

# [ED269] Inserindo e removendo numa lista

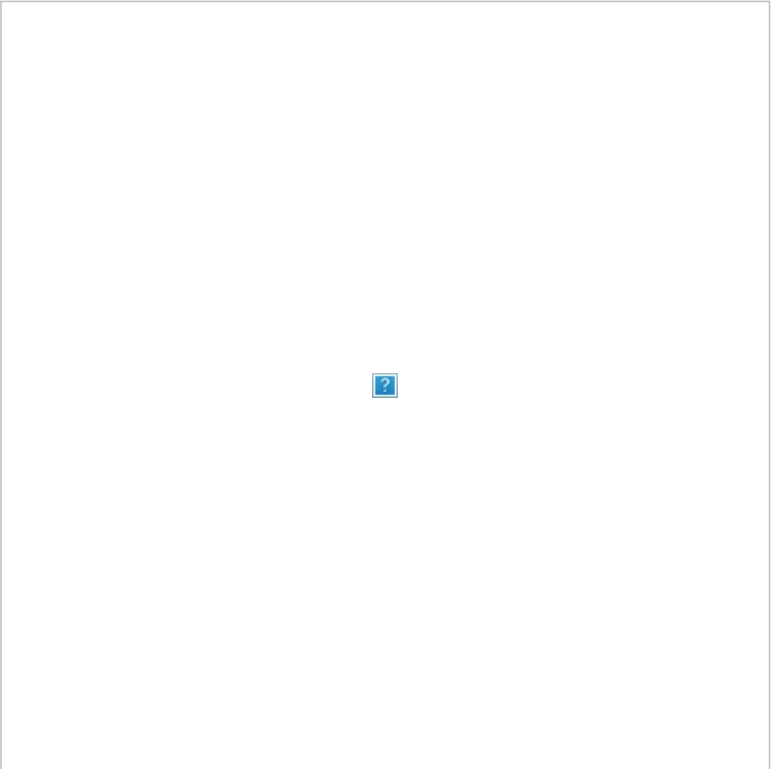
Neste problema deverá apenas submeter uma classe `SinglyLinkedList<T>` (e não um programa completo).

## Código Base

Use como base a classe `SinglyLinkedList<T>` ([ver código](#) | download de [Node.java](#) e [SinglyLinkedList.java](#)), que representa uma lista ligada simples e tem disponíveis métodos para adicionar ou remover um elemento no início ou no final, devolver o tamanho, saber se a lista está vazia ou retornar representação em *string* para escrita (tal como dado nas aulas).

## Métodos a Implementar

Deve acrescentar à classe dada os seguintes métodos (**não modificando nenhum dos métodos já existentes no código base**):



- public void duplicate(int pos)** (40% da cotação)  
Deve **duplicar o nó da posição *pos***, modificando a própria lista. Por outras palavras, deve criar um novo nó contendo o mesmo valor do nó da posição *pos* e inseri-lo imediatamente a seguir. Por exemplo, se *list* for `{'a','b','c'}`, uma chamada a `list.duplicate(0)` deve fazer com que *list* fique a ser `{'a','a','b','c'}`, uma chamada a `list.duplicate(1)` deve fazer com que *list* fique a ser `{'a','b','b','c'}` e uma chamada a `list.duplicate(2)` deve fazer com que *list* fique a ser `{'a','b','c','c'}`. Assuma que as posições começam em zero. É também garantido que nos testes feitos ao seu método a posição é válida, ou seja,  $0 \leq pos < tamanho\_da\_lista$ .
- public SinglyLinkedList<T> remove(int[] pos)** (60% da cotação)  
Deve **devolver uma lista que é uma cópia da original, mas com os elementos das posições correspondentes aos inteiros contidos no array *pos[]* removidos**. Por exemplo, se *list* for `{2,4,6,8,10}`, uma chamada a `list.remove([0,1,3])` deve devolver uma nova lista com conteúdo `{6,10}` e uma chamada a `list.remove([2,3,4])` deve devolver uma nova lista com conteúdo `{2,4}`. A lista inicial não deve ser modificada. Assuma que as posições começam em zero. É também garantido que **os números do array *pos[]* vêm por ordem estritamente crescente** e todas as posições são válidas (existem na lista). Se o array de posições for vazio, a lista devolvida deve ser uma cópia da original.

## Notas

- Pode submeter código com apenas um dos métodos implementados (para obter pontuação parcial).
- Não se esqueça de garantir que o atributo `size` fica correto.
- Em todos os casos de teste as listas e arrays têm tamanho máximo de 50 elementos, com a exceção do **último caso de teste do método *remove* (valendo 10% da cotação)**, onde a lista e o array podem ter até 50 mil elementos, pelo que nesse caso a sua solução não poderá ser quadrática (ou pior) no número de elementos da lista original ou array para passar no tempo limite.
- Pode implementar métodos auxiliares, se quiser.
- Para testar na sua máquina deve criar uma lista (pode criar no código ou ler a partir de um input) e chamar o método correspondente.

## Exemplos de Input/Output para o método *duplicate*

Lista inicial	Chamada	Estado da lista depois da chamada
<code>list = {'a','b','c','d'}</code>	<code>list.duplicate(0)</code>	<code>list = {'a','a','b','c','d'}</code>
<code>list = {'a','b','c','d'}</code>	<code>list.duplicate(1)</code>	<code>list = {'a','b','b','c','d'}</code>

list = {'a','b','c','d'}	list.duplicate(3)	list = {'a','b','c','d','d'}
list = {2,4,42,8,10}	list.duplicate(2)	list = {2,4,42,42,8,10}

Exemplos de Input/Output para o método *remove*

Lista inicial	Chamada	O que deve ser devolvido
list = {2,4,8,6,10,12}	list.remove([])	new_list = {2,4,8,6,10,12}
list = {2,4,8,6,10,12}	list.remove([0,1,2,3,4,5])	new_list = {}
list = {2,4,8,6,10,12}	list.remove([0,2,4])	new_list = {4,6,12}
list = {2,4,8,6,10,12}	list.remove([1,3,4,5])	new_list = {2,8}
list = {'a','b','c','d'}	list.remove([1,2])	new_list = {'a','d'}
list = {"quarenta","e","dois"}	list.remove([0,2])	new_list = {"e"}