Exclusão mútua com coleções e locks de leitura-escrita

Grupo de Sistemas Distribuídos
Universidade do Minho

Objectivos

Uso das facilidades de biblioteca em Java relativas a locks para resolver problemas de exclusão mútua envolvendo coleções de objectos. Locks que distinguem leituras e escritas.

Mecanismos

- Interface `Lock`; métodos `lock`, `unlock`;
- Classe `ReentrantLock`;
- Classe `ReentrantReadWriteLock`; métodos `readLock`, `writeLock`.

Exercícios propostos

1 Partindo do código fornecido, sem controlo de concorrência, relativo a uma nova versão do banco que permite a criação e fecho de contas, modifique este de modo a permitir que o banco seja acedido por múltiplas threads, fazendo uso de `ReentrantLock`. As operações são:

```
int createAccount(int balance);
int closeAccount(int id);
int balance(int id);
boolean deposit(int id, int value);
boolean withdraw(int id, int value);
boolean transfer(int from, int to, int value);
int totalBalance(int[] ids);
```

em que criar uma conta devolve um identificador de conta, para ser usado em outras operações, e fechar uma conta devolve o saldo desta; é ainda possível obter a soma do saldo de um conjunto de contas. Algumas operações devolvem `false` ou 0 se um identificador de conta não existir ou não houver saldo suficiente.

A implementação deve permitir concorrência, mas garantir que os resultados sejam equivalentes a ter acontecido uma operação de cada vez. Por exemplo, ao somar os saldos de um conjunto de contas, faça com que ela seja equivalente a uma "fotografia instantânea", não permitindo, e.g., que

sejam usados montantes a meio de uma transferência (depois de retirar da conta origem e antes de somar à conta destino).

2 Conceba cenários de teste, especialmente envolvendo fecho de contas. Modifique o código de teste da versão anterior do banco, de modo a testar a nova implementação.

3 Utilize os locks de leitura-escrita disponibilizados em `ReentrantReadWriteLock` de `java.util.concurrent.locks`, para permitir mais concorrência, nomeadamente não bloqueando o banco todo sempre que diferentes threads tentam manipular as mesmas contas. Dado o custo maior destes locks, tente limitar o seu uso, tendo em conta a expectativa de não haver grande probabilidade de contenção nas contas (pensando o caso de um banco com muitas contas).

Exercícios Adicionais

4 Com locks de exclusão mútua, e seguindo o *2 Phase Locking*, o banco é todo bloqueado quando 2 threads colidem em alguma conta. Explore possibilidades alternativas para permitir mais concorrência apenas com o uso de `ReentrantLock` (ou seja, sem usar `ReentrantReadWriteLock`). Explore nomeadamente a possibilidade de libertar o lock do banco mais cedo do que o indicado pelas regras do *2 Phase Locking*. Descubra qual seria o comportamento errado introduzido. Tente descobrir algum modo "artesanal" de o corrigir.