

UD2. Boletín 4.Métodos de la clase Thread

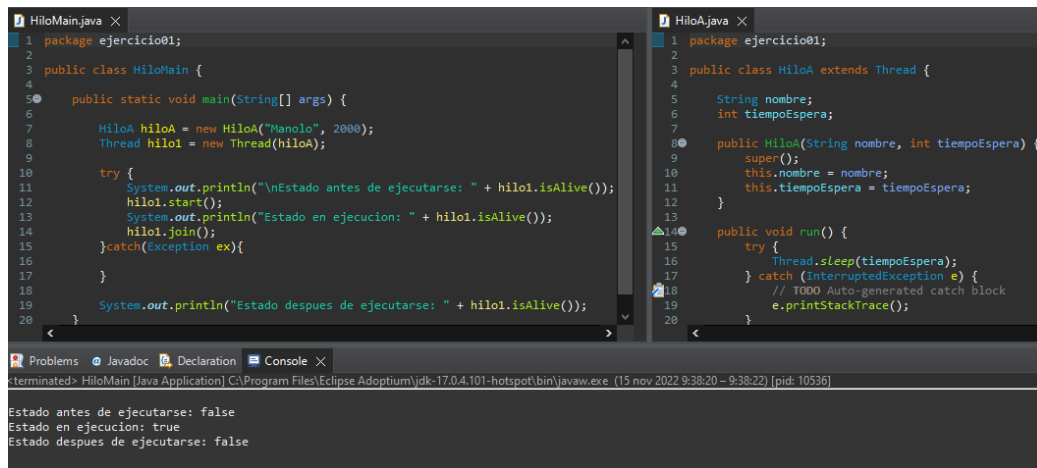
1. Crear un hilo cuyo método run() deberá dormirlo durante unos 2000 milisegundos (el método irá encerrado en un bloque try... catch (una excepción).

Luego en el programa principal quiero que me muestres el estado del hilo antes de ejecutarse (indicará false) y durante su ejecución (indicará que está vivo, true). Luego con el método join() esperaremos a que el hilo muera y se mostrará su estado (deberá estar muerto, false).

El método hilo.isAlive() me dice cuál es el estado del hilo.

Run:

```
Estado antes de iniciarse: false
Estado en ejecución: true
Estado después de ejecutarse: false
```



```
1 package ejercicio01;
2
3 public class HiloMain {
4
5     public static void main(String[] args) {
6
7         HiloA hiloA = new HiloA("Manolo", 2000);
8         Thread hilo1 = new Thread(hiloA);
9
10        try {
11            System.out.println("\nEstado antes de ejecutarse: " + hilo1.isAlive());
12            hilo1.start();
13            System.out.println("Estado en ejecucion: " + hilo1.isAlive());
14            hilo1.join();
15        } catch (Exception ex) {
16
17        }
18
19        System.out.println("Estado despues de ejecutarse: " + hilo1.isAlive());
20    }
21 }
```

```
1 package ejercicio01;
2
3 public class HiloA extends Thread {
4
5     String nombre;
6     int tiempoEspera;
7
8     public HiloA(String nombre, int tiempoEspera) {
9         super();
10        this.nombre = nombre;
11        this.tiempoEspera = tiempoEspera;
12    }
13
14    public void run() {
15        try {
16            Thread.sleep(tiempoEspera);
17        } catch (InterruptedException e) {
18            // TODO Auto-generated catch block
19            e.printStackTrace();
20        }
21    }
22 }
```

terminated> HiloMain [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.4-hotspot\bin\javaw.exe (15 nov 2022 9:38:20 - 9:38:22) [pid: 10536]

```
Estado antes de ejecutarse: false
Estado en ejecución: true
Estado despues de ejecutarse: false
```

2. Crea 3 hilos que se dormirán un tiempo comprendido entre 0 y 10 segundos (tiempo calculado aleatoriamente).

El método `run()` mostrará un mensaje similar a:

El *hiloX* va a estar dormido durante 7675 milisegundos.

Luego lo dormiremos (esto deberá estar incluido en un bloque `try {} catch`, atrapar una excepción).

Y finalmente indicaremos: *hiloX1 ha despertado*

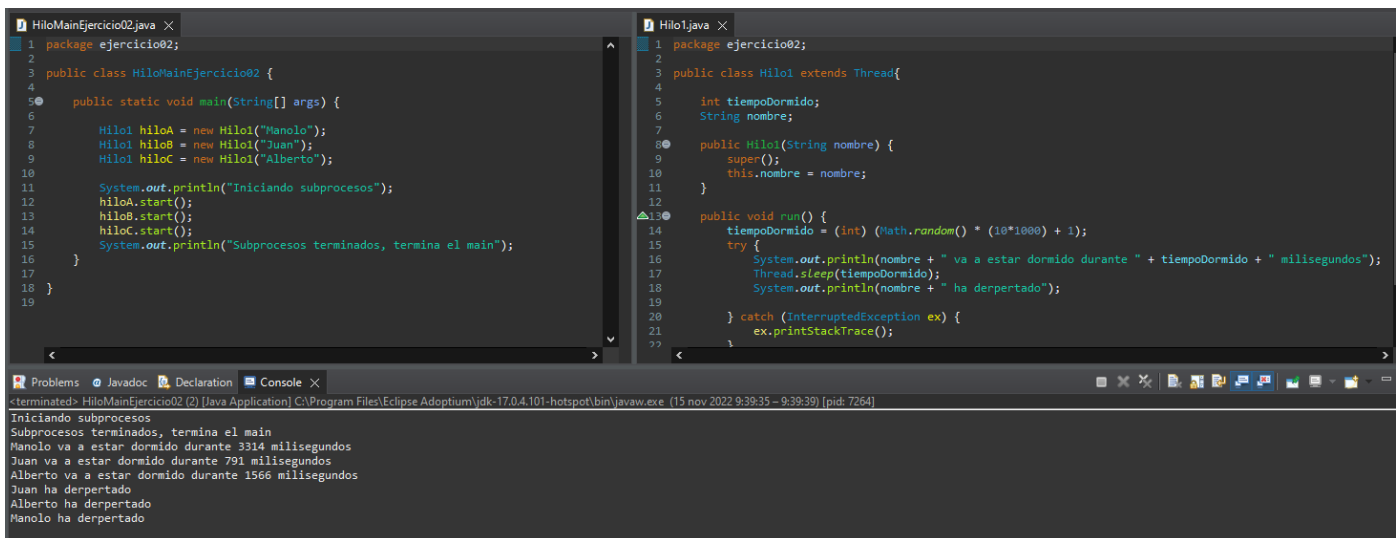
El tiempo que estará dormido el hilo lo calcularemos en el constructor mediante la fórmula:

$$tiempoDormido = ((int) (Math.random() * 10000));$$

Quando lo ejecutes comprueba que los hilos van despertando poco a poco.

Run:

```
Iniciando superprocesos
Subprocesos iniciados termina el main
hilo1 va a estar dormido durante 1026 milisegundos.
hilo2 va a estar dormido durante 6538 milisegundos.
hilo3 va a estar dormido durante 1614 milisegundos.
hilo1 ha despertado
hilo3 ha despertado
hilo2 ha despertado
```



```
1 HiloMainEjercicio02.java X
2 package ejercicio02;
3 public class HiloMainEjercicio02 {
4
5     public static void main(String[] args) {
6
7         Hilo1 hiloA = new Hilo1("Manolo");
8         Hilo1 hiloB = new Hilo1("Juan");
9         Hilo1 hiloC = new Hilo1("Alberto");
10
11         System.out.println("Iniciando subprocesos");
12         hiloA.start();
13         hiloB.start();
14         hiloC.start();
15         System.out.println("Subprocesos terminados, termina el main");
16     }
17 }
18
19
20 Hilo1.java X
21 package ejercicio02;
22 public class Hilo1 extends Thread{
23
24     int tiempoDormido;
25     String nombre;
26
27     public Hilo1(String nombre) {
28         super();
29         this.nombre = nombre;
30     }
31
32     public void run() {
33         tiempoDormido = (int) (Math.random() * (10*1000) + 1);
34         try {
35             System.out.println(nombre + " va a estar dormido durante " + tiempoDormido + " milisegundos");
36             Thread.sleep(tiempoDormido);
37             System.out.println(nombre + " ha despertado");
38         } catch (InterruptedException ex) {
39             ex.printStackTrace();
40         }
41     }
42 }
```

```
<terminated> HiloMainEjercicio02 (2) [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.4-hotspot\bin\javaw.exe. (15 nov 2022 9:39:35 - 9:39:39) [pid: 7264]
Iniciando subprocesos
Subprocesos terminados, termina el main
Manolo va a estar dormido durante 3314 milisegundos
Juan va a estar dormido durante 791 milisegundos
Alberto va a estar dormido durante 1566 milisegundos
Juan ha despertado
Alberto ha despertado
Manolo ha despertado
```

3. Construcción de un Thread implementando la interface Runnable.

Código del método run():

Mensaje indicando que se está ejecutando el hilo

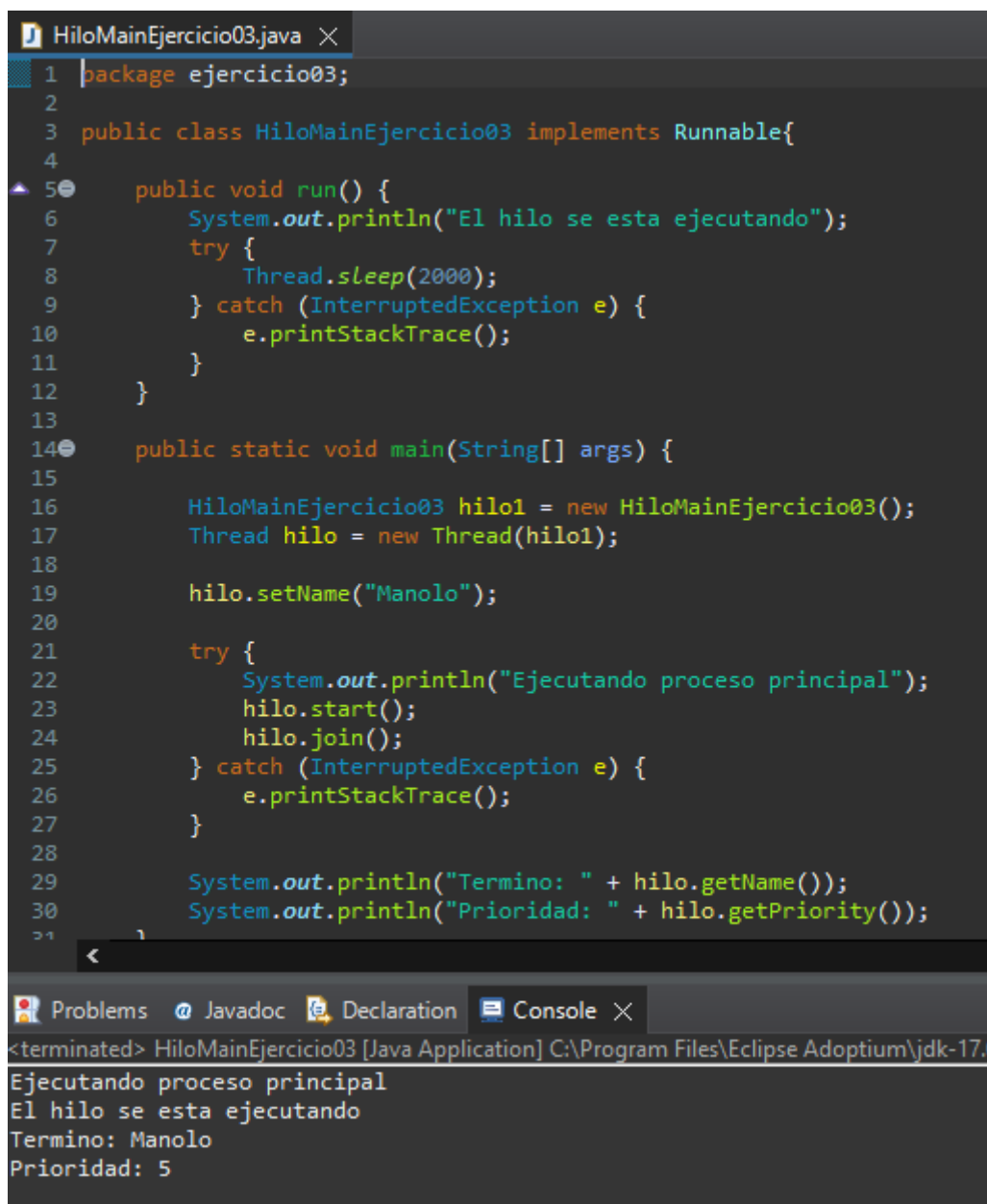
Se dormirá el thread durante 2 segundos ()

En el main:

Mensaje indicando la ejecución del proceso principal, asignarle un nombre al hilo (método setName) mensaje indicando que el hilo terminó se mostrará la prioridad del hilo.

Run:

```
Ejecutando proceso principal  
Hilo en ejecución  
Terminó: Hiliño  
Prioridad: 5
```



```
HiloMainEjercicio03.java X  
1 package ejercicio03;  
2  
3 public class HiloMainEjercicio03 implements Runnable{  
4  
5     public void run() {  
6         System.out.println("El hilo se esta ejecutando");  
7         try {  
8             Thread.sleep(2000);  
9         } catch (InterruptedException e) {  
10             e.printStackTrace();  
11         }  
12     }  
13  
14     public static void main(String[] args) {  
15  
16         HiloMainEjercicio03 hilo1 = new HiloMainEjercicio03();  
17         Thread hilo = new Thread(hilo1);  
18  
19         hilo.setName("Manolo");  
20  
21         try {  
22             System.out.println("Ejecutando proceso principal");  
23             hilo.start();  
24             hilo.join();  
25         } catch (InterruptedException e) {  
26             e.printStackTrace();  
27         }  
28  
29         System.out.println("Termino: " + hilo.getName());  
30         System.out.println("Prioridad: " + hilo.getPriority());  
31     }  
32 }  
<  
Problems Javadoc Declaration Console X  
<terminated> HiloMainEjercicio03 [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.  
Ejecutando proceso principal  
El hilo se esta ejecutando  
Termino: Manolo  
Prioridad: 5
```

4. Crea un hilo en cuyo método run() muestre los valores entre 1 y 10. Quiero que en el main, tras instanciar la subclase hilo anterior arranques la ejecución del hilo y con `hilo.join()` hagas esperar indefinidamente al hilo que se esté ejecutando en ese momento (el del main) hasta que finalice el nuestro, es decir, *hilo*

Opcion A -> sin usar join

Run:

```
Aún no se ha iniciado el hilo
hilo1 ha terminado
Fin del programa
1 2 3 4 5 6 7 8 9 10
```

Opcion B -> usando join

Run:

```
Aún no se ha iniciado el hilo
1 2 3 4 5 6 7 8 9 10 hilo1 ha terminado
Fin del programa
```

¿Entiendes el funcionamiento del join()?

```
1 package ejercicio04;
2
3 public class HiloMainEjercicio04 {
4
5     public static void main(String[] args) {
6
7         HiloA hilo = new HiloA();
8
9         System.out.println("Aun no se ha iniciado el hilo");
10        // hilo.start();
11        // System.out.println(hilo.getName() + " ha terminado");
12        // System.out.println("Fin del programa");
13
14        System.out.println("Aun no se ha iniciado el hilo");
15
16        try {
17            hilo.start();
18            hilo.join();
19            System.out.println(hilo.getName() + " ha terminado");
20        } catch (InterruptedException e) {
21            e.printStackTrace();
22        }
23
24        System.out.println("Fin del programa");
25    }
26 }
```

```
1 package ejercicio04;
2
3 public class HiloA extends Thread{
4
5     public void run() {
6         for(int i = 0; i <= 10; i++) {
7             System.out.print(i + " ");
8         }
9     }
10
11 }
12 }
```

Problems | Javadoc | Declaration | Console X

<terminated> HiloMainEjercicio04 (2) [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.4-hotspot\bin\javaw.exe (15 nov 2022 9:41:37 - 9:41:37) [pid: 4128]

```
Aun no se ha iniciado el hilo
0 1 2 3 4 5 6 7 8 9 10 Thread-0 ha terminado
Fin del programa
```

La funcionalidad del join es esperar a que acabe el subproceso antes de continuar con el proceso principal

5. A continuación te muestro un ejemplo similar pero con 2 hilos, transcríbelo y a ver si comprendes el funcionamiento de join()

```
1 package PaqueteMetodos;
2
3 public class MetodoJoin extends Thread {
4     MetodoJoin (String nom) {
5         super (nom);
6     }
7
8     public void run() {
9         int i;
10        for (i=1; i<=10; i++)
11            System.out.print(i+ " ");
12    }
13
14 }
15
16
17 package PaqueteMetodos;
18
19 public class MainMetodoJoin2 {
20
21     public static void main(String[] args)
22         throws InterruptedException {
23         MetodoJoin hilo1 = new MetodoJoin("hilo1");
24         MetodoJoin hilo2 = new MetodoJoin("hilo2");
25
26         System.out.println("Aún no se han iniciado los hilos");
27
28         hilo1.start();
29         hilo1.join();
30
31         System.out.println(hilo1.getName()+ " ha terminado");
32
33         hilo2.start();
34         hilo2.join();
35
36         System.out.println(hilo2.getName()+ " ha terminado");
37
38         System.out.println("Fin del programa");
39     }
40 }
41
42 }
```

Opcion 1 -> házlo suponiendo que no hay join()

Run:

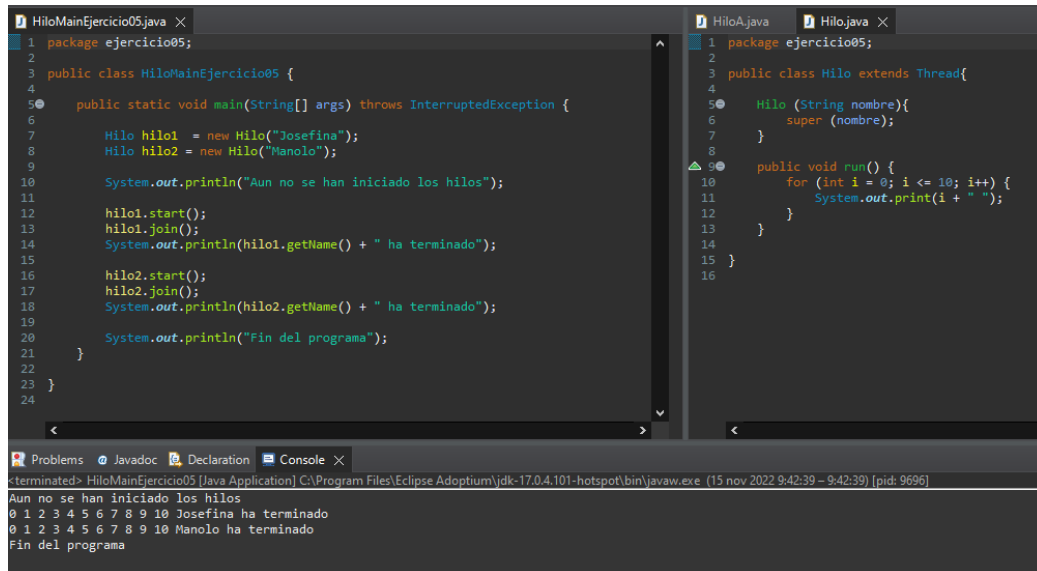
```
Aún no se han iniciado los hilos
hilo1 ha terminado
1 2 3 4 5 6 7 8 9 10 hilo2 ha terminado
Fin del programa
1 2 3 4 5 6 7 8 9 10
```

Opcion 2 -> hazlo con el join, la salida será ésta

Run:

```
Aún no se han iniciado los hilos
1 2 3 4 5 6 7 8 9 10 hilo1 ha terminado
1 2 3 4 5 6 7 8 9 10 hilo2 ha terminado
Fin del programa
```

¿Entiendes el funcionamiento del join()?



```
package ejercicio05;

public class HiloMainEjercicio05 {

    public static void main(String[] args) throws InterruptedException {

        Hilo hilo1 = new Hilo("Josefina");
        Hilo hilo2 = new Hilo("Manolo");

        System.out.println("Aun no se han iniciado los hilos");

        hilo1.start();
        hilo1.join();
        System.out.println(hilo1.getName() + " ha terminado");

        hilo2.start();
        hilo2.join();
        System.out.println(hilo2.getName() + " ha terminado");

        System.out.println("Fin del programa");
    }
}

package ejercicio05;

public class Hilo extends Thread{

    Hilo (String nombre){
        super (nombre);
    }

    public void run() {
        for (int i = 0; i <= 10; i++) {
            System.out.print(i + " ");
        }
    }
}
```

Problems Javadoc Declaration Console

<terminated> HiloMainEjercicio05 [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.4-hotspot\bin\javaw.exe (15 nov 2022 9:42:39 – 9:42:39) [pid: 9696]

```
Aun no se han iniciado los hilos
0 1 2 3 4 5 6 7 8 9 10 Josefina ha terminado
0 1 2 3 4 5 6 7 8 9 10 Manolo ha terminado
Fin del programa
```

La funcionalidad del join es esperar a que acabe el subproceso antes de continuar con el proceso principal