Entregable 2 - Concurrencia

Una empresa de comercio electrónico procesa cientos de pedidos diariamente. Cada pedido pasa por tres etapas principales:

1. **Procesamiento de Pago**: Verificar y autorizar las transacciones de pago, lo cual requiere una serie de validaciones y comprobaciones.
2. **Empaquetado de Pedidos**: Preparar el pedido para el envío, lo que incluye la impresión de etiquetas y la preparación de los artículos.
3. **Envío**: Generar etiquetas de envío y asignar el pedido a un socio logístico.

El desafío para esta empresa es procesar múltiples pedidos de forma eficiente y en el menor tiempo posible, manejando las tres etapas de manera concurrente, para que varias tareas puedan ocurrir al mismo tiempo sin que ninguna de ellas bloquee a las demás. Además, los pedidos urgentes deben procesarse con prioridad.

Se debe diseñar e implementar un sistema que pueda manejar varios pedidos simultáneamente, optimizando el uso de los recursos (hilos) disponibles. Se deberá experimentar con diferentes estrategias para asignar hilos a las distintas tareas y justificar las decisiones en un documento breve.

Detalle de las etapas:

- Utilice la clase ExecutorService de Java para gestionar la ejecución concurrente de las tareas de Procesamiento de Pago, Empaquetado de Pedidos y Envío.
- Diseñe un *pool* de hilos (*thread pool*) que gestione múltiples pedidos de forma eficiente, evitando bloqueos innecesarios.

- La etapa de Empaquetado de Pedidos puede realizarse en paralelo cuando hay varios pedidos por procesar. Utilice *streams* paralelos o *ForkJoinPool* para optimizar esta etapa.
- Justifique qué estrategia de paralelización utilizó en caso de hacerlo.

- El sistema debe ser capaz de procesar los pedidos urgentes con prioridad. Implemente un mecanismo para asegurarse de que estos pedidos se procesan antes que los normales.
- El sistema debe seguir funcionando eficientemente para los pedidos normales mientras maneja los pedidos urgentes.

- Decida cuántos hilos asignar a cada etapa. Esta decisión debe basarse en la complejidad de cada tarea y la carga de trabajo.
- Experimente con diferentes configuraciones y analice el impacto en el rendimiento. Comente brevemente sobre las distintas configuraciones probadas.

Requisitos:

- Debe modelar la realidad de los pedidos para poder crear la lógica de negocio. Cumpliendo con los requisitos funcionales, se dispone de total libertad para diseñar el sistema.
- Implemente un mecanismo de cierre ordenado que asegure que todas las tareas se completen antes de detener los hilos del sistema.
- El sistema debe ser capaz de manejar al menos 100 pedidos concurrentes sin problemas de rendimiento.
- Se debe asegurar el acceso seguro a recursos compartidos.
- La solución debe ser modular y seguir las mejores prácticas de concurrencia en Java, incluyendo la gestión de excepciones y el uso de estructuras de datos seguras para hilos.
- Deben realizarse tests unitarios para todas las operaciones, teniendo un 80% de cobertura como mínimo.

Además de un *zip* con el código fuente, debe entregarse un documento de 1 a 2 páginas explicando las decisiones de diseño tomadas, la cantidad de hilos asignados a cada tarea, la implementación de priorización de pedidos urgentes y un análisis del rendimiento del sistema.

La entrega se realizará vía *moodle*, siendo la fecha límite de entrega el domingo 29 de septiembre a las 23:59.