**Universidade Federal do Rio de Janeiro**

2014.1

**Projeto Elevadores**

**Disciplina Computação Concorrente**

**Patrícia S. Ghiraldelli**

**Carlos Eduardo S. Martins**

SUMÁRIO

Visão geral do projeto #

Página 3

conteúdo do projeto #

Página 6

Principais desafios do projeto #

Página 7

casos de teste #

Página 8

como compilar o projeto #

Página 9

Esse relatório foi criado com a intenção de apresentar o projeto Elevadores criado por Carlos Eduardo S. Martins e Patrícia S. Ghiraldelli para a disciplina Computação Concorrente da Universidade Federal do Rio de Janeiro, ministrada pela professora Silvana Rossetto, no período de 2014.1.

Aqui será possível entender a lógica usada para a criação de cada etapa do projeto, assim como conhecer a estrutura do projeto.

* Visão geral do projeto

O projeto contém quatro classes:

* Main
* Monitor de Tarefas
* Elevador
* Tarefa

Vida de cada classe:

Classe Main

1. Lê o arquivo de entrada linha por linha, armazenando as informações de cada linha em variáveis distintas.
2. Cria uma nova instância do MonitorDeTarefas.
3. Cria as threads responsáveis por processar as tarefas em questão(elevadores).
4. Dá início a execução das threads.
5. Dá início ao monitor, passando o vetor de tarefas do arquivo lido.
6. Espera todas as threads terminarem para finalizar o programa.

Classe Elevador

1. Chama o método do MonitorDeTarefas para escolher uma tarefa.
2. Verifica se a Tarefa escolhida é nula. Se sim, finaliza a execução. Se não, prossegue.
3. Armazena as informações da Tarefa escolhida em variáveis.
4. Imprime o status da Tarefa Bruta (como veio do arquivo lido).
5. Processa tarefa (enquanto há requisição: atualiza o andar atual para o andar da primeira requisição, remove o andar que foi atualizado e imprime a requisição).
6. Chama o método do MonitorDeTarefas para finalizar a Tarefa.
7. Pede por mais Tarefa.

Classe MonitorDeTarefas

1. Recebe um vetor de string de tarefas da Main e o armazena numa variável de tarefasBrutas.
2. Cria tarefas: sinaliza que começou a criar tarefas, percorre o vetor de string das tarefas. A cada iteração cria uma nova instância do objeto Tarefa com um id específico (nesse caso, o número do loop em que se encontra) e trata a tarefa bruta da iteração(a partir do vetor de string recebido armazena as informações na instância do objeto Tarefa). Antes de armazenar as requisições, as mesmas passam pelos métodos de ordenarRequisições e excluirRepetições) . No final, sinaliza que acabou de criar as tarefas.
3. Aguarda chamadas dos métodos escolherTarefa e finalizarTarefa.

Classe Tarefa

Objeto que armazena todas as informações necessárias para o processamento de uma tarefa.

Para complementar a documentação do código, sugerimos abrir os arquivos do Javadoc, que se encontra na pasta doc, em algum browser de sua preferência.

* Conteúdo do projeto

O projeto “Elevadores” contém 4 pastas:

* Src : contém os arquivos (.java) de todas as classes.
* Doc: contém todos os arquivos referentes ao javadoc do projeto. Ao acessar doc -> resources-> index.xhtml e abrir o arquivo em um browser de preferência, será possível navegar entre todas as páginas com as explicações de todos as classes e/ou métodos e/ou variáveis do projeto.
* Log: contém dois arquivos (logExecucaoElevador0.txt e logExecucaoElevador1.txt), que são os arquivos de saída de cada elevador.
* Testes: contém 7 pastas (teste0 até teste6). Cada subpasta possui uma pasta chamada log (contém os arquivos de saída de cada elevador do teste) e um arquivo entrada.txt(entrada utilizada para teste).
* Principais desafios do projeto

As especificações do projeto diziam que era necessário que as threads fossem criadas antes da criação das tarefas. Portanto, quando damos início a uma thread, a mesma logo pede ao monitor uma nova tarefa. Porém, como ainda não foi criada nenhuma tarefa, a thread elevador deveria ficar bloqueada até que alguma tarefa fosse criada. Para resolver o problema do bloqueio das threads, optamos por acordar a todas as threads toda vez que uma nova tarefa é criada. Assim, a cada tarefa uma nova thread ganha a oportunidade de executar.

Após resolvido tal problema ainda enfrentamos outro. Caso houvesse menos tarefas do que elevadores, seria mais eficiente que a tarefa escolhesse um elevador mais apropriado dentre todos que estão aguardando. Porque, se fosse ao contrário, o elevador necessariamente escolheria uma tarefa, mas talvez essa tarefa não fosse a melhor para esse elevador.

Para resolver esse problema usamos de uma variável booleana que nos “avisaria” quando a criação de tarefas teria terminado, para testarmos quem deveria escolher quem (elevador escolhe tarefa ou vice-versa).

O último grande problema a ser enfrentado seria o de terminar a execução das threads. As mesmas deveriam finalizar somente quando nenhuma tarefa estivesse sendo criada e a lista de tarefas estivesse vazia. Para contornar tal situação, fizemos com que o método escolherTarefa retornasse nulo quando houvesse erro ou quando a condição lógica descrita anteriormente fosse alcançada.

* Casos de Testes

Realizamos 7 testes diferentes. As entradas e respectivas saídas se encontram na pasta “teste” (**ver conteúdo do projeto).**

* Como compilar o projeto

Para obter uma perfeita execução da aplicação pelo terminal deve-se:

1. Ir até o diretorio trabalho-compconc/src. Lá se encontram as classes do projeto.
2. No terminal rodar o comando

javac Main.java

1. O comando ls –la mostrará que todas as classes foram compiladas e criados arquivos .class para cada uma delas.
2. Agora basta rodar java Main para que a aplicação execute. A pasta com os arquivos de saída será criada automaticamente no mesmo diretório.

Nota adicional sobre a compilação:

Para rodar a aplicação com outros casos de teste basta substituir no diretório do projeto o arquivo “entrada” por outro de mesmo nome.