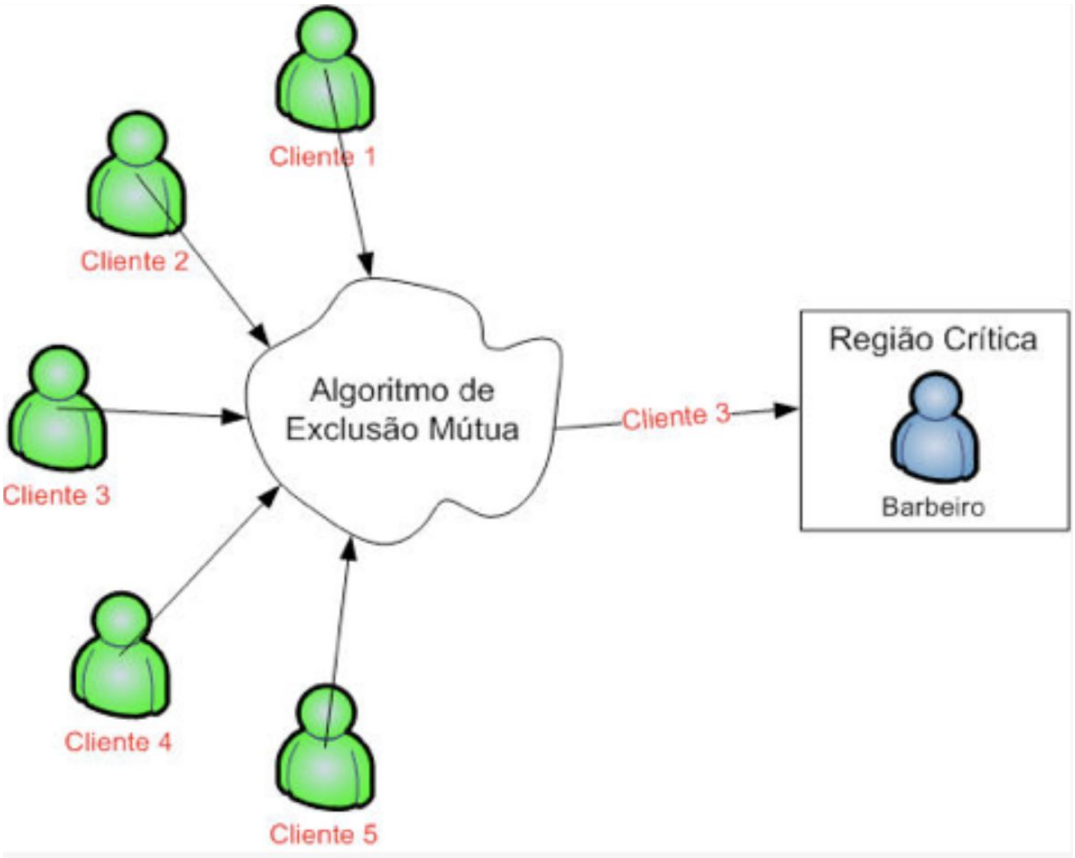
Trabalho: Exclusão Mútua Distribuída

Problema:

Neste trabalho, será desenvolvido 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:

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 (p.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.

Interface IDL dos Clientes:

```
struct Mensagem {
   short id;
    string rc;
    short cont;
};

interface Cliente {
    oneway void Concorrer(Mensagem msg);
    oneway void msgOK(short id);
};
```

Interface IDL do Servidor Barbeiro:

```
interface Barbeiro {
    boolean cortarBarba();
    boolean cortarCabelo();
    boolean cortarBigode();
};
```

Apresentação:

A atividade pode ser desenvolvida **em dupla**. O programa deve ser apresentado ao professor no laboratório ou na sala do professor **até o dia 07/12**. Os dois componentes do grupo devem estar presentes. Será verificado o funcionamento do programa e em seguida os alunos devem responder a questões sobre a forma como foram utilizados os mecanismos de comunicação e sincronização entre os processos no programa.

Podem ser atribuídas notas diferentes aos alunos de um grupo, dependendo das respostas às perguntas sobre o código do programa efetuadas pelo professor. Caso um dos alunos não esteja presente ou demonstra não conhecer o código do programa, será atribuída nota zero à atividade. Em caso de **cópia** do código de outro grupo, ambos terão nota igual a **zero**.