Sistemas Operacionais - 2º Trabalho de Programação

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

Período: 2019/1

Data de Entrega: 23/06/2019 (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 1**"
  - 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.

Descrição do Trabalho

Pretende-se construir uma aplicação Java em que threads *Producers* enviam mensagens para threads *Consumers*. As mensagens são colocadas em 4 filas, cada uma associada a um nível de prioridade de 0 a 3, em que 0 é a mais prioritária (podíamos considerar mensagens com quatro níveis de urgência). As mensagens são todas do mesmo tamanho. As filas de mensagens têm espaço para 3 mensagens cada.

Quando um *Producer* pretende escrever a mensagem e a respectiva fila está cheia, fica bloqueado. Um processo *Consumer* deve retirar sempre a mensagem da fila mais prioritária. Se não houver nenhuma mensagem (em nenhuma das filas), ele deve ficar bloqueado.

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 um buffer de 4 filas. Nessa classe vocês devem implementar os dois métodos que permitam inserir e retirar mensagens:

```
void inserir (Message mensagem)

Message retirar()
```

Observem que vocês também deverão implementar a classe `Message`, que deve conter um inteiro para indicar a prioridade de uma mensagem, e uma string correspondendo à mensagem em si.

Sua aplicação deverá criar 10 threads *Producers* e 10 threads *Consumers* que devem ficar em loop produzindo ou consumindo. *Producers* devem produzir mensagens a cada *x* segundos (*x* é um valor aleatório entre 1 e 5). A prioridade da mensagem também deve ser definida de forma aleatória. *Consumers*, após consumirem uma mensagem devem, esperar um tempo aleatório (entre 1 e 5 seg.) para tentar consumir uma nova mensagem.

O programa deverá imprimir informações na saída padrão, sempre que:

- uma mensagem for produzida ou consumida, identificando: (i) a mensagem, (ii) sua prioridade e (iii) a thread que a produziu ou consumiu;

- uma thread for bloqueada ou desbloqueada.

ATENÇÃO: Monitores em Java não consideram “ordem de chegada” quando se executa `notify()` ou `notifyAll()`. Portanto, vocês necessitam de alguma estrutura de dados auxiliar para garantir que as mensagens sejam inseridas nas filas na ordem em que os *Producers* “chegam”. No caso dos *Consumers*, não é necessário garantir ordem de chegada, o importante é consumir as mensagens mais prioritárias primeiro (e garantindo a ordem das mensagens dentro da mesma fila)…

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