MAC 438 - Programação Concorrente - Primeiro Semestre de 2002

Terceiro Exercício-Programa: O Problema dos Barbeiros

Data de Entrega: 27 de junho de 2002

Este exercício deve ser desenvolvido em equipes de duas pessoas, a fim de suscitar discussão. Dúvidas sobre o enunciado devem ser enviadas para `reverbel-mac438@ime.usp.br`.

1 O problema

Você deve implementar uma solução para um problema clássico de comunicação entre processos: o Problema dos Barbeiros. Vimos em aula uma solução para uma versão desse problema na qual havia um único barbeiro e não havia limite no número de clientes esperando atendimento na barbearia.

Neste exercício você deve lidar com n barbeiros trabalhando simultaneamente (ou seja, em vez de ter um barbeiro e uma cadeira de barbeiro, a barbearia agora tem n barbeiros e n cadeiras de barbeiro). Considere também que a barbearia tem m assentos para os clientes que estão esperando que algum barbeiro os atenda. Quando um cliente chega à barbearia ele verifica se ela está lotada ou não. Se a barbearia estiver lotada (se todos os n barbeiros estiverem atendendo clientes e todos os m assentos de espera estiverem ocupados), ele sai sem ter seu cabelo cortado. Se a barbearia não estiver lotada, ele entra e espera pelo corte de seu cabelo.

2 Seu arsenal

Você deve fazer este EP em Java, usando threads Java e os recursos para sincronização entre threads oferecidos pela linguagem (métodos/blocos `synchronized`, operações `wait()`, `notify()` e `notifyAll()`). Se possível evite o uso de `notifyAll()`, que facilita os algoritmos mas tende a penalizar o desempenho.

Note que a solução vista em classe para o caso de um barbeiro só (seção 5.2.5 do livro do Andrews) usou um monitor com três variáveis de condição. No caso mais geral de n barbeiros você certamente vai precisar de mais de uma variável de condição. O desafio aqui é implementar em Java algo equivalente a um monitor com múltiplas variáveis de condição. Ler a seção 3.7 (e especificamente o ítem 3.7.3) do livro do Doug Lea deve ajudar você nessa tarefa.

3 Organização do programa

Crie n threads "barbeiro" e k threads "cliente". A vida de um cliente é muito monótona: ele vai até a barbearia para cortar seu cabelo (pode ser que não consiga, caso a barbearia esteja cheia), depois faz outras coisas menos importantes (ou seja, espera um tempo aleatório), depois vai cortar seu cabelo novamente. A vida de um barbeiro também é monótona: ele pega um cliente, corta o cabelo dele (ou seja, espera um tempo aleatório), avisa esse cliente que o corte acabou e passa para o próximo cliente.

A saída do seu programa deve indicar que barbeiro está atendendo a que cliente (ou seja, o programa associa uma identificador a cada barbeiro e a cada cliente).

Uma das classes do programa deve implementar conceitualmente um monitor que encapsula todos os acessos à barbearia. O "conceitualmente" aqui indica que você provavelmente não fará uma tradução direta do monitor em classe Java, com `synchronized` em todos os métodos públicos, pois isso não é adequado para o caso de múltiplas condições (a menos também se use chamadas `notifyAll()`). Esse monitor tem os métodos indicados abaixo:

```
monitor Barbearia {

    // Campos (variáveis privadas) do monitor
    ...

    // Operação chamada pelos clientes:

    boolean cortaCabelo() { ... } // se a barbearia não estiver lotada, espera
                                  //     que o corte seja feito e retorna true
                                  // se a barbearia estiver lotada, retorna false


    // Operações chamadas pelos barbeiros:

    void proximoCliente() { ... } // pega o próximo cliente (dentro desta chamada
                                  // o barbeiro pode dormir esperando um cliente)

    void corteTerminado() { ... } // o barbeiro acorda o cliente que está na sua
                                  // cadeira e espera que ele saia da barbearia
                                  // (tome cuidado para acordar o cliente certo)
}
```

Se achar conveniente, adicione aos métodos `proximoCliente()` e `corteTerminado()` um parâmetro que identifica o barbeiro que atenderá ou atendeu o cliente:

```
void proximoCliente(int idBarbeiro) { ... }

void corteTerminado(int idBarbeiro) { ... }
```

4 Sobre a entrega

Você deverá entregar tres coisas:

- um arquivo tar.gz contendo sua solução (arquivos-fonte, makefile, README, ...);
- uma listagem impressa dos seus arquivos-fonte;
- um relatório impresso.

Por favor, entregue o relatório e a listagem **na secretaria do MAC**, em um saco plástico devidamente fechado, contendo também um disquete com o arquvo tar.gz.

EPs atrasados não serão aceitos!

Bom trabalho!