PSP_UT2_Foro Entrenamiento examen

Implementar un nuevo programa *Ejercicio03.java* modificando el programa principal del ejercicio 2 para usar instancias de la clase *HIloDormilonExclusivo* en lugar de instancias de la clase *HiloDormilon*. Para ello deberás implementar la clase *HiloDormilonExclusivo* donde tendrás que **ejecutar en exclusión mutua** todo el bloque de código que va desde el mensaje de inicio del hilo hasta el mensaje de fin del hilo. En este caso el **recurso compartido** por todos los hilos dormilones que debe ejecutarse en exclusión mutua será **la pantalla** (el objeto ***System.out***).

El resultado en pantalla de la ejecución del programa principal *Ejercicio03* debería ser el siguiente:

```
HILOS DORMILONES CONCURRENTES CON EXCLUSIÓN MUTUA
-------------------------------------------------
Iniciando hilo dormilón exclusivo 1 que va a dormir 5 segundos y bloquea al resto de
hilos que intenten usar la pantalla.
Finalizando hilo dormilón exclusivo 1 que acaba de dormir 5 segundos.
Iniciando hilo dormilón exclusivo 5 que va a dormir 2 segundos y bloquea al resto de
hilos que intenten usar la pantalla.
Finalizando hilo dormilón exclusivo 5 que acaba de dormir 2 segundos.
Iniciando hilo dormilón exclusivo 4 que va a dormir 4 segundos y bloquea al resto de
hilos que intenten usar la pantalla.
Finalizando hilo dormilón exclusivo 4 que acaba de dormir 4 segundos.
Iniciando hilo dormilón exclusivo 3 que va a dormir 3 segundos y bloquea al resto de
hilos que intenten usar la pantalla.
Finalizando hilo dormilón exclusivo 3 que acaba de dormir 3 segundos.
Iniciando hilo dormilón exclusivo 2 que va a dormir 1 segundos y bloquea al resto de
hilos que intenten usar la pantalla.
Finalizando hilo dormilón exclusivo 2 que acaba de dormir 1 segundos.
```

En este caso, como cada vez que se ejecuta un hilo de tipo "*dormilón exclusivo*", se "apropia" de la pantalla, ningún otro hilo que vaya a usar la pantalla podrá ejecutarse hasta que no finalice la parte de exclusión mutua del hilo anterior. Esto hará que la ejecución sea prácticamente "secuencial", pues hasta que el hilo 1 no duerma sus 5 segundos, muestre su mensaje de finalización por pantalla y abandone su sección de exclusión mutua, el hilo 2 no va a poder mostrar su mensaje de inicio y así sucesivamente con el resto de hilos. Observarás que el programa tardará mucho más en ejecutarse, pues el "dormir" no va a ser concurrente sino secuencial debido a que se encuentra dentro de un bloque de exclusión mutua (*synchronized*) para el objeto *System.out*.

```java
package Ejercicio2;

import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Main {

    public static void main(String[] args) {
        ArrayList<Thread> hilosDormilones = new ArrayList<>();
        long tiempo[] = {5, 1, 3, 4, 2};

        System.out.println("HILOS DORMILONES CONCURRENTES CON EXCLUSION MUTUA");
        System.out.println("------------------------------------------------");

        for (int i = 0; i < tiempo.length; i++) {
            HiloDormilonExclusivo hiloDormilon = new HiloDormilonExclusivo(String.valueOf(i + 1),
tiempo[i]);
            hilosDormilones.add(hiloDormilon);
        }
        for (Thread hilo : hilosDormilones) {
            hilo.start();
            try {
                Thread.sleep(100);
            } catch (InterruptedException ex) {
                Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }
}
```

```java
package Ejercicio2;

public class HiloDormilonExclusivo extends Thread {

    private final long tiempoEspera;
    private final String nombre;

    public HiloDormilonExclusivo(String nombre, long tiempo) {
        this.tiempoEspera = tiempo;
        this.nombre = nombre;
    }
    @Override
    public void run() {
        synchronized (System.out) {
            System.out.printf("Iniciando hilo dormilón exclusivo %s que va a "
                    + "dormir %d segundos y bloquea al resto de hilos que "
                    + "intentan usar la pantalla.\n",
                    this.nombre, this.tiempoEspera);
            try {
                Thread.sleep(this.tiempoEspera * 1000);
            } catch (InterruptedException ex) {
            }

            System.out.printf("Finalizando hilo dormilón exclusivo %s "
                    + "que acaba de dormir %d segundos.\n",
                    this.nombre, this.tiempoEspera);
        }
    }
}
```

Comentarios sobre hilos dormilones y sincronización...

CON JOIN y CON SYNCHRONIZED (System.out)
la salida es siempre la misma INDEPENDIENTEMENTE DEL MULTIPLICADOR DE TIEMPO.
la salida es 5, 1, 3, 4, 2 (es decir, primero entra el hilo de los 5 segundos y finaliza, después entra el de 1 segundo y finaliza,....) es decir, los hilos se ejecutan tal y como se leen desde los datos del array.
Lo del multiplicador de tiempo es que el *1000 lo he cambiado y probado con *1, *10 y *100 por si hubiera alguna diferencia
CON JOIN Y SIN SYNCHRONIZED (System.out)
la salida es siempre la misma que con join y synchronized. Al igual que en el caso anterior, la salida es INDEPENDIENTEMENTE DEL MULTIPLICADOR DE TIEMPO.
la salida es 5, 1, 3, 4, 2, es decir, los hilos se ejecutan tal y como se leen desde el array.

Las diferencias aparecen si pongo solo el **SYNCHRONIZED (System.out)**
con multiplicador *1000 y *100 la salida es: 5, 2, 4, 3, 1
con multiplicados *70 la salida es: 5, 4, 3, 1, 2
con multiplicador *50 la salida es: 5, 3, 1, 2, 4
con multiplicador *30, *10, *1 la salida es: 5, 1, 3, 4, 2

como veis, las únicas diferencias aparecen si no esta el join() y ademas la salida depende del tiempo que el hilo está dormido. Yo creo que añadir join() es incorrecto, ya que realmente no se sincroniza el recurso, simplemente espera a que termine un hilo para dar comienzo a otro, pero el recurso no se bloquea ni se sincroniza ni hay bloqueos o comunicación entre hilos, que creo que es lo que se pretendía. Lo dejo por aquí a ver que opináis vosotros y por si Fran nos diera su opinión y nos aclarara el tema.