Atividade subturma A - Listas simplesmente encadeadas

Objetivo

O objetivo desta atividade é praticar a implementação e debugação de operações fundamentais em listas simplesmente encadeadas em Java. Os alunos trabalharão com métodos para adicionar e remover elementos, além de calcular o tamanho da lista de maneira iterativa. Esta atividade ajudará a reforçar o entendimento sobre manipulação de ponteiros e estruturas de dados dinâmicas.

Contexto

A classe `Lista` representa uma lista simplesmente encadeada com uma cabeça e cauda fictícias (`dummy nodes`), facilitando algumas operações. Cada nó (`No`) da lista armazena um valor inteiro e um ponteiro para o próximo nó. A lista mantém um atributo `tamanho` para rastrear o número de elementos contidos nela.

Estrutura da Lista

A classe `Lista` já contém métodos para adicionar um elemento no início da lista e imprimir todos os elementos da lista. Você irá trabalhar na implementação e correção de outros métodos essenciais para a manipulação da lista.

Tarefas

Exercício 1: Correção do Método `adicionarFim`

O método `adicionarFim` foi projetado para adicionar um novo elemento ao final da lista. No entanto, há um erro na lógica que impede que o método funcione corretamente em certas condições. Sua tarefa é debugar e corrigir esse erro.

Dicas:

- Verifique o loop que busca o último nó válido antes da cauda.
- Considere o caso especial quando a lista está vazia.

Exercício 2: Implementar o Método `obtemTamanho`

Complete o método `obtemTamanho` para calcular e retornar o tamanho da lista sem utilizar o atributo `tamanho`. Esse método deve iterar por toda a lista, contando os nós até alcançar a cauda fictícia.

Dicas:

- Inicialize uma variável de contagem local no início do método.
- Utilize um loop para percorrer a lista e incrementar a contagem para cada nó encontrado.

Exercício 3: Completar o Método `removerInicio`

O método `removerInicio` deve remover o primeiro elemento da lista. No entanto, o trecho responsável por atualizar os ponteiros da lista está faltando. Complete este método, garantindo que a lista mantenha sua integridade após a remoção.

Dicas:

- Verifique se a lista não está vazia antes de tentar remover um elemento.
- Atualize o ponteiro do nó cabeça para apontar para o novo primeiro elemento da lista.

Classe `No`

Lembre-se de que a classe `No` já está definida e não necessita de modificações. Cada instância de `No` representa um elemento da lista, contendo um valor inteiro (`valor`) e um ponteiro para o próximo nó (`proximo`).

Instruções Gerais

- Trabalhe em cada exercício sequencialmente, começando com o Exercício 1.
- Use a impressão de depuração (`System.out.println`) para ajudar a identificar o comportamento dos métodos durante a execução.
- Após concluir cada exercício, teste sua solução com vários casos de teste para garantir que seu método funciona corretamente em diferentes condições.

Avaliação

Sua solução será avaliada com base na corretude dos métodos implementados e corrigidos, bem como na eficiência da sua lógica de programação. A clareza

do seu código e a utilização de boas práticas de programação também serão consideradas.

Boa sorte, e divirtam-se aprimorando suas habilidades de programação e debugação!