**Documentação de Sistema**

Trabalho Prático Final

Autores:

Chrystian Henrique Fonseca de Souza

Samara Hellen Ferreira Chaves

Belo Horizonte

Junho 2023

# Documentação do Sistema

[Documentação do Sistema 2](#_Toc121578588)

[Introdução 3](#_Toc121578589)

[Escopo do Sistema 3](#_Toc121578590)

[Nome do sistema e de seus componentes principais 3](#_Toc121578591)

[Missão ou objetivo do sistema 3](#_Toc121578592)

[Arquitetura do Sistema 3](#_Toc121578593)

[Funcionalidades do produto 4](#_Toc121578594)

[Usuários e sistemas externos 4](#_Toc121578595)

[Descrição 4](#_Toc121578596)

[Documentação do código 4](#_Toc121578597)

[Documentação da Estrutura de dados geral do software 4](#_Toc121578598)

[Procedimento <mostra\_tipeeq> 5](#_Toc121578599)

[Procedimento <setCodTE> 5](#_Toc121578600)

[Função <getCodTE> 5](#_Toc121578601)

[Procedimento <setDescricaoTE> 5](#_Toc121578602)

[Função <getDescricaoTE> 5](#_Toc121578603)

[Procedimento <mostra\_departamento> 5](#_Toc121578604)

[Procedimento <setCodDep> 5](#_Toc121578605)

[Função <getCodDep> 5](#_Toc121578606)

[Procedimento <setNomeDep> 6](#_Toc121578607)

[Função <getNomeDep> 6](#_Toc121578608)

[Procedimento <setDescricaoDep> 6](#_Toc121578609)

[Função <getDescricaoDep> 6](#_Toc121578610)

[Procedimento <setLocalDep> 6](#_Toc121578611)

[Função <getLocalDep> 6](#_Toc121578612)

[Procedimento <mostra\_EQ> 6](#_Toc121578613)

[Procedimento <setCodEQ> 6](#_Toc121578614)

[Função <getCodEQ> 7](#_Toc121578615)

[Procedimento <setNomeEQ> 7](#_Toc121578616)

[Função <getNomeEQ> 7](#_Toc121578617)

[Procedimento <setPrecoEQ> 7](#_Toc121578618)

[Função <getPrecoEQ> 7](#_Toc121578619)

[Função <Verifica\_Strings> 7](#_Toc121578620)

[Procedimento <alterar\_string> 7](#_Toc121578621)

[Função <tem\_cadastro\_EQ> 8](#_Toc121578622)

[Função <tem\_cadastro\_DEP> 8](#_Toc121578623)

[Função <tem\_cadastro\_TE> 8](#_Toc121578624)

[Procedimento <menuEQ> 8](#_Toc121578625)

[Procedimento <menuTE> 8](#_Toc121578626)

[Procedimento <menuDEP> 8](#_Toc121578627)

[Função <\*Busca\_TE > 8](#_Toc121578628)

[Função <\*Busca\_TE > 9](#_Toc121578629)

[Testes do sistema 9](#_Toc121578630)

[Casos de testes do sistema: Procedimento < Verifica\_Strings > 9](#_Toc121578631)

[Casos de testes do sistema: Procedimento < alterar\_string > 9](#_Toc121578632)

[Casos de testes do sistema: função < tem\_cadastro\_EQ > 9](#_Toc121578633)

[Casos de testes do sistema: função < tem\_cadastro\_TE > 10](#_Toc121578634)

[Casos de testes do sistema: função <tem\_cadastro\_DEP> 10](#_Toc121578635)

[Casos de testes do sistema: Procedimento < menuEQ> 10](#_Toc121578636)

[Casos de testes do sistema: Procedimento <menuDep> 11](#_Toc121578637)

[Casos de testes do sistema: função <menuTE> 12](#_Toc121578638)

## Introdução

### Escopo do Sistema

#### Nome do sistema e de seus componentes principais

|  |
| --- |
| Trabalho Prático Final  Classes: ArvoreAVL, ListaDupla, No, Palavra, Classe Principal |

#### Missão ou objetivo do sistema

|  |
| --- |
| O objetivo do sistema é ler um arquivo e armazenar as palavras do arquivo por ordem lexicográfica em uma arvore, depois que todos as palavras são adicionadas na arvore, percorremos a arvore com caminhamento central e adicionamos as palavras em uma lista duplamente encadeada, sendo que serão exibidas em forma decrescente, da palavra que mais apareceu para a que menos apareceu. |

**Descrição:**

De forma resumida, o algoritmo se inicia com a leitura de um arquivo txt. Após isso, é criada uma árvore AVL que irá armazenar as palavras contidas no arquivo, dada as condições apresentadas no trabalho. Teremos então o bloco try, que será utilizado para ler de forma geral, e os blocos de catch e finally para tratar as exceções encontradas.

Dentro do bloco try, então, teremos a leitura do arquivo com a função BufferReader e a FilerReader que são da biblioteca “java.io.BufferedReader”. Foi colocado um while com o objetivo de que fosse lido todo o arquivo, e enquanto ele lê, ele irá desprezar os sinais de pontuação e espaços, e também converte letras maiúsculas em minúsculas, essas funcionalidades foram feitas com as funções replaceAll e toLowerCase, que são da biblioteca citada acima. Então é chamada a classe palavra, que recebe cada palavra do arquivo, e armazena na árvore. Depois disso, a árvore é imprimida na tela e após, é criada uma lista dupla para armazenar todas as palavras contidas na árvore e exibir em ordem decrescente.

Nos blocos catch, é tratada exceções como se o arquivo não for encontrado no sistema, e também para caso tenha algum erro durante a leitura do arquivo. Tudo isso será exibido para o usuário caso ocorra.

O bloco finally será executado independente das exceções ou não. Ele é utilizado para limpeza do sistema, ou seja, garantir que o buffer reader seja fechado, caso ocorra alguma exceção ou não.

### Documentação do código

#### Documentação da Estrutura de dados geral do software

**Classes:**

Departamento: Classe que possui os atributos (todos privados) de int código, char nome do departamento, char descrição do departamento e int localização do departamento (prédio onde estará), possui também um método construtor onde os valores são iniciados, os métodos get/set para utilização, sem que acesse os atributos privados diretamente.

Tipo de Equipamento: Classe que possui os atributos (todos privados) de int código do tipo de equipamento, char descrição do tipo de equipamento. Possui também um método construtor para inicialização dos valores, e os métodos get/set para utilização, sem que acesse os atributos privados diretamente.

Equipamento: Classe que possui os atributos (todos privados) de int código do equipamento, char nome do equipamento e o float preço. Através do relacionamento de classes também foi atribuído um char nome do departamento onde o equipamento está localizado e também, o char descrição do tipo de equipamento. Também foi feito um método construtor para inicialização dos valores e o get/set para utilização sem que se acesse os atributos privados diretamente.

**Vetores:**

Vetores globais de utilização ao decorrer do código, TE (tipo de equipamento), EQ (equipamento), DP (departamento).

#### Procedimento <mostra\_tipeeq>

Descrição: Procedimento de assinatura mostra\_tipeeq utilizada para mostrar os tipos de equipamentos cadastrados.

Parâmetros: Não possui.

Retorno: N/A

#### Procedimento <setCodTE>

Descrição: Procedimento de assinatura setcodTE utilizado para fazer o set do código do tipo de equipamento.

Parâmetro: int idTE.

Retorno: N/A

#### Função <getCodTE>

Descrição: Função do tipo int de assinatura getcodTE utilizado para fazer o get do código do tipo de equipamento.

Parâmetros: Não possui.

Retorno: int codTE.

#### Procedimento <setDescricaoTE>

Descrição: Procedimento de assinatura setdescricaoTE utilizado para fazer o set da descrição do tipo de equipamento.

Parâmetro: char descTE

Retorno: N/A

#### Função <getDescricaoTE>

Descrição: Função do tipo char de assinatura getdescricaoTE utilizado para fazer o get da descrição do tipo de equipamento.

Parâmetros: Não possui.

Retorno: descricaoTE.

#### Procedimento <mostra\_departamento>

Descrição: Procedimento de assinatura mostra\_departamento utilizada para mostrar os tipos de departamentos cadastrados.

Parâmetros: Não possui.

Retorno: N/A

#### Procedimento <setCodDep>

Descrição: Procedimento de assinatura setCodDep utilizado para fazer o set do código do departamento.

Parâmetro: int IdDep.

Retorno: N/A

#### Função <getCodDep>

Descrição: Função do tipo int de assinatura getCodDep utilizado para fazer o get do código do departamento.

Parâmetros: Não possui.

Retorno: int codDep.

#### Procedimento <setNomeDep>

Descrição: Procedimento de assinatura setNomeDep utilizado para fazer o set do nome do departamento.

Parâmetro: char NDep.

Retorno: N/A

#### Função <getNomeDep>

Descrição: Função do tipo char de assinatura getNomeDep utilizado para fazer o get do nome do departamento.

Parâmetros: Não possui.

Retorno: char NomeDep.

#### Procedimento <setDescricaoDep>

Descrição: Procedimento de assinatura setDescricaoDep utilizado para fazer o set da descrição do departamento.

Parâmetro: char DescDep.

Retorno: N/A

#### Função <getDescricaoDep>

Descrição: Função do tipo char de assinatura getDescricaoDep utilizado para fazer o get da descrição do departamento.

Parâmetros: Não possui.

Retorno: char DescricaoDep

#### Procedimento <setLocalDep>

Descrição: Procedimento de assinatura setLocalDep utilizado para fazer o set do número do prédio do departamento.

Parâmetro: int LocDep.

Retorno: N/A

#### Função <getLocalDep>

Descrição: Função do tipo int de assinatura getLocalDep utilizado para fazer o get do do número do prédio do departamento.

Parâmetros: Não possui.

Retorno: int LocalizacaoDep.

#### Procedimento <mostra\_EQ>

Descrição: Procedimento de assinatura mostra\_EQ utilizada para mostrar os equipamentos cadastrados.

Parâmetros: Não possui.

Retorno: N/A

#### Procedimento <setCodEQ>

Descrição: Procedimento de assinatura setcodEQ utilizado para fazer o set do código do equipamento.

Parâmetro: int idEQ

Retorno: N/A

#### Função <getCodEQ>

Descrição: Função do tipo int de assinatura getcodEQ utilizado para fazer o get do código do equipamento.

Parâmetros: Não possui.

Retorno: int codEquip.

#### Procedimento <setNomeEQ>

Descrição: Procedimento de assinatura setNomeEQ utilizado para fazer o set do nome do equipamento.

Parâmetro: char NomeEQ

Retorno: N/A

#### Função <getNomeEQ>

Descrição: Função do tipo char de assinatura getNomeEQ utilizado para fazer o get do nome do equipamento.

Parâmetros: Não possui

Retorno: char NomeEquip.

#### Procedimento <setPrecoEQ>

Descrição: Procedimento de assinatura setPrecoEQ utilizado para fazer o set do preço do equipamento.

Parâmetro: float PrecoEQ,

Retorno: N/A

#### Função <getPrecoEQ>

Descrição: Função do tipo int de assinatura getPrecoEQ utilizado para fazer o get do preço do equipamento.

Parâmetros: Não possui.

Retorno: float PrecoEquip

#### Função <Verifica\_Strings>

Descrição: Função de tipo inteiro e assinatura Verifica\_Strings que recebe duas strings por parâmetro e compara as duas e retorna 0 se elas forem iguais ou 1 se elas forem diferentes.

Parâmetros: char StrA e char StrB

Retorno: int Result;

#### Procedimento <alterar\_string>

Descrição: Procedimento de assinatura alterar\_string que compara o tamanho de duas strings e altera o valor conforme necessário.

Parâmetros: char nome1 e char nome2

Retorno: N/A.

#### Função <tem\_cadastro\_EQ>

Descrição: Função do tipo bool de assinatura tem\_cadastro\_EQ que verifica se possui cadastro de equipamento.

Parâmetros: int ContEQ

Retorno: um valor booleano 0 ou 1

#### Função <tem\_cadastro\_DEP>

Descrição: Função do tipo bool de assinatura tem\_cadastro\_DEP que verifica se possui cadastro de departamento.

Parâmetros: int ContDEP

Retorno: um valor booleano 0 ou 1

#### Função <tem\_cadastro\_TE>

Descrição: Função do tipo bool de assinatura tem\_cadastro\_TE que verifica se possui cadastro de tipo de equipamento.

Parâmetros: int ConTE

Retorno: um valor booleano 0 ou 1

#### Procedimento <menuEQ>

Descrição: Procedimento para implementação do menu de equipamentos junto com todas as funcionalidades

Parâmetros: Não possui.

Retorno: N/A

#### Procedimento <menuTE>

Descrição: Procedimento para implementação do menu do tipo de equipamentos junto com todas as funcionalidades

Parâmetros: Não possui.

Retorno: N/A

#### Procedimento <menuDEP>

Descrição: Procedimento para implementação do menu dos departamentos junto com todas as funcionalidades

Parâmetros: Não possui.

Retorno: N/A

#### Função <\*Busca\_TE >

Descrição: Função para buscar o código de um tipo de equipamento previamente cadastrado.

Parâmetros: Tipo\_Equipamento TE[Max], int num

Retorno: &TE[i]

#### Função <\*Busca\_TE >

Descrição: Função para buscar o código de um departamento previamente cadastrado.

Parâmetros: Departamento DP[Max], int num2

Retorno: &DP[i]