BCC - Estruturas de Dados
Lab 03 - Listas Simplesmente Encadeadas - Parte 2
Versão orientada a objetos

Prof. Dr. Paulo César Rodacki Gomes
IFC - Instituto Federal Catarinense

15 de março de 2023

1 Objetivo

O objetivo desta atividade prática em laboratório é realizar a segunda etapa de implementação de listas simplesmente encadeadas. Por motivo de simplicidade, vamos implementar listas encadeadas para armazenar valores inteiros.

A atividade consiste em continuar a implementação em uma linguagem de programação orientada a objetos das classes `Lista` e `NoLista`, iniciada na atividade **lab02** de acordo com o diagrama de classes da figura 1.

A seguir, temos uma breve descrição dos novos métodos a serem implementados na atividade de hoje. Note que os novos métodos aparecem em negrito no diagrama de classes da figura 1.

Observação: o método `insereOrdenado` não precisa ser implementado.

Classe `Lista`:

1. `void retira(int v)`: remove da lista o primeiro nó que contiver o valor igual a v. Se nenhum nó com esse valor for encontrado, o método não retira nenhum nó da lista;
2. `libera`: este método destrói toda a lista. Note que, dependendo da linguagem de programação adotada, você precisará liberar a memória de cada um dos nós;

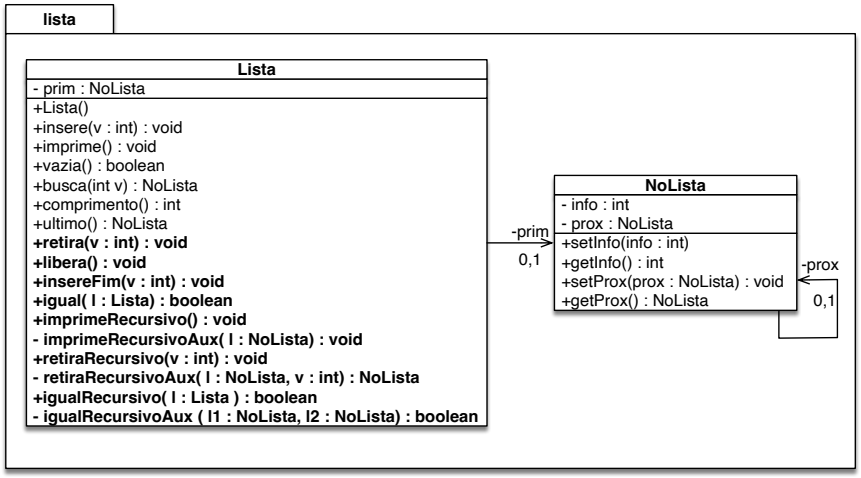

Figura 1: Diagrama de classes a ser implementado

3. `insereFim`: este método deve inserir o valor no final da lista. Você pode usar o método `ultimo` para chegar ao final da lista. Não se esqueça de verificar o caso de chamar este método em uma lista vazia;

4. `boolean igual(Lista l)`: verifica se a lista que executa o método (this ou self) é igual à lista `l` passada como parâmetro do método. Para as duas listas serem iguais, elas devem armazenar valores iguais e na mesma ordem;

Os demais métodos devem ser implementados com recursividade, e devem ter funcionamento equivalente as versões iterativas implementadas anteriormente.

Por exemplo: os métodos `public int comprimentoRec()` e `private int comprimentoRecAux(NoLista no)` são a versão recursiva do método `comprimento` implementado na lista de exercícios anterior. Nas operações implementadas recursivamente, sempre vamos ter um método público não recursivo que chama o método privado recursivo.

Observação: após implementar a lista, implemente uma classe chamada `ListaMain`, que deve conter o método `main` para testar e demonstrar o funcionamento da lista implementada.

IMPORTANTE: entregue somente os arquivos fonte separadamente no Google Classroom, sem "zipá-los".