Sistemas Operacionais - 2º Trabalho de Programação

Profa. Roberta Lima Gomes - email: soufes@gmail.com

Período: 2019/2

Data de Entrega: 01/12/2019 (até meia-noite)
Composição dos Grupos: até 3 pessoas

Material a Enviar

- Por email: enviar um email para **soufes@gmail.com** seguindo o seguinte formato:
  - Subject do email: "**Trabalho 2**"
  - Corpo do email: lista contendo os **nomes completos dos componentes** do grupo em ordem alfabética
  - Em anexo: um arquivo compactado com o seguinte nome "**nome_do_grupo.zip**" (*ex: joao-maria-jose.zip*). Este arquivo deverá conter todos os arquivos fonte da sua solução.

Valendo ponto: clareza, endentação e comentários no programa.
Atrasos implicarão em descontos de 1 ponto por dia...

Descrição do Trabalho

Você deve implementar uma solução para um problema clássico de comunicação entre processos: o **Problema dos Barbeiros**. No livro do *A. Tanenbaum* você pode encontrar uma solução para uma versão desse problema na qual existe um único barbeiro e não há limite no número de clientes esperando atendimento na barbearia.

Neste trabalho você deve lidar com **n** barbeiros trabalhando simultaneamente (ou seja, em vez de ter 1 barbeiro e 1 cadeira de barbeiro, a barbearia agora tem **n** barbeiros e **n** cadeiras de barbeiro). Considere também que a barbearia agora tem **m** assentos para os clientes que estão esperando que algum barbeiro os atenda (lembre-se de garantir a ordem de chegada… caso contrário os clientes podem ficar nervosos!). 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 se todos os **m** assentos estiverem ocupados), ele sai sem ter seu cabelo cortado. Se a barbearia não estiver lotada, ele entra e espera pelo corte do seu cabelo.

Qualquer mecanismo de sincronização deve ser implementado com base exclusivamente nos monitores Java. NÃO é permitido utilizar nenhuma Classe auxiliar para implementar a sincronização/bloqueio das threads.

Neste trabalho vocês devem implementar uma classe que funcione como a Barbearia. Nessa classe vocês devem implementar os seguintes métodos:

```
// Operação chamada pelos clientes:
     public boolean cortaCabelo(Cliente c) { ... }
          // 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:

    public Cliente proximoCliente() { ... }

        // Seleciona o próximo cliente (dentro desta chamada o
        // barbeiro pode dormir esperando um cliente).


    public void corteTerminado(Cliente c) { ... }

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

Além disso, vocês devem implementar as classes, Pessoa, Cliente e Barbeiro (sendo que Cliente e Barbeiro são sub-classes de Pessoa). Pessoa deve definir um atributo int id, e o método público int getID().

Já, as classes Cliente e Barbeiro devem ser executadas com threads.

Barbeiro deve ficar em loop, chamando os métodos `proximoCliente()` e `corteTerminado()` esperando um tempo aleatório entre 1 e 3 seg entre uma chamada e outra.

Cliente deve ficar em loop chamando `cortaCabelo()`. Dê um intervalo entre 3 e 5 seg a cada loop.

Seu programa deverá receber como parâmetro (na linha de comando)

- o número **n** de Barbeiros;
- o número **m** de cadeiras na barbearia;
- o número total de clientes.

O programa deverá iniciar as threads em seguida. Durante a execução, seu programa deverá imprimir informações na saída padrão, sempre que Clientes e Barbeiros mudarem de estado:

- “Cliente X esperando corte...”
- “Cliente X cortando cabelo com Barbeiro Y”
- “Cliente X terminou o corte… saindo da barbearia!”
- “Cliente X tentou entrar na barbearia, mas está lotada… indo dar uma voltinha”
- “Barbeiro Y indo dormir um pouco… não há clientes na barbearia...”
- “Barbeiro Y acordou! Começando os trabalhos!”

ATENÇÃO: Monitores em Java não consideram “ordem de chegada” quando se executa `notify()` ou `notifyAll(`. Portanto, vocês necessitam de estruturas de dados auxiliares para garantir que os Clientes tenham seus cabelos cortados na ordem em que eles chegaram na barbearia…

PROBLEMAS COM JAVA?

https://docs.oracle.com/javase/tutorial/essential/concurrency/sync.html
http://www.primeuniversity.edu.bd/160517/vc/eBook/download/IntroductiontoJava.pdf
http://campus.murraystate.edu/academic/faculty/wlyle/325/ch32.pdf