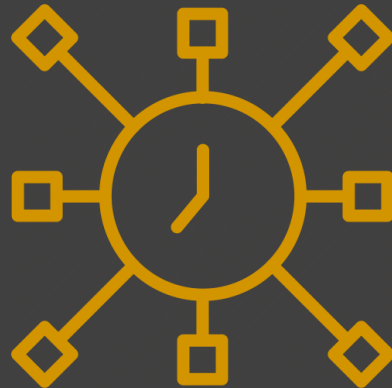


Hilos (Thread)



¿Qué son los hilos?

Objetos que dan la capacidad de hacer más de una tarea al mismo tiempo!



Características



La Máquina Virtual Java (JVM) es un sistema multi-thread capaz de ejecutar varias tareas (o subprogramas) simultáneamente

Java soporta Thread con algunas clases e interfaces y con métodos específicos en la clase Object

La JVM gestiona todos los detalles, asignación de tiempos de ejecución, prioridades, de forma similar a como gestiona un Sistema Operativo

Ejemplos de Thread

```
class CalculoThread extends Thread {  
    @Override  
    public void run() {  
        // hacer algo ...  
    }  
}  
  
class EjemploThread {  
    public static void main(String[] args) {  
        CalculoThread cal = new CalculoThread();  
        cal.start();  
    }  
}
```

Ejemplos de Thread

```
class Tarea implements Runnable {  
    @Override  
    public void run() {  
        // hacer algo ...  
    }  
}  
  
class EjemploThread {  
    public static void main(String[] args) {  
        Thread t = new Thread(new Tarea());  
        t.start();  
    }  
}
```

Ciclo de vida de un Thread: NEW

- Un hilo NEW es uno que se ha creado pero que aún no se ha iniciado con el método `start()`:

```
Runnable runnable = new Tarea();  
Thread t = new Thread(runnable);  
System.out.println(t.getState()); // NEW
```

Ciclo de vida de un Thread: **RUNNABLE**

- Un hilo **RUNNABLE** es uno que se ha creado e iniciado con el `start()`:

```
Runnable runnable = new Tarea();  
Thread t = new Thread(runnable);  
t.start();  
System.out.println(t.getState()); // RUNNABLE
```

Ciclo de vida: **BLOCKED**

- Un hilo está en estado BLOCKED cuando actualmente no es elegible para ejecutarse.
- Entra en este estado cuando está esperando un bloqueo del monitor e intenta acceder a una sección de código que está bloqueada por algún otro hilo en un método sincronizado.

```
public static void main(String[] args) throws InterruptedException {  
    Runnable runnable = ()-> recurso();  
  
    Thread t1 = new Thread(runnable);  
    Thread t2 = new Thread(runnable);  
    t1.start();  
    t2.start();  
    Thread.sleep(1000);  
    System.out.println(t2.getState()); // BLOCKED  
}  
  
public static synchronized void recurso() {  
    // realizando algun proceso compartido entre hilos  
    while(true) {}  
}
```


Ciclo de vida: **WAITING**

- Un hilo está en estado **WAITING** cuando está esperando que otro hilo realice una acción en particular
- Un hilo puede entrar en este estado llamando a cualquiera de los dos métodos `wait()` y `join()`.

```
public static void main(String[] args) {  
    Thread t1 = Thread.currentThread();  
    Thread t2 = new Thread() -> {  
        // realizando alguna tarea costosa  
        try {  
            Thread.sleep(5000);  
        } catch (InterruptedException e) { ... }  
  
        System.out.println(t1.getState()); // WAITING  
    }  
  
    t2.start();  
    t2.join();  
}
```

Ciclo de vida: **TERMINATED**

- Este es el estado de un hilo muerto. Está en el estado **TERMINATED** cuando ha finalizado la ejecución o se terminó de forma anormal.
- También podemos usar el método `isAlive()` para determinar si el hilo está vivo o no.

```
public static void main(String[] args) throws InterruptedException{
    Thread t1 = new Thread(() -> {
        // realizando alguna tarea rápida
    });

    t1.start();
    Thread.sleep(1000);

    System.out.println(t1.getState()); // TERMINATED
    System.out.println(t1.isAlive());  // false
}
```

Métodos wait(), notify() y notifyAll()

- La clase Object tiene tres métodos que permiten que los hilos se sincronicen y comuniquen sobre el estado bloqueado de un recurso.
- wait(): libera el bloqueo para que otros hilos tengan la oportunidad de acceder a un recurso compartido (método sincronizado) y queda esperando indefinidamente hasta que otro hilo invoca notify() o notifyAll().
- notify() y notifyAll() se usa para despertar los hilos que están esperando un acceso a un recurso compartido (monitor).

Método sleep()

- Thread.sleep () envía el hilo actual al estado "TIMED_WAITING" durante algún tiempo.
- Permanece dormido hasta que el tiempo expire o se llame al método interrupt()

```
public static void main(String[] args) throws InterruptedException{  
    Thread t1 = new Thread(() -> {  
        // realizando alguna tarea costosa  
        try {  
            Thread.sleep(5000);  
        } catch (InterruptedException e) { ... }    });  
  
    t1.start();  
    Thread.sleep(1000);  
    System.out.println(t1.getState()); // TIMED_WAITING  
}
```