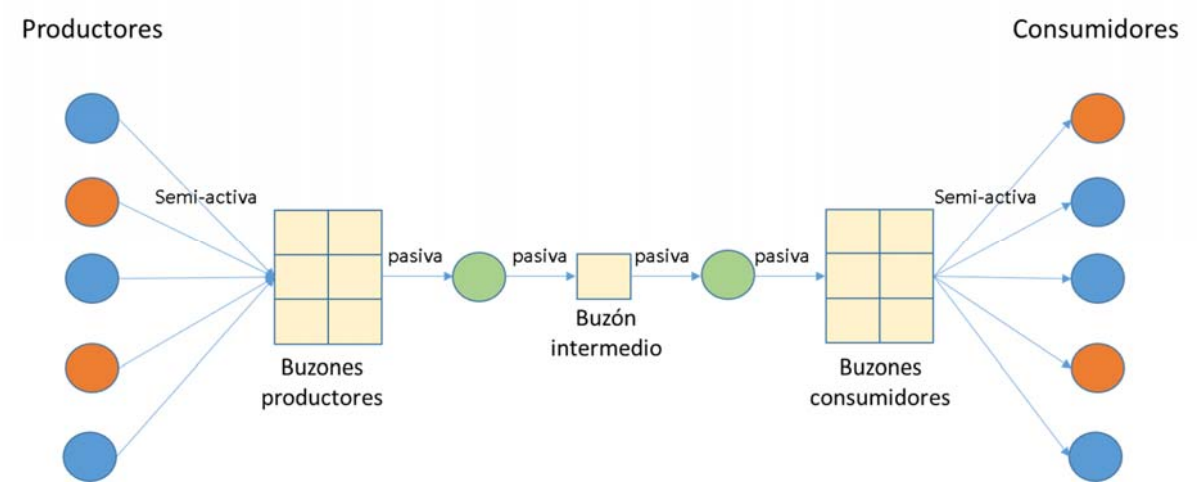
Universidad de los Andes
Departamento de Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional

Caso 1
Manejo de la Concurrencia

Queremos implementar una arquitectura de productores y consumidores como la que muestra la figura.:

- Hay un número determinado de productores que generan sus productos y los dejan en los buzones a su derecha. Un productor que logra almacenar su producto, continúa produciendo hasta que termina de generar y almacenar todos sus MAX_PRODUCTOS productos. Si no puede almacenar porque no hay espacio en los buzones, libera el procesador y lo vuelve a intentar más tarde de acuerdo con las políticas del sistema operativo (espera semi-activa: yield).
- Hay el mismo número de consumidores que de productores que consumen sus productos de los buzones a su izquierda (no hacen nada con ellos en este ejemplo). Un consumidor busca productos de un tipo determinado y si lograr retirarlo, continúa para consumir el siguiente producto hasta que termina de consumir todos sus MAX_PRODUCTOS productos. Si no puede retirar porque no hay productos para consumir (del tipo que espera), libera el procesador y lo vuelve a intentar más tarde de acuerdo con las políticas del sistema operativo (espera semi-activa: yield).
- Hay dos intermediarios (color verde en la figura): Uno que retira los productos de los buzones de los productores y lo almacena en el buzón (capacidad de un solo producto) de comunicaciones con el otro. Tanto el retiro de los buzones de los productores como el almacenamiento en el buzón intermedio lo hace con espera pasiva (wait)
- El segundo intermediario retira un producto del buzón intermedio y lo almacena en los buzones de los consumidores. Tanto el retiro de del buzón intermedio como el almacenamiento en los buzones de los consumidores lo hace con espera pasiva (wait).



Objetivo

Diseñar un mecanismo de comunicación para implementar la arquitectura descrita. Para este caso, los productores, los intermediarios y los servidores serán *threads* en la misma máquina (en realidad debería ser un sistema distribuido; este es solo un prototipo).

El proyecto debe ser realizado en java, usando *threads*. Para la sincronización solo pueden usar directamente las funcionalidades básicas de Java: `synchronized, wait, notify, notifyAll, yield` y `join` (o las `CyclicBarrier`).

Funcionamiento

Cada *thread* productor genera un producto, lo deposita y continúa. Todos los productores generan el mismo número de productos. Una vez un productor ha depositado todos sus productos, termina. El número de productores y el número de productos a generar deben ser parámetros de configuración.

Un thread consumidor, por su parte, retira un producto y continúa. El número de consumidores es el mismo de productores y cada consumidor consume el mismo número de productos que produce un productor. Una vez un consumidor ha retirado todos sus productos, termina.

Los threads intermedios deben ser capaces de mover todos los productos generados por todos los productores y terminar una vez lo hayan hecho.

Los productos son de dos tipos: A y B. Su solución debe implementar que la mitad de los productores produce solo productos tipo A y la otra mitad produce solo productos tipo B (puede suponer un número par de threads). De manera simétrica, la mitad de los consumidores consume solo productos tipo A y la otra mitad consume solo productos tipo B. Un consumidor que intenta retirar un producto y encuentra que en sus buzones no hay del tipo que él consume, implementa su espera semi-activa. Los intermediarios mueven productos indistintamente de su tipo.

El número de productores/consumidores, los tamaños de los buzones de productores y de consumidores (que pueden ser diferentes) y el número de productos a producir o consumir por cada thread productor/consumidor deben estar en un archivo, el cual debe ser procesado por el `main` del programa.

Se debe definir la manera de evidenciar que el programa funciona correctamente. La salida debe ser clara para que se puede validar fácilmente la correcta operación del sistema.

La configuración para la ejecución debe estar almacenada en un archivo de propiedades. Un ejemplo de dicho archivo sería:

```
concurrencia.numProdCons = 4
concurrencia.numProductos = 7
concurrencia.buzonesProd = 3
concurrencia.buzonesCons = 8
```

Condiciones de entrega

- En un archivo .zip entregar el código fuente del programa, y un documento word explicando el diseño y funcionamiento del programa, así como la validación realizada. En particular, para cada pareja de objetos que interactúan, explique cómo se realiza la sincronización, así como el funcionamiento global del sistema. El nombre del archivo debe ser: caso1_*login1*_*login2*.zip
- El trabajo se realiza en grupos de máximo **2** personas de la misma sección. No debe haber consultas entre grupos.
- El grupo responde solidariamente por el contenido de todo el trabajo, y lo elabora conjuntamente (no es trabajo en grupo repartirse puntos o trabajos diferentes).
- Se puede solicitar una sustentación a cualquier miembro del grupo sobre cualquier parte del trabajo. Dicha sustentación puede afectar la nota de todos los miembros.

- El proyecto debe ser entregado por Sicua+ por uno solo de los integrantes del grupo. **Al comienzo del documento word, deben estar los nombres y carnés de los integrantes del grupo**. Si un integrante no aparece en el documento entregado, el grupo podrá informarlo posteriormente, sin embargo, habrá una penalización: la calificación asignada será distribuida (dividida de forma equitativa) entre los integrantes del grupo.
- **Se debe entregar por Sicua+ a más tardar el martes 23 de febrero a las 23:55 (p.m.)**