Juan Esteban Jiménez Benavides (201922487)

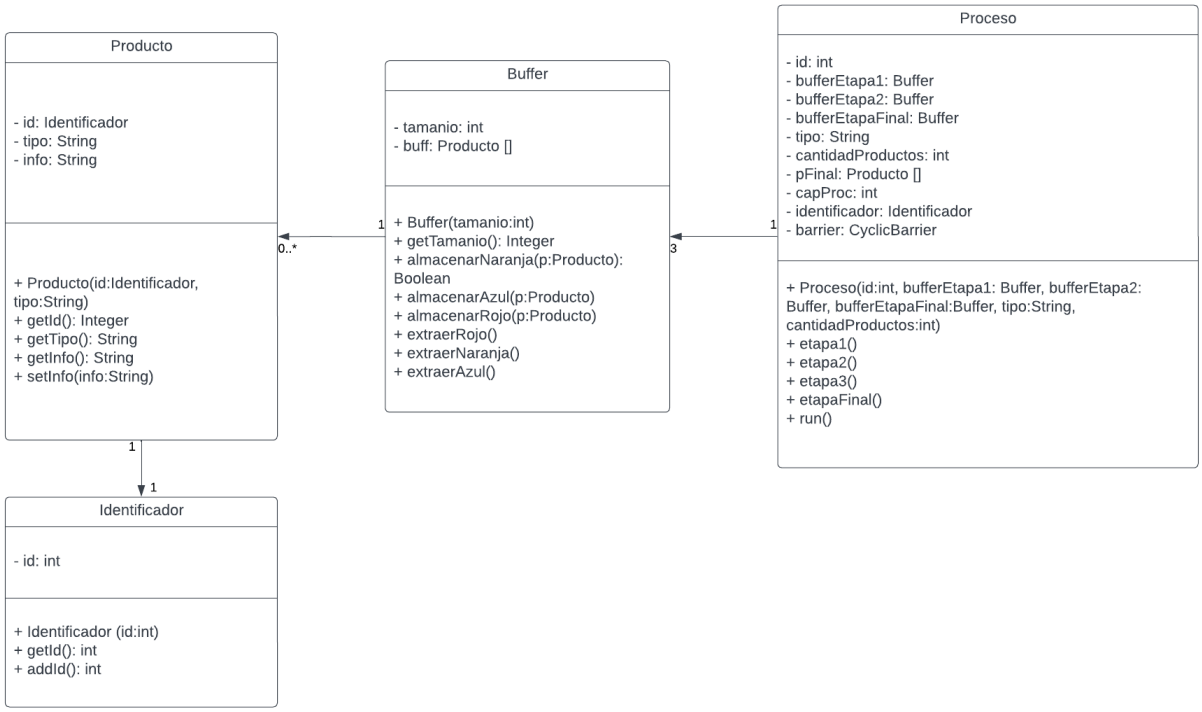
Nicolás Diaz Montaña (202021006)

Thais Tamaio Ramírez (202022213)

Infraestructura computacional Caso 1

Manejo de la Concurrencia

- ***Diagrama de clase:***



/

- ***Funcionamiento del programa:***
    - Nuestro programa utiliza diferentes clases y objetos para modelar un proceso productivo de tres etapas que involucra la producción y transformación de productos de tres tipos diferentes: **naranja, azul y rojo**.

- El programa tiene una clase principal llamada App qué instancia varios objetos de la clase Buffer, que se utilizan como "**contenedores"** para el tránsito de los productos en diferentes etapas del proceso de producción. También crea un objeto de la clase **CyclicBarrier**, con el fin de sincronizar los hilos que representan los diferentes procesos de producción, para que todos los productos lleguen a la etapa final antes de salir a "distribución"/imprimir.
- La clase **Buffer** tiene varios métodos que permiten almacenar y extraer productos de diferentes tipos, y que se utilizan para modelar las diferentes etapas del proceso productivo. Cada **objeto Buffer** tiene una lista de productos y un tamaño máximo, y los métodos que se encargan de almacenar productos (**naranja, azul o rojo**) verifican si hay espacio disponible en la lista antes de agregar un nuevo producto.
- La **clase Identificador se** utiliza para asignar un **identificador** (ID) único a cada producto que se produce en el proceso(también con la finalidad de identificarlos de manera más eficiente para fines de análisis), y se utiliza en la creación de objetos Producto en la clase **Proceso**.
- La **clase Proceso** representa un hilo que ejecuta una de las etapas del proceso productivo. Cada hilo tiene acceso a **tres objetos Buffer** (para almacenar productos en diferentes etapas del proceso), un tipo de producto (naranja, azul o rojo), una cantidad de productos a producir y un objeto **CyclicBarrier**. Cada hilo ejecuta un ciclo que produce una cantidad fija de productos de su tipo, utilizando los objetos **Buffer** correspondientes para almacenar los productos en diferentes etapas del proceso.
- Al final de cada ciclo, el hilo espera en el objeto **CyclicBarrier** para sincronizarse con los demás hilos, antes de continuar con la etapa final. Cuando se completa la última etapa del proceso (representada por el hilo que produce **productos rojos**), se termina el programa e imprime los productos.

- ***Validación realizada: Los siguientes son escenarios de pruebas.***
  - Tamaño **Buffer** 10 cantidad **procesos** 10 cantidad **productos** 10, resultado = 100 productos impresos.

- Tamaño **Buffer** 3 cantidad **procesos** 20 cantidad **productos** 15, resultado = 60 productos impresos.
- Tamaño **Buffer** 20 cantidad **procesos** 4 cantidad **productos** 4, resultado = 80 productos impresos.
- Tamaño **Buffer** 50 cantidad **procesos** 50 cantidad **productos** 15, resultado = 2500 productos impresos.
- Tamaño **Buffer** 3 cantidad **procesos** 3 cantidad **productos** 2, resultado = 9 productos impresos.
- Tamaño **Buffer** 300 cantidad **procesos** 300 cantidad **productos** 80, resultado = 90.000 productos impresos.

- ***Sincronización Pareja de objetos:***
  - En nuestro código se utilizan varios mecanismos de **sincronización** para asegurar que los diferentes hilos que se ejecutan en paralelo puedan acceder a los recursos compartidos de manera segura.
  - La **clase Buffer** es utilizada para implementar un **buffer compartido** entre los diferentes hilos que se ejecutan en paralelo. Para asegurar que los diferentes hilos no accedan al buffer al mismo tiempo, los métodos que manipulan el **buffer** (almacenar y extraer productos) están definidos como **sincronizados**. Además, se utiliza el mecanismo de **espera-pausa** (**wait-notify**) para asegurar que los hilos que intentan extraer productos del buffer esperen cuando el **buffer** está vacío, y se les notifique cuando se agregue un nuevo producto al **buffer**.
  - La **clase Identificador** es utilizada para generar identificadores únicos para cada uno de los productos que se procesan en el sistema. Para asegurar que los diferentes hilos no accedan al identificador al mismo tiempo, el método getId() está definido como sincronizado.
  - La clase **Proceso** es la que implementa el procesamiento de los productos. Para asegurar que los diferentes hilos no accedan a los buffers de manera conflictiva, se utiliza el mecanismo **Synchronized del buffer**. Esta clase permite que los diferentes hilos se sincronicen al interactuar con el **Buffer**. En este caso, se utiliza

una barrera cíclica para asegurar que todos los hilos hayan completado de la etapa 1 a la 3 antes de avanzar a la etapa final.

- En general, se utilizan diferentes mecanismos de sincronización para asegurar que los diferentes hilos que se ejecutan en paralelo puedan acceder a los recursos compartidos de manera segura y sin conflictos.

- ***Funcionamiento global del sistema:***
  - En primer lugar, durante la etapa 1 se crea la cantidad deseada de productos, a cada producto se le asigna un ID para su identificación y se almacenan en el **Buffer** respectivo.
  - Luego los productos avanzan a la 2 y 3 etapa donde se "transforman" para lograr el producto final, al final de la etapa 3 se extraen para ser llevados al **Buffer Rojo** para marcar el fin del proceso y por último imprimir los productos que se obtuvieron 1 a 1.