Este segundo trabalho de programação em Java consiste na implementação parcial de uma lista paramétrica recorrendo a algoritmos recursivos para fazer a travessia da lista. E de um programa de testes unitários para validar a correção dos métodos já implementados ou por implementar.

Começe por ler a descrição da classe ***IntId*** (documento **IntId.html**). Em caso algum deve alterar este tipo de dados que serve para testar os algoritmos da lista ligada.

Pretende-se implementar o tipo de dados ***RecLinkedList*** utilizando programação defensiva com lançamento de excepções no caso de situações de dados anormais que violam as pré-condições e/ou as pós-condições dos métodos. Para esse efeito é disponibilizada uma classe de excepção designada por *EmptyListException* para indicar quando uma lista está vazia. As outras classes de excepção utilizadas *IndexOutOfBoundsException* e *NoSuchElementException* já existem na linguagem Java.

1. Começe por ler a descrição da classe ***RecLinkedList*** (documento **RecLinkedList.html**). Depois implemente as funções privadas auxiliares que fazem efetivamente as travessias recursivas da lista: *insertPos*; *copy*; *reverse*; *searchDown*; *searchUp*; *removeAllElements*; *removeFirstElement*; *removeLastElement*; *firstIndexOf*; e *lastIndexOf*. Algumas das quais já deve ter implementado no ano passado em Programação II. Em caso algum deve alterar os restantes métodos.

2. Complete o programa de testes unitários **TestRecList.java** que testa alguns dos métodos da lista recursiva. Acrescente testes utilizando a instrução *assert* para validar a correção dos métodos em falta ou parcialmente testados. Não se esqueça também de testar todos os métodos para uma lista vazia, por exemplo num programa de simulação à parte, para assegurar que os métodos estão implementados de forma robusta.

Nota muito importante: tenha em atenção que as funções auxiliares recursivas ***remove*** além de desligarem o elemento pretendido da lista trazem a sua referência para que o método invocador possa devolver ao programa o elemento removido da lista. Para ser possível devolver simultanemente a referência do elemento e a referência do elemento seguinte da lista para refazer a ligação da lista é preciso usar um tipo de dados interno (**private class DelNode**) que armazena duas referências. Para perceber como se pode implementar uma função desta maneira fica como exemplo resolvido o método ***removePosElement***. Só desta forma é que se pode implementar o método ***removeLastElement***.

O trabalho deve ser entregue até 5 de Junho de 2022. Apenas um dos alunos do grupo deve enviar por email para adrego@ua.pt um arquivo zip contendo apenas os ficheiros finais da classe e do programa de testes unitários: **RecLinkedList.java** e **TestRecList.java**.

Para quaisquer questões de Java e de listas ligadas pode e deve contactar-me por *email* com dúvidas acerca do trabalho.