PSP Tema 1: Programación concurrente

Además del correcto funcionamiento de los ejercicios se valorará:

- **La legibilidad del código: código autoexplicativo y/o comentarios.**
- **La calidad del código: sin código duplicado, sin código superfluo, desacoplado.**
- **La estructura del código: código ordenado.**
- **Seguir la convención estándar para Java: https://www.oracle.com/docs/tech/java/codeconventions.pdf**

Ejercicio 1 (4 puntos)

Crea un programa que simule como 3 jardineros hacen trabajos en 10 jardines de una localidad. Los jardineros escogen un jardín, lo trabajan y pasan a trabajar en otro jardín. En un momento dado hay un sólo jardinero por jardín. Trabajar en un jardín cuesta un día y la simulación dura 30 días. Cada día de la simulación equivale a 1 segundo real.

El funcionamiento del programa será el siguiente siguiente:

- Hay 10 jardines y 3 jardineros.
- Los jardineros empiezan a trabajar:
  - Buscan un jardín libre de manera aleatoria. No debe estar siendo trabajado en este momento por otro jardinero.
  - Empieza a trabajarlo (durante un día / segundo)
  - Termina liberando el jardín. Y a partir de aquí se repite el proceso hasta que termina la simulación.
- El programa principal espera a que hayan terminado todos los hilos.

El programa irá mostrará la siguiente información:

- Cada hilo mostrará el siguiente mensaje cuando se empiece a trabajar en un jardín: “El jardinero 3 ha empezado a trabajar en el jardín 9”. Siendo 3 el identificador único del hilo y 9 el identificador numérico que asignamos a mano a cada jardín.
- Cada hilo mostrará el fin de los trabajos de cada jardinero. Por ejemplo: “El jardinero 3 ha terminado de trabajar en el jardín 9”. Siendo 3 el identificador único del hilo y 9 el identificador numérico que asignamos a mano a cada jardín.
- El **programa principal** esperará que finalice la simulación y mostrará cuántos jardines ha trabajado cada jardinero.

El jardinero 1 ha trabajado 30 jardines.
El jardinero 2 ha trabajado 30 jardines.
El jardinero 3 ha trabajado 30 jardines.

Ejercicio 2 (4 puntos)

Partiendo del ejercicio anterior:

- Vamos a añadir a 5 ciudadanos. Los ciudadanos cada 2 días visitan un jardín al azar (hasta que termina la simulación) y se pasan medio día en el jardín. Si ese jardín hace más de 2 días que ha sido trabajado, no se encuentra en buen estado y por tanto el ciudadano hace una queja formal que se mostrará al finalizar el programa.
  "El jardín X está en mal estado." Donde X es el identificador del jardín.
- Modifica la selección del siguiente jardín a trabajar por cada jardinero. Deben escoger el jardín que lleva más tiempo descuidado para minimizar las quejas.

El programa principal esperará a que termine la simulación y mostrará la siguiente información:

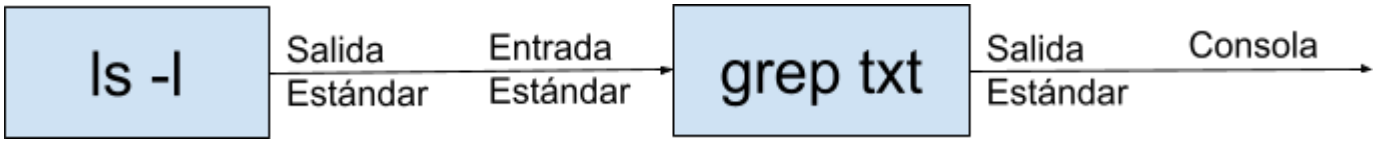
Trabajadores:
El jardinero 1 ha trabajado X jardines.
El jardinero 2 ha trabajado X jardines.
El jardinero 3 ha trabajado X jardines.

Quejas recibidas:
El jardín 3 está en mal estado.
El jardín 12 está en mal estado.

Ejercicio 3 (2 puntos)

Crea un programa que leerá por consola dos comandos con parámetros, creará dos procesos y los conectará simulando el comportamiento de una tubería. Finalmente mostrará por pantalla la salida estándar del segundo comando / proceso.

Si al programa le pasamos los siguientes comandos:
ls -l
grep txt
El resultado será el mismo que al ejecutar:
ls -l | grep txt
Este comando muestra todos los nombres de ficheros y propiedades listados por ls -l que contengan txt.



Recuerda que una tubería lo que hace es unir dos procesos del siguiente modo: la **salida estándar del primer proceso**, se convierte en la **entrada estándar del segundo proceso**.
Ten en cuenta también que para notificar que has terminado la escritura por la entrada estándar tienes que hacer **close()** del stream, writter o fichero que estés usando para escribir.