Instituto Federal de Sergipe
Análise e Desenvolvimento de Sistemas
Disciplina: Estrutura de Dados I

**Professor:** Francisco Rodrigues Santos
**Semestre:** 2024.1

**Nome:** _____

# — AV 1—

---

**- Os códigos deverão ser enviados para o *github* e compartilhados com o usuário *frchico*.**
**- É permitido o uso de códigos próprios.**
**- Será considerada somente a primeira resposta, para códigos com alto grau de semelhança entre alunos.**
**- Não será permitido o uso de coleções implementadas pelo Java, como ArrayList.**
**- Os nomes dos métodos, bem como os tipos de dados utilizados precisam ser os mesmos definidos na prova.**

---

**Implemente uma versão da lista simplesmente encadeada (`ListaEnc` e `No`) para que seja possível atender as especificações das classes/interfaces definidas (`Lista` e `IProva`), conforme solicitado abaixo:**

1. (1,0) Implemente o método `compareTo` na classe `No`.
2. (1,0) Implemente os métodos básicos da classe `ListaEnc`.
3. (2,0) Implemente os métodos que informam se a lista está ou não ordenada.
4. (3,0) Implemente o método `void importarListas(Lista<T> l1, Lista<T> l2)` da interface `IProva`. O objetivo é permitir a inclusão dos valores oriundo de outras duas listas, que foram recebidas via parâmetro (`l1` e `l2`), ao conteúdo atual da lista;

   `void importarListas(Lista<T> l1, Lista<T> l2) throws Exception;`

5. (3,0) Implemente a sobrecarga do método `importarListas` para permitir a inclusão da interface `IProva`. O objetivo é realizar a adição dos valores oriundo de outras duas listas, que foram recebidas via parâmetro (`l1` e `l2`), ao conteúdo atual da lista **GARANTINDO**, quando o parâmetro `manterOrdenacao` é **verdadeiro**, as seguintes observações:

   (a) Todas as listas envolvidas sejam também ordenadas. Em caso de falha lançar uma exceção;
   (b) O Resultado seja também uma lista ordenada

`void importarListas(Lista<T> l1, Lista<T> l2, boolean manterOrdenacao) throws Exception;`

Boa prova!

```java
package br.edu.ifs.cads.ed1;

public interface IProva<T extends Comparable> {
    /**
     * Adiciona o conteúdo da <param>l1</param> e da lista <param>l2</param> à lista atual.
     * @param l1 primeira lista contendo valores
     * @param l2 segunda lista contendo valores
```

```
8       */
9       void importarListas(Lista<T> l1, Lista<T> l2) throws Exception;
10
11      /**
12       * Adiciona o conteúdo da <param>l1</param> e da lista <param>l2</param> à lista
             atual.
13       * Se o <param>manterOrdenacao</param> for verdadeiro, verificar se a lista de
             destino (a atual),
14       * as listas <param>l1</param> e <param>l2</param> também estão ordenadas.
15       * Caso negativo lançar um erro.
16       * @param l1 primeira lista contendo valores
17       * @param l2 segunda lista contendo valores
18       * @param manterOrdenacao parâmetro que indica se é necessário manter a ordenação
             dos dados. Caso seja positivo,
19       *                        todas as listas envolvidas devem ser ordenadas.
20       *                        Caso o parâmetro tenha seu valor como false, não há
             necessidade de verificação se as
21       *                        listas afetadas estão, originalmente, ordenadas.
22       */
23      void importarListas(Lista<T> l1, Lista<T> l2, boolean manterOrdenacao) throws
            Exception;
24      /**
25       * Inpeciona a lista atual para saber se ela está ou não ordenada.
26       * @return true em caso da lista ordenada e false se ela não estiver ordenada.
27       * @throws Exception
28       */boolean estahOrdenada()  throws Exception;
29
30      /**
31       * Inpeciona uma lista para saber se ela está ou não ordenada.
32       * @param lista lista a inspecionada
33       * @return true em caso da lista ordenada e false se ela não estiver ordenada.
34       * @throws Exception
35       */
36      boolean estahOrdenada(Lista<T> lista)  throws Exception;
37
38  }
```

Interface IProva ordenada

```
1   package br.edu.ifs.cads.ed1;
2
3   public class No<T extends Comparable> implements Comparable<T>{
4       private T dado;
5       private No<T> proximo;
6       public No(T item){
7           this.dado = item;
8       }
9
10      public T getDado() {
11          return dado;
12      }
13
14      public void setDado(T dado) {
15          this.dado = dado;
16      }
17
18      public No<T> getProximo() {
19          return proximo;
20      }
21
22      public void setProximo(No<T> proximo) {
23          this.proximo = proximo;
24      }
25
26      @Override
27      /**
```

```
28      * Método a ser implementado na prova.
29      */
30     public int compareTo(T o) {
31         return 0;
32     }
33 }
```

Classe No. Personalize caso necessário.

```
1  package br.edu.ifs.cads.ed1;
2
3  public abstract class Lista<T extends Comparable>  {
4
5
6      /**
7       * Adiciona um elemento no final da lista
8       * @param elemento
9       * @throws Exception
10      */
11     public abstract void incluir(T elemento) throws Exception;
12
13     /**
14      * Adiciona um elemento no início da lista
15      * @param elemento
16      * @throws Exception
17      */
18     public abstract void incluirInicio(T elemento) throws Exception;
19
20     /**
21      * Adiciona um elemento em uma posição específica da lista
22      * @param elemento
23      * @param posicao
24      * @throws Exception
25      */
26     public abstract void incluir(T elemento, int posicao) throws Exception;
27
28     /**
29      * Retorna o elemento que está na posição
30      * @param posicao
31      * @return
32      * @throws Exception
33      */
34     public abstract T get(int posicao)  throws Exception;
35
36     /**
37      * Retorna a posição do elemento
38      * @param elemento
39      * @return
40      * @throws Exception
41      */
42     public abstract int getPosElemento(T elemento)  throws Exception;
43
44     /**
45      * Remove o elemento da posição
46      * @param posicao
47      * @throws Exception
48      */
49     public abstract void remover(int posicao) throws Exception;
50
51     /**
52      * Remove todos os elementos da lista.
53      */
54     public abstract void limpar();
55
56     /**
57      * Retorna a quantidade de elementos na lista
```

```
58      * @return Quantidade de elementos na lista
59      */
60     public abstract int getTamanho();
61 
62     /**
63      * Indica se contém ou não o elemento na lista
64      * @param elemento
65      * @return Veradeiro, se achou ou falso caso contrário.
66      * @throws Exception
67      */
68     public abstract boolean contem(T elemento) throws Exception;
69 }
```

Classe Lista.

```
1  package br.edu.ifs.cads.ed1;
2  
3  public class ListaEnc<T extends Comparable<T>> extends Lista<T> implements IProva<T> {
4  
5      public ListaEnc(){
6  
7      }
8  
9      @Override
10     public void incluir(T elemento) throws Exception {
11         throw new Exception("Não implementado");
12     }
13 
14 
15     public T get(int posicao)  throws Exception {
16         throw new Exception("Posição solicitada não existe na lista");
17     }
18 
19 
20     public int getPosElemento(T elemento)  throws Exception {
21         throw new Exception("Elemento não localizado");
22     }
23 
24     @Override
25     public void incluirInicio(T elemento) throws Exception {
26         throw new Exception("Não implementado");
27     }
28 
29     @Override
30     public void incluir(T elemento, int posicao) throws Exception {
31         throw new Exception("Não implementado");
32     }
33 
34     @Override
35     public void remover(int posicao) throws Exception {
36         throw new Exception("Não implementado");
37     }
38 
39     @Override
40     public int getTamanho() {
41         return Integer.MIN_VALUE;
42     }
43 
44     public void limpar() {
45     }
46 
47 
48 
49     @Override
50     public boolean contem(T elemento) throws Exception {
51         throw new Exception("Não implementado");
```

```
52      }
53
54      @Override
55      public void importarListas(Lista<T> l1, Lista<T> l2) throws Exception {
56
57      }
58
59      @Override
60      public void importarListas(Lista<T> l1, Lista<T> l2, boolean manterOrdenacao)
            throws Exception {
61
62      }
63
64      @Override
65      public boolean estahOrdenada() throws Exception {
66          return false;
67      }
68
69      @Override
70      public boolean estahOrdenada(Lista<T> lista) throws Exception {
71          return false;
72      }
73  }
```

Classe Lista Encadeada.