Gerenciamento de Concorrência com ReentrantLock - Sincronização de Acesso a Dados com Read/Write Locks

STATUS **DESENVOLVIDA**

Objetivo do Projeto

Este relatório demonstra o uso de ReentrantLock em Java para gerenciar o acesso concorrente a uma fila de impressão compartilhada. Ele implementa um sistema onde múltiplas threads tentam imprimir documentos simultaneamente, destacando a diferença entre o uso de locks justos (FIFO) e não justos.

Estrutura do Código

Job.java: Representa um trabalho de impressão enviado para a fila. PrintQueue.java: Gerencia a fila de impressão utilizando ReentrantLock para controlar o acesso. Main.java: Cria e executa 10 threads que simulam solicitações de impressão concorrentes.

Funcionalidade

Threads solicitam impressão e são atendidas conforme a configuração do lock. O lock pode ser justo (fair = true), garantindo atendimento por ordem de chegada (FIFO), ou não justo (fair = false), permitindo atendimento fora de ordem. Cada impressão é dividida em duas etapas, ambas protegidas pelo lock.

🗒️ Cenários teste

Cenário 1: Executar com 10 threads e lock justo. Verificar se a ordem de execução segue FIFO. Lock Justo (fair = true):

Threads são atendidas na ordem de chegada. Nenhuma thread fica bloqueada por longos períodos. Execução pode ser ligeiramente mais lenta devido ao gerenciamento FIFO.

=======

Cenário 2: Executar com 10 threads e lock não justo. Observar se algumas threads são privilegiadas. Lock Não Justo (fair = false): Algumas threads podem ser atendidas antes de outras, independentemente da ordem de chegada. Execução pode ser mais eficiente, mas pode levar a "starvation" (algumas threads ficam presas por mais tempo).

=======

Cenário 3: Aumentar o número de threads para testar o impacto da contenção no desempenho. Teste com maior quantidade de threads: Aumentar o número de threads permite avaliar o impacto da contenção e do desempenho.

🔧 Conclusão:

🔗

O uso de ReadWriteLock permite um acesso eficiente aos preços, permitindo múltiplas leituras simultâneas e garantindo exclusividade durante a escrita. Os testes realizados confirmam que o mecanismo de bloqueio está funcionando corretamente, evitando condições de corrida e garantindo a consistência dos dados no sistema.

💻 Desenvolvedor

🔗

Marcio Fonseca
👨‍💻

=======