AVA3

Algoritmos e Estrutura de Dados - Noturno

Iniciado em	quinta, 21 mar 2024, 20:26
Estado	Finalizada
Concluída em	quinta, 21 mar 2024, 21:10
Tempo empregado	43 minutos 56 segundos
Avaliar	**0.25** de um máximo de 4.00(**6**%)

AVA3

Completo

Atingiu 0.25 de 1.50

A partir da lista de exercícios L02, adicione o seguinte método:

`public void inserir(int posicao, T valor)`

O objetivo deste método será adicionar `valor` na lista, na posição indicada pelo parâmetro `posicao`. Os elementos a partir da posição indicada deverão ser deslocados para uma posição adiante.

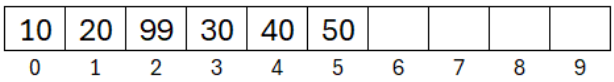
Por exemplo, considere o seguinte fragmento de código:

```
ListaEstatica<Integer> lista = new ListaEstatica<>();

lista.inserir(10);
lista.inserir(20);
lista.inserir(30);
lista.inserir(40);
lista.inserir(50);

lista.inserir(2, 99);
```

Após a chamada do método `inserir(2, 99)`, o vetor `info` encapsulado por `lista` deve conter os seguintes dados:



Além disso, o tamanho da lista deve ser alterado para 6, para indicar que existem 6 dados armazenados.

Sua implementação deverá recusar posições inválidas na lista (índice negativo), mas deve aceitar índice que corresponda à primeira posição livre da lista. Ou seja, dada a lista acima, deve ser admissível invocar `inserir(6, 88)`, por exemplo, já que o índice 6 é a primeira posição livre. Valores de índice superior a 6 deve causar o lançamento de exceção. Para recusar posições inválidas, lance a exceção IndexOutOfBoundsException.

O novo método `inserir(int,T)` deve considerar a possibilidade de expandir o vetor caso não haja posição disponível.

```
public void inserir(int posicao, T valor){
    if(posicao < 0 || posicao > tamanho + 1){
        throw new IndexOutOfBoundsException();
    }
    Object[] novo;
    if(tamanho == info.length){
        novo = new Object[info.length+10];
    }
    else{
        novo = new Object[info.length];
    }
    for(int i = 0; i< posicao; i ++){
        novo[i] = info[i];
    }
    novo[posicao] = valor;
```

Comentário:
*** Casos que falharam:
Inserir no início da estrutura
Inserir no meio da estrutura
Inserir no final da estrutura
Inserir exigindo redimensionamento

AVA3

Completo

Atingiu 0.00 de 2.50

A partir da lista L04 acrescente o seguinte método na classe `ListaDupla`:

`public ListaDupla<T> criarSubLista(int inicio, int fim)`

Este método deve criar uma nova lista encadeada, sendo que seu conteúdo deve ser originado da lista corrente. O conteúdo da nova lista deve ser constituído dos dados armazenados nos nós cujos índices estejam compreendidos entre `inicio` e `fim` (inclusive estes). Por exemplo, se `inicio` for 2 e `fim` for 5, devem ser copiados os dados do 3º, 4º, 5º e 6º nós para a nova lista. Ao copiar os dados para a nova lista, os dados devem ser mantidos na mesma ordem da lista original. O método deve retornar a nova lista construída.

Por exemplo, considere a lista criada abaixo:

```
ListaDupla<Integer> lista1 = new ListaDupla<>();
lista1.inserir(70);
lista1.inserir(60);
lista1.inserir(50);
lista1.inserir(40);
lista1.inserir(30);
lista1.inserir(20);
lista1.inserir(10);
```

Observe que a `lista1` contém os seguintes dados (nesta ordem): `10,20,30,40,50,60,70`.

Considere que logo em seguida seja executado:

```
ListaDupla<Integer> lista2;
lista2 = lista1.criarSubLista(2,5);
```

Ao executar o método `criarSubLista()` uma nova lista deve ser criada. No caso, a lista criada é referenciada por `lista2` e irá conter os valores `30,40,50,60` (nesta ordem. Esta seria a mesma saída de `toString()`).

a exceção `IndexOutOfBoundsException`. A lista original não pode ser modificada. Será avaliada a eficiência da implementação.

Publique aqui o método `criarSubLista()` que você criou.

```
public ListaDupla<T> criarSubLista(int inicio, int fim) {
    NoListaDupla<T> p = primeiro;
    ListaDupla<T> lista = new ListaDupla<>();
    int tamanho = 0;
    while (p != null) {
        p = p.getProximo();
        tamanho++;
    }
    if (inicio > tamanho || inicio < 0 || fim > tamanho ||
        throw new IndexOutOfBoundsException();
    }
    int tamanhoSub = 0;
    while (p != null) {
        if (tamanhoSub >= inicio && tamanhoSub <= fim) {
            lista.inserir(p.getInfo());
```

Comentário:
A lista resultante sempre está vazia.

É desnecessário calcular a quantidade total de nós para criar a sublista.

Atividade anterior

Seguir para...

AVA3

Resumo de retenção de dados
Baixar o aplicativo móvel.