ATIVIDADE DE PARTICIPAÇÃO 03 - RELATÓRIO TÉCNICO

Estrutura de Dados

Mateus de Sousa Moura

Mateus Weslley de Oliveira Freitas

Pablo Carvalho Magalhães

Segue o relatório sobre a atividade de participação 03:

```java
public class No {

    3 usages
    public boolean excluido;
    3 usages
    Pessoa item;
    9 usages
    No proximo = null;
    2 usages
    public No(String nome) {
        item = new Pessoa(nome);
        proximo = null;
        boolean excluido = false;
    }
```

Foi criada uma classe No com os atributos item, proximo e excluído, para compor os elementos da lista.

```java
public class Pessoa {

    3 usages
    private String nome;

    public Pessoa() {
    }
```

A lista vai ser composta pelo objeto pessoas.

```java
public void inserirNoInicio(String nome) throws IOException { // Para inserir no inicio preciso parar todos os
    boolean valor = false; // nenhum nome igual ao que está passado por parâmetro
    No novaPessoa = new No(nome); // Aloca espaço para o novo nó
    if(isEmpty() == false) //
        valor = buscaNome(novaPessoa.getName());
    if (valor == false) {
        if (isEmpty()) { // Se a lista estiver vazia o fim nó vai ser o nó atual.
            inicio = novaPessoa;
            fim = novaPessoa;
            novaPessoa.proximo = null;
        } else {
            novaPessoa.proximo = inicio;
            inicio = novaPessoa;
        }
        System.out.println("Nome cadastrado com sucesso!");
    } else {
        System.out.println(
                "Não possível cadastrar esse nome!\nNome já cadastrado!");
    }
}
```

O método InserirNoInicio insere elementos no início da lista, verificando se a lista está vazia inicialmente através do método isEmpty(), caso a lista não esteja vazia fazemos a busca do nome passado por parâmetro dentro da lista usando o método buscaNome(). Se o método buscaNome() retorna false para a variável valor significa que não foi encontrado nenhum nome igual ao que está sendo passado dentro da lista, logo então, fazemos a inserção dentro da lista. Se a lista estiver vazia o descritor de início e fim recebe o nó auxiliar novaPessoa. Caso contrário, novaPessoa será inserido no início da lista

```java
public void inserirNoFim(String nome) {
    boolean valor = false; // Vamos assumir inicialmente que o valor não está na lista
    No novaPessoa = new No(nome); // Aloca espaço para o novo nó
    if(isEmpty() == false)
        valor = buscaNome(novaPessoa.getName());
    if (valor == false) {// se o nome não foi encontrado no arquivo
        if (isEmpty()) {
            inicio = novaPessoa;
        } else {
            fim.proximo = novaPessoa; // O antigo ultimo nó agora aponta para o novo nó e esse novo nó será o novo último elemento da lista
        }
        fim = novaPessoa; // Declaramos oficialmente que a novaPessoa vai ser o último elemento da lista
        System.out.println("Nome cadastrado com sucesso!");
    } else {
        System.out.println("Não possível cadastrar esse nome!\nNome já cadastrado!");
    }
}
```

O método inserirNoFim opera de maneira similar ao método InserirNoInício, a diferença é que o descritor de fim que recebe o valor de novaPessoa.

```java
public void alterarNome(String nomeAntigo, String novoNome) {
    No auxiliar = inicio;
    boolean valor = false;
    if(isEmpty()) {
        System.out.println("Não é possível alterar nome\nLista vazia!");
    } else {
        valor = buscaNome(nomeAntigo);
        if(valor == true || auxiliar.excluido == true) {
            while(auxiliar!=null) {
                if(auxiliar.getName().equalsIgnoreCase(nomeAntigo)) {
                    auxiliar.setName(novoNome);
                    System.out.println("Nome alterado com sucesso!");
                    break;
                } else {
                    auxiliar = auxiliar.proximo;
                }
            }
        } else {
            System.out.println("Esse nome que você deseja alterar não foi encontrado na lista!");
        }
    }

}
```

O método alterarNome recebe duas Strings, nomeAntigo que recebe o nome que irá ser alterado e novoNome que será o nome que irá substituir o nome. O método verifica se existe elementos na lista, se o elemento que foi requistado a alteração existe e o elemento não foi marcado como excluído, ele troca o nomeAntigo pelo novoNome.

```java
public void excluiNomes(String nome){
   No auxiliar = inicio;

    if(buscaNome(nome)){
        while(auxiliar!= null) {
            if(auxiliar.getName().equalsIgnoreCase(nome)){
                auxiliar.excluido = true;
                System.out.println("Nome excluído com sucesso!");
            }
                auxiliar = auxiliar.proximo;


        }
    } else {
        System.out.println("Esse nome que você deseja exluir não foi encontrado na lista!");
    }
}
```

O método exclui nomes verifica a existência do nome passado por parâmetro, percorre a lista através de um loop, quando ele encontra o nome igual na lista usando o método equalsIgnoreCase() ele altera o descritor excluido para true.

```
public void listaNomes(ListaEncadeada lista){
    No auxiliar = inicio;
     while(auxiliar != null) {
         if (auxiliar.excluido == false){
             System.out.println(auxiliar.getName());
     }

         auxiliar = auxiliar.proximo;

     }
}
```

O método listaNomes() percorre a lista e imprimindo no console todos os elementos nos quais o descritor excluido esteja com o valor *false*.