**1.Introducción**

Este proyecto consiste en el desarrollo de una aplicación Java que implementa el clásico problema de productor-consumidor. En este contexto, se crean múltiples hilos que actúan como productores y consumidores, los cuales interactúan con un buffer común. Los productores generan datos o ítems y los almacenan en el buffer, mientras que los consumidores extraen y procesan estos ítems. El desafío principal radica en gestionar el acceso concurrente al buffer de manera eficiente y segura, evitando condiciones de carrera y asegurando una sincronización adecuada entre productores y consumidores.

**2.Objetivos del Proyecto**

Los objetivos principales del proyecto son:

1. Desarrollar un entendimiento práctico del manejo de hilos y la sincronización en Java.

2. Implementar una solución eficaz para el problema de productor-consumidor, utilizando mecanismos de concurrencia como `wait()` y `notifyAll()`.

3. Garantizar que la aplicación funcione correctamente bajo condiciones de concurrencia, sin bloqueos innecesarios ni condiciones de carrera.

4. Explorar el uso de herramientas de desarrollo como NetBeans y Maven para la gestión y construcción del proyecto.

**3.Importancia o Relevancia del Proyecto**

El proyecto tiene una importancia significativa tanto desde un punto de vista educativo como práctico. Desde la perspectiva educativa, ofrece una oportunidad valiosa para comprender los desafíos y soluciones relacionados con la programación concurrente, un aspecto fundamental en el desarrollo de software moderno. Desde una perspectiva práctica, proporciona experiencia en la implementación de patrones de diseño concurrente que son ampliamente aplicables en el desarrollo de sistemas de tiempo real, sistemas operativos, aplicaciones de servidor y en el procesamiento de grandes volúmenes de datos. Por lo tanto, este proyecto no solo fortalece las habilidades de programación en Java, sino que también prepara al desarrollador para enfrentar problemas de concurrencia reales en entornos de producción.

**4. Diseño del Sistema**

El sistema se basa en una arquitectura de productor-consumidor. Se diseñaron tres clases principales:

* **Productor:** Genera ítems y los agrega al buffer.
* **Consumidor:** Extrae ítems del buffer y los procesa.
* **Buffer:** Almacena los ítems producidos y maneja el acceso concurrente.

*Incluye diagramas de clases o secuencia aquí si es posible.*

**5. Implementación**

Se puso especial énfasis en la sincronización y la gestión de hilos. Se emplearon métodos **wait()** y **notifyAll()** para manejar el acceso al buffer. Un problema significativo fue la sincronización incorrecta del buffer, que se resolvió asegurando que las operaciones de poner y obtener estuvieran correctamente sincronizadas.

**6. Resultados y Pruebas**

El sistema se comporta de manera estable en ejecución, manteniendo un equilibrio entre los productores y consumidores. Se realizaron pruebas de carga para validar la funcionalidad, asegurando que no ocurrieran bloqueos o condiciones de carrera incluso bajo alta concurrencia.