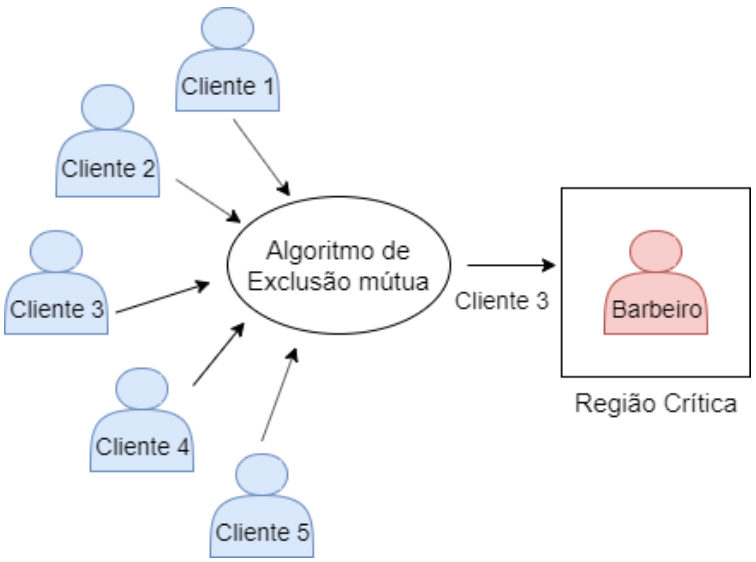
Trabalho Prático - Coordenação e Acordo
Exclusão Mútua Distribuída

- Trabalho em dupla ou individual
- Códigos iguais acarreta nota zero para ambas as equipes
- Pode ser utilizado qualquer linguagem de programação tal como: python, c, c++, javascript.

Descrição

Neste trabalho, desenvolva um sistema para gerenciar o uso de um recurso (um objeto servidor) que não pode ser utilizado de forma compartilhada (exclusão mútua) pelos clientes. Crie cinco objetos clientes que ficam competindo para acessar esse recurso. O recurso pode ser implementado na forma de um objeto *Barbeiro* que fornece três métodos: *cortarCabelo(), cortaBarba(), cortarBigode()*. Cada um desses métodos leva 3, 4 e 5 segundos, respectivamente, para processar a operação (use o *sleep* para gastar esse tempo). Como só tem um barbeiro, esses três métodos devem ser acessados de forma exclusiva, ou seja, o objeto *Barbeiro* só pode realizar apenas uma operação por vez: corta cabelo, barba ou bigode. O algoritmo de exclusão mútua deve, portanto, controlar o acesso ao objeto *Barbeiro*. Os objetos clientes devem tentar cortar o cabelo primeiro, depois a barba e, por fim, o bigode, nessa sequência repetidas vezes (até o limite de 20 ciclos). Quando um cliente obtém o privilégio para acessar o objeto *Barbeiro*, ele só pode acessar um dos serviços de corte e, em seguida, liberar o objeto *Barbeiro*, voltando a competir com os outros clientes.



Implementação

Requisitos da aplicação:

- Utilizar um *middleware* (Java RMI, Web Services, MOM ou RPC/XML) para prover a comunicação.

O algoritmo de exclusão mútua a ser implementado usando o algoritmo de Exclusão Mútua Distribuído dado em aula. Portanto, não é preciso utilizar qualquer mecanismo de controle de concorrência do Java (ex. monitores, *locks*, semáforos, etc.). Implemente primeiro o algoritmo de ordenação de eventos do *Lamport*, colocando um contador em cada cliente, para que possam ter um número de sequência nas suas mensagens no algoritmo de Exclusão Mútua Distribuído.

Quando um cliente necessitar de um serviço de corte deve invocar o método *Concorrer (short id, string rc, short cont)* nos outros clientes do grupo enviando uma mensagem contendo seu *id*, o recurso *rc* que quer utilizar e o contador *cont* da mensagem (segundo o algoritmo de *Lamport*). Após enviar essa mensagem cria uma *Thread* para coletar as mensagens *OK* dos outros clientes para poder acessar a região crítica.