INSTITUTO FEDERAL
Rio Grande do Sul | Campus Farroupilha

ATIVIDADE PRÁTICA 3 - DESENVOLVIMENTO DE TAD PARA LISTA ENCADEADA (ESTRUTURA SEQUENCIAL DINÂMICA)

CURSO: TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
DISCIPLINA: ESTRUTURAS DE DADOS
PERÍODO LETIVO: 2024-01
PROFESSOR: FELIPE MARTIN SAMPAIO

INSTRUÇÕES:

- Atividade avaliativa que será contabilizada como parte do instrumentos "Atividades práticas" da etapa N1;
- Deve ser enviado na tarefa disponível no Moodle:
  - Códigos .java com a implementação do TAD e função principal para testes;
- Prazo para entrega: **xxxxxxxx**;

OBJETIVOS:

- Compreender o desenvolvimento de um Tipo Abstrato de Dado por meio do paradigma de orientação a objetos;
- Aplicar a compreensão sobre o funcionamento das operações planejadas para a estrutura sequencial dinâmica (listas encadeadas) na implementação dos métodos na linguagem de programação Java.

ESPECIFICAÇÃO:

1. Desenvolver um Tipo Abstrato de Dado (TAD), utilizando a linguagem Java, para a estrutura de dados sequencial dinâmica (lista encadeada).

2. Este TAD deve utilizar como armazenamento interno dos elementos da estrutura:

```
public class Nodo {
    public int elem;
    public Nodo ant, prox;
    (...)
```

```
public class ListaTAD {
    private Nodo inicio, fim;
    private int numElementos;
    (...)
```

3. Além disso, o TAD desenvolvido deve oferecer as seguintes operações:

   a. Inicialização da lista vazia

```
public ListaTAD() { … }
```

b. Acesso ao tamanho da estrutura (número de elementos)

```
public int tamanho() { … }
```

c. Teste se a estrutura encontra-se vazia

```
public boolean estaVazia() { … }
```

d. Impressão dos elementos da estrutura na tela

```
public void imprime() { … }
```

e. Impressão dos elementos da estrutura na tela em ordem reversa

```
public void imprimeReverso() { … }
```

f. Criação de uma string representando os elementos da estrutura

```
public String toString() { … }
```

g. Acesso a um elemento específico na estrutura a partir de uma posição informada como parâmetro
Observação: verificar se a posição solicitada está dentro dos limites da estrutura.

```
public int acessa(int pos) { … }
```

h. Inserção de um elemento na estrutura:

I. No final (como último elemento)

```
public void insereFinal(int valor) { … }
```

II. No início (como primeiro elemento)

```
public void insereInicio(int valor) { … }
```

III. Em qualquer posição (a partir de uma posição informada como parâmetro)

```
public void insere(int pos, int valor) { … }
```

i. Remoção de um elemento da estrutura:
Observação: caso a estrutura esteja vazia, deve ser reportado um erro indicando que não há elementos a serem removidos.

I. Do final (último elemento)

```
public void removeFinal() { … }
```

II. Do início (primeiro elemento)

```
public void removeInicio() { … }
```

III. De qualquer posição (a partir de uma posição informada como parâmetro)

```
public void remove(int pos) { … }
```

j. Pesquisa por um elemento

```
public boolean pesquisa(int valor) { … }
```

k. Altera o valor de um elemento (a partir de uma posição informada como parâmetro)
Observação: verificar se a posição solicitada está dentro dos limites da estrutura.

```
public void altera(int pos, int valor) { … }
```

l. Limpa a estrutura (remove todos os elementos)

```
public void limpa() { … }
```

4. Implementar uma função principal (`main`) para a criação de um objeto do TAD desenvolvido, como forma de testar as implementações das operações. Exemplo:

```
    public static void main(String[] args) {
        ListaTAD tad = new ListaTAD();

        //TODO implementar

    }
```