Comportamento de Concorrência com e sem ReentrantLock - Sincronizando um Bloco de Código com um Lock

STATUS **DESENVOLVIDA**

Objetivo do Projeto

Este teste visa demonstrar as diferenças no comportamento de concorrência entre threads quando usamos o controle de concorrência com ReentrantLock e quando não o utilizamos em um cenário com múltiplas threads tentando acessar um recurso compartilhado (neste caso, uma fila de impressão).

🗒️ Cenários teste

Cenário 1: Sem Uso de Lock

Descrição:

No primeiro cenário, removemos o controle de concorrência usando ReentrantLock. Nesse caso, várias threads tentam acessar o método printJob simultaneamente sem nenhum tipo de sincronização. Isso pode resultar em interferência entre threads e dados inconsistentes.

Teste Realizado: 10 threads tentando acessar o método printJob simultaneamente, sem controle de concorrência.

Resultado Observado: Interferência entre Threads: As mensagens de início e término da impressão podem se sobrepor, resultando em saídas confusas. Inconsistência de Dados: O tempo de impressão pode ser exibido de maneira incorreta, devido à falta de sincronização.

=======

Cenário 2: Com Uso de ReentrantLock

Descrição:

Neste segundo cenário, utilizamos o ReentrantLock para garantir que apenas uma thread por vez possa acessar o método printJob. Isso impede a interferência entre as threads e mantém os dados consistentes.

Teste Realizado: 10 threads tentam acessar o método printJob, mas apenas uma thread pode executá-lo por vez, graças ao ReentrantLock.

Resultado Observado: Controle de Concorrência: As mensagens de início e término da impressão são exibidas de maneira ordenada. Consistência de Dados: O tempo de impressão é exibido corretamente, sem interferência entre as threads.

🔧 Conclusão:

🔗

Conclusão Sem Lock: A execução é inconsistente e as threads interferem umas nas outras, causando saídas confusas e problemas de dados. Com ReentrantLock: O uso do ReentrantLock garante que as threads sejam executadas de forma ordenada e eficiente, mantendo a consistência dos dados e o controle de concorrência. O uso de ReentrantLock é fundamental em ambientes multithreaded para evitar problemas como interferência entre threads e inconsistências nos dados.

💻 Desenvolvedor

🔗

Marcio Fonseca
👨‍💻

=======