Caso 1

Angela Liliana Jiménez Flechas 201912941

Miguel Ángel Acosta Walteros 201914976

**Funcionamiento Global del Sistema**

El sistema consta de tres tipos de threads lanzados desde la clase principal “Mercado”:

* Threads Productores: Son threads encargados de depositar n productos en el buzón de productores llamado ColaEsperaConsumidores. Cada uno de estos threads termina su ejecución cuando termina de depositar n productos.
* Threads Intermedios (ThreadIntermediario y ThreadIntermediario2) : Son 2 threads que van a estar encargados de mover uno a uno los productos (n productos por m productores) entre buzones. Uno que va a sacar productos del buzón de productores y los va a poner en el buzón intermedio y otro que va a sacar productos del buzón intermedio y los va a poner en el buzón de consumidores, terminan su ejecución cuando los n productos por m productores terminan de transportarse.
* Threads Consumidores: Son threads encargados de recoger n productos del buzón o arreglo de consumidores llamado ColaEsperaProductores. Cada uno de estos threads termina su ejecución cuando termina de recoger n productos de su tipo.

Estos buzones van a ser herramientas de almacenamiento:

* ColaEsperaIntermedia es un un objeto de tipo Buzón que permite regular el movimiento de un buzón de productores a un buzón de consumidores. Este movimiento está siendo realizado por los threads intermediarios y es gracias a este buzón que se logra sincronizar y regular la retirada y sacada de productos de forma tal que se genera un efecto parecido al semáforo binario, aplicando el bloqueo mutuo.
* ColaEsperaConsumidores y ColaEsperaProductores son listas de k Productos.

Así mismo los Productos son un Objeto con un único atributo que es del tipo de producto (String) que puede ser de “Tipo\_A” o de “Tipo\_B”.

Inicialmente todos los threads son creados y lanzados desde la clase mercado y en las ejecuciones de estos threads van a acceder a los buzones e ir tratando de mover los productos hasta que todos lleguen a su destino y sean consumidos.

**Sincronización Productores - Intermediario 1**

Los threads productores pondrán productos en una lista llamada ColaEsperaProductores y el thread intermediario 1 sacará productos de ahí, para esto los threads harán lo siguiente:

Los threads productores realizarán el siguiente proceso sobre el buzón de productores:

1. Revisan si siguen teniendo productos para producir.
2. Revisan si el buzón (en este caso nuestra lista ColaEsperaProductores) está ocupado por otro thread, si no lo está lo ocupamos nosotros (para esto usamos synchronized sobre la lista).
3. El thread que ocupe el buzón pregunta si no excede el tamaño del buzón, en caso de excederlo el thread activa yield y abandona la producción, en caso de no excederlo agrega un producto, queda a la espera de ingresar cuando el buzón tenga espacio.
4. Finalmente cuando un thread no tenga más productos para producir terminará su ejecución.

Por otro lado el thread Intermediario 1 hará lo siguiente sobre el buzón de productores:

1. Verifica que la lista de productores tenga algo adentro y que tiene espacio para ingresar un elemento a la lista intermedia.
2. Sacará un producto del buzón si cumple con los requisitos de arriba.

**Sincronización Intermediario 1 – Intermediario 2**

Para la sincronización entre intermediario 1 y 2 se usó la clase Buzon, la cual cuenta con diferentes elementos que nos permiten coordinar los wait y notify para llevar una espera pasiva. En la clase Buzon se manejan dos objetos para regular el movimiento de los productos: salida y entrada.

El proceso que lleva el Intermediario 1 es el siguiente:

1. Una vez tiene el producto y lo va a depositar al buzón intermediario, ingresa al método Buzon.meter(Producto x) el cual se sincroniza con el objeto de entrada y mientras la lista no esté en su tamaño máximo (y no quepa más) se hará entrada.wait y se irá a esperar activamente.
2. Si pasó de ahí es porque se puede agregar, por lo que se hace un .add a la lista y se le notifica a la salida que ha ingresado un producto (por lo que se hace un salida.notify() ).

El proceso que lleva a cabo el intermediario 2 es el siguiente:

1. Se sincroniza con la lista para asegurarse que solo 1 thread haga este proceso a la vez.
2. Verifica que hay algo en la lista intermedia en la que el Intermediario 1 deposito su producto así como que la cola de consumidores no haya superado su máximo.
3. En caso tal de cumplir las condiciones, remueve el producto de la lista intermediaria y lo agrega a la lista. Este proceso está sincronizado en el objeto de salida de la clase Buzón mencionado anteriormente, por lo que al ser notificado verifica la condición de que la lista del intermedio no sea 0, en caso tal de que lo sea, esperará, pero si no es el caso, se retirará sin problema y se le notificará al objeto de entrada que ya puede ingresar. Por lo que permite el acceso al siguiente producto a la lista intermedia.
4. Se incrementa el contador de productos que pasaron por el intermediario 2 y se imprime el proceso.

**Sincronización Intermediario 2 – Consumidores**

El thread intermediario 2 pone productos en una lista llamada ColaEsperaConsumidores y los threads consumidores sacan productos de ahí, para esto los threads harán lo siguiente:

El Thread Intermediario 2 hace lo siguiente sobre el buzón de consumidores:

1. Mete un producto al buzón si este tiene espacio para el producto.

Los threads consumidores realizarán el siguiente proceso sobre el buzón de consumidores:

1. Revisan si el Buzón (en este caso nuestra lista) está ocupado por otro thread, si no lo está lo ocupamos nosotros (para esto usamos synchronized sobre el objeto y de esta forma evitamos que diferentes consumidores intenten sacar el mismo producto).
2. Revisan si el buzón de consumidores tiene algún producto en él
3. Como el consumidor solo consumirá productos de su mismo tipo, este revisa si el buzón tiene algún producto que sea del mismo tipo que él, si hay algún producto del mismo tipo, entonces lo consume (lo remueve del arreglo y aumenta su contador en uno), en caso contrario activa el yield.
4. Finalmente cuando no tenga más productos para consumir de su tipo, terminará su ejecución.

**Modelo Conceptual**

