



GRADO EN INGENIERÍA INFORMÁTICA

PROGRAMACIÓN CONCURRENTES Y TIEMPO REAL

DEPARTAMENTO DE INGENIERÍA INFORMÁTICA

Práctica 7

Autor:

Pablo Velicias Barquín

Fecha:

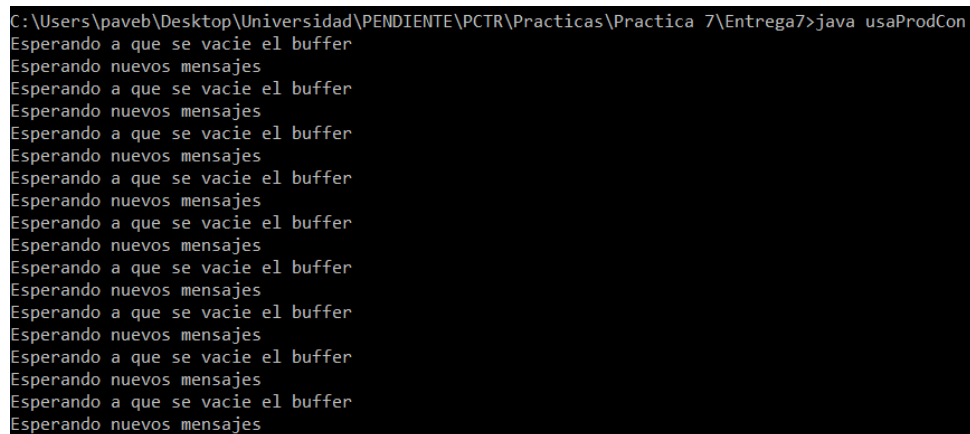
03 de Diciembre de 2021

1. Ejercicio Producto-Consumidor.

Para la realización de este ejercicio, he utilizado **PCMonitor.java** y he editado el código dentro de este .java.

1.1. Reduzca el tamaño del buffer a uno y vea qué ocurre

Si reducimos el buffer y ejecutamos el código, podemos ver como de primeras el código termina sin ningún problema la ejecución. Sin embargo, si añadimos código para imprimir por pantalla a medida que se ejecuta el código, obtenemos los siguientes resultados:



```
C:\Users\paveb\Desktop\Universidad\PENDIENTE\PCTR\Practicas\Practica 7\Entrega7>java usaProdCon
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
Esperando a que se vacie el buffer
Esperando nuevos mensajes
```

Figura 1: Impresión por pantalla durante la ejecución

Como podemos ver, las hebras entran al wait-set mediante **wait()** y salen de ella continuamente debido al **notifyall()**. Es un poco ineficiente en temas de ejecución.

1.2. Ejecute un productor y varios consumidores y vea qué ocurre.

En este caso, si ejecutamos un productor y dos consumidores, cada uno solamente 1 iteración, y con el tamaño de buffer de $N=1$, podemos ver como el que esta como productor termina de ejecutar su parte. Sin embargo, uno de los consumidores se queda a la espera de que otro productor realice el envío, ya que el buffer está vacío y no puede realizar la acción. El programa se queda en estado muerto.



```
C:\Users\paveb\Desktop\Universidad\PENDIENTE\PCTR\Practicas\Practica 7\Entrega7>java usaProdCon
Esperando nuevos mensajes
```

Figura 2: Resultado 1 productor y 2 consumidores

1.3. Ejecute varios productores y un consumidor y vea qué ocurre.

Si lo hacemos al revés, ejecutando varios productores y un consumidor, cada uno solamente 1 iteración, y con el tamaño de buffer de $N=1$, podemos ver como esta vez el consumidor hace su trabajo, pero un productor espera a que el buffer tenga hueco para enviar, es decir, esta esperando a otro consumidor, por lo tanto el programa se queda en estado muerto.



```
C:\Users\paveb\Desktop\Universidad\PENDIENTE\PCTR\Practicas\Practica 7\Entrega7>java usaProdCon
Esperando a que se vacie el buffer
```

Figura 3: Resultado 2 productores y 1 consumidor