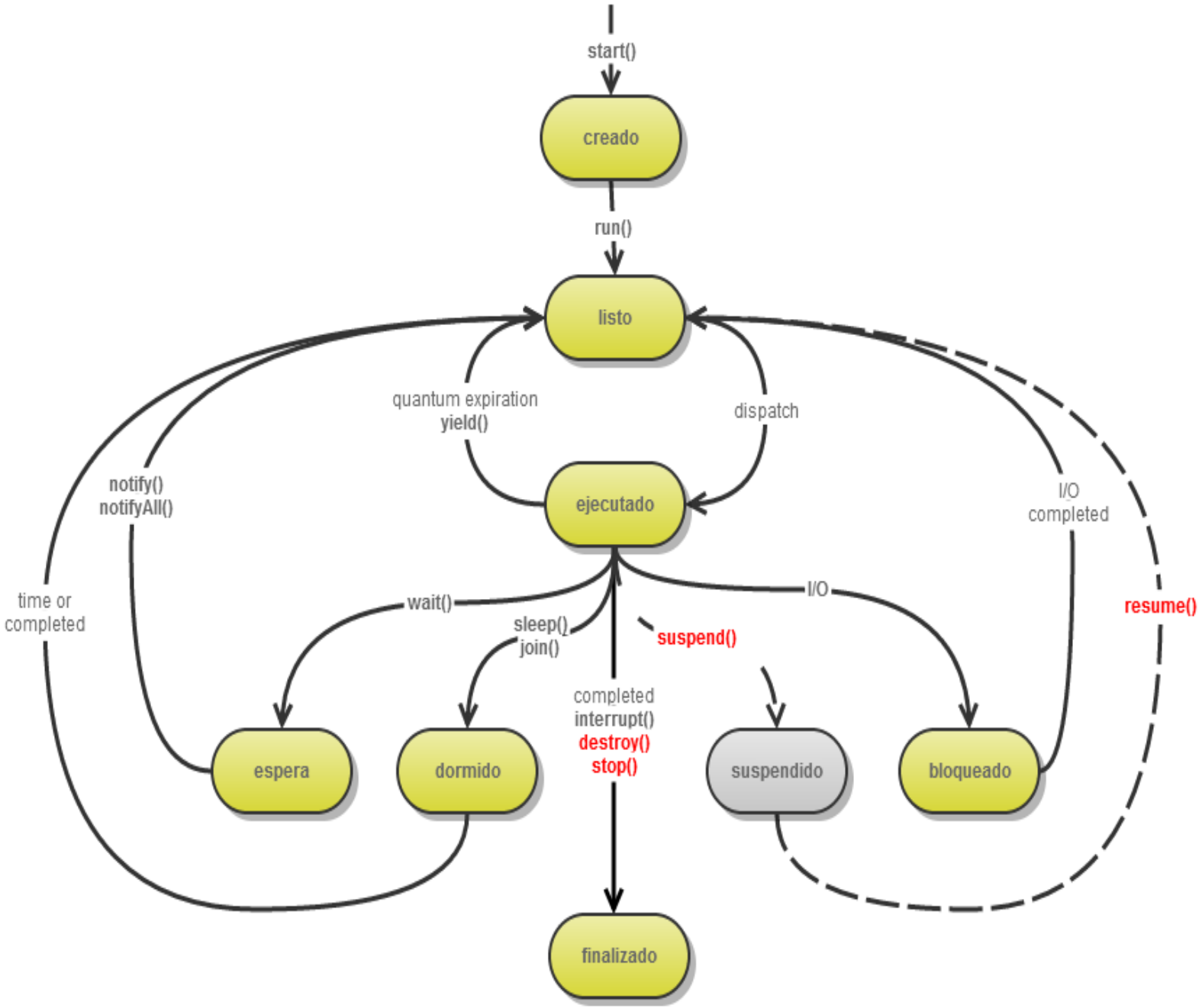
Estados y métodos de los Hilos en Java

Programación de servicios y procesos


start()
creado
run()
listo
quantum expiration
yield()
dispatch
notify()
notifyAll()
ejecutado
I/O
completed
time or
completed
wait()
sleep()
join()
suspend()
I/O
resume()
completed
interrupt()
destroy()
stop()
espera
dormido
suspendido
bloqueado
finalizado

Métodos I

- **interrupt()**, interrumpe un hilo, el hilo interrumpido debe capturar la excepción **InterruptedException** o comprobar si ha sido interrumpido con **Thread.interrupted()** o **isInterrupted()** y terminar su ejecución
- **isAlive()**, comprueba si el hilo sigue activo

Métodos II

- **join()**, espera a que termine otro hilo, se puede indicar también un tiempo máximo de espera
- **notify()**, despierta un hilo que esté esperando (wait())
- **notifyAll()**, despierta todos los hilos que estén esperando (wait())
- ~~**resume()**, deprecated~~

Métodos III

- **sleep()**, suspende temporalmente un hilo
- **start()**, lanza un hilo
- ~~**stop()**, deprecated~~
- ~~**suspend()**, deprecated~~
- **wait()**, el hilo esperará hasta recibir un notify() o notifyAll()
- **yield()**, petición de abandoner el procesador

Métodos IV

- **Thread.currentThread().getName()**, nombre del hilo
- **Thread.interrupted()**, comprueba si el hilo ha sido interrumpido y restablece el *flag* de interrupción
- **isInterrupted()**, comprueba si el hilo ha sido interrumpido

Ejemplo: pausar un hilo

Thread.sleep()

```
public void run() {
 System.out.println("antes");
  try {
    Thread.sleep(3000);
  } catch (InterruptedException ex) {
    return;
  }
  System.out.println(“después");
}
```

Interrumpir un hilo

- El método **interrupt()** permite finalizar la ejecución de un hilo.
- El hilo se deberá programar de modo que sea capaz de detectar que se ha tratado de interrumpir.
- Una forma de detectar la interrupción es capturando **InterruptedException**.
- Otra forma de detectar la interrupción consiste en ver si se ha tratado de interrumpir con **Thread.interrupted()** o **isInterrupted()**.

Thread.currentThread().interrupt();

- La mayor parte de la bibliografía recomienda que se utilice este método una vez que se haya detectado la interrupción de la hebra.

Ejemplo: interrumpir un hilo

InterruptedException

```
try {
    …
} catch (InterruptedException ex) {
    Thread.currentThread().interrupt();
    return;
}
```

Thread.interrupted()

```
…
if(Thread.interrupted()){
    Thread.currentThread().interrupt();
    return;
}
```

Esperar la finalización de un hilo

- Se puede forzar que la ejecución del hilo principal se detenga hasta que haya finalizado la ejecución de un hilo. Esto lo hace el método **join()**.
- También se puede esperar la finalización del hilo, indicando un tiempo de espera máximo usando **join(milisegundos)**.

Ejemplo: esperar un hilo

join()

```
public class UsoJoin implements Runnable {
    @Override
    public void run() {
        System.out.println(Thread.currentThread().getName());
        try {
            Thread.sleep(4000);
        } catch (InterruptedException ex) {
            return;
        }
        System.out.println(Thread.currentThread().getName());
    }
    public static void main(String args[]) throws InterruptedException {
        UsoJoin uj=new UsoJoin();
        Thread t = new Thread(uj);
        t.start();
        t.join();
        System.out.println(Thread.currentThread().getName());
    }
}
```

Ejemplo: hilo activo

join(milisegundos), isAlive(), interrupt()

```
t.start();
t.join(2000);
if(t.isAlive()){
    t.interrupt();
    j.join(); //es opcional
}
```

Se espera hasta un máximo de 2 segundos a que el hilo finalice. Si el hilo sigue activo, se interrumpe. El método join() final espera hasta que la hebra haya finalizado efectivamente.