Trabalho Prático
Mecanismos de Sincronização

Aluno: Alexandre Dantas dos Santos

Introdução

O objetivo deste trabalho é estimular o projeto, implementação e avaliação de soluções para problemas por meio de programação concorrente, em especial colocando em prática os conceitos e mecanismos de sincronização de threads.

Considere uma lista simplesmente encadeada cujo acesso é compartilhado por três tipos de threads: o tipo B realiza operações de busca sobre a lista, o tipo I realiza operações de inserção de itens no final da lista e o tipo R realiza operações de remoção de itens a partir de qualquer posição da lista. Threads do tipo B meramente realizam operações de leitura sobre a lista e, portanto, podem ser executadas de forma simultânea com as outras. Por sua vez, as operações de inserção realizadas pelas threads do tipo I devem ser mutuamente exclusivas a fim de impedir que duas threads estejam inserindo itens no final da lista ao mesmo tempo. Por fim, no máximo uma thread do tipo R pode acessar a lista por vez para realizar remoção de itens e essa operação deve ser mutuamente exclusiva com relação às demais (busca e inserção).

Solução

Este trabalho foi desenvolvido na linguagem de programação Java (Versão 11) e fez uso dos recursos fornecidos pela linguagem para criação e gerenciamento de threads. O projeto foi desenvolvido junto a IDE Eclipse.

Na implementação proposta foram desenvolvidas três classes, uma para cada tipo de operação solicitada: BuscarItemThread, RemoverItemThread e InserirItemThread. Estas classes possuem uma instância da classe ListaCompartilhada como atributo. Durante a execução cada thread invoca seu método correspondente na lista compartilhada, formada integralmente por inteiros. Para inserir, buscar e/ou remover elementos, um número com valor até 50 é gerado randomicamente e é repassado como parâmetro.

A classe Main é responsável por criar, iniciar e sincronizar (com ela mesmo) os três tipos de threads e a lista encadeada de inteiros a ser compartilhada. Neste ponto são criadas três threads de cada tipo. O número de threads é controlado através da constante NUM_THREADS, que para a solução apresentada possui valor igual a 4.

Para solução do problema foram utilizados o uso de bloqueios, fazendo uso de *ReentrantReadWriteLock* e *ReentrantLock* do Java. Deste modo, as operações de inserção e busca recebem um lock do *ReentrantReadWriteLock* do tipo leitura (readLock()), haja visto que ambas as operações podem ser executadas em paralelo, conforme as diretrizes da atividade. Para a operação de remoção foi utilizado um lock do tipo escrita (writeLock()) do *ReentrantReadWriteLock*, haja visto que para este caso é necessário garantir total exclusão mútua entre threads. Por fim, também foi utilizado um lock do *ReentrantLock* na operação de inserção de itens, a fim de garantir também a exclusão mútua entre as inserções. É importante salientar também que os *locks* foram instanciados com política de justiça ativa (policy fairness) na declaração do construtor da lista compartilhada.

Foram executadas várias repetições/simulações a fim de identificar se a solução proposta atende ao solicitado. Percebe-se que múltiplas threads de busca e inserção conseguem a trava para execução simultaneamente. A imagem abaixo representa essa situação:

```
ThreadInserção 4 adquiriu acesso para inserção do item 49.
ThreadBusca 1 adquiriu acesso para consulta do item 40.
Elemento 40 não foi localizado na lista.
Elemento 49 inserido na lista.
ThreadBusca 1 finalizou e liberou acesso.
```

Por outro lado, quando uma thread de remoção consegue acesso, apenas esta executa até haver uma nova liberação. Neste ponto foram garantidas também as condições de exclusão mútua e ausência das condições de deadlock e starvation.

Dificuldades

A maior dificuldade ocorreu na abstração do problema e seu uso com threads, o que por si só não é algo tão simples. Foi necessária uma extensa consulta a materiais complementares e a documentação do próprio Java para melhor entendimento da solução apresentada.

Execução

Para execução do projeto basta acessar a pasta /bin presente na raiz do projeto e digitar em seguida (As classes estão presentes na pasta bin/core):

- **Java core.Main**

Outra forma de executar o projeto é através da IDE Eclipse. Para testar, basta importar o projeto para a IDE e executá-lo diretamente através da aplicação.

Referências

LinkedList (Java Platform SE 7) (oracle.com)
ReentrantLock (Java Platform SE 7) (oracle.com)
ReadWriteLock (Java Platform SE 7) (oracle.com)