Instituto Superior de Engenharia do Porto

**Licenciatura em Engenharia Informática**

**ESINF 2020/2021**

# Trabalho Prático 3

**Autores:**

1190402 António Fernandes

1191045 Rui Soares

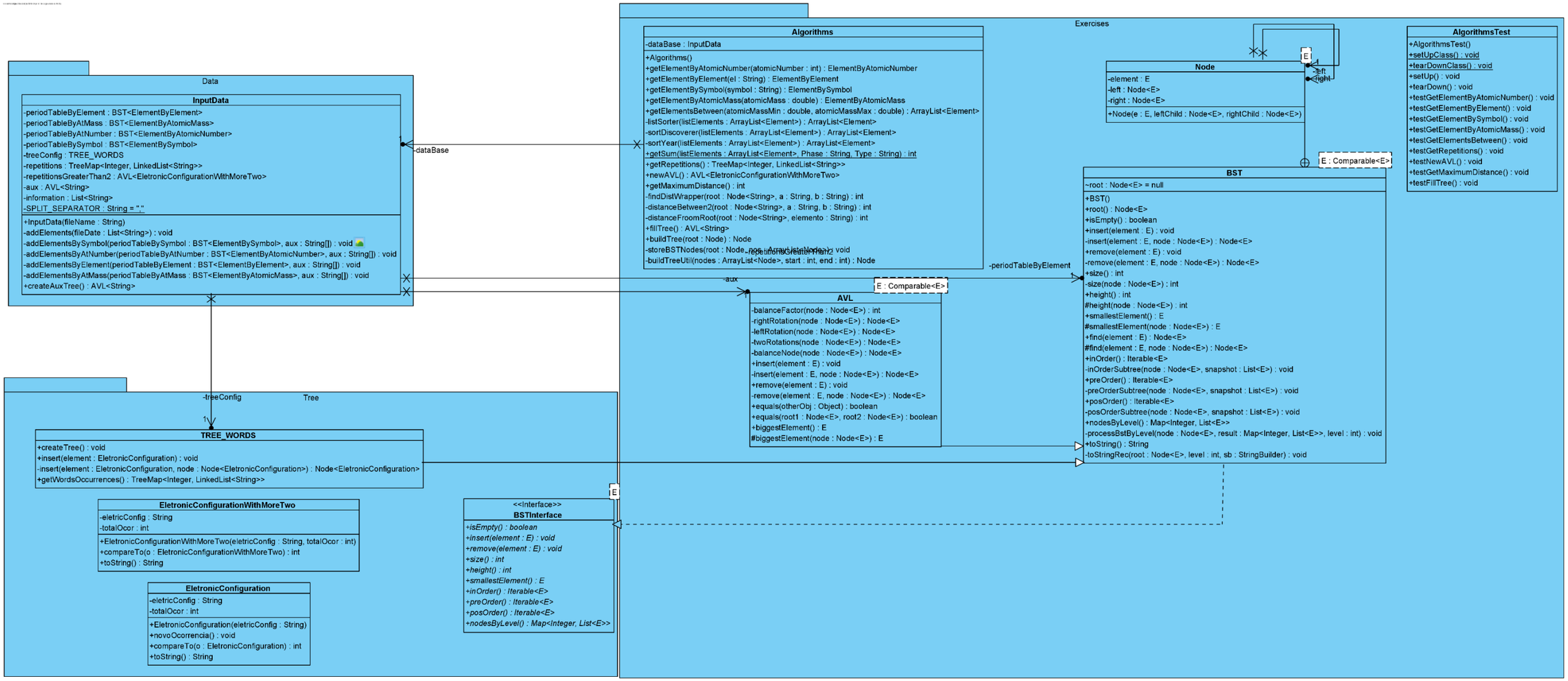
**Turma:** 2DK

**Data:** 23/12/20

**Docente:** Ana Madureira AMD

# Índice:

# Diagrama de Classes



# Análise de Complexidade das Funcionalidades Implementadas

No exercício 1

a) Implementamos métodos que através das 4 árvores BST iam pesquisar o elemento em questão por massa atómica, nº atómico, por elemento e por símbolo.

Complexidade->Todos os métodos desta alínea são O(n), percorre a lista toda

__

b) Implementamos métodos que partindo da árvore BST organizada por massa atómica através dos valores mínimos e máximos retornamos os que pertencem a esse intervalo ordenado por descobridor (crescente) e ano de descoberta (decrescente) juntamente com um sumário do número de elementos devolvidos agrupados por Type e Phase.

Complexidade->

O(n^2)

__

No exercício 2

a) Recorremos à classe TREE_WORD para ler do ficheiro as configurações eletrónicas e criamos uma classe que guardava as strings e o nº de ocorrências depois teríamos criado um TreeMap com toda a informação e retornamos esse tree map.

Complexidade->O(n^2)

__

b) A partir do tree map da alínea anterior tiramos as entradas repetidas duas ou menos vezes e inserimos numa AVL.

Complexidade-> Pior Caso:O(n^2 log^2(n))) Melhor Caso:O(1)

c) Criamos um método que calculava os elementos mais distantes a partir da árvore AVL criada na alínea anterior, sendo esses os seguintes: getMaxDist (), findDistW (),e outros dois métodos para calcular a distância ao root e entre dois elementos distRoot (), distBet2 ()

Complexidade-> O(h) onde h é a altura da árvore binária de pesquisa.__

d) Criamos métodos que transformam a árvore obtida alínea anterior numa árvore binária completa, inserindo nestas possíveis configurações eletrónicas únicas recorrendo aos métodos:

Complexidade->

Uma solução eficiente pode construir BST balanceado em tempo O (n) com a altura mínima possível. Abaixo estão as etapas. Percorra o BST fornecido na ordem e armazene o resultado em uma matriz. Esta etapa leva tempo O (n). Observe que essa matriz seria classificada como a travessia em ordem do BST sempre produz uma sequência classificada. Construa um BST balanceado a partir da matriz classificada criada acima usando a abordagem recursiva discutida aqui. Esta etapa também leva tempo O (n), pois atravessamos cada elemento exatamente uma vez e o processamento de um elemento leva tempo O (1).

—

## Melhoramentos Possíveis

Alguns melhoramentos que seriam talvez possíveis seriam melhorar a complexidade dos algoritmos que produzimos.