Procesos, hilos y multihilos


Parallel code execution
S.KSHIRSAGAR

Objetivos

- Describir el concepto de hilos, procesos y multi hilos.
- Identificar los estados y control de hilos.

Procesos

¿Qué es un proceso?

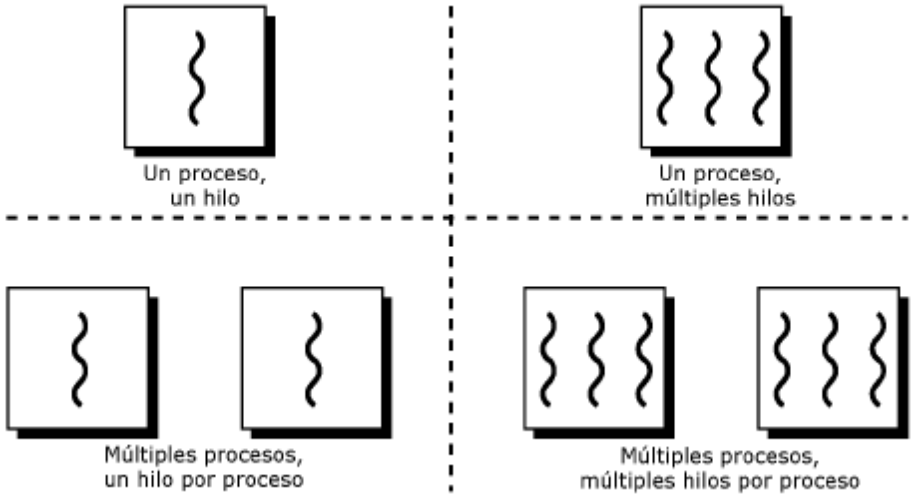
"Un proceso es un programa ejecutándose dentro de su propio espacio de direcciones".

"Son instrucciones de un programa destinadas a ser ejecutadas por el microprocesador"

Procesos

SISTEMAS DISTRIBUIDOS

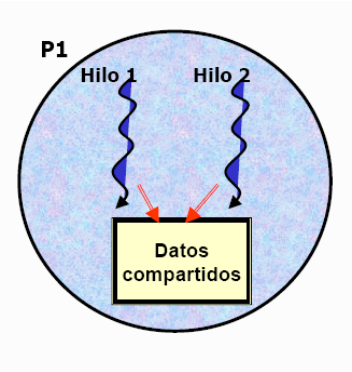
Se puede decir que un proceso es un **supervisor** de hilo(s) de ejecución.



Hilos

SISTEMAS DISTRIBUIDOS

- Un hilo es una secuencia de código en ejecución dentro del contexto de un proceso.
- Los hilos no pueden ejecutarse ellos solos.
- Requieren la supervisión de un proceso padre para correr.
- Dentro de cada proceso hay un hilo o varios hilos ejecutándose.



Ventajas de hilos

SISTEMAS DISTRIBUIDOS

- La ventaja que proporcionan los hilos es la capacidad de tener más de un camino de ejecución en un mismo programa.
- Multihilos en aplicaciones Cliente-Servidor
- Agilizar los tiempos de retraso de la comunicación cliente-servidor.

Ejemplo de hilos

SISTEMAS DISTRIBUIDOS

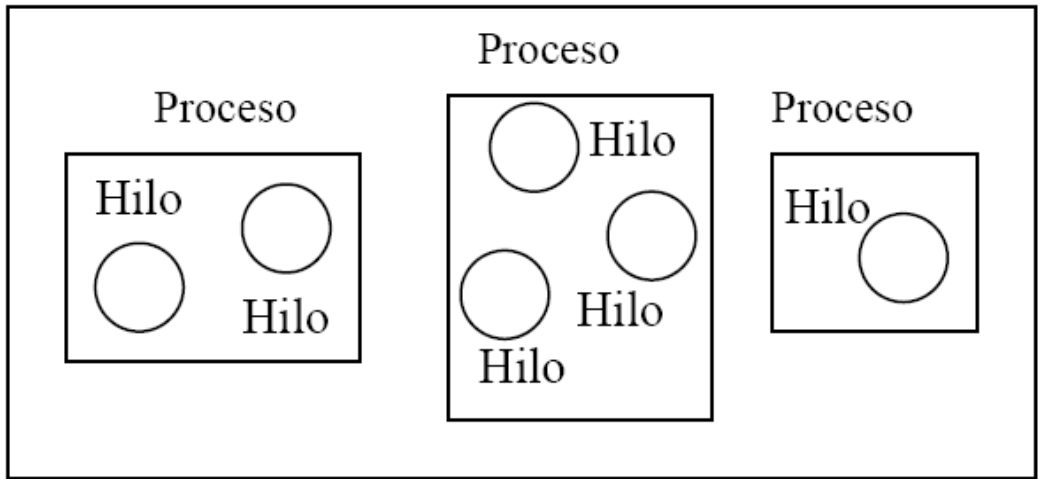
- Word puede tener un hilo en *background* chequeando automáticamente la gramática de lo que se escribe, mientras otro hilo puede estar guardando automáticamente los cambios del documento en el que se trabaja.

Hilos y procesos

SISTEMAS DISTRIBUIDOS

- Los hilos a menudo son conocidos o llamados procesos ligeros.

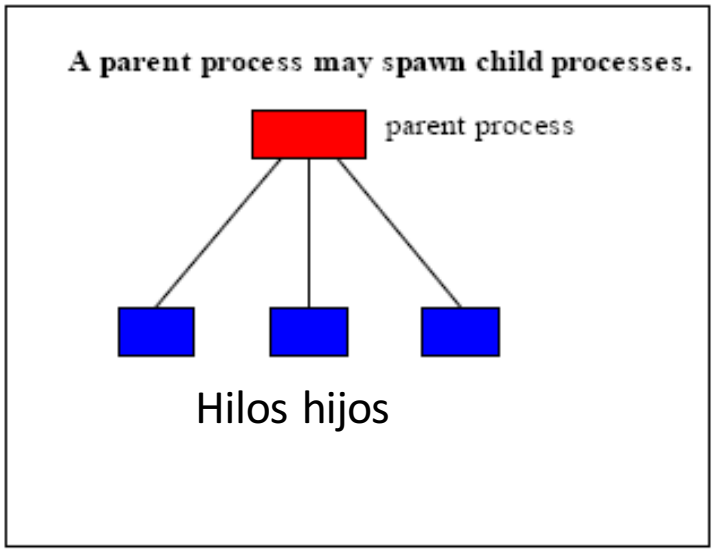
- Los hilos dependen de un programa padre en lo que se refiere a recursos de ejecución.

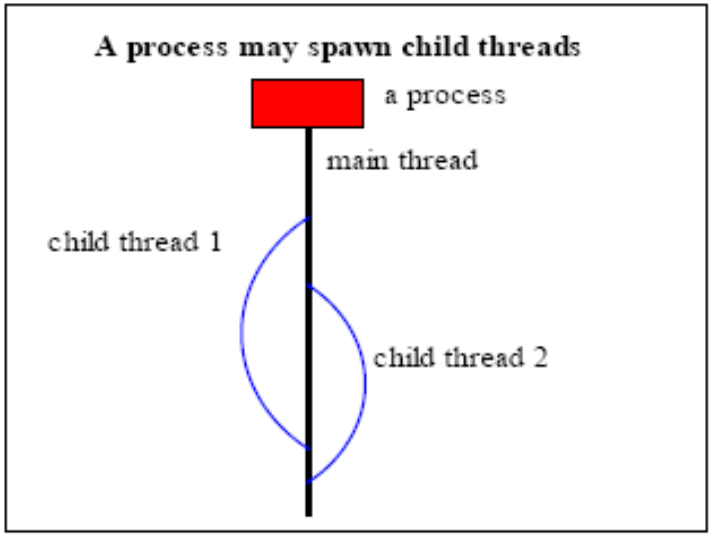


Hilos y procesos

SISTEMAS DISTRIBUIDOS

- Los hilo siempre existen dentro de un proceso, lo necesitan.
- Java Virtual Machine implementa le gestión de hilos.



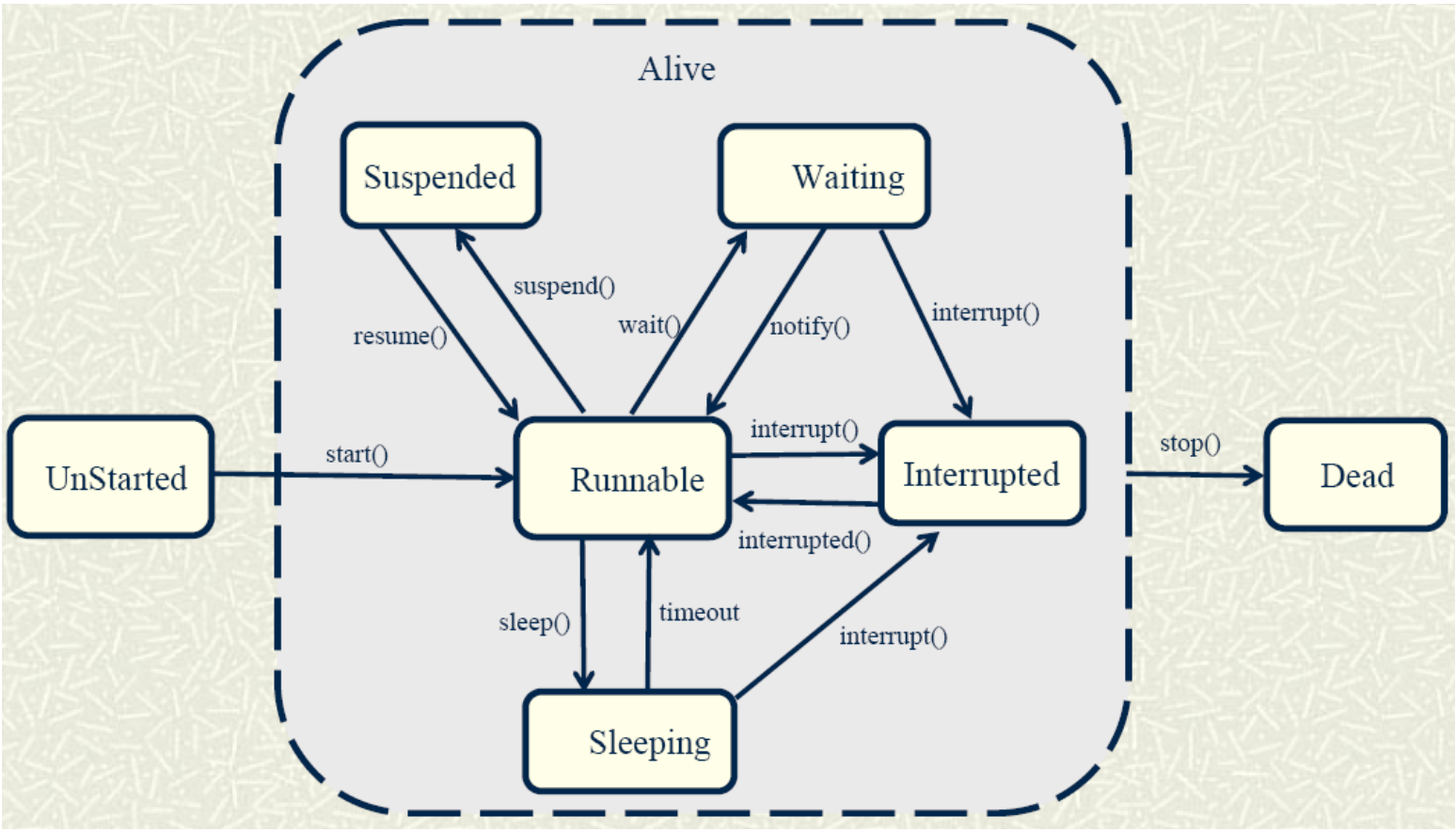


Estados y control de hilos

Estados de un hilo

SISTEMAS DISTRIBUIDOS

Semántica de los estados de un hilo.

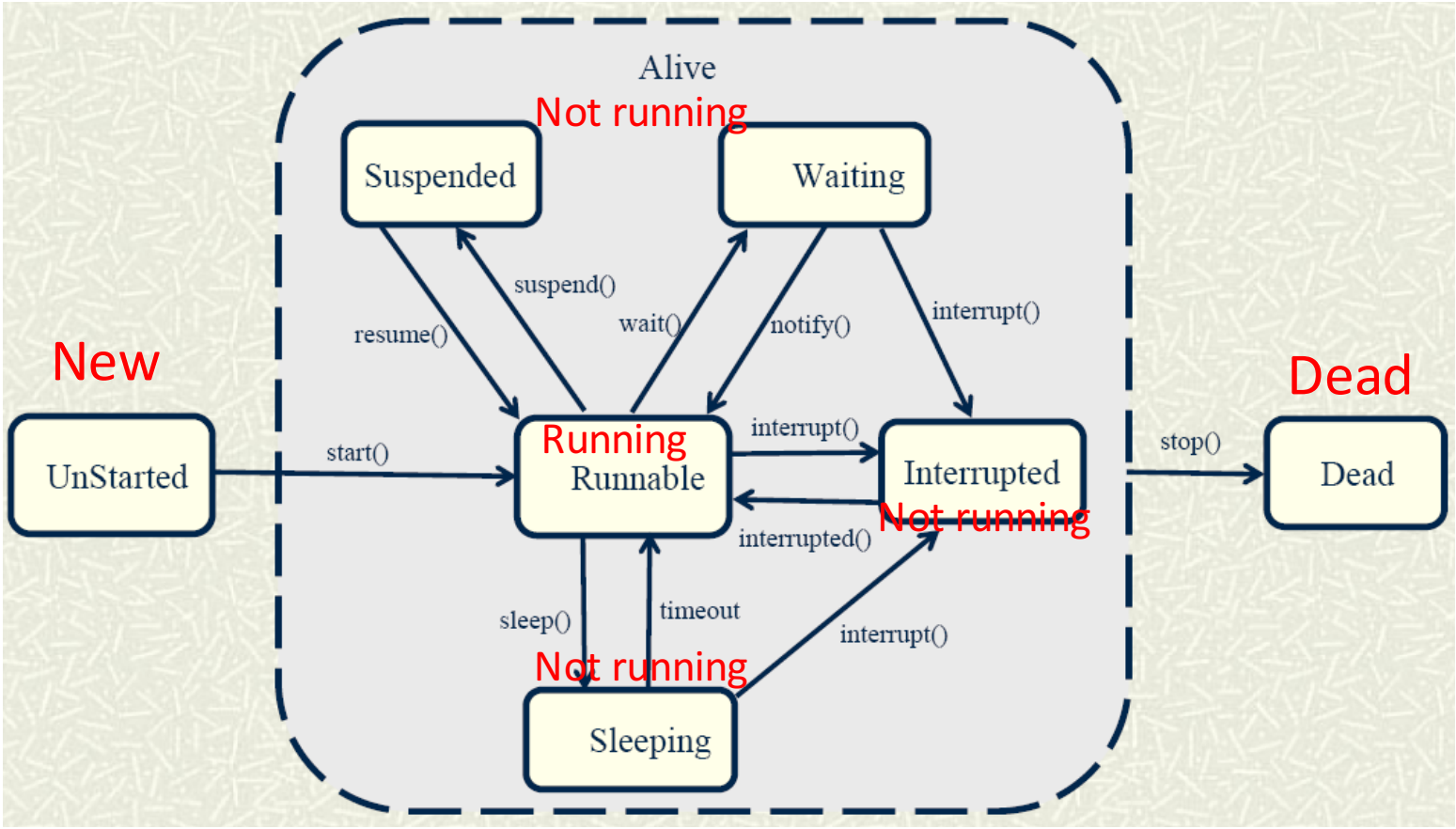


Estados de un hilo

El comportamiento de un hilo depende del estado en que se encuentre, este estado define su modo de operación actual. Los estados en los que puede estar un hilo Java:

- New
- Running
- Not running
- Dead

Estados de un hilo



Estados New

Un hilo se encuentra en el estado *new* la primera vez que se crea y hasta que el método *start* es llamado.

Los hilos en estado *new* ya han sido inicializados y están listos para empezar a trabajar, pero aún no han sido notificados para que empiecen a realizar su trabajo.

Estados Running

Cuando se llama al método *start* de un hilo nuevo, el método *run* es invocado y el hilo entra en el estado *running*.

Este estado podría llamarse “running” porque la ejecución del método *run* significa que el hilo esta corriendo.

Estados Not Running

El estado *not running* se aplica a todos los hilos que están parados por alguna razón. Cuando un hilo está en este estado, está listo para ser usado y es capaz de volver al estado *running* en un momento dado. Los hilos pueden pasar al estado *not running* a través de varias vías.

- El método *suspend()*
- El método *sleep ()*
- El método *wait()*
- El método *interrupt()*

Estados Dead

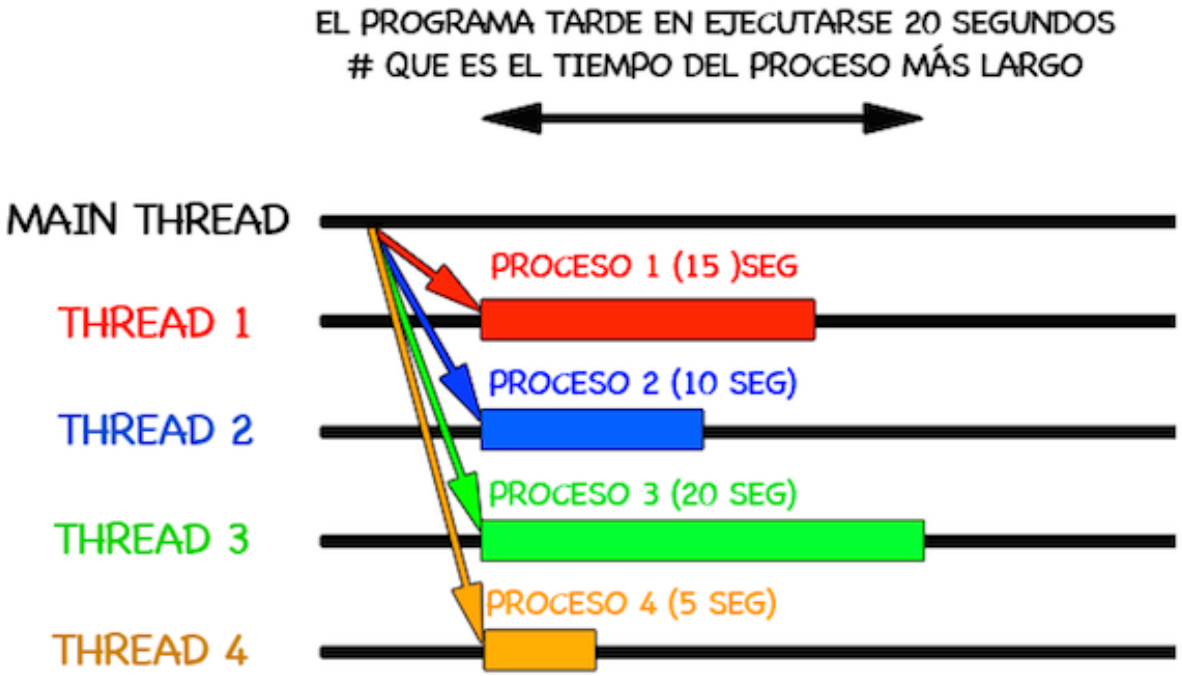
Un hilo entra en estado *dead* cuando ya no es un objeto necesario. Los hilos en estado *dead* no pueden ser resucitados y ejecutados de nuevo. Un hilo puede entrar en estado *dead* a través de dos vías:

- El método *run* termina su ejecución.
- El método *stop* es llamado.

La primera opción es el modo natural de que un hilo muera.

La segunda es una muerte causada.

Hilos en Java



Estados

SISTEMAS DISTRIBUIDOS

```
        // Hilos concurrentes mediante la clase Thread

        MiHilo h1= new MiHilo("Hilo1");


        h1.start();
```

```
    // cuerpo del hilo
    public void run() {
        try{

            for(int i=1; i<=10; i++)
            {

                System.out.println(nombre+ ": "+ i);

                Thread.sleep(1000);
            }

            System.out.println("Sale del hilo "+nombre);

        } catch(InterruptedException e){
            System.out.println(nombre+"Interrupcion del hilo"+nombre);
        }
    }

}
```

Dead

Running

New

Not Running

Estados

SISTEMAS DISTRIBUIDOS

```
        // Hilos concurrentes mediante la clase Thread

        MiHilo h1= new MiHilo("Hilo1");

        h1.start();
```

New

```
    // cuerpo del hilo
    public void run() {
        try{

            for(int i=1; i<=10; i++)
            {

                System.out.println(nombre+ ": "+ i);

                Thread.sleep(1000);
            }

            System.out.println("Sale del hilo "+nombre);

        } catch(InterruptedException e){
            System.out.println(nombre+"Interrupcion del hilo"+nombre);
        }
    }

}
```

Running

Not Running

Dead

Control de hilos

Arranque de un hilo

start()

- Este método indica al intérprete de Java que cree un contexto del hilo del sistema y comience a ejecutarlo.
- A continuación, el método *run()* de este hilo será invocado en el nuevo contexto del hilo.

- *start*, en realidad es un método oculto en el hilo que llama al método *run*.

Control de hilos

Manipulación del hilo

run()

- El método *run()* constituye el cuerpo de un hilo en ejecución. Este es el único método del interfaz **Runnable**.
- Es llamado por el método *start()* después de que el hilo apropiado del sistema se haya inicializado.
- Siempre que el método *run()* devuelva el control, el hilo actual se detendrá.

Control de hilos

Parada de un hilo

stop()

- Este método provoca que el hilo se detenga de manera inmediata. A menudo constituye una manera brusca de detener un hilo, especialmente si este método se ejecuta sobre el hilo en curso.

- Una forma más elegante de detener un hilo es utilizar alguna variable que ocasione que el método *run()* termine de manera ordenada.

- En realidad, nunca se debería recurrir al uso de este método (se encuentra **deprecated = desaprobado**)

Control de hilos

Suspensión de hilos

suspend()

- El método *suspend()* es distinto de *stop()*.
- *suspend()* toma el hilo y provoca que se detenga su ejecución sin destruir el hilo de sistema subyacente.
- Si la ejecución de un hilo se suspende, puede llamarse a *resume()* sobre el mismo hilo para lograr que vuelva a ejecutarse de nuevo.
- Es parecido al sleep(time) con la diferencia que suspend() es por tiempo indefinido.

Control de hilos

resume()

- El método *resume()* se utiliza para revivir un hilo suspendido.
- No hay garantías de que el hilo comience a ejecutarse inmediatamente, ya que puede haber un hilo de mayor prioridad en ejecución actualmente, pero *resume()* ocasiona que el hilo vuelva a ser un candidato a ser ejecutado.

Control de hilos

SISTEMAS DISTRIBUIDOS

Prioridades en hilos

setPriority(int)

- El método *setPriority()* asigna al hilo la prioridad indicada por el valor pasado como parámetro.

- Hay constantes predefinidas para la prioridad, definidas en la clase **Thread**, tales como MIN_PRIORITY, NORM_PRIORITY y MAX_PRIORITY, que toman los valores 1, 5 y 10, respectivamente.

Control de hilos

Prioridades en hilos

getPriority()

- Este método devuelve la prioridad del hilo de ejecución en curso, que es un valor comprendido entre uno y diez.

Control de hilos

Identificación de hilos

setName(String)

- Este método permite identificar al hilo con un nombre. De esta manera se facilita la depuración de programas multihilo.
- El nombre aparecerá en todas las líneas de trazado que se muestran cada vez que el intérprete Java imprime excepciones no capturadas.

getName()

- Este método devuelve el valor actual, de tipo cadena, asignado como nombre al hilo en ejecución mediante *setName()*.

Control de hilos

SISTEMAS DISTRIBUIDOS

Identificación de hilos

boolean ***isAlive****()*

Retorna true si el *thread* se encuentra en el estado Alive (en alguno de sus subestados), esto es, ya ha comenzado y aun no ha terminado.