2.5.3. PARADA DE UN HILO

El método stop() detiene la ejecución de un hilo de forma permanente y esta no se puede reanudar con el método **start():**

h. stop;

Este método al igual que suspend(), **resume()** y destroy() han sido abolidos en Java 2 para reducir la posibilidad de interbloqueo. El método **run()** no libera los bloqueos que haya adquirido el hilo, y si los objetos están en un estado inconsistente los demás hilos podrán verlos y modificarlos en ese estado. En lugar de usar este método se puede usar una variable como se vio en el estado **Dead** del hilo.

El método isAlive() devuelve *true* si el hilo está vivo, es decir ha llamado a su método **run()** y aún no ha terminado su ejecución o no ha sido detenido con **stop();** en caso contrario devuelve *false*.

El metodo **interrupt()** envía una petición de interrupción a un hilo. Si el hilo se encuentra bloqueado por una llamada a **sleep()** o **wait()** se lanza una excepción *InterruptedException*, El método **isInterrupted()** devuelve *true* si el hilo ha sido interrumpido, en caso contrario devuelve *false*. El siguiente ejemplo usa interrupciones para detener el hilo. En el método **run()** se comprueba en el bucle while si el hilo esta interrumpido, si no lo esta se ejecuta el código. El método *interrumpir()* ejecuta el método **interrupt()** que lanza una interrupción que es recogida por el manejador **(catch):**

```
public class HiloEjemploInterrup extends Thread
public void run()    {
try        {
while    (!isInterrupted())   {
System.out.println("En el Hilo"); Thread.sleep(10);
}
catch    (InterruptedException e)    {
System.out.println("HA OCURRIDO UNA EXCEPCION"); }
System.out.println("FIN HILO"); }

public void interrumpir()    {
interrupt();


public static void main(String[] args)  {
HiloEjemploInterrup h        = new HiloEjemploInterrupO; h.start();
for(int    1=0;   i<1000000000;   i++)    ;//no hago nada h.interrumpir();
}//
}//
```

Un ejemplo de ejecución muestra la siguiente información:

```
En el Hilo
En el Hilo
HA OCURRIDO UNA EXCEPCION FIN HILO
```

Si en el código anterior quitamos la línea *Thread.sleep(10);* también hay que guitar el bloque **try-catch,** la interrupción será recogida por el método **isInterrupted(),** que será *true* con lo que la ejecución del hilo terminara ya que finaliza el método **run().**

El método join() provoca que el hilo que hace la llamado espere la finalización de otros hilos. Por ejemplo, si en el hilo actual escribo hl *join(),* el hilo se queda en espera hasta que muera el

```
Fin Bucle Hilo2
Hilo3:    6
```

Hilo3: 7
Fin Bucle Hilo3
FINAL DE PROGRAMA

Si en el ejemplo anterior quitamos los **join()** veremos que el texto FINAL DE PROGRAMA no se mostrará al final. El método **join()** puede lanzar la excepción *InterruptedException*, por ello se incluye en un bloque **try-catch.**

ACTIVIDAD 2.5

Realiza una pantalla grafica para iniciar dos hilos y finalizar su ejecución usando interrupciones. Se deben mostrar varios botones, el **botón *Comenzar Proceso*** crea los dos hilos y lanza su ejecución, los hilos solo se crean una vez, el botón se desactivará al pulsarle. Cada hilo tendrá su **botón para interrumpir su ejecución**. Se debe mostrar un mensaje en pantalla que indique si el hilo está corriendo o se ha sido interrumpida su ejecución. El **botón *Finalizar Proceso*** detiene los dos hilos y muestra en consola el valor final de cada contador. El cierre de la ventana hace lo mismo. El constructor del hilo recibe dos parámetros, uno con el nombre del hilo y el segundo la cantidad de milisegundos que permanece el hilo dormido.