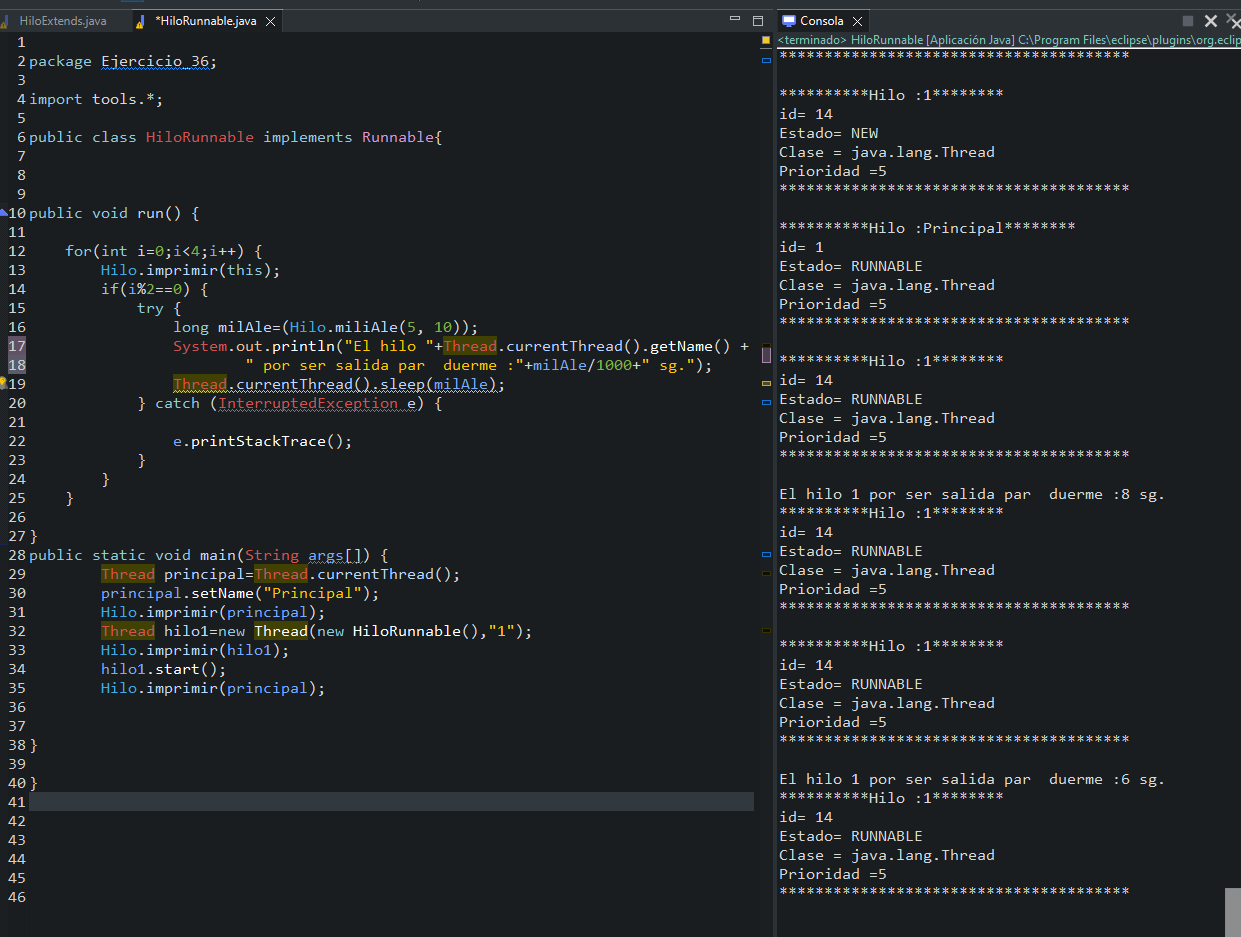
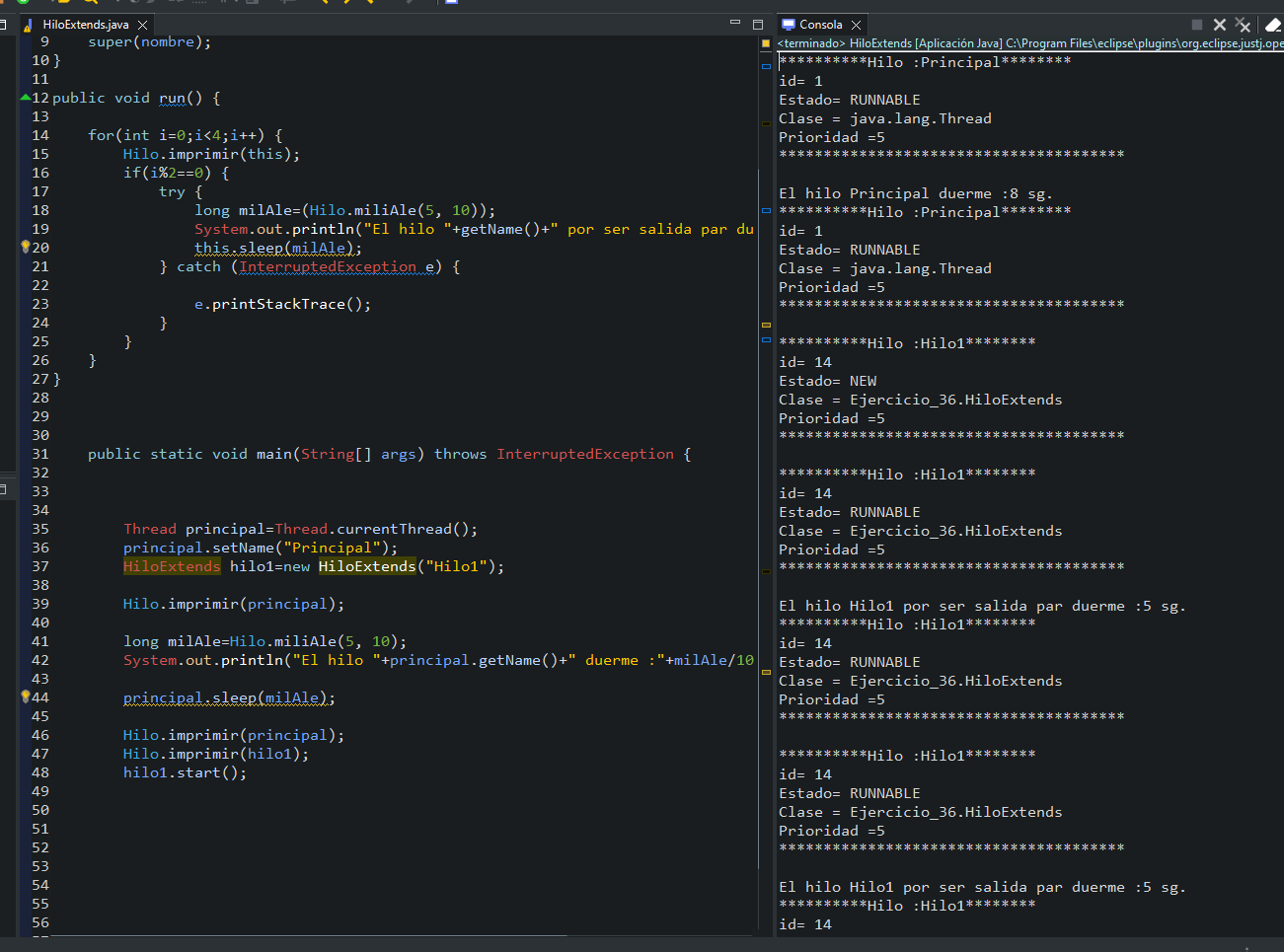
# Hilos

## Implemntacion Runnable y herencia de Thread

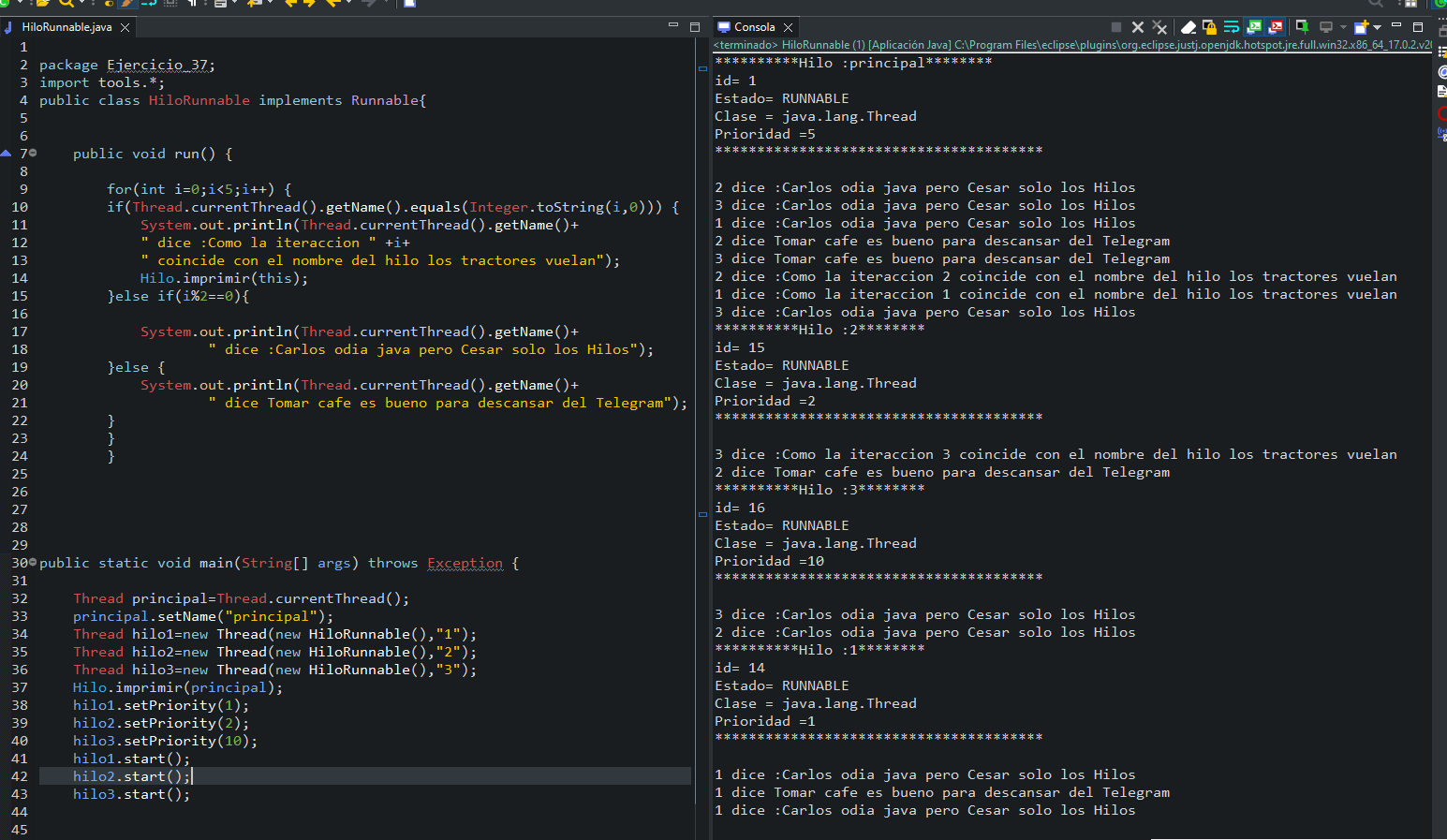
### Crea una clase que implemente Runnable en el que, durante X veces, se muestre el nombre del hilo, se duerma ese hilo durante un tiempo aleatorio y al finalizar se muestre que el hilo ha terminado .



### Crea una clase que extienda de Thread en el que, durante X veces, se muestre el nombre del hilo, se duerma ese hilo durante un tiempo aleatorio y al finalizar se muestre que el hilo ha terminado En el hilo principal (main) vete viendo como va cambiando de estado las instancias de las clases anteriores (puedes dormir e hilo principal durante algún segundo para verlo más claro) .



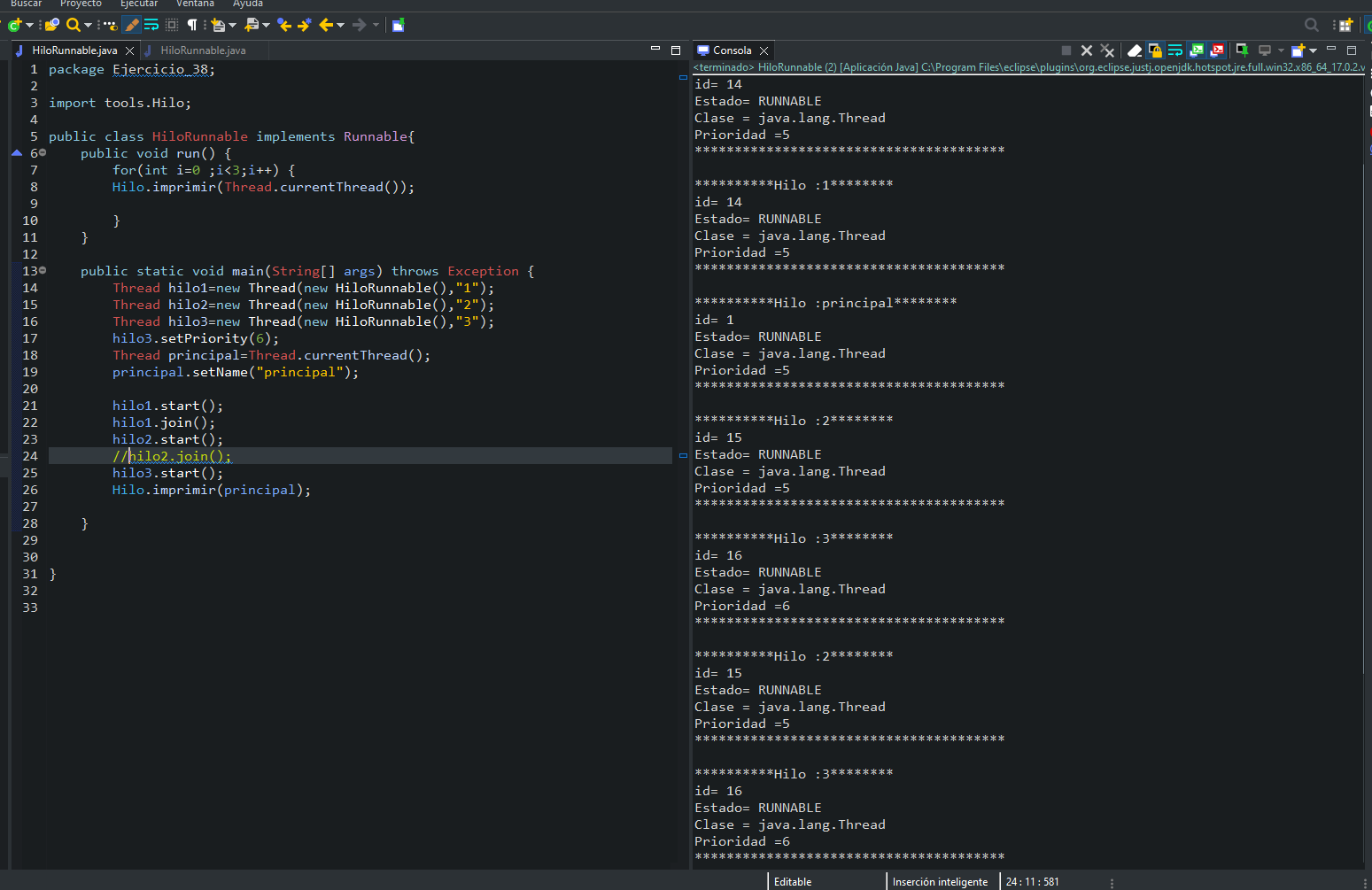
## Crea 3 hilos de la clase que implementa Runnable (ejercicio 36) y arráncalos. Interpreta lo que salga por pantalla



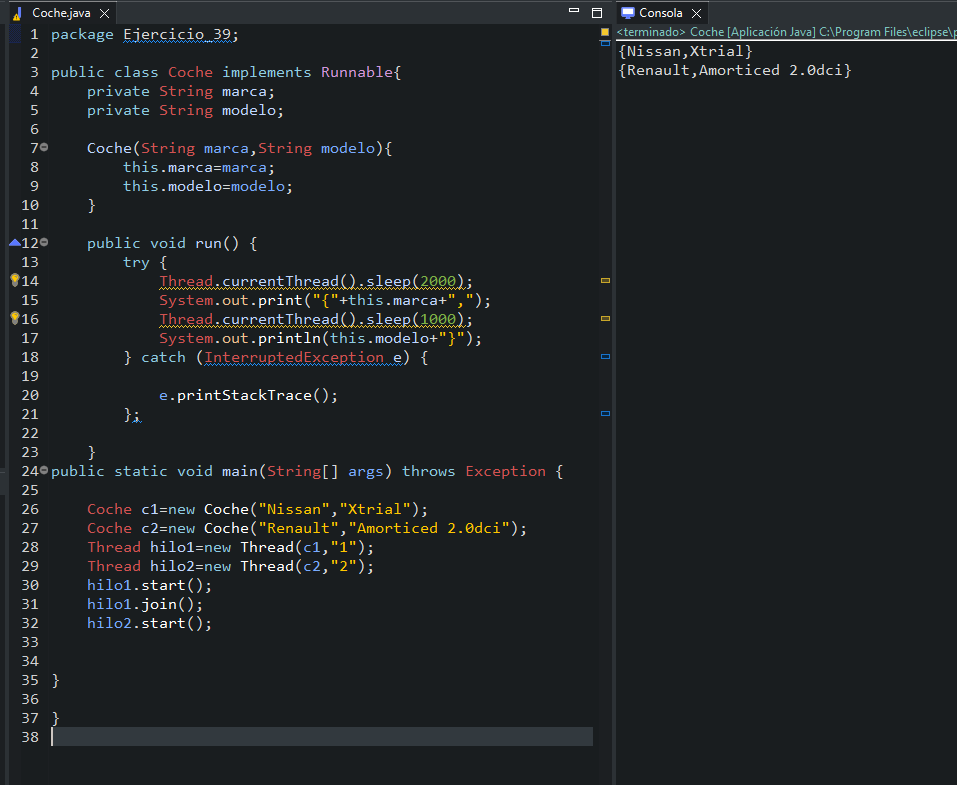
Las prioridades son +- dependiendo de cuando pase al estado del listo el hilo ,dando preferencia al valor de la prioridad.???? Ciertamente es así

## Siguiendo con el ejercicio anterior, usa el método join e interpreta la salida por pantalla.

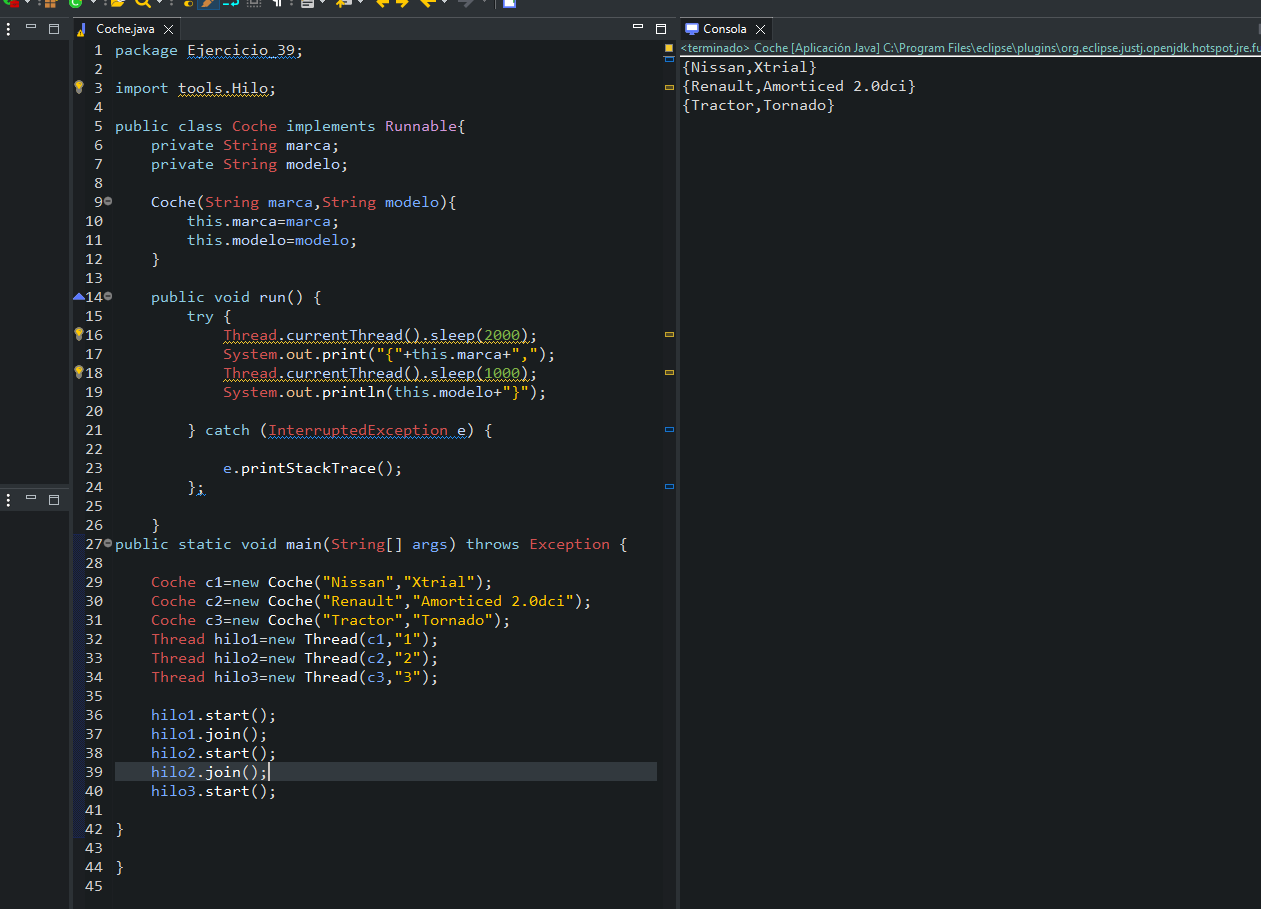
Void Join():Hace que el thread que se está ejecutando actualmente pase a el estado “esperando” indefinidamente hasta que se muera el hilo sobre el que se realiza el Join, en este caso el que está ejecutándose es el hilo principal el cual se para hasta que se termine o muera el hilo 1 desde el cual se invoca.



## Crea la clase coche que implemente runnable con marca y modelo como atributos. En el método run muestra los valores de los atributos separados en un espacio de tiempo (duerme el hilo por ejemplo 300 milisegundos o 1 segundo).



### Arranca en el método main 3 hilos de esta clase y observa su comportamiento con la salida por pantalla.



Se pude hacer sin Joins????

Ejercicio 40: Crea una clase distribuidor de coches donde se va a almacenar cuantos coches ha vendido y si tiene stock o no. Además contendrá la funcionalidad de poner un coche a la venta y otra que será la de vender un coche. Crea una clase fabricante de automóvil, que cada cierto tiempo (por ejemplo entre 10 y 15 segundos), intente entregar un coche al distribuidos de coches (esto durante un número determinado de veces) Crea una clase comprador de coches que, durante un número determinado de veces, intente comprar un coche. Desde el método main inicialízalos y observa qué va ocurriendo.

Ejercicio 41: Crea una tarea futura que muestre algún mensaje por pantalla

Ejercicio 42: Crea una tarea futura recurrente 5 veces por ejemplo (usa el AtomicInteger) Ejercicio 43: Crea una tarea que muestre el inicio y fin de la misma (con una diferencia de X segundos/milisegundos) y lánzala dentro de un SingleThreadExecutor Ejercicio 44: Crea dos tareas que muestren el inicio y fin de cada una de ellas (con una diferencia de X segundos/milisegundos) y devuelvan un numero al azar. Lánzalas dentro de un SingleThreadExecutor y comprueba si esa tarea ya se ha completado o no y cuál es el resultado( usa Future<?>) Ejercicio 45: Haz lo mismo que en el ejercicio anterior, pero añade las tareas a un FixedThreadPool. Observa las diferencias si las hay.

Ejercicio 44:

Crea dos tareas que muestren el inicio y fin de cada una de ellas (con una diferencia de X segundos/milisegundos) y devuelvan un numero al azar. Lánzalas dentro de un SingleThreadExecutor y comprueba si esa tarea ya se ha completado o no y cuál es el resultado(usa Future<?>)

Ejercicio 45:

Haz lo mismo que en el ejercicio anterior, pero añade las tareas a un FixedThreadPool. Observalas diferencias si las hay.