# *Semana 7 Productor/Consumidor*

## P4

En la siguiente entrega se plantea el clásico problema de Productor/Consumidor. Para ello creamos 2 hilos con dichos nombres y un buffer en el que intentan leer (reducir la lista) y escribir (aumentar la lista) garantizando que no se escribe cuando el tamaño de la lista esté por encima de un numero fijado (en mi caso la capacidad es 10) ni se lee por debajo de 0.

Se ha impleméntalo la lectura y escritura como una LinkedList que simula una cola (FIFO).

Se adjunta una muestra del resultado obtenido:

Consumidor is reading, Product size is: 1

Productor is writting, Product size is: 2

Consumidor is reading, Product size is: 1

Productor is writting, Product size is: 2

Consumidor is reading, Product size is: 1

Productor is writting, Product size is: 2

Productor is writting, Product size is: 3

Consumidor is reading, Product size is: 2

Consumidor is reading, Product size is: 1

Productor is writting, Product size is: 2

Consumidor is reading, Product size is: 1

Productor is writting, Product size is: 2

Productor is writting, Product size is: 3

Consumidor is reading, Product size is: 2

Consumidor is reading, Product size is: 1

Productor is writting, Product size is: 2

Aquí se puede ver como aleatoriamente el productor y consumidor añaden y eliminan objetos del buffer. La aleatoriedad de las ejecuciones depende de una pausa que se hace durante un tiempo aleatorio.

El deadlock (también llamado bloqueo mutuo o abrazo mortal) es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos. ​

En esta práctica se podría generar un deadlock si tenemos 2 monitores A y B en los que hay sincronizaciones anidadas, donde un objeto B, que es el encargado de hacer .notify() a A, está esperando al objeto A. En este caso se queda bloqueado y nunca despertará.