Unión Europea

Fondo Social Europeo

*El FSE invierte en tu futuro*

# UD 2 Sincronizar HILOS (PSP)

CFGS Desarrollo de Aplicaciones Multiplataforma

Pepe

GENERALITAT VALENCIANA

Conselleria d'Educació, Investigació, Cultura i Esport

GOBIERNO DE ESPAÑA

MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

## Continguts

# 1 EL MÉTODO JOIN

## 1.1 Objetivo del método

El método *join()* provoca que el hilo que hace la llamada espere la finalización de otros hilos. Por ejemplo, si en el hilo actual escribo "fil1.join()", el hilo se queda en espera hasta que muera el hilo sobre el que se realiza el join(), en este caso, el fil1.

```
public class Hilo extends Thread {

    public void run() {
        for(int i=0; i<=5; i++)
        System.out.println("Hola soy un hilo "+Thread.currentThread().
            getName()+" "+i);
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Hilo fil1=new Hilo();

        fil1.start();

        try {
            fil1.join();
        } catch (InterruptedException e) {
        // TODO Auto-generated catch block
            e.printStackTrace();
        }


        //imprime el nombre del hilo en la Salida (función getName())
        System.out.println("El hilo se llama " + Thread.currentThread() +
            "\n");

    }

}
```

```
Hola soy un hilo Thread-0 0
Hola soy un hilo Thread-0 1
Hola soy un hilo Thread-0 2
Hola soy un hilo Thread-0 3
Hola soy un hilo Thread-0 4
Hola soy un hilo Thread-0 5
El hilo se llama Thread[#1,main,5,main]
```

(a) con join

```
Hola soy un hilo Thread-0 0
Hola soy un hilo Thread-0 1
El hilo se llama Thread[#1,main,5,main]

Hola soy un hilo Thread-0 2
Hola soy un hilo Thread-0 3
Hola soy un hilo Thread-0 4
Hola soy un hilo Thread-0 5
```

(b) sin join

Gracias al método join() le decimos al programa Principal que tiene que esperarse a que finalice el hilo

*fil1* para poder mostrar el mensaje de “El hilo se llama”, fíjate en la diferencia entre utilizar join y no utilizarlo.

Otra opción seria utilizar el método `run`, el hilo principal quedaria bloqueado hasta que terminen los hilos

```
public static void main(String[] args) {
        // TODO Auto-generated method stub
        Thread t1 = new Thread(() -> System.out.println("Hola 1"));
        Thread t2 = new Thread(() -> System.out.println("Hola 2"));
        Thread t3 = new Thread(() -> System.out.println("Hola 3"));

        System.out.println("Inicio");
        t1.run();
        t2.run();
        t3.run();
        System.out.println("Fin");


    }
```

```
Inicio
Hola 1
Hola 2
Hola 3
Fin
```

(c) con run

## 2 EL MÉTODO INTERRUPT

Interrumpimos, pramos la ejecución de un hilo.

```
public class Hilo extends Thread {

    public void run() {
        for(int i=0; i<=5; i++) {
            System.out.println("Hola soy un hilo "+Thread.currentThread().
               getName()+" "+i);
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {
```

```
                // TODO Auto-generated catch block
                //e.printStackTrace();
                System.out.println("I'm resumed");
                return;
            }
        }
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Hilo fil1=new Hilo();

        fil1.start();

        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }//pausar el hilo principal

      fil1.interrupt(); // si no ha finalizado fil1 lo paro yo.

    }

}
```

Veamos en un ejemplo práctico donde un hilo titene que ejcutar varias tareas *(los mensajes)* en un periodo de tiempo y si sobre pasa un tiempo estipulado, *deadline*, abortamos.

```
public class Interrupciones {

    public static void main(String[] args) throws InterruptedException {
        // TODO Auto-generated method stub
        List<String> messages =Arrays.asList("Mensaje1","Mensaje2","
           Mensaje3","Mensaje4");
        long starTime= System.currentTimeMillis();
        long deadLine =8000;

        Thread fil2 = new Thread(()->{
            Thread.currentThread().setName("fil2");
            System.out.println(Thread.currentThread().getName()+"
               Iniciando Trabajo");
            try {
                for(String msg:messages) {
                    Thread.sleep(4000);
                    String nom= Thread.currentThread().getName();
                    System.out.println( nom+" "+msg);
                    //print(msg);
                }
                System.out.println(Thread.currentThread().getName()+" Fin
```

```
                    Trabajo");
                }catch(InterruptedException e) {
                    //throw new RuntimeException(e);
                    System.out.println(" Hilo Parado");
                }

            });

            fil2.start();

            while(fil2.isAlive() ) { //comprobamos que el hilo esta vivo...
                podria finalizaar antes
                System.out.println(Thread.currentThread().getName()+"
                    Esperando");
                fil2.sleep(1000);
                long endTime= System.currentTimeMillis();
                long lapso= endTime-starTime;

                if(lapso>deadLine && fil2.isAlive()) { //comprobamos los
                    tiempos  y si está vivo... en caso afirmativo abortamos.
                    System.out.println(Thread.currentThread().getName()+"
                        Sacabo");
                    fil2.interrupt();
                }

            }

            System.out.println(Thread.currentThread().getName()+" fin");

        }

        static void print(String message) {
            String nom= Thread.currentThread().getName();
            System.out.format("%s: %s%n", nom, message);
        }

}
```

## 3 Ejercicio

Vamos a siumlar en Java un programa para sabotear los servidores de la sede de defensa del gobierno de Estados Unidos con solo unas líneas de código malicioso.

Para ello, el programa solicitará al usuario el número de virus a inyectar. Por cada uno de ellos, el programa funcionará del siguiente modo:

- Lanzará un hilo de ejecución (codificado en forma de **expresión lambda**) informando el número

de virus (Virus 1, Virus 2, etc…).

- En la misma línea, se mostrará una barra de progreso en la que, en cada segundo, acumularemos de forma aleatoria un 5 o un 10, para de esa forma dotar de más realismo, si cabe.
- Cuando sumemos un 5 al total, imprimiremos una "X" en la barra de progreso y cuando sumemos 10, imprimiremos "XX", esperando medio segundo entre impresión e impresión para ir viendo en tiempo real como se va completando.
- Cuando lleguemos al 100%, imprimiremos el valor "100%" al lado de la barra de progreso y pasaremos a darle el control al siguiente virus, que se comportará exactamente igual.
- Cuando se hayan cargado todos los virus, el programa imprimirá el mensaje **"HAS SIDO INFECTADO"**. Un ejemplo de ejecución sería el siguiente: si queremos cargar 4 virus en el Pentágono, primero se carga el virus 1, luego el virus 2…

```
Introduzca numero de virus a cargar...
3

Virus 1:XXXXXXXXXXXXXXXXXXX 100%
Virus 2:XXXXXXXXXXXXXXXXXXX 100%
Virus 3:XXXXXXXXXXXXXXXXXXX 100%

HAS SIDO INFECTADO!
```