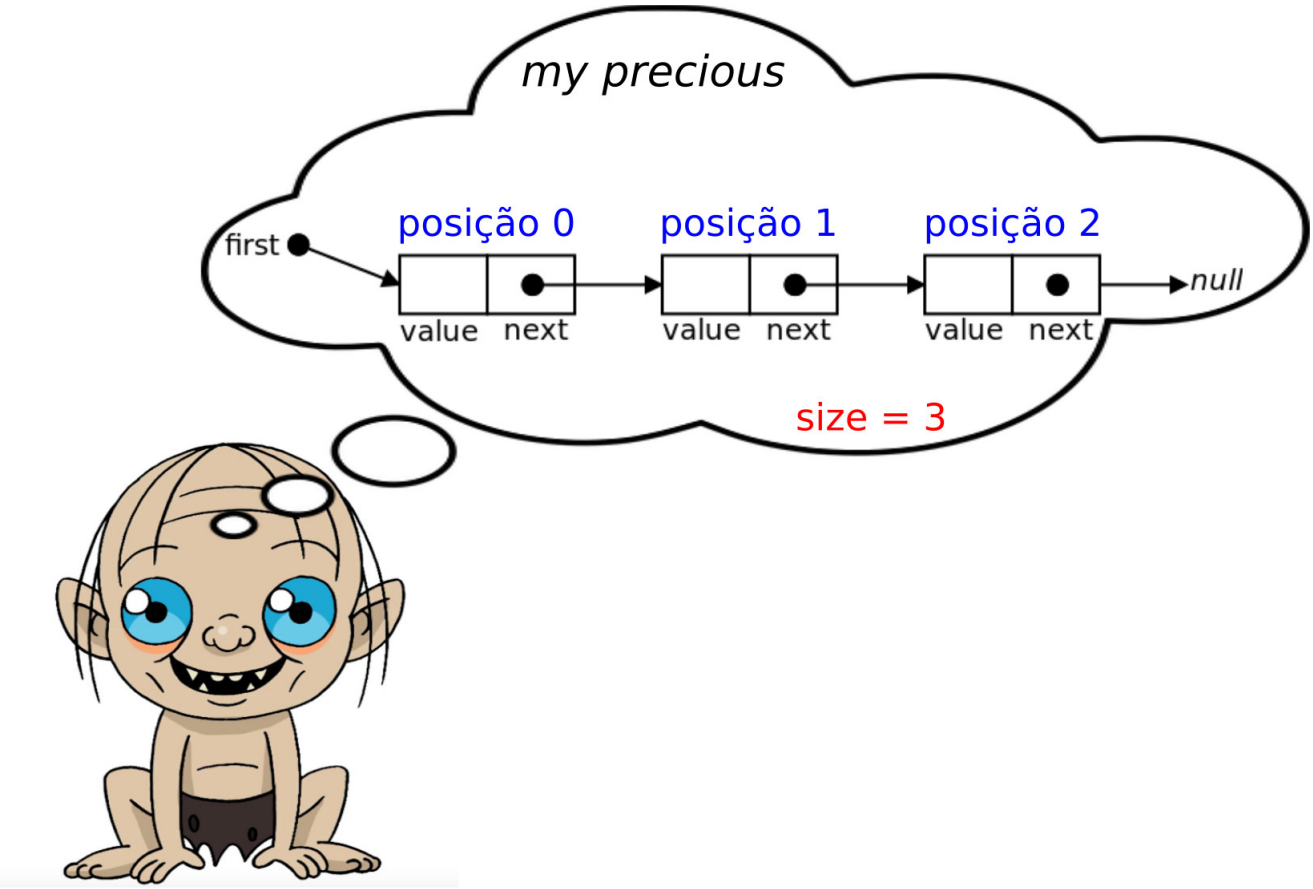
Para efeitos da nota atribuida à resolução de exercícios ao longo do semestre - **Submeter até 23:59 de 22 de Maio**
(o problema continuará depois disponível para submissão, mas sem contar para a nota)
[para perceber o contexto do problema deve ler o guião da aula #09]

[ED232] O Senhor das Listas

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).

Problem

One linked list to rule them all,
one linked list to find them,
One linked list to bring them all
and in the darkness bind them.

A população da *Terra dos Dados* deposita toda a sua fé em ti! Depois de teres conseguido obter o código base da classe *SinglyLinkedList<T>*, arrancada das mãos dos docentes que a admiravam dizendo *"my precious"*, tens agora de provar que sabes dominar o seu poder, implementando três novos métodos da classe.

Métodos a Implementar

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

- **public SinglyLinkedList<T> reverse()** *(30% da cotação)*
  Deve **devolver uma nova lista que é igual à lista original, mas invertida**. Por exemplo, se *list* for {2,4,6,8}, uma chamada a *list.reverse()* deve devolver uma nova lista com conteúdo {8,6,4,2}. A lista inicial não deve ser modificada.

- **public int[] occurrences(T elem)** *(30% da cotação)*
  Deve **devolver um array contendo as posições (por ordem crescente) de todas as ocorrências** do elemento *elem* na lista. Se não existir nenhuma ocorrência deve devolver *null*. As posições da lista começam em 0. Por exemplo, se *list* for {2,5,1,1,2,1}, uma chamada a *list.occurrences(1)* deve devolver [2,3,5], uma chamada a *list.occurrences(2)* deve devolver [0,4] e uma chamada a *list.occurrences(3)* deve devolver *null*. A lista inicial não deve ser modificada. **Dica: para comparar valores genéricos deve usar o .equals e não o ==**

- **public void remove(SinglyLinkedList<T> toRemove)** *(40% da cotação)*
  Deve **remover da lista todos os elementos que estão na lista *toRemove***. Por exemplo, se *list* for {'a','b','d','a','c'}, uma chamada a *list.remove({'c','a'})* deve fazer com que *list* fique a ser {'b','d'}, uma chamada a *list.remove({'a'})* deve fazer com que *list* fique a ser {'b','d','c'} e uma chamada a *list.remove({'a','b','c','d','e'})* deve fazer com que *list* fique a ser {} (lista vazia).

Notas

- Pode submeter código com apenas alguns dos métodos implementados (para obter pontuação parcial).
- Em todos os casos de teste as listas têm tamanho máximo de 100 elementos, com a excepção do último caso de teste do método *remove* (valendo 10% da cotação), onde a lista inicial pode ter 50 mil elementos (a lista dos elementos a remover continua limitada a tamanho 100), pelo que nesse caso a sua solução não poderá ser quadrática (ou pior) no número de elementos da lista original para passar no tempo limite.
- Não se esqueça de garantir que no final o atributo *size* também fica correto .
- Cuidado com os casos limites (ex: lidar com listas vazias, remover no início ou no final da lista; etc).
- 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 *reverse*

Lista inicial	Chamada	O que deve ser devolvido
list = {2,4,6,8}	list.reverse()	new_list = {8,6,4,2}
list = {'a','b','c'}	list.reverse()	new_list = {'c','b','a'}
list = {"edados"}	list.reverse()	new_list = {"edados"}
list = {}	list.reverse()	new_list = {}

Exemplos de Input/Output para o método *occurrences*

Lista inicial	Chamada	O que deve ser devolvido
list = {2,5,1,1,2,1}	list.occurrences(1)	[2,3,5]
list = {2,5,1,1,2,1}	list.occurrences(2)	[0,4]

list = {2,5,1,1,2,1}	list.occurrences(3)	*null*
list = {'a','a','a','a'}	list.occurrences('a')	[0,1,2,3]
list = {"estruturas","de","dados"}	list.occurrences("dados")	[2]
list = {'a','n','a','n','a','s',}	list.occurrences('a')	[0,2,4]

Exemplos de Input/Output para o método *remove*

Lista inicial	**Chamada**	**Estado da lista depois da chamada**
list = {'a','b','d','a','c'}	list.remove({'c','a'})	list = {'b','d'}
list = {'a','b','d','a','c'}	list.remove({'a'})	list = {'b','d','c'}
list = {'a','b','d','a','c'}	list.remove({'a','b','c','d','e'})	list = {}
list = {42,22,42,42,22,42}	list.remove({42,1})	list = {22,22}
list = {"ola","ola","mundo","ola"}	list.remove("ola")	list = {"mundo"}
list = {1,2,3,3,2,1}	list.remove(3)	list = {1,2,2,1}
