Juan Manuel Perez – 202021827

Daniela Camacho – 202110975

Giovanny Parra - 202014668

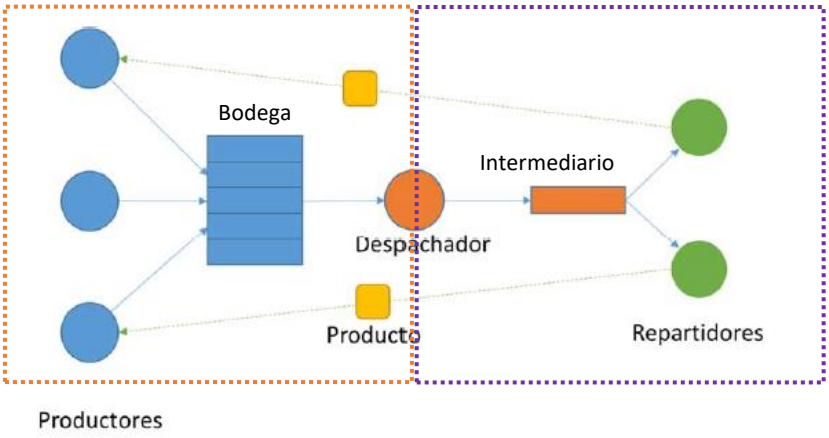
Caso 1

Documento de Informe

Arquitectura de la operación

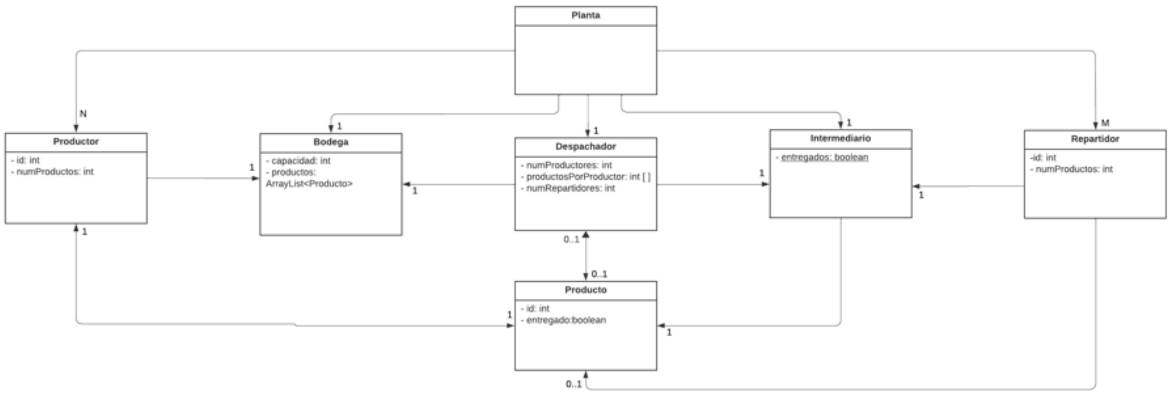
Como primer análisis se miró el funcionamiento de la planta de producción descrito en la Ilustracion1. En este momento se observó que la arquitectura que mejor se ajustaba a la planta de producción era un **Productor – Consumidor** anidado. El primer patrón de **Productor – Consumidor** está en la línea naranja entre los productores, la bodega y el despachador. En este caso, la bodega debe tener una capacidad *TAM* que es un parámetro que ingresa el usuario, los productores toman el papel de *Productor* pues son estas las entidades que crean y añaden productos a la bodega. Mientras que el despachador toma el papel de *consumidor* y es el que saca los productos de la bodega. El segundo patrón de **Productor – Consumidor** es visible en la línea morada entre el despachador, el intermediario y los repartidores. En este otro caso, el despachador cumple con el rol de *Productor* al añadir productos en el intermediario que tiene una capacidad de 1 producto. Para los repartidores su rol es el de sacar productos del intermediario para repartirlos.

Ilustración 1. Diagrama de funcionamiento planta producción



Después de lo anterior se diseñó el siguiente diagrama de clases UML para representar las clases y sus atributos necesarias para la solución del problema. El diagrama se observa en la Ilustración 2, la raíz del diagrama es la *Planta* que tiene N productores, una bodega, un despachador, un intermediario y M repartidores. El *Productor* tiene un id, el número de productos que debe producir (*numProductos*), debe conocer la bodega y el producto que creo actualmente. La *Bodega* tiene una capacidad y una lista con los productos que se encuentran actualmente (*productos*). El *Despachador* conoce el número de productores (*numProductores*), la cantidad de productos que produce cada productor (*productosPorProductor)*, el número de repartidores (*numRepartidores*), la bodega, el intermediario y el producto que tiene actualmente. El *Intermediario* tiene un producto actual y un indicador si el despachador ya no tiene más productos para despachar en la ejecución (*entregados*). El *Repartidor tiene* un id, un contador de los números de productos que ha entregado (*numProductos*) y el producto que tiene actualmente.

Ilustración 2. Diagrama UML planta de producción



Ejecución de la solución

- *Planta*

La planta es el lugar en donde está el método *Main( )* de la solución propuesta. Acá se crean los buffers de la Bodega y del Intermediario. También, se le pide al usuario los parámetros para conocer el número de productores (*N*), numero de repartidores (***M*)**, capacidad de la Bodega (*TAM*) y el número de productos que se deben producir (*C*). En este método calcula el número de productos que debe producir cada productor dependiendo de los valores que ingreso el usuario y además se crean los *N* Threads de los productores, el Thread del despachador y los *M* Threads de los repartidores.

- *Productor*

Es necesario que esta clase extienda la clase Thread. El constructor del productor tiene los parámetros clásicos del patrón **Productor – Consumidor**, pues este tiene un id del productor, la bodega y el número de productos que debe producir. El productor tiene un método de creación de productos (*crearProducto(i)*). Mientras que la sobreescritura del método *Run ( )* inicia con un ciclo el cual finaliza cuando el productor produzca el número de productos que debe producir, para cada producto creado se busca almacenar el producto llamando el método almacenar de la Bodega (*BODEGA.almacenar( )*) si este método es no es exitoso el productor se duerme en la bodega y espera a que haya espacio, mientras que si el método es exitoso el productor se duerme en el producto llamando el método del producto (*producto.dormirSobreProducto( )*) esperando hasta que el repartidor lo entregue y lo despierte para terminar la ejecución o crear otro producto.

- Bodega

La bodega actúa como un monitor para controlar el almacenamiento y despachamiento de los productos. El constructor de la bodega tiene como parámetro la capacidad que va a tener. Dado el patrón de **Productor – Consumidor** el primer método sincronizado es el de almacenar exclusivamente para los productores (*almacenar(Producto)*), este método revisa activamente la capacidad de la bodega y si no está llena almacena correctamente el producto que entra por parámetro, en el caso que este llena se duerme el productor con un *wait( )*. El otro método que debe tener la bodega según el patrón es el de retirar productos exclusivamente para los despachadores (*retirar( )*). Sin embargo, para este problema

no era necesario hacer la verificación de que la bodega no estuviera desocupada en este método, pues de esto se va a encargar despachador mediante su espera activa esto quiere decir que los despachadores no se van a dormir mediate *wait( )* y siempre va a haber por lo menos un producto cuando se ejecute, por lo que saca el producto de la bodega y despierta a todos los productores para avisar que hay espacio libre.

- Despachador:

El despachador extiende la clase de thread. Este cumple el rol de consumidor en el patrón de **Productor – Consumidor**, por lo que cuenta con los siguientes atributos: La bodega, la cual es compartida con el productor; el número de productores y los productos por productor, estos dos atributos son usados dentro del método *run()* con el fin de realizar un ciclo el cual pare en el momento que todos los productos ya hayan sido despachados; el producto a despachar, el cual se puede encontrar en *null* lo que quiere decir que no tiene un producto a despachar, si tiene un objeto de tipo *Producto* quiere decir que este es el que se encuentra en el proceso de despacho; el intermediario que es el segundo buffer de la aplicación donde se realiza el segundo patrón de **Productor – Consumidor**. El método *run ()* se sobrescribió de tal manera que el despachador siga corriendo hasta que la totalidad de los productos haya sido despachada, dentro del ciclo se llama a la función *despachar (),* dentro de este método se verifica la ocupación de la bodega, en caso de ser 0 el despachador entre en espera activa hasta que algún producto se encuentre en la bodega, luego de esto llama a la función del intermediario *recibirProducto (productoADespachar())* y luego de esto asigna el producto a despachar como *null.*

- Intermediario:

Como se mencionó previamente, estamos tratando con un buffer de tamaño 1 que es compartido entre el despachador y el repartidor. Sus funciones, *recibirProducto()* y *darProducto()*, están sincronizadas. La primera se encarga de recibir productos del despachador, mientras que la segunda se encarga de entregárselos al repartidor.

Dado que solo hay un Thread de ejecución para el despachador, en la función *recibirProducto(),* no es necesario verificar si el intermediario está ocupado, ya que el despachador espera hasta que el producto haya sido entregado antes de continuar. Por otro lado, en la función *darProducto(),* se verifica si hay un producto en el buffer para repartir. Además, el intermediario conoce si todos los productos ya han sido entregados a través de una variable booleana, que es modificada por el despachador cuando ha terminado de entregar todos los productos al intermediario mediante la función *setEntregados()*. El intermediario es responsable de notificar a los repartidores si pueden finalizar su ejecución a través de esta variable y la función *getEntregados()*. Estas dos últimas funciones también están sincronizadas debido a la variable compartida llamada entregados.

- Repartidor:

El repartidor desempeña el papel de consumidor en este patrón. Conoce al intermediario del cual recogerá los productos a repartir, y estos productos se almacenan en la variable *productoARepartir*. Si esta variable es *null,* significa que el repartidor no tiene ningún producto que repartir. Además, el repartidor lleva un contador que registra la cantidad de productos entregados por él.

El método run ha sido sobrescrito y contiene un ciclo que verifica la condición de la variable entregados del intermediario a través del método *getEntregados()*. Este ciclo se ejecuta mientras esta variable sea *false*, lo que indica que aún hay productos por entregar. Dentro de este ciclo, se llama a la función

repartir(), que se encarga de recoger el producto del intermediario y notificar al usuario sobre su entrega.

- Producto:

El producto debe actuar como monitor pues acá es donde se duermen los productores luego de entregar el producto en la bodega. Es por esto que el método constructor recibe como parámetro el productor y el id del producto, además el atributo entregado empieza en falso. El método de dormir el producto sobre el producto (*dormirSobreProducto( )*) es sincronizado y duerme pasivamente con un *wait( )* el thread del productor cuando se hace el llamado, para cumplir el requerimiento que el productor no produzca más productos hasta que el repartidor entregue su producto. Asimismo, hay un método sincronizado exclusivamente para los repartidores para despertar el thread de los productores cuando el producto se entregó (*setEntregado( )*) mediante un *notify( )* y cambia el atributo a entregado como verdadero.

Caso de prueba

A partir de las siguientes entradas del programa:

```
Ingrese el número de productores:
2
Ingrese el número de repartidores:
2
Ingrese la capacidad de la bodega:
1
Ingrese el número de productos:
5
```

Se recibe la siguiente salida:

```
Despachador en espera activa, no hay nada en la bodega por lo que sumo 5 + 5  = 10
Soy el productor: 0 y produzco: 3
Soy el productor: 1 y produzco: 2
```

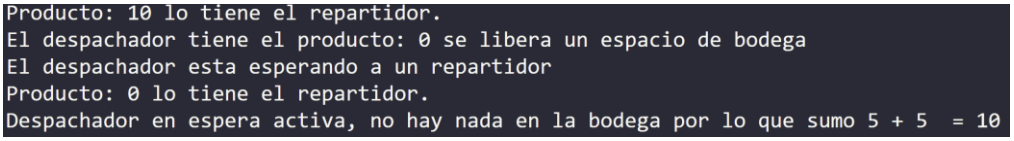
El despachador se encuentra en espera activa ya que no hay ningún producto almacenado en la bodega. Mientras que a los productores se les asigno la cantidad de productos a producir.

```
El productor: 1 creo el producto 10
El productor: 0 creo el producto 0
Se almaceno el producto: 10
El productor: 1 se durmio sobre el producto: 10
No hay espacio, el productor: 0 se duerme sobre la bodega
```

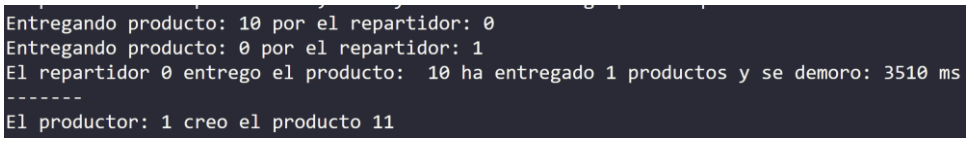
El productor 1 y 0 crearon el producto 10 y 0 respectivamente. El producto 10 se almacena en la bodega y su productor queda dormido sobre él. Debido a que la bodega es de tamaño 1 el productor 0 queda en espera.

```
El despachador tiene el producto: 10 se libera un espacio de bodega
El despachador esta esperando a un repartidor
Se almaceno el producto: 0
El productor: 0 se durmio sobre el producto: 0
```

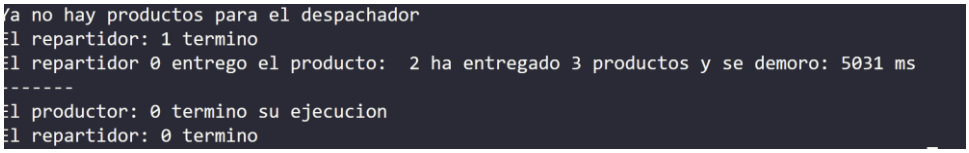
Se despacha el producto 10 y el despachador queda en espera de un repartidor. Como se liberó un espacio en bodega el producto 0 se almacena en esta y el producto queda dormido sobre el hasta que este sea entregado.



El producto 10 lo tiene un repartidor por lo que el despachador se despierta y despacha al producto 0 de la bodega y queda en espera de un repartidor. Un repartidor tiene el producto 0. No se encuentra ningún producto en la bodega por lo que el despachador entra en espera activa.



Los dos productos son entregados. Por lo que los productores se despiertan y continúan con la producción de los productos, los cuales tienen el mismo proceso ya descrito.



Después de realizar todo el proceso de entrega de los procesos, el despachador avisa que se entregó la totalidad de los productos y tanto los productores como repartidores terminan su ejecución después de finalizar la entrega de los productos.