JAVA CODE CHALLENGE

Crear una API REST que simule un sistema de procesamiento de pedidos de alta concurrencia. El sistema debe ser capaz de manejar múltiples solicitudes simultáneas de manera eficiente, evitando problemas comunes de concurrencia como condiciones de carrera, deadlocks y uso ineficiente de recursos.

Requisitos del Proyecto:

1. **Construcción de la API REST:**
    - Desarrollar una API REST utilizando Spring Boot.
    - La API debe tener un endpoint /processOrder que acepte solicitudes POST con un JSON que represente un pedido.
    - El pedido debe incluir información como el orderId, customerId, orderAmount y orderItems.
2. **Simulación de Procesamiento de Pedidos:**
    - Implementar una lógica que simule el procesamiento de pedidos. Esto puede incluir operaciones como validación de pedidos, cálculo de precios, y almacenamiento de los pedidos procesados en una estructura de datos (como una lista o un mapa).
    - El procesamiento de cada pedido debe simular un retraso aleatorio (por ejemplo, de 100 a 500 milisegundos) para representar operaciones de negocio complejas.
3. **Manejo de Concurrencia:**
    - El sistema debe ser capaz de manejar hasta 1000 solicitudes concurrentes sin pérdida de datos ni inconsistencias.
    - Implementar un mecanismo eficiente de gestión de hilos utilizando ExecutorService, ThreadPoolExecutor, o cualquier otra técnica que consideres apropiada.
    - Asegurar que la aplicación maneje correctamente situaciones de alta carga sin bloquearse o desperdiciar recursos.
4. **Optimización de Rendimiento:**
    - La API debe ser diseñada para ser altamente performante. Debe manejar eficientemente la asignación y liberación de recursos.

- Implementar técnicas de optimización (se valorará el uso de CompletableFuture, ForkJoinPool, o el patrón Reactor para mejorar la latencia y throughput de la aplicación)
- Monitorear y loggear el tiempo de procesamiento de cada solicitud para identificar cuellos de botella y áreas de mejora.

5. **Pruebas y Validación:**
    - Escribir pruebas unitarias y de integración que cubran la lógica de negocio y las condiciones de concurrencia.
    - Simular cargas concurrentes en un entorno de prueba y ajustar la configuración del pool de hilos para encontrar la configuración óptima.
    - Proveer un informe que describa los resultados de las pruebas, incluyendo métricas de rendimiento como tiempo de respuesta promedio, throughput, y uso de recursos.

Criterios de Evaluación:

6. **Eficiencia en la Gestión de Concurrencia:**
    - Capacidad para manejar múltiples solicitudes concurrentes de manera eficiente y segura.
    - Uso correcto de estructuras de datos concurrentes y mecanismos de sincronización.
7. **Optimización del Rendimiento:**
    - Capacidad para optimizar la aplicación para manejar cargas altas sin comprometer la estabilidad.
    - Implementación de técnicas avanzadas de concurrencia y asincronía en Java.
8. **Calidad del Código y Documentación:**
    - Claridad, limpieza y organización del código.
    - Comentarios y documentación que expliquen decisiones de diseño y técnicas utilizadas.
9. **Pruebas y Robustez:**

- Cobertura de pruebas adecuada para asegurar el funcionamiento correcto bajo condiciones de alta carga.
- Manejo adecuado de excepciones y errores.

Entregables:

- Código fuente completo de la aplicación Java publicados en repositorio GitHub.
- Pruebas unitarias y de integración.

- Un informe de rendimiento y optimización.
- Instrucciones para desplegar y probar la API.