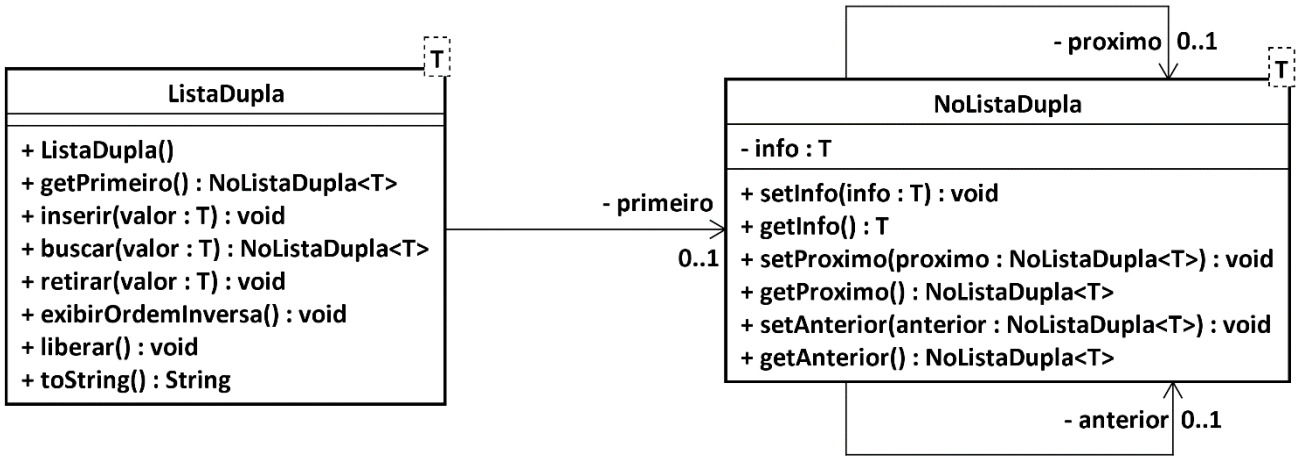
UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO
PROFESSOR GILVAN JUSTINO
ALGORITMOS E ESTRUTURAS DE DADOS

Lista de Exercícios 04

Questão 1

Implemente o diagrama de classes a seguir para exercitar a manipulação de uma lista duplamente encadeada.



A descrição dos métodos da classe `ListaDupla` consta a seguir:

a) `ListaDupla()`:construtor da classe. Deve estabelecer que a lista está vazia;
b) `getPrimeiro()`: método *getter* da variável `primeiro`;
a) `inserir(T)`: Deve armazenar o dado fornecido como argumento na estrutura de dados;
b) `buscar(T)`: Deve procurar na lista encadeada se há um nó cujo conteúdo seja igual à variável `valor` (utilizar igualdade de valores). Caso seja localizado, deverá retornar este nó (objeto da classe `NoListaDupla`). Se não for localizado, deverá retornar `null`;
c) `retirar(T)`: Deve retirar um nó da lista que contenha o valor informado como parâmetro para este método;
d) `exibirOrdemInversa()`: deve exibir o conteúdo armazenado nos nós da lista encadeada de forma que primeiro seja exibido o valor do último nó da lista e por último seja exibido o valor do primeiro nó da lista.
e) `liberar()`: Deverá limpar a estrutura de dados. Ao invés de simplesmente atribuir `null` para a variável de instância `primeiro`, remova todos os encadeamentos dos nós, isto é, atribua `null` para a associação `proximo` e `anterior` em todos os nós da lista;
f) `toString()`: deve retornar os valores armazenados na lista, desde o primeiro nó até o último, separando-os por vírgula.

Questão 2

Implemente o seguinte plano de testes:

Plano de testes PL01 – Validar funcionamento da classe ListaDupla			
Caso	Descrição	Entrada	Saída esperada
1	Testar método de inclusão de dados na lista encadeada, validando que as ligações proximo e anterior estejam consistentes	Criar uma lista duplamente encadeada. Adicionar os dados 5, 10, 15 e 20.	Percorrer a lista do primeiro até o último nó. A lista deverá retornar os nós contendo 20, 15, 10 e 5, nesta ordem. Percorrer a lista do último nó até o primeiro, os nós encontrados na navegação devem possuir os dados 5, 10, 15 e 20, nesta ordem
2	Buscar elemento no início	Criar uma lista duplamente encadeada. Adicionar os	Buscar(20) deve resultar no nó contendo 20.

	da estrutura de dados	dados 5, 10, 15 e 20.	
3	Buscar elemento no meio da estrutura de dados	Criar uma lista duplamente encadeada. Adicionar os dados 5, 10, 15 e 20.	Buscar(10) deve resultar no nó contendo 10
4	Remover elemento no início da lista	Criar uma lista duplamente encadeada. Adicionar os dados 5, 10, 15 e 20. Remover o elemento contendo 20	Percorrer a lista encadeada do início até o fim. Deve encontrar os nós contendo 15, 10 e 5, nesta ordem. Navegar na estrutura do último até o primeiro. Deve encontrar os nós contendo 5, 10 e 15, nesta ordem.
5	Retirar elemento no meio da lista	Criar uma lista duplamente encadeada. Adicionar os dados 5, 10, 15 e 20. Remover o elemento contendo 10	Percorrer a lista encadeada do início até o fim. Deve encontrar os nós contendo 20, 15 e 5, nesta ordem. Navegar na estrutura do último até o primeiro. Deve encontrar os nós contendo 5, 15 e 20, nesta ordem.
6	Retirar elemento no fim da lista	Criar uma lista duplamente encadeada. Adicionar os dados 5, 10, 15 e 20. Remover o elemento contendo 5	Percorrer a lista encadeada do início até o fim. Deve encontrar os nós contendo 20, 15 e 10, nesta ordem. Navegar na estrutura do último até o primeiro. Deve encontrar os nós contendo 10, 15 e 20, nesta ordem.
7	Liberar dados da lista	Criar uma lista duplamente encadeada. Adicionar os dados 5, 10, 15 e 20. Liberar dados da lista.	Buscar os nós que guardam os dados 5, 10, 15 e 20. Em seguida, liberar os dados da lista. Verificar que os nós obtidos antes da liberação possuem referência de anterior e proximo igual a null.