

# APUNTES TEMA 2 DE PROCESOS

Juan José Rojano Doncel

## Programación multihilo

### ¿Qué es un hilo con tus propias palabras Juan José?

Un hilo es una secuencia de código que se ejecuta de manera paralela juntos a otros hilos.

### ¿Porqué o cómo ejecutarías un hilo?

Pues yo creo que cuando necesitamos ejecutar varias tareas al mismo tiempo y no queremos que hagan cola, no queremos que la secuencia uno termine y entonces, la siguiente comienza, podemos añadir que el propio hilo puede dar a lugar a otros.

### ¿Qué características y problemas tienen?

Bueno pues resulta que los hilos tienden a competir por los recursos ejemplo:

```
Int a=2;
```

```
Función sincronizada {Int b=3;}
```

Si un hilo trata de acceder a la variable “a” sumando +1, pero otro hilo intenta restar -1, tenemos un serio problema y es que al tener esa característica tan peculiar, al intentar acceder un hilo sobrescribe el trabajo de otro.

Para solucionar esto, debemos hacer el recurso esté sincronizado, de esta manera siguen un orden y evitamos errores de persistencia de datos como pasaría con el caso de la variable “b”.

## Implementación de hilos

### ¿Cómo podemos entonces crear esos hilos?

Pues tenemos dos opciones, podemos extender un objeto de la clase **Thread**, o mi favorita, implementar un interfaz Runnable puesto que esta opción me resulta mucho más clara.

### Antes de entrar en detalle en ambos métodos de crear hilos, ¿Cómo podemos ejecutarlos y en que se diferencian?

Es posible utilizar 2 métodos claves run() y start() **PERO OJO** el método start() es el más importante puesto que run(), rompe con la idea de lo que trata de hacer un hilo, provoca que los hilos se ejecuten de manera **secuencial y no paralela**.

### Como funciona o que características tiene extender de Thread

Para ello debemos crear una clase para el hilo que extienda de thread y debemo sobrescribir la función run().

### Como funciona o que características tiene crear la interfaz

La clase implementa Runnable, permite heredar de otra clase.

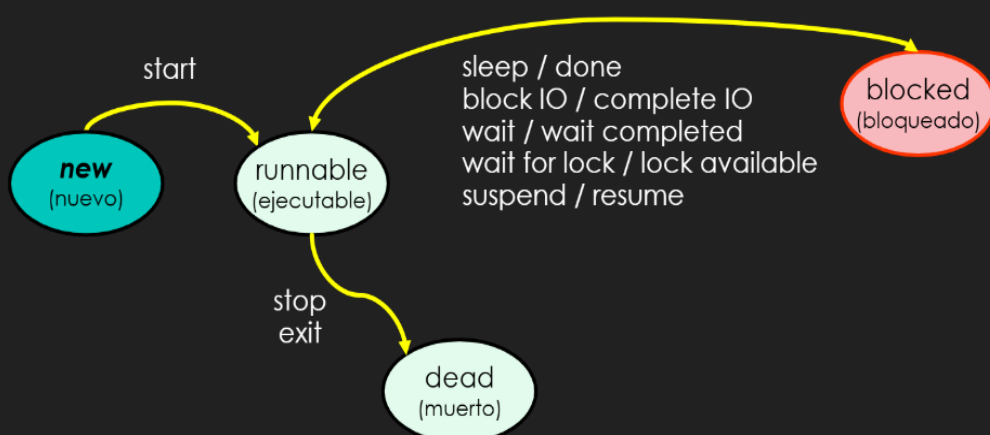
## Estados del hilo

### ¿Qué estados tiene un hilo?

Pues un hilo tiene los siguientes estados posibles (Según damos en clase):

- Nuevo/new: Justo al crear el hilo y nada más terminar el constructor de construirlo, el hilo obtiene este estado identificándolo por así decirlo como hilo recién creado.
- Ejecutable/runnable: Tras invocar start() **NUNCA USAR RUN** el hilo obtiene este estado, es una manera de decir que se está ejecutando, podemos añadir que llama al método stop().
- Muerto/dead: Finaliza el método start() **NUNCA USAR RUN** es una manera de decir que el hilo termina su actividad antes de tiempo.
- Bloqueado/blocked: El hilo suspende su ejecución, no es igual que el estado de muerto pues la espera se da a lugar durante x tiempo y o condiciones, tras la espera es capaz de continuar y podemos añadir que cuando ocurre llama al método sleep() tras ello, queda bloqueado y llama al método wait().

▪ Diagrama de estados de **ejecución** de un hilo (simplificado)



## Sincronización

### ¿Cómo sincronizamos?

Para ello usamos la palabra clave synchronized.

```
Public synchronized void incrementar(){  
variable ++;  
}
```

### Sincronización y sus métodos

Si queremos usar métodos relacionados usamos:

- `.await()`: suspende el hilo hasta ejecutar `notify` o `notifyAll`.
- `.notify()`: despierta a solo un hilo en espera, no es aconsejable.
- `.notifyAll()`: despierta TODOS los hilos que esperan por un objeto de manera que un hilo puede acceder al recurso y luego hacer esperar al resto.

### Otras funciones útiles

- `isAlive()`: Devuelve true si el hilo ha comenzado y su ejecución no ha terminado.
- `toString()`: Devuelve un string con información del hilo.
- `getId()`: Devuelve un id único.
- `yield()`: Pasa el hilo a runnable permitiendo otros hilos ejecutarse.
- `setPriority()`: cambia la prioridad del hilo pero no necesariamente tiene que ser respetado, añadimos que permite números enteros de mínimo 1 hasta el máximo 10.

- `Interrupt()`: Debe ser usado en `trycatch`, le dice al hilo que va a estar en modo interrupción `.sleep()` o `.wait()`, si ya estaba interrumpido lanza `InterruptedException` es por ello que necesitamos un `trycatch`.
- `join()`: el hilo llamado espera a que otro termine.

## Productor consumidor

### ¿Qué es?

Es un modelo que presta un servicio como pueda ser una librería.

Por ejemplo tenemos un hilo que ofrece un servicio como un barista repartiendo cervezas y unos clientes que son quienes la consumen, devolviendo sus vasos.