

Lista de Exercícios 3

1. Implemente uma árvore binária encadeada de inteiros. Sua solução deve conter duas classes, Position e LinkedBinaryTree. A primeira representa um objeto posição da árvore, enquanto que a segunda representa a árvore como um todo. Abaixo, a lista de atributos e métodos que você deve escrever, para cada classe.

Position

- Atributos
  - Elemento
  - Pai
  - Filho Esquerdo
  - Filho Direito

LinkedBinaryTree

- Atributos
  - Raíz
  - Tamanho
- Métodos
  - Getters e Setters
  - boolean isEmpty() //retorna se a árvore é vazia
  - boolean isInternal(Position v) //retorna se o nó é interno
  - boolean isExternal(Position v) //retorna se o nó é externo
  - boolean isRoot(Position v) //retorna se o nó é raiz
  - boolean hasLeft(Position v) //retorna se o nó tem filho esquerdo
  - boolean hasRight(Position v) //retorna se o nó tem filho direito
  - Position left(Position v) //retorna o filho esquerdo de um nó
  - Position right(Position v) //retorna o filho direito de um nó
  - Position parent(Position v) //retorna o pai de um nó
  - List<Position> children(Position v) //retorna uma lista com os filhos de um nó
  - void addRoot(int i) //adiciona a raiz da árvore
  - void insertLeft(Position v, int i) //adiciona um elemento na esquerda do nó
  - void insertRight(Position v, int i) //adiciona um elemento na direita do nó
  - String toStringPreOrder() // retorna uma String com os dados da árvore em pré-ordem
  - String toStringPosOrder() // retorna uma String com os dados da árvore em pós-ordem
  - int detph(Position v) // retorna a profundidade de um nó