Escola Superior Dom Helder Câmara
Disciplina: Algoritmo e Estruturas de Dados II
Professor: Diego Silva Caldeira Rocha

Lista 10 – Lista Linear e Lista Dupla

Atenção Para soluções propostas a seguir não pode utilizar tipos abstratos de dados nativos da linguagem JAVA como: ArrayList, Stack, List, Queue, LinkedList e demais.
O objetivo desta prática é criar uma versão simplificada do ArrayList e LinkedList do JAVA.

Não esqueça de lançar uma mensagem de exceção no caso de operações incorretas e uso do encapsulamento nas boas práticas da programação.

1. Crie uma lista linear com tipo genérico com os métodos: inserir no início, remover no início, inserir no fim e remover no fim, remover na posição *i*, inserir na posição *i*, tamanho da lista, mostra elementos da lista, booleano que verifica se a lista está vazia..
Exemplo de prototipação:

```
public class ListaLinear<T> {
   private T[] array;
   private int n;
(..)
}
```

2. Crie um procedimento para realocação de memória; isto é, caso o usuário tente inserir um elemento com a lista cheia, você deverá clonar os dados existentes. Em seguida, crie um array com o dobro do tamanho do existente e copie os elementos clonados para o novo vetor.

3. Crie uma rotina de teste em uma main para os tipos *String*, *Integer*, *Float* e *Character*

4. Crie uma célula duplamente encadeada de tipo genérico, ou seja, é capaz de assumir tipo de dado (*Integer, Float, Long* e etc.).

Exemplo de prototipação:

```
lass CelulaDupla <T> {
        private T elemento;
        private CelulaDupla <T>ant;
        private CelulaDupla <T> prox;
(..)
}
```

6. Crie uma lista duplamente encadeada flexível com tipo genérico como descrito na questão anterior com os métodos: inserir no início, remover no início, inserir no fim e remover no fim, remover na posição *i*, inserir na posição *i*, tamanho da lista, mostra elementos da lista, booleano que verifica se a lista está vazia. Não esqueça de fazer o uso da célula implementada na questão anterior.

Exemplo de prototipação:

```
class ListaDupla<T> {
   private CelulaDupla primeiro;
   private CelulaDupla ultimo;
(..)
}
```

7. Faça uma nova implementação dos métodos: remover na posição *i* e inserir na posição *i,* que faça um número reduzido de operações, ou seja, verifica se o elemento a ser modificado está mais próximo do último ou do primeiro elemento e, posteriormente fazer a modificação.

8. Crie uma rotina de teste em uma *main* que você consiga validar todas as operações da classe Lista implementada nas questões anteriores.

Informações sobre cópias

As questões são individuais. Em caso de cópias de trabalho a pontuação será zero para os autores originais e copiadores. Não serão aceitas justificativas como: “Fizemos o trabalho juntos, por isso estão idênticos”.