Relatório Técnico

Nome: João Gabriel Pilger dos Santos

Matrícula: 0197359

Matéria: Processamento Paralelo

Introdução

Este relatório foi feito para a matéria Processamento Paralelo, que é lecionada pelo professor Paulo Ricardo.

Foi proposto a resolução de dois problemas computacionais, sendo eles o “Ajudantes do Papai-Noel” e “Barbeiros Dorminhocos”, usando o paradigma de Thread e programação concorrente, onde resolvi os mesmos usando a linguagem Java.

Problemas no caminho

Eu particularmente não obtive tantos problemas, o que me causou um bom tempo de reflexão foi o método " wait(); " usado na thread que iria esperar determinada ação.

Seria utilizada nas classes: elfos, renas, papai noel, barbeiros e clientes.

A verdadeira dificuldade foi na chamada do " notify() " e " notifyAll() ", que não consegui

voltar a execução as threads que eu queria que voltassem. Então para corrigir essa questão, usei variaveis do tipo "boolean" para controle dos estados, deixando cada thread independente.

Após isso somente erros de estruturação.

Soluções

Como dito acima, usei variáveis booleanas para o controle dos estados, sendo identificados por:

**Elfos:**

- emConsultoria (usado para quando o noel estiver atendendo ele)

**Renas:**

- ajudandoNoel (usado para quando a rena estiver ajudando o noel a entregar os presentes)

**Noel:**

- estados de atendendo elfos ou renas e estado de dormindo.

**Barbeiros:**

- dormindo (para quando estiver dormindo, entao estará aguardando por atendimento por cliente ou preparado para trocar de lugar com o outro barbeiro)

-cortando (não estara disponivel para atender outro cliente e para trocar de lugar com outro barbeiro)

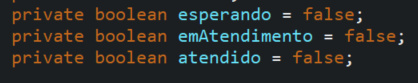
-organizando (estará disponível para trocar de lugar com outro barbeiro e não estará disponível para atender)

**Cliente:**

-esperando (aguardando atendimento)

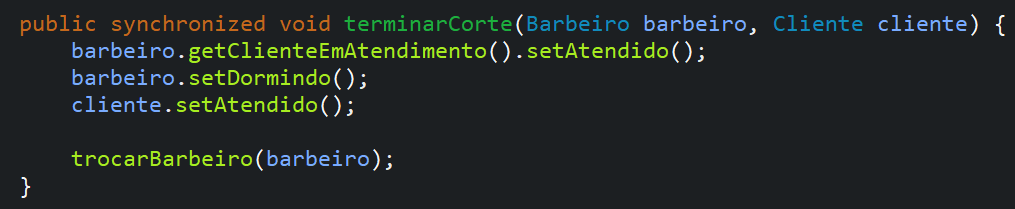
-emAtendimento (sendo atendido)

-atendido (quando o barbeiro finaliza o corte e libera o cliente)





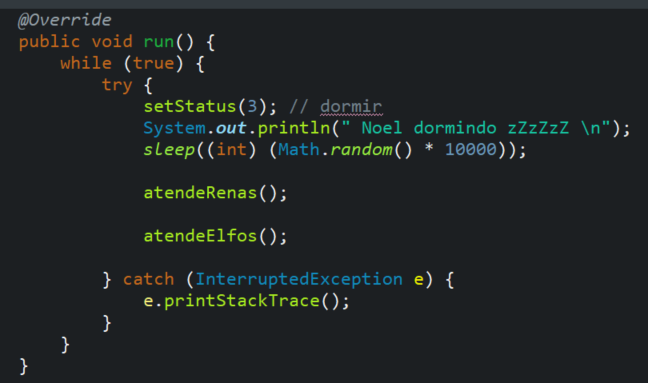
O print acima retrata o método run da classe “Cliente”, que ao invés de usar o método “wait”, usei um loop na variável de controle. A mudança de estado se dá através da classe de controle “Barber”, que usei como um singleton.



A thread “Barbeiro” chama esse método acima ao terminar o corte do cliente, onde a classe “Barber” irá alterar as variáveis de controle das threads em execução, que será do barbeiro que finalizou o corte e o cliente que irá embora (thread interrupt).

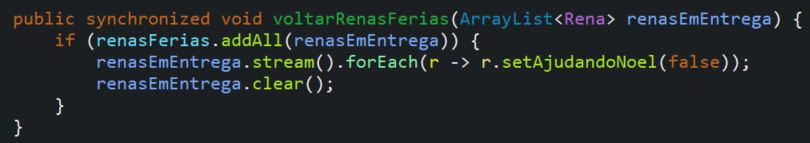


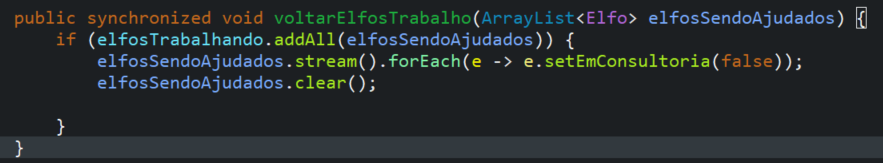
No run do Barbeiro (print acima) segue a mesma lógica do Cliente, porém trabalhando um pouco mais o console, printando a etapa em que se encontra. Começa verificando se está ‘organizando’ e caso esteja, só sairá depois de um barbeiro terminar um corte. E assim a mesma lógica para dormindo, porém só sairá se tiver cliente esperando na fila. E quando estiver cortando o cabelo, ficará um tempo randômico no “sleep” e depois dispensará o cliente e irá trocar de lugar com o barbeiro que estiver organizando.



No método acima, o “run” do Papai Noel, cabe a própria thread procurar os recursos que precisam de atenção, caso as 9 renas estejam prontas, será executado o método “atendeRenas”, ou se tiver 3 elfos aguardando atendimento, no método “atendeElfos”.

E então depois de executar, irá chamar o método da classe “Controller”, que também é um singleton, para as threads voltarem a executar em seus estados normais.





Exemplo

Podemos pegar de exemplo o Node.js, que é um interpretador de JavaScript assíncrono orientada a eventos. Então uma das suas função é por exemplo, quando há um processo de I/O, ele é deixado em ‘wait’ até que esse I/O seja feito, enquanto isso outros processos são executados sem que fiquem preso por outro que esteja aguardando algo.