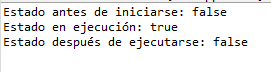
**UD2. Boletín 4.Métodos de la clase Thread**

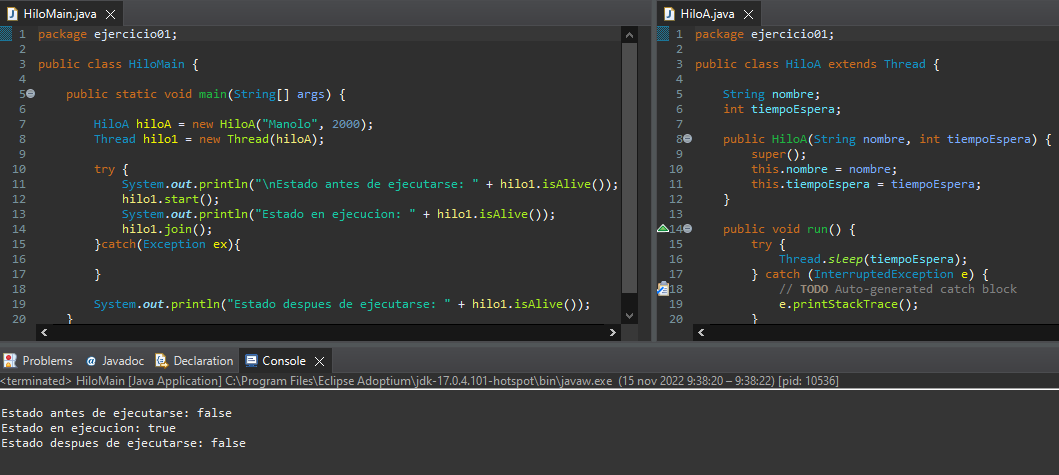
1. Crear un hilo cuyo método run() deberá dormirlo durante unos 2000 milisegundos (el método irá encerrado en un bloque try… catch (una excepción).

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:





1. 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.

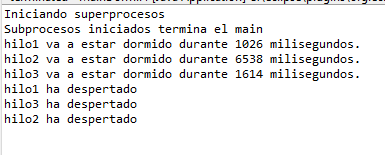
Y finalmente indicaremos: *hiloX1 ha despertado*

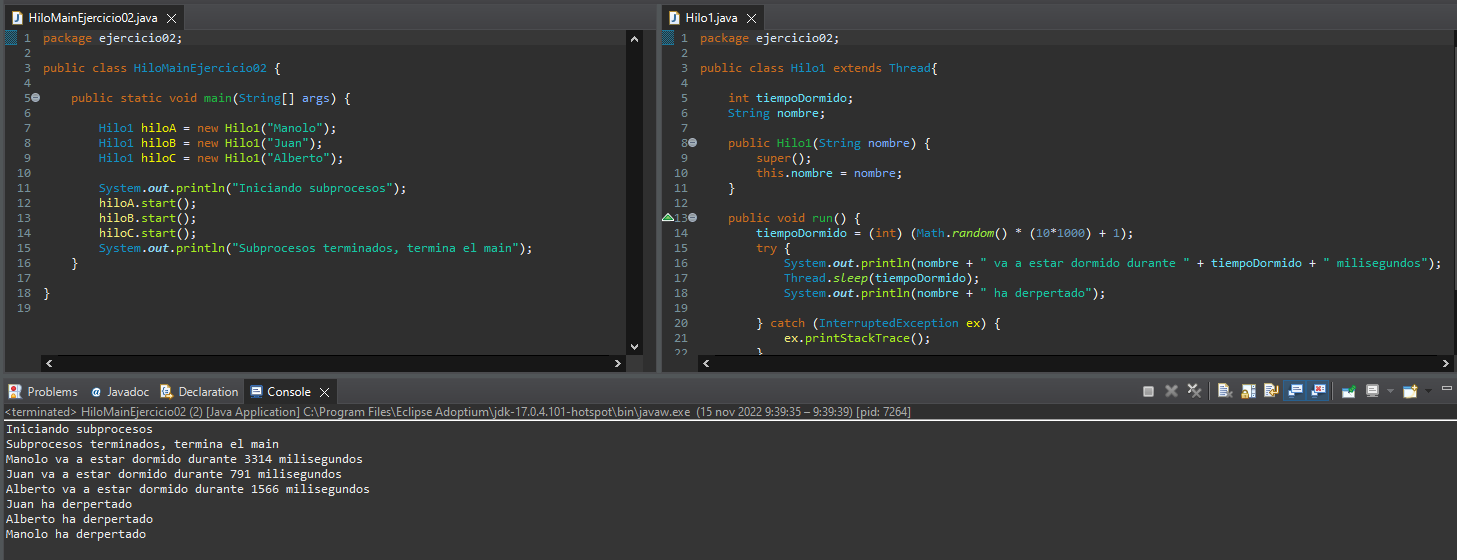
El tiempo que estará dormido el hilo lo calcularemos en el constructor mediante la fórmula:

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

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

Run:





1. Construcción de un Thread implementando la interface Runnable.

Código del método run():

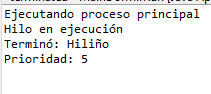
Mensaje indicando que se está ejecutando el hilo

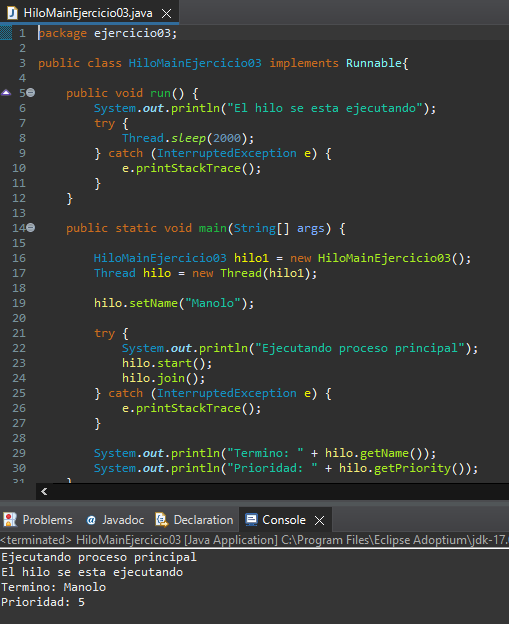
Se dormirá el thread durante 2 segundos ()

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:

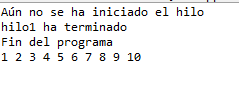




1. Crea un hilo en cuyo método run() muestre los valores entre 1 y 10. Quiero que en el main, tras instanciar la subclase hilo anterior arranques la ejecución del hilo y con *hilo.join()* hagas esperar indefinidamente al hilo que se esté ejecutando en ese momento (el del main) hasta que finalice el nuestro,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

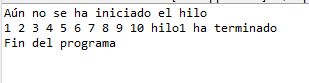
***Opcion A -> sin usar join***

Run:

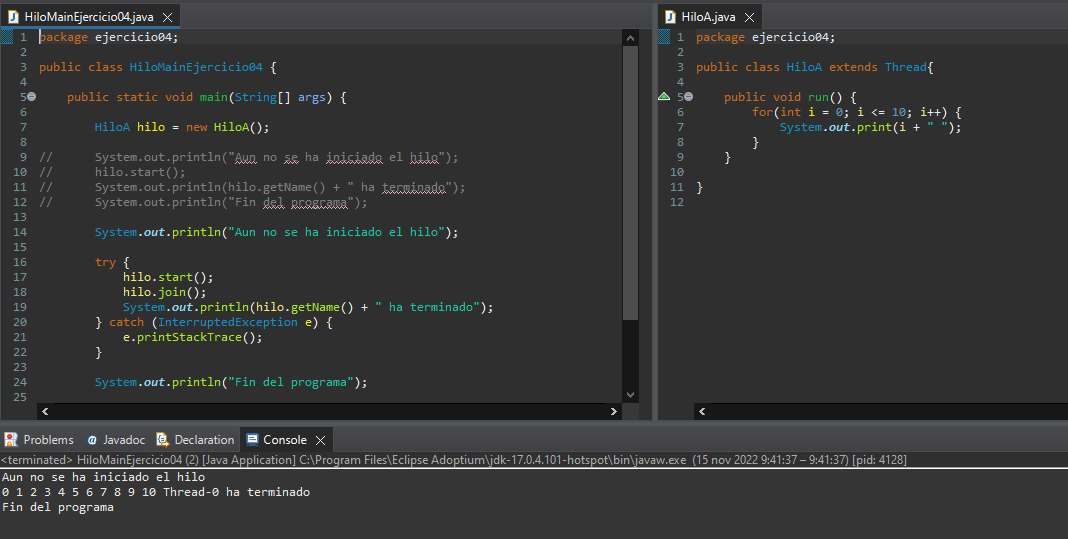


***Opcion B -> usando join***

Run:

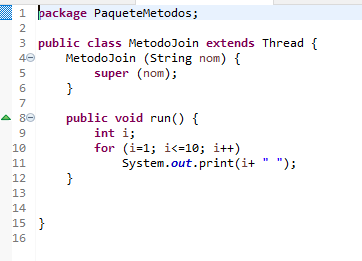


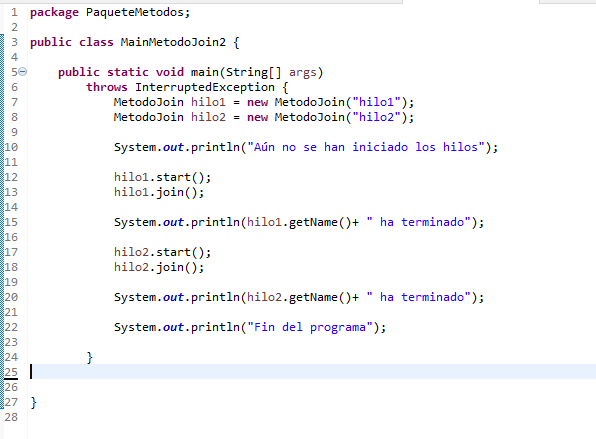
¿Entiendes el funcionamiento del join()?



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

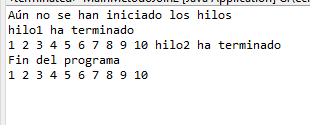
1. A continuación te muestro un ejemplo similar pero con 2 hilos, transcríbelo y a ver si comprendes el funcionamiento de join()





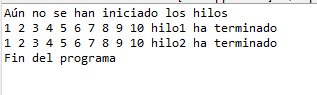
***Opcion 1 -> házlo suponiendo que no hay join()***

Run:

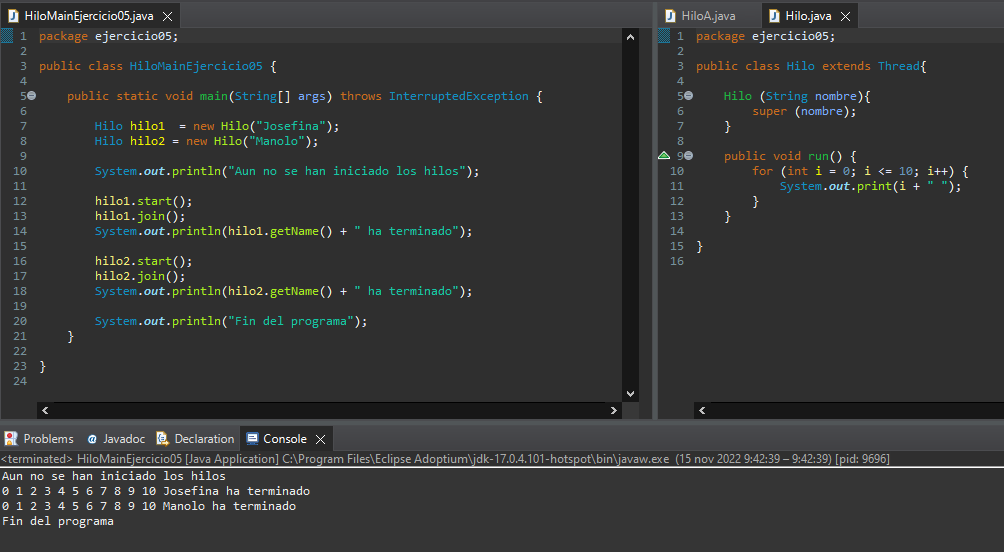


***Opcion 2 -> hazlo con el join, la salida será ésta***

Run:



¿Entiendes el funcionamiento del join()?



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