Índice Remissivo Utilizando Listas Encadeadas

Carlos André - carlos.sousa@acad.pucrs.br

Diéferson Marinho - dieferson.marinho@acad.pucrs.br

Estruturas de dados utilizadas

Para realização do trabalho foram utilizadas estruturas encadeadas em forma de lista (LinkedList) e a sua implementação foi baseada nos algoritmos vistos em aula. As estruturas estão apresentadas no diagrama de classes ao fim do relatório.

Basicamente, foram utilizadas 6 classes principais, onde há também “classes internas” em algumas delas, correspondentes aos “Nodos” comuns às estruturas encadeadas. A classe app apenas dá início a execução do programa, instanciando um objeto da classe Interface, que por sua vez irá instanciar objetos das outras classes primárias, Stopwords, Texto e IndiceRemissivo. É na classe Interface que se dá toda a interação com o usuário, sendo que todas as operações pertinentes ao fluxo de execução partem dessa classe.

Os formalismos de UML não foram seguidos à risca para simplificar a visualização da estrutura.

Execução para cada funcionalidade

Item 1

Para cumprir a tarefa de exibir o índice em ordem alfabética, foi escolhido inserir as palavras ordenadamente durante a montagem do índice com estruturas encadeadas. Após, apenas foi usado o método toString() da classe IndiceRemissivo para a exibição ao usuário.

Item 2

Para contabilizar o número total de palavras, bem como quantas dessas palavras são stopwords, foram criados dois atributos na classe Interface. Tais atributos são atualizados durante a montagem do índice remissivo no método “loadIndice()”. O método “menuPercentualStopwords”, que calcula a relação entre os dois atributos e exibe ao usuário, é chamado pelo menu.

Item 3

Foi criado um método palavraMaisFrequente() na classe IndiceRemissivo, que devolve a palavra mais frequente acompanhada da quantidade de vezes em que a mesma aparece no texto. Tal método é chamado no menu e seus dados exibidos ao usuário.

Item 4

Os dados são solicitados ao usuário no método menuPesquisarPalavra() na classe Interface, que por sua vez chama o método getPalavra() do índice para exibir as páginas em que a palavra aparece no texto. Em seguida, é chamado o método pesquisarPalavra() presente na classe Texto, que exibe a página correspondente, destacando a palavra informada com [ ].

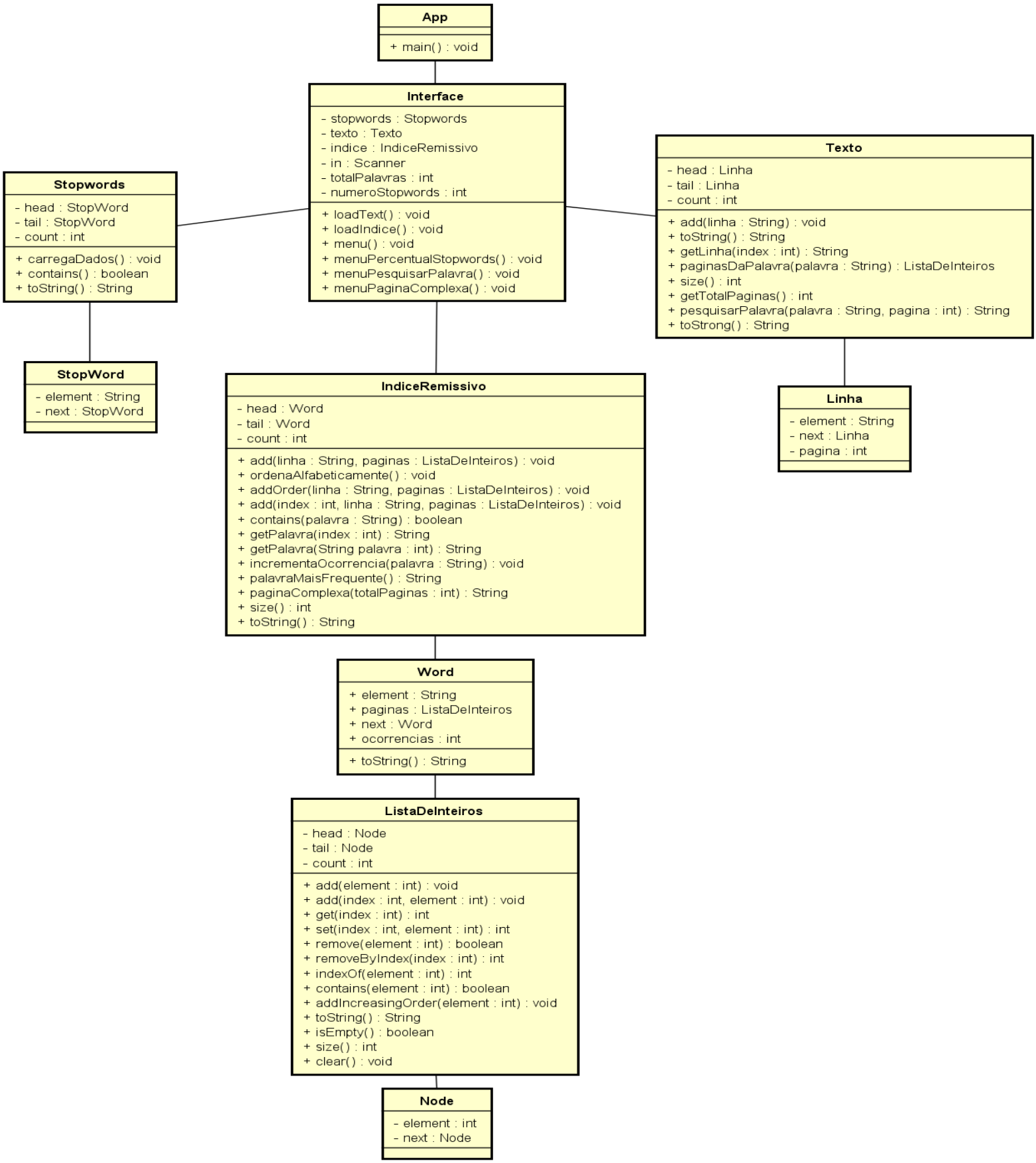
Item 5

Foi utilizado um método menuPaginaComplexa() na classe interface, onde irá fazer o tratamento de dados vindos do método paginaComplexa() da classe IndiceRemissivo, e em seguida exibirá tais informações ao usuário. No método paginaComplexa(), a estratégia utilizada foi verificar qual a página que mais aparece entre todos os endereços de páginas indexadas por todas as palavras do índice remissivo.

Análise de complexidade do item 5

A execução do item 5 tem início quando o método menuPaginaComplexa() é chamado na classe Interface, que por sua vez chama o método paginaComplexa() no índice. O primeiro loop acontece no caminhamento pelos índices das páginas (dado do tipo int), onde para cada página (ou índice da página), haverá outro loop que percorrerá todas as palavras presentes no índice remissivo, sendo que para cada palavra, haverá também um loop que irá verificar se o índice de página atual está presente entre os índices das páginas em que a palavra do índice remissivo ocorre. Portanto, de acordo com essas observações é possível caracterizar a complexidade envolvida na execução do item 5 como sendo $\mathbf{O(N^3)}$.

Modelagem de classes



Comentários

O tempo de execução para o arquivo java.txt disponibilizado para testes é praticamente instantâneo, levando em conta o tamanho do arquivo. Porém, os arquivos maiores, como por exemplo fiveweeksinaballoon.txt, o tempo de execução ultrapassou 3 minutos inserindo ordenadamente as palavras no índice remissivo.