the Java virtual machine but it may be waiting for other resources from the operating system such as a processor.

`public static final Thread.State RUNNABLE`

## 3. Bloqueado

Estado de hilo para un hilo esperando a un monitor lock. Un hilo bloqueado espera a un monitor lock para llamar métodos sincronizados (synchronized) o volver a llamarlos después de que llamara `Object.wait()`.

`public static final Thread.State BLOCKED`

## 4. En espera

Estado de hilo para un hilo en espera. Un hilo espera al llamar a uno de los siguientes métodos:

*`Object.wait sin temporizador`*

*`Thread.join sin temporizador`*

*`LockSupport.park`*

`public static final Thread.State WAITING`

## 5. Espera por tiempo

Estado de hilo para un hilo esperando un tiempo fijo. Un hilo espera un tiempo fijo al llamar a alguno de los siguientes métodos con un tiempo positivo de espera especificado:

*`Thread.sleep`*

*`Object.wait con temporizador`*

*`Thread.join con temporizador`*

*`LockSupport.parkNanos`*

*`LockSupport.parkUntil`*

`public static final Thread.State TIMED_WAITING`

## 6. Terminado

Estado de hilo para un hilo terminado. El hilo terminó su ejecución

`public static final Thread.State TERMINATED`

Deadlock en programación multihilo en Java

Una situación Deadlock en Java ocurre cuando se tienen varios hilos interactuando con

recursos compartidos a través de métodos `synchronized`, cuando los hilos permanecen en espera para acceder a un método pero jamás reciben la notificación de que deben ejecutarse nuevamente, por ejemplo si un hilo ha terminado de utilizar el método sincronizado y no llama a `notify()` o `notifyAll()`.

Igualmente, cuando hay un hilo 1 con el monitor lock de un método requerido por otro hilo 2 bloqueado, y éste intenta llamar a un método ocupado por el hilo 2, entonces ambos hilos se bloquean, esperan a que el otro libere el método que necesitan y se llega a un punto muerto.

Detectar Deadlocks es difícil, sin embargo hay recomendaciones que reducen su riesgo de aparición:

Evitar locks anidados, métodos sincronizados que llamen a otros métodos sincronizados y puedan ocupar múltiples locks para un solo hilo.

Evitar locks innecesarios: métodos que no necesiten un acceso organizado no necesitan ser Sincronizados.

usar `Thread.join`: utilizar `Thread.join` en un hilo con un parámetro de tiempo límite puede sacarlo del estado de espera infinito si sufre este problema.

