Práctica – 9

Coordinación y cooperación entre procesos.

Al programa concurrente de *Productores-Consumidores* presentado en *matBH*, se le han hecho algunos cambios, en cuanto a la cantidad de bienes que caben en el almacén. REVISE los programas Java.

Ahora desde la creación de madriguera se indica cuántos bienes, máximo, pueden llegar a almacenarse temporalmente en el almacén. Antes sólo había lugar para un bien almacenado. El almacén esta implementado como una pila (stack).

Otro cambio es que la impresión de los bienes producidos o consumidos ya no se hace desde los Productores o Consumidores, sino desde los mismos métodos sincronizados *get* y *put* asegurando que la impresión corresponde al orden en que se fueron efectuando el tomar y poner. Esta impresión desde estos métodos no es lo más aconsejable, dado que las E/S son muy tardadas, pero con tal de ver el funcionamiento correcto, recurriremos a esta solución, en este ejercicio.

Haga los siguiente incisos, comprobando y / o contestando a las preguntas.

Entre lo que sigue igual es la cantidad total de bienes que los productores producen y los consumidores consumen: 10 cada uno.

**Los siguientes ejercicios valen 4 puntos del total de la práctica.**

1. (\*) Ejecute el programa *ProCon* en diferentes corridas para 1 y 30 lugares de almacenamiento. Nota usted que siempre que algún thread consumidor reporta un *bien* es por que la impresión del thread productor ya apareció.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (\*) Los métodos sincronizados *get* y *put* cada vez que se invocan ¿cuántos bienes toman o ponen?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (\*) Ejecute el programa *ProCon* en diferentes corridas para 1, 3, 20 y 30 lugares de almacenamiento.

¿En cuales de estas corridas seguro el almacén se llenará durante la corrida, en cuáles corridas puede que el almacén llegue a llenarse durante la corrida y en cuáles corridas jamás se llenará el almacén durante la corrida?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (\*) Modifique en el programa del Consumidor la instrucción libHilos.hacerTiem(1,1000); para que cada consumidor tarde más tiempo entre que toma y toma; p.e. libHilos.hacerTiem(8,1000); .

Ejecute *ProCon* para 20 lugares de almacenamiento. ¿Que diferencias nota?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Después de la prueba regrese a libHilos.hacerTiem(1,1000).

1. (\*) Lea el archivo *Monitors.pptx* (en modo notas de página).

Cuál(es) estructura(s), de la aplicación *Productores-Consumidores*, corresponde(n) a una estructura *monitor*.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

¿Qué clase implementa el almacén?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. (\*) ¿Cuál es la utilidad de los métodos de instancia *vacio* y *lleno* en la clase Pila? ¿Para qué se usan?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**El siguiente ejercicio vale 6 puntos del total de la práctica.**

1. (\*) Modifique el ejercicio de Productores-Consumidores para que ahora, además de sillas también se producirán mesas y sofás. Esto implicará tres áreas de almacenamiento (madrigueras –monitores-), una por cada tipo de bien. Así habrá un monitor para las *sillas*, otro monitor para las *mesas* y finalmente un tercer monitor para los *sofás*.

Por cada tipo de bien se producirá y almacenará vía su propio monitor-madriguera, entre todos los productores correspondientes, un máximo de 20 unidades, que a su vez los respectivo consumidores tomarán y consumirán.

Ahora habrá tres productores por cada tipo de bien. Un productor producirá 8 unidades, otro, 5 unidades y el último 7 (8+5+7=20). Entonces para producir todos los bienes arrancarán un total de nueve productores. Cada productor deberá indicar el momento en que termina. Mantenga el mismo esquema de nombre de cada thread productor que ya se tenía.

En cuanto a los consumidores habrá cuatro por cada tipo de bien. Un consumidor tomará 4 unidades, otros dos consumidores 5 unidades y el último 6 unidades (4+5+5+6=20). Habrá un total de doce threads consumidores. Cada consumidor deberá indicar el momento en que termina.

Como siempre el thread *main( )* deberá terminar después de todos los threads.

¿Bajo cuáles condiciones este programa concurrente podría quedarse bloqueado?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

No se olvide de subir la práctica reportada con los programas java, en un zip.