Universidade de Itaúna
Faculdade de Engenharia
Bacharelado em Ciência da Computação
Disciplina: Lab. de Computação Paralela e Sistemas Distribuídos
Professor: Zilton Cordeiro Júnior **Data:** 05/09/2016

Aluno: _____ Nota: _____

Threads

Este trabalho prático tem por objetivo explorar temas referentes a programação concorrente através do emprego de threads. Serão explorados problemas (regiões críticas) e soluções (sincronismo).

1. Especificação do problema

Um bar resolveu liberar um número específico de rodadas grátis para seus N clientes presentes no estabelecimento. Esse bar possui X garçons. Cada garçom consegue atender a um número limitado (C) de clientes por vez. Como essas rodadas são liberadas, cada garçom somente vai para a cozinha para buscar o pedido quando todos os C clientes que ele pode atender tiverem feito o pedido ou não houver mais clientes a serem atendidos. Após ter seu pedido atendido, um cliente pode fazer um novo pedido após consumir sua bebida (o que leva um tempo aleatório) e a definição de uma nova rodada liberada. Uma nova rodada somente pode ocorrer quando foram atendidos todos os clientes que fizeram pedidos. Por definição, nem todos os clientes precisam pedir uma bebida a cada rodada.

2. Construção da solução

Implemente uma solução que permita a passagem por parâmetro (i) o número de clientes presentes no estabelecimento, (ii) o número de garçons que estão trabalhando, (iii) a capacidade de atendimento dos garçons e (iv) o número de rodadas que serão liberadas no bar.
Cada garçom e cada cliente devem ser representados por threads, estruturalmente definidos como os pseudo-códigos que seguem:

```
thread cliente {
  while (!fechouBar){
    fazPedido();
    esperaPedido();
    recebePedido();
    consomePedido();//tempo variável
  }
}
```

```
thread garçom{
  while(existemClientesNoBar){
    recebeMaximoPedidos();
    registraPedidos();
    entregaPedidos();
    rodada++; //serve como parâmetro para
              // fechar o bar
  }
}
```

- Este trabalho deve ser implementado em Java, utilizando monitores e threads.
  - O principal no programa é o uso de monitores para controlar a concorrência e a sincronização entre as threads.
- A ordem de chegada dos pedidos dos clientes na fila de pedidos de cada garçom deve ser respeitada.
  - Sua solução não deve permitir que clientes furem essa fila.
- O garçom só pode ir para a cozinha quando tiver recebido seus C pedidos.
- O programa deve mostrar a evolução, portanto planeje bem o que será apresentado na saída (tela). Deve ficar claro o que está acontecendo no bar a cada rodada. Os pedidos dos clientes, os atendimentos pelos garçons, os deslocamentos para o pedido, a garantia de ordem de atendimento, etc.
- O desenvolvimento do trabalho deverá ser iniciado na sala de aula.

3. Informações adicionais

- O trabalho deve ser submetido através do Portal da Universidade até a data definida pelo professor.
  - Caso o portal não esteja disponível, a entrega deve ser feita na aula, via pendrive.

- O trabalho deverá ser submetido como um arquivo compactado contendo o código do projeto.

- Não serão aceitos trabalhos entregues fora do prazo.
- Trabalhos que não compilam ou que não executam receberão nota ZERO, bem como trabalhos que sejam considerados como plágio.
- Deve ser adicionado no zip um arquivo README para a execução do trabalho.
- O trabalho deve ser implementado em Java.

Os itens para avaliação são:

1. Funcionamento do programa,
2. Execução das threads sem ocorrência de deadlocks e/ou outros problemas de sincronização,
3. Saída do programa (de modo a permitir a avaliação de seu funcionamento),
4. Clareza do código (utilização de comentários e nomes de variáveis adequadas).
5. Organização geral do trabalho.

Bons estudos!