PSP ENTRENAMIENTO FEBRERO

EJERCICIO 1 – HILOS DORMILONES

Dada la clase de tipo hilo *HiloDormilon.java*, escribir un hilo principal (programa principal ***Ejercicio02.java***) que **implemente un bucle donde se lancen tantas instancias de ese hilo "dormilón" como elementos tenga el array *arrayTiempos***, usando cada elemento (número entero) como parámetro para el constructor y dejando entre cada lanzamiento un pequeño lapsus (*Thread.sleep*) de 100 milisegundos para que dé tiempo a cada hilo a comenzar su ejecución y mostrar su mensaje de inicio sin que se lance antes otro.

Cada hilo de tipo dormilón que se ejecute:

1. mostrará un mensaje de inicio "*Iniciando hilo dormilón xxx que va a dormir zzz segundos*", donde xxx será el número de hilo (1, 2, 3, etc.) que se está lanzando y zzz será la cantidad de segundos que va a dormir;
2. dormirá la cantidad de segundos que se haya indicado en su constructor;
3. mostrará un mensaje de finalización "*Finalizando hilo dormilón xxx que acaba de dormir zzz segundos*".

El resultado en pantalla de la ejecución del programa principal *Ejercicio02* debería ser el siguiente:

```
HILOS DORMILONES CONCURRENTES
-----------------------------
Iniciando hilo dormilón 1 que va a dormir 5 segundos.
Iniciando hilo dormilón 2 que va a dormir 1 segundos.
Iniciando hilo dormilón 3 que va a dormir 3 segundos.
Iniciando hilo dormilón 4 que va a dormir 4 segundos.
Iniciando hilo dormilón 5 que va a dormir 2 segundos.
Finalizando hilo dormilón 2 que acaba de dormir 1 segundos.
Finalizando hilo dormilón 5 que acaba de dormir 2 segundos.
Finalizando hilo dormilón 3 que acaba de dormir 3 segundos.
Finalizando hilo dormilón 4 que acaba de dormir 4 segundos.
Finalizando hilo dormilón 1 que acaba de dormir 5 segundos.
```

Lo más probable es que siempre obtengas este resultado (quizá con alguna mínima variación), pues es razonable que primero vayan terminando los hilos que deben dormir menos tiempo.

CLASE HILO DORMILON (me la dan..)

```
package Ejercicio7;


public class HiloDormilon extends Thread {


    private final long tiempoEspera;
    private final String nombre;


    public HiloDormilon(String nombre, long tiempo) {
        this.tiempoEspera = tiempo;
        this.nombre = nombre;
    }


    @Override
    public void run() {
        System.out.printf("Iniciando hilo dormilón %s que va a dormir %d segundos.\n",
                this.nombre, this.tiempoEspera);
        try {
            Thread.sleep(this.tiempoEspera * 1000);
        } catch (InterruptedException ex) {
        }
        System.out.printf("Finalizando hilo dormilón %s que acaba de dormir %d segundos.\n",
                this.nombre, this.tiempoEspera);
    }


}
```

CLASE Main (propuesta compañero..)

```
public class Ejercicio02 {


	public static void main(String[] args) {
		int[] arrayTiempos = {5, 1, 3, 4, 2}; //Array con el tiempo a dormir de cada hilo
		HiloDormilon[] hilosDormilones = new HiloDormilon[arrayTiempos.length]; //Array de hilos

		//Rellena el array de HiloDormilon
		for (int i = 0; i < arrayTiempos.length; i++) {
			hilosDormilones[i] = new HiloDormilon(String.valueOf(i+1), arrayTiempos[i]);
			try{
				Thread.sleep(100);
			} catch (InterruptedException e){
				Logger.getLogger(Ejercicio02.class.getName()).log(Level.SEVERE, null, e);
			}
		}

		//Ejecuta cada hilo.
		for (HiloDormilon hd : hilosDormilones) {
			hd.start();
		}
	}
}
```

EJERCICIO 2 – HILOS DORMILONES en EXCLUSION MUTUA

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*.

Me dan…

```
package Ejercicio8;


public class HiloDormilonExclusivo extends Thread {


    // Aquí tienes que escribir tu solución al ejercicio

    // ...

    // ...


}
```

CLASE Main (propuesta compañero..)

```
public class Ejercicio02 {


	public static void main(String[] args) {
		int[] arrayTiempos = {5, 1, 3, 4, 2}; //Array con el tiempo a dormir de cada hilo
		HiloDormilon[] hilosDormilones = new HiloDormilon[arrayTiempos.length]; //Array de hilos

		//Rellena el array de HiloDormilon
		for (int i = 0; i < arrayTiempos.length; i++) {
			hilosDormilones[i] = new HiloDormilon(String.valueOf(i+1), arrayTiempos[i]);
			try{
				Thread.sleep(100);
			} catch (InterruptedException e){
				Logger.getLogger(Ejercicio02.class.getName()).log(Level.SEVERE, null, e);
			}
		}

		//Ejecuta cada hilo.
		for (HiloDormilon hd : hilosDormilones) {
			hd.start();
		}
	}
}
```

CLASE HiloDormilonExclusivo (propuesta compañero)

```
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\n"
					+ "que intenten usar la pantalla.\n\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\n",
					this.nombre, this.tiempoEspera);
		}
	}
}
```

EJERCICIO 3 – Lanzar programas a ejecución. Escribir en texto errores.

Crea un programa llamado *Ejercicio1.java* que reciba desde los argumentos del main() un nombre y lo lance a su ejecución, con las siguientes características:

- Si el programa finaliza correctamente utiliza *System.exit(1)*, en caso que no se hayan introducido los argumentos correctamente utiliza *System.exit(-1)*.

- Comprueba el valor de salida del proceso que se ejecuta, de tal manera que:

- Si se ejecuta correctamente muestra en pantalla la información que devuelve el programa lanzado (es importante invocar programas que generen algo, como “DIR”).

- Si no se ejecuta correctamente el lanzamiento, debe mostrar los errores en un archivo llamado “errores.txt”.

CLASE Main (propuesta compañero. Es correcto?)

package Ejercicio3;

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Ejercicio1 {

    public static void main(String[] args) {
        File file = null;
        FileWriter fw = null;

        try {
            String comando = args[0];//recogemos el comando que se pasa por argumentos

            Process process = Runtime.getRuntime().exec(new String[]{"cmd.exe", "/c", comando});
            //creamos proceso para que ejecute en la consola el comando almacenado

            process.waitFor();//esperamos hasta que termine el proceso
            //usamos un bufferedReader para leer la salida del proceso

            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));

            String line;

            while ((line = reader.readLine()) != null) {//leemos cada línea
                System.out.println(line); // y la pintamos
            }

            System.exit(1);

        } catch (IOException e) {//en caso de comando incorrecto
            try {
                file = new File("errores.txt");//se crea fichero
                fw = new FileWriter(file);//accedemos a él para escribir
                fw.write(e.getMessage());// y escribiremos el contenido del error
                fw.close();
                System.exit(-1);
            } catch (IOException ex) {
                Logger.getLogger(Ejercicio1.class.getName()).log(Level.SEVERE, null, ex);
            }

        } catch (InterruptedException ex) {
            Logger.getLogger(Ejercicio1.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
}
```