Sistema de folha de pagamento

AUTORES

AUGUSTO VINÍCIUS FERREIRA DE SALES Matrícula; *20200001829*
LEANDRO LUCAS DE OLIVEIRA BANDEIRA Matrícula : *20210025029*
RENATA AVELINO DE ANDRADE Matrícula: *20210025421*

Versão
Quarta, 22 de Junho de 2022

Sumário

Tabela de conteúdo

Índice Hierárquico

Hierarquia de Classes

Esta lista de hierarquias está parcialmente ordenada (ordem alfabética):

Índice dos Componentes

Lista de Classes

Aqui estão as classes, estruturas, uniões e interfaces e suas respectivas descrições:

Índice dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos e suas respectivas descrições:

TentativaAbrirArquivo.h

Classes

Referência da Classe Arquivo

`#include <Arquivo.h>`

Membros Públicos

Arquivo ()
Construtor()

void **atualizaArquivoFolha** (std::vector< **Funcionario** * > &funcionariosVec)
atualizaArquivoFolha

void **atualizaArquivoFolha** (**Funcionario** *)
atualizaArquivoFolha

void **AtualizaBaseDadosCsv** (const std::vector< **Funcionario** * > &funcionariosVec)
atualizaBaseDadosCsv

void **addPresidenteBaseDadosCsv** (**Funcionario** *)
addPresidenteBaseDadosCsv

void **criaArquivoBaseDadosZerado** ()
criaArquivoBaseDadosZerado

void **carregaDadosCsv** (std::vector< **Funcionario** * > &operadores, std::vector< **Funcionario** * > &gerentes, std::vector< **Funcionario** * > &diretores, **Funcionario** **presidente)
CarregaDadosCsv.

HistoricoArquivo * **getHistoricoArquivo** ()
getHistoricoArquivo

Construtores e Destrutores

Arquivo::Arquivo ()

Construtor()
Responsável por carregar o arquivo de folha.csv

Funções membros

void Arquivo::addPresidenteBaseDadosCsv (Funcionario * *presidente*)

addPresidenteBaseDadosCsv

Parâmetros

ponteiro	para um tipo de funcionário

Responsável por atualizar o arquivo de base de dados com apenas um funcionario

Retorna

Nada

void Arquivo::atualizaArquivoFolha (Funcionario * *presidente*)

atualizaArquivoFolha

Parâmetros

ponteiro	para um tipo de funcionário

Responsável por atualizar o arquivo de folha.csv com apenas um funcionário

Retorna

Nada

void Arquivo::atualizaArquivoFolha (std::vector< Funcionario * > & *funcionariosVec*)

atualizaArquivoFolha

Parâmetros

lista	lista de funcionários

Responsável por atualizar o arquivo de folha.csv

Retorna

Nada

void Arquivo::AtualizaBaseDadosCsv (const std::vector< Funcionario * > & *funcionariosVec*)

atualizaBaseDadosCsv

Parâmetros

lista	lista de funcionários para salvar na base de dados

Responsável por atualizar a base de dados

Retorna

Nada

void Arquivo::carregaDadosCsv (std::vector< Funcionario * > & *operadores*, std::vector< Funcionario * > & *gerentes*, std::vector< Funcionario * > & *diretores*, Funcionario ** *presidente*)

CarregaDadosCsv.

Parâmetros

lista	de operadores
lista	de gerentes
lista	de diretores
ponteiro	para um tipo de funcionário

Responsável por carregar os dados e inicializar nas listas

Retorna

Nada

void Arquivo::criaArquivoBaseDadosZerado ()

criaArquivoBaseDadosZerado

Responsável por criar o arquivo zerado

Retorna

Nada

HistoricoArquivo * Arquivo::getHistoricoArquivo ()

getHistoricoArquivo

Responsável por retornar o endereço de um objeto do tipo Historico

Retorna

endereço de um objeto historicoArquivo

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Arquivo.h
Arquivo.cpp

Referência da Classe CadastrarFuncionarioException

Exception personalidado usado para detectar e informar os erros que podem ocorrer durante o cadastramento de um funcionário.

`#include <CadastrarFuncionarioException.h>`

Membros Públicos

CadastrarFuncionarioException (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Exception personalidado usado para detectar e informar os erros que podem ocorrer durante o cadastramento de um funcionário.

Construtores e Destrutores

CadastrarFuncionarioException::CadastrarFuncionarioException (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * CadastrarFuncionarioException::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

Atributos

char CadastrarFuncionarioException::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

- **CadastrarFuncionarioException.h**
- **CadastrarFuncionarioException.cpp**

Referência da Classe CEPException

Exception personalizado criado para reconhecer e informar ao usuário possíveis erros durante o cadastro do CEP do funcionário.

`#include <CEPException.h>`

Membros Públicos

CEPException (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Exception personalizado criado para reconhecer e informar ao usuário possíveis erros durante o cadastro do CEP do funcionário.

Construtores e Destrutores

CEPException::CEPException (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * CEPException::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

Atributos

char CEPException::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

CEPException.h
CEPException.cpp

Referência da Classe CPFException

Exception personalizado criado para reconhecer e informar ao usuário possíveis erros durante o cadastro do CPF do funcionário.

`#include <CPFException.h>`

Membros Públicos

CPFException (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Exception personalizado criado para reconhecer e informar ao usuário possíveis erros durante o cadastro do CPF do funcionário.

Construtores e Destrutores

CPFException::CPFException (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * CPFException::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

Atributos

char CPFException::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

CPFException.h
CPFException.cpp

Referência da Classe Data

`#include <Data.h>`

Membros Públicos

Data ()
Data (int dia, int mes, int ano)
Construtor.

int **getDia** ()
Métodos Get.

int **getMes** ()
int **getAno** ()
void **setMes** (int mes)
Métodos Set.

void **setDia** (int dia)
void **setAno** (int ano)
int **getQuantidadeDiaMes** (int mes)
getQuantidadeDiaMes()

void **exibeData** ()
Exibe a data armazenada.

std::string **retornaStringData** ()
retornaStringData()

std::string **retornaDataComputador** ()
retornaDataComputador()

bool **comparaDatas** (int *data)
comparaDatas()

Construtores e Destrutores

Data::Data ()

Data::Data (int *dia*, int *mes*, int *ano*)

Construtor.
Inicializa o dia, mês e ano de uma data.

Parâmetros

int	dia do ano
int	mes do ano
int	refere-se ao ano da data

Funções membros

bool Data::comparaDatas (int * *data*)

comparaDatas()

Parâmetros

data	lista de uma data, contendo dia, mes e ano. Compara a data armazenada com aquela que recebe como parâmetro.

Retorna

valor bool que indica se são iguais ou não.

void Data::exibeData ()

Exibe a data armazenada.

int Data::getAno ()

int Data::getDia ()

Métodos Get.

Parâmetros

getDia	retorna o dia
getMes	retorna o mes
getAno	retorna o ano

Retorna

um inteiro

int Data::getMes ()

int Data::getQuantidadeDiaMes (int *mes*)

getQuantidadeDiaMes()

Retorna a quantidade de dia do mes

Parâmetros

mes	recebe mes do ano

Retorna

inteiro

std::string Data::retornaDataComputador ()

retornaDataComputador()

Retorna

Retorna o formato string da data atual do computador

std::string Data::retornaStringData ()

retornaStringData()

Retorna

retorna o formato string da data armazenada, na forma xx/xx/xxxx

void Data::setAno (int *ano*)

void Data::setDia (int *dia*)

void Data::setMes (int *mes*)

Métodos Set.

Parâmetros

setDia	armazena o dia
setMes	armazena o mes
setAno	armazena o ano

Retorna

void

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Data.h
Data.cpp

Referência da Classe Diretor

Diretor é uma classe filha de Funcionário.
`#include <Diretor.h>`

Membros Públicos

Diretor ()
Construtor.

Diretor (int codigo, std::string nome, std::string CPF, int idade, std::string endereco, int numeroResidencia, std::string telefone, int *data, int designacao, std::string areaSupervisao, std::string areaFormacao)
Construtor.

std::string **getAreaSupervisao** ()
Métodos Get.

std::string **getAreaFormacao** ()
void **setAreaSupervisao** (std::string areaSupervisao)
Métodos Set.

void **setAreaFormacao** (std::string areaFormacao)
void **calcularSalarioMensal** (int mes)
calcularSalarioMensal(int)

Outros membros herdados

Descrição detalhada

Diretor é uma classe filha de Funcionário.

Construtores e Destrutores

Diretor::Diretor ()

Construtor.

Construtor padrão sem argumentos

Diretor::Diretor (int *codigo*, std::string *nome*, std::string *CPF*, int *idade*, std::string *endereco*, int *numeroResidencia*, std::string *telefone*, int * *data*, int *designacao*, std::string *areaSupervisao*, std::string *areaFormacao*)

Construtor.

Inicializa os atributos do diretor

Parâmetros

int-codigo	código do diretor
string-nome	nome completo do diretor
string-CPF	CPF do diretor
int-idade	idade do diretor
string-endereco	CEP da residência do diretor
int-numeroResidencia	número da residência do diretor
string-telefone	telefone do diretor
int-data*	array de inteiros referentes a dia/mes/ano do ingresso do diretor
int-designacao	designacao do diretor (2)
string-areaSupervisao	área de supervisão do diretor
string-areaFormacao	área de formação do diretor

Funções membros

void Diretor::calcularSalarioMensal (int *mes*) `[virtual]`

calcularSalarioMensal(int)

Calcula o salário mensal do diretor de acordo com o mês que recebe como parâmetro

Parâmetros

int-mes	mês que se deseja calcular o salário

Implementa **Funcionario** (*p.*).

std::string Diretor::getAreaFormacao ()

std::string Diretor::getAreaSupervisao ()

Métodos Get.

Parâmetros

getAreaSupervisao	retorna a área de supervisão como uma string
getAreaFormacao	retorna a área de formação como uma string

void Diretor::setAreaFormacao (std::string *areaFormacao*)

void Diretor::setAreaSupervisao (std::string *areaSupervisao*)

Métodos Set.

Parâmetros

setAreaSupervisao	instancia a área de supervisão

setAreaFormacao	instancia a área de formação

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Diretor.h
Diretor.cpp

Referência da Classe Empresa

`#include <Empresa.h>`

Membros Públicos

Empresa ()
virtual **~Empresa** ()
int **getQtdFuncionarios** (int tipoFuncionario)
Métodos Get.

void **addFuncionario** (**Funcionario** *funcionario, int tipoFuncionario)
addFuncionario()

void **modificarFuncionario** (int codigo, int opcao, std::string valor)
modificarFuncionario()

void **modificarFuncionario** (int codigo, int opcao, int valor)
modificarFuncionario()

void **modificarFuncionario** (int codigo, int *valor)
modificarFuncionario()

void **excluirFuncionario** (int codigo)
excluirFuncionario()

void **exibirFuncionario** (int codigo)
exibirFuncionario()

void **exibirTodosFuncionarios** ()
exibirTodosFuncionarios()

void **exibirFuncionariosPorTipo** (int tipoFuncionario)
exibirFuncionariosPorTipo()

void **concederAumentoSalarial** ()
concederAumentoSalarial()

void **calcularFolhaSalarial** (int mes)
calcularFolhaSalarial()

void **imprimirFolhaSalarialFuncionario** (int codigo)
imprimirFolhaSalarialFuncionario()

void **imprimirFolhaSalarialFuncionario** (std::string nome)

imprimirFolhaSalarialFuncionario()

void **imprimirFolhaSalarialEmpresa** (int opcao)
imprimirFolhaSalarialEmpresa()

Funcionario * buscarFuncionario (int codigo)
buscarFuncionario()

Funcionario * buscarFuncionario (int codigo, int *indice, int *designacao)
buscarFuncionario()

void **buscarFuncionariosIntervaloTempo** (int *dataInicial, int *dataFinal)
buscarFuncionariosIntervaloTempo()

void **buscarFuncionariosParcial** (std::string informacao, int opcao)
buscarFuncionariosParcial()

Construtores e Destrutores

Empresa::Empresa ()

Empresa::~Empresa () `[virtual]`

Funções membros

void Empresa::addFuncionario (Funcionario * *funcionario*, int *tipoFuncionario*)

addFuncionario()

Adiciona um ponteiro de funcionário ao vector de sua designação, ou no caso do presidente faz ele apontar para o funcionário, e as quantidades dos funcionários de cada tipo é atualizada.

Parâmetros

ponteiro	Para um tipo de funcionário.

Retorna

Void.

Funcionario * Empresa::buscarFuncionario (int *codigo*)

buscarFuncionario()

Busca um funcionário utilizando o código solicitado.

Parâmetros

int	Código do funcionário a ser buscado.

Retorna

Retorna um ponteiro para o funcionário encontrado.

Funcionario * Empresa::buscarFuncionario (int *codigo*, int * *indice*, int * *designacao*)

buscarFuncionario()

Busca um funcionário utilizando o código solicitado.

Parâmetros

int	Código do funcionário a ser buscado.
ponteiro	Ponteiro para apontar para o indice do funcionário encontrado no vector.
ponteiro	Ponteiro para apontar para a designação do funcionário encontrado

Retorna

Retorna um ponteiro para o funcionário encontrado.

void Empresa::buscarFuncionariosIntervaloTempo (int * *dataInicial*, int * *dataFinal*)

buscarFuncionariosIntervaloTempo()

Busca e exibe funcionários que possuem a data de ingresso no intervalo de tempo solicitado.

Parâmetros

ponteiro	Aponta para o array que armazena a data inicial do intervalo a se buscado.
ponteiro	Aponta para o array que armazena a data final do intervalo a se buscado.

Retorna

Void.

void Empresa::buscarFuncionariosParcial (std::string *informacao*, int *opcao*)

buscarFuncionariosParcial()

Busca e exibe funcionários que possuem o nome ou o endereço correspondentes a informação solicitada para buscar.

Parâmetros

string	Informação usada para buscar funcionários.
int	A opção se a busca parcial vai ser pelo nome ou endereço.

Retorna

Void.

void Empresa::calcularFolhaSalarial (int *mes*)

calcularFolhaSalarial()

Calcula a folha salarial do mês solicitado para os funcionários já cadastrados, mas só se não tiver sido calculada anteriormente.

Parâmetros

int	Mês a se calculado a folha salarial.

Retorna

Void.

void Empresa::concederAumentoSalarial ()

concederAumentoSalarial()

Concede aumento a todos os funcionários já cadastrados, as taxas de aumento são fixas para cada designação.

Retorna

Void.

void Empresa::excluirFuncionario (int *codigo*)

excluirFuncionario()

Exclui o funcionário do codigo solicitado do seu vector, e depois libera a memória alocada para ele.

Parâmetros

int	Código do funcionário a ser excluido.

Retorna

Void.

void Empresa::exibirFuncionario (int *codigo*)

exibirFuncionario()

Exibe os atributos do funcionário correspondente ao código solicitado.

Parâmetros

int	Código do funcionário a ser exibido.

Retorna

Void.

void Empresa::exibirFuncionariosPorTipo (int *tipoFuncionario*)

exibirFuncionariosPorTipo()

Exibe todos os funcionários do tipo de designação solicitado que já foram cadastrados.

Parâmetros

int	Tipo de designação.

Retorna

Void.

void Empresa::exibirTodosFuncionarios ()

exibirTodosFuncionarios()

Exibe todos os funcionários que já foram cadastrados

Retorna

Void.

int Empresa::getQtdFuncionarios (int *tipoFuncionario*)

Métodos Get.

Parâmetros

getQtdFuncionarios	Retorna a quantidade de funcionários dependendo da designação.

Retorna

Retorna um inteiro.

void Empresa::imprimirFolhaSalarialEmpresa (int *opcao*)

imprimirFolhaSalarialEmpresa()

Dependendo da opção imprimi a folha salarial da empresa para o ano ou para um mês especifico.

Parâmetros

int	Opção de impressão da folha salarial da empresa.

Retorna

Void.

void Empresa::imprimirFolhaSalarialFuncionario (int *codigo*)

imprimirFolhaSalarialFuncionario()

Imprimi a folha salarial dos meses já calculados do funcionário correspondente ao código solicitado.

Parâmetros

int	Código do funcionário a ser impresso a folha salarial.

Retorna

Void.

void Empresa::imprimirFolhaSalarialFuncionario (std::string *nome*)

imprimirFolhaSalarialFuncionario()

Imprimi a folha salarial dos meses já calculados do funcionário correspondente ao nome solicitado.

Parâmetros

string	Nome do funcionário a ser impresso a folha salarial.

Retorna

Void.

void Empresa::modificarFuncionario (int *codigo*, int * *valor*)

modificarFuncionario()

Modifica a data de ingresso do funcionário solicitado através do código.

Parâmetros

int	Código do funcionário a ser modificado.
ponteiro	Aponta para um array que está armazenando a nova data.

Retorna

Void.

void Empresa::modificarFuncionario (int *codigo*, int *opcao*, int *valor*)

modificarFuncionario()

Modifica um atributo int do funcionário solicitado através do código.

Parâmetros

int	Código do funcionário a ser modificado.
int	Opção de qual atributo vai ser modificado.
int	O novo valor para o atributo int.

Retorna

Void.

void Empresa::modificarFuncionario (int *codigo*, int *opcao*, std::string *valor*)

modificarFuncionario()

Modifica um atributo string do funcionário solicitado através do código

Parâmetros

int	Código do funcionário a ser modificado.
int	Opção de qual atributo vai ser modificado.
string	O novo valor para o atributo string.

Retorna

Void.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Empresa.h
Empresa.cpp

Referência da Classe Endereco

Classe criada para facilitar o amarzenamento de dados do endereço.
`#include <Endereco.h>`

Membros Públicos

Endereco ()
Construtor.

void **setNumero** (int numero)
Métodos Set.

void **setEndereco** (std::string CEP)
std::string **getRua** ()
Métodos Get.

std::string **getBairro** ()
std::string **getCidade** ()
std::string **getEstado** ()
std::string **getCEP** ()
int **getNumero** ()
void **validaCEP** (std::string CEP)
validaCEP(string)

std::string **getInformacao** ()
getInformacao()

bool **comparaEndereco** (std::string endereco)
comparaEndereco(string)

int **validaNumero** (std::string numeroStr)
validaNumero(string)

Descrição detalhada

Classe criada para facilitar o amarzenamento de dados do endereço.

Construtores e Destrutores

Endereco::Endereco ()

Construtor.

Construtor padrão sem argumentos

Funções membros

bool Endereco::comparaEndereco (std::string *endereco*)

comparaEndereco(string)

Método que verifica se o endereço do argumento é igual ao endereço do funcionário

Parâmetros

string-endereco	CEP que deseja ser comparado digitado pelo usuário

std::string Endereco::getBairro ()

std::string Endereco::getCEP ()

std::string Endereco::getCidade ()

std::string Endereco::getEstado ()

std::string Endereco::getInformacao ()

getInformacao()

Método que retorna o endereço completo em forma de string

int Endereco::getNumero ()

std::string Endereco::getRua ()

Métodos Get.

Parâmetros

getRua	retorna a rua do endereço em forma de string
getBairro	retorna o bairro do endereço em forma de string
getCidade	retorna a cidade do endereço em forma de string
getEstado	retorna o estado do endereço em forma de string
getCEP	retorna o CEP do endereço em forma de string
getNumero	retorna o número da residência em forma de inteiro

void Endereco::setEndereco (std::string *CEP*)

void Endereco::setNumero (int *numero*)

Métodos Set.

Parâmetros

setNumero	instancia o número da residência

setEndereco	através do CEP, instancia os dados referentes ao endereço (rua, bairro, ...)

void Endereco::validaCEP (std::string *CEP*)

validaCEP(string)

Método que recebe um CEP e verifica se ele é válido e já salva

Parâmetros

string-CEP	CEP digitado pelo usuário

int Endereco::validaNumero (std::string *numeroStr*)

validaNumero(string)

Método que valida o número da residência do funcionário

Parâmetros

string-numeroStr	número digitado pelo usuário

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Endereco.h
Endereco.cpp

Referência da Classe FolhaSalarial

`#include <FolhaSalarial.h>`

Membros Públicos

FolhaSalarial ()
FolhaSalarial (int designacao)
virtual ~**FolhaSalarial** ()
void **setSalarioLiquido** (double salario)
Métodos Set publics.

double **getSalarioBase** ()
Métodos Get publics.

double **getSalarioLiquido** ()
double **getDescontoImpostoRenda** ()
double **getDescontoPrevidencia** ()
void **aumentarSalarioBase** ()
aumentarSalarioBase()

Construtores e Destrutores

FolhaSalarial::FolhaSalarial ()

FolhaSalarial::FolhaSalarial (int *designacao*)

FolhaSalarial::~FolhaSalarial () `[virtual]`

Funções membros

void FolhaSalarial::aumentarSalarioBase ()

aumentarSalarioBase()

Aumenta o salario base de acordo com a taxa de aumento.

Retorna

void.

double FolhaSalarial::getDescontoImpostoRenda ()

double FolhaSalarial::getDescontoPrevidencia ()

double FolhaSalarial::getSalarioBase ()

Métodos Get publics.

Parâmetros

getSalarioBase	retorna o valor do salario base.
getSalarioLiquido	retorna o valor do salario liquido.
getDescontoImpostoRenda	retorna o valor do desconto do imposto de renda no salario liquido.
getDescontoPrevidencia	retorna o valor do desconto da previdencia social no salario liquido.

Retorna

Retorna um double.

double FolhaSalarial::getSalarioLiquido ()

void FolhaSalarial::setSalarioLiquido (double *salario*)

Métodos Set publics.

Parâmetros

setSalarioLiquido	seta o salario liquido.

Retorna

Void.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

FolhaSalarial.h
FolhaSalarial.cpp

Referência da Classe Funcionario

Funcionario é uma classe filha de pessoa.
`#include <Funcionario.h>`

Membros Públicos

Funcionario ()
Construtor.

Funcionario (int codigo, std::string nome, std::string CPF, int idade, std::string CEP, int numeroResidencia, std::string telefone, int *data, int designacao)
Construtor.

virtual **~Funcionario** ()
Destrutor.

virtual void **calcularSalarioMensal** (int mes)=0
*virtual **calcularSalarioMensal(int)** = 0*

int **validaDesignacao** (std::string designacaoStr)
validaDesignacao(string)

void **validaDataIngresso** (std::string dataStr, int *dataInt)
validaDataIngresso(string, int)*

int **validaCodigoFuncionario** (std::string codigoStr)
validaCodigoFuncionario(string)

int **getHorasTrabalhadas** (int mes)
Métodos Get.

int **getCodigoFuncionario** ()
std::string **getDesignacaoStr** ()
int **getDesignacaoInt** ()
FolhaSalarial * getFolhaSalarial (int mes)
Data getDataIngresso ()
void **setHorasTrabalhadas** (int mes, int valor)
Métodos Set.

void **setDesignacao** (int designacao)
void **setDataIngresso** (int *data)
void **setCodigoFuncionario** (int codigo)

Membros Protegidos

int **gerarAleatorio** (int intervaloMax)

gerarAleatorio(int)

Descrição detalhada

Funcionario é uma classe filha de pessoa.

Construtores e Destrutores

Funcionario::Funcionario ()

Construtor.

Construtor padrão sem argumentos

Funcionario::Funcionario (int *codigo*, std::string *nome*, std::string *CPF*, int *idade*, std::string *CEP*, int *numeroResidencia*, std::string *telefone*, int * *data*, int *designacao*)

Construtor.

Inicializa os atributos do funcionário

Parâmetros

int-codigo	código do funcionário
string-nome	nome completo do funcionário
string-CPF	CPF do funcionário
int-idade	idade do funcionário
string-endereco	CEP da residência do funcionário
int-numeroResidencia	número da residência do funcionário
string-telefone	telefone do funcionário
int-data*	array de inteiros referentes a dia/mes/ano do ingresso do funcionário
int-designacao	designacao do funcionário (de 0 a 3)

Funcionario::~Funcionario () `[virtual]`

Destrutor.

Destrutor padrão

Funções membros

virtual void Funcionario::calcularSalarioMensal (int *mes*) `[pure virtual]`

virtual **calcularSalarioMensal(int)** = 0

Método virtual puro para calcular o salário mensal do funcionário de acordo com sua designação

Parâmetros

int-mes	mês que seja calcular o salário

Implementado por **Diretor** (*p.*), **Presidente** (*p.*), **Gerente** (*p.*) e **Operador** (*p.*).

int Funcionario::gerarAleatorio (int *intervaloMax*) `[protected]`

gerarAleatorio(int)

Método para gerar um número aleatório dentro do intervalo digitado

Parâmetros

int-intervaloMax	intervalo máximo que deseja se gerar o número aleatório

int Funcionario::getCodigoFuncionario ()

Data Funcionario::getDataIngresso ()

int Funcionario::getDesignacaoInt ()

std::string Funcionario::getDesignacaoStr ()

FolhaSalarial * Funcionario::getFolhaSalarial (int *mes*)

int Funcionario::getHorasTrabalhadas (int *mes*)

Métodos Get.

Parâmetros

getHorasTrabalhadas	retorna as horas trabalhadas do mês
getCodigoFuncionario	retorna o código do funcionário
getDesignacaoStr	retorna a designação em forma de string
getDesignacaoInt	retorna a designação em forma de int
getFolhaSalarial	retorna um ponteiro para a folha salarial do mês desejado
getDataIngresso	retorna um objeto data com a data de ingresso do funcionário

void Funcionario::setCodigoFuncionario (int *codigo*)

void Funcionario::setDataIngresso (int * *data*)

void Funcionario::setDesignacao (int *designacao*)

void Funcionario::setHorasTrabalhadas (int *mes*, int *valor*)

Métodos Set.

Parâmetros

setHorasTrabalhadas	instancia as horas trabalhadas do mês desejado

setDesignação	instancia a designação do funcionário
setDataIngreso	instancia a data de ingresso do funcionário
setCodigoFuncionario	instancia o código do funcionário

int Funcionario::validaCodigoFuncionario (std::string *codigoStr*)

validaCodigoFuncionario(string)

Método para validar o código digitado pelo usuário

Parâmetros

string-codigoStr	código digitado pelo usuário

void Funcionario::validaDataIngresso (std::string *dataStr*, int * *dataInt*)

validaDataIngresso(string, int*)

Método para validar e instanciar a data digitada pelo usuário

Parâmetros

string-dataStr	data digitada pelo usuário
int-dataInt*	ponteiro para a data que deseja ser instanciada

int Funcionario::validaDesignacao (std::string *designacaoStr*)

validaDesignacao(string)

Método que vê se a designação digitada pelo usuário é válida

Parâmetros

string-DesignacaoStr	designação digitada pelo usuário

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Funcionario.h
Funcionario.cpp

Referência da Classe FuncionarioJaCadastradoExcept

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar um código que já está em uso.

`#include <FuncionarioJaCadastradoExcept.h>`

Membros Públicos

FuncionarioJaCadastradoExcept (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar um código que já está em uso.

Construtores e Destrutores

FuncionarioJaCadastradoExcept::FuncionarioJaCadastradoExcept (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * FuncionarioJaCadastradoExcept::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

Atributos

char FuncionarioJaCadastradoExcept::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

FuncionarioJaCadastradoExcept.h
FuncionarioJaCadastradoExcept.cpp

Referência da Classe FuncionarioNaoEstaCadastradoExcept

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar algum método que exija a existência de um funcionário.

`#include <FuncionarioNaoEstaCadastradoExcept.h>`

Membros Públicos

FuncionarioNaoEstaCadastradoExcept (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar algum método que exija a existência de um funcionário.

Construtores e Destrutores

FuncionarioNaoEstaCadastradoExcept::FuncionarioNaoEstaCadastradoExcept (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * FuncionarioNaoEstaCadastradoExcept::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

Atributos

char FuncionarioNaoEstaCadastradoExcept::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

FuncionarioNaoEstaCadastradoExcept.h
FuncionarioNaoEstaCadastradoExcept.cpp

Referência da Classe Gerente

Gerente é uma classe filha de Funcionário.
`#include <Gerente.h>`

Membros Públicos

Gerente ()
Construtor.

Gerente (int codigo, std::string nome, std::string CPF, int idade, std::string endereco, int numeroResidencia, std::string telefone, int *data, int designacao, std::string areaSupervisao)
Construtor.

std::string **getAreaSupervisao** ()
Métodos Get.

void **setAreaSupervisao** (std::string areaSupervisao)
Métodos Set.

void **calcularSalarioMensal** (int mes)
calcularSalarioMensal(int)

Outros membros herdados

Descrição detalhada

Gerente é uma classe filha de Funcionário.

Construtores e Destrutores

Gerente::Gerente ()

Construtor.

Construtor padrão sem argumentos

Gerente::Gerente (int *codigo*, std::string *nome*, std::string *CPF*, int *idade*, std::string *endereco*, int *numeroResidencia*, std::string *telefone*, int * *data*, int *designacao*, std::string *areaSupervisao*)

Construtor.

Inicializa os atributos do gerente

Parâmetros

int-codigo	código do gerente
string-nome	nome completo do gerente
string-CPF	CPF do gerente
int-idade	idade do gerente
string-endereco	CEP da residência do gerente
int-numeroResidencia	número da residência do gerente
string-telefone	telefone do gerente
int-data*	array de inteiros referentes a dia/mes/ano do ingresso do gerente
int-designacao	designacao do gerente (1)
string-areaSupervisao	área de supervisão do gerente

Funções membros

void Gerente::calcularSalarioMensal (int *mes*) `[virtual]`

calcularSalarioMensal(int)

Calcula o salário mensal do gerente de acordo com o mês que recebe como parâmetro

Parâmetros

int-mes	mês que se deseja calcular o salário

Implementa **Funcionario** (*p.*).

std::string Gerente::getAreaSupervisao ()

Métodos Get.

Parâmetros

getAreaSupervisao	retorna a área de supervisão como uma string

void Gerente::setAreaSupervisao (std::string *areaSupervisao*)

Métodos Set.

Parâmetros

setAreaSupervisao	instancia a área de supervisão

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Gerente.h
Gerente.cpp

43

Referência da Classe HistoricoArquivo

`#include <HistoricoArquivo.h>`

Membros Públicos

HistoricoArquivo ()
Construtor.

void **setDataModificacao** (int)
setDataModificacao()

void **setModificacao** (int tipoFuncionario, std::string)
setModificacao()

void **setCodigo** (int tipoFuncionario, int codigo)
setCodigo()

int **getCodigo** (int)
getCodigo()

void **setNome** (int tipoFuncionario, std::string nome)
setNome()

std::string **getNome** (int)
getNome()

void **escreveArquivoModificacoes** (int)
escreveArquivoModificacoes()

virtual **~HistoricoArquivo** ()

Construtores e Destrutores

HistoricoArquivo::HistoricoArquivo ()

Construtor.
Inicializa o arquivo de historico

HistoricoArquivo::~HistoricoArquivo () `[virtual]`

Funções membros

void HistoricoArquivo::escreveArquivoModificacoes (int *tipoFuncionario*)

escreveArquivoModificacoes()

Parâmetros

int	tipo de funcionário

Escreve as modificações referente ao tipo de funcionário

int HistoricoArquivo::getCodigo (int *tipoFuncionario*)

getCodigo()

Parâmetros

int	tipo de funcionário

Retorna

retorna o codigo do tipo de funcionário que foi modificado

std::string HistoricoArquivo::getNome (int *tipoFuncionario*)

getNome()

Parâmetros

int	tipo de funcionário

Retorna

retorna o nome do funcionário

void HistoricoArquivo::setCodigo (int *tipoFuncionario*, int *codigo*)

setCodigo()

Parâmetros

int	Tipo de funcionário
int	Codigo do funcionário Armazena o codigo do funcionário que foi realizada a modificação

void HistoricoArquivo::setDataModificacao (int *tipoFuncionario*)

setDataModificacao()

Parâmetros

int	Tipo de funcionário

Armazena a data de modificações dependendo do tipo de funcionário

void HistoricoArquivo::setModificacao (int *tipoFuncionario*, std::string *modificacao*)

setModificacao()

Parâmetros

int	Tipo de funcionário
string	Modificao realizada no funcionário Armazena a modificação referente ao tipo do funcionário

void HistoricoArquivo::setNome (int *tipoFuncionario*, std::string *nome*)

setNome()

Parâmetros

int	Tipo de funcionário
string	Nome do funcionário Armazena a data de modificações dependendo do tipo de funcionário

---

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

HistoricoArquivo.h
HistoricoArquivo.cpp

Referência da Classe Interface

`#include <Interface.h>`

Membros Públicos

int **menu** ()
menu()

void **menuTexto** ()
menuTexto()

Funcionario * lerAtributosFuncionario ()
lerAtributosFuncionario()

int **lerCodigoParaModificarFuncionario** ()
lerCodigoParaModificarFuncionario()

int **lerOpcaoParaModificarFuncionario** ()
lerOpcaoParaModificarFuncionario()

void **opcaoParaModificarFuncionarioTexto** ()
opcaoParaModificarFuncionarioTexto()

std::string **lerNovoAtributoStrParaModificarFuncionario** (int opcao)
lerNovoAtributoStrParaModificarFuncionario()

int **lerNovoAtributoIntParaModificarFuncionario** (int opcao)
lerNovoAtributoIntParaModificarFuncionario()

void **lerNovaDataParaModificarFuncionario** (int *data)
lerNovaDataParaModificarFuncionario()

int **lerCodigoParaExcluirFuncionario** ()
lerCodigoParaExcluirFuncionario()

int **lerCodigoParaExibirFuncionario** ()
lerCodigoParaExibirFuncionario()

int **lerTipoParaExibirFuncionarios** ()
lerTipoParaExibirFuncionarios()

int **lerMesParaCalcularFolhaSalarialEmpresa** ()
lerMesParaCalcularFolhaSalarialEmpresa()

std::string **lerNomeParaImprimirFolhaSalarialFuncionario** ()
lerNomeParaImprimirFolhaSalarialFuncionario()

int **lerCodigoParaImprimirFolhaSalarialFuncionario** ()
lerCodigoParaImprimirFolhaSalarialFuncionario()

int **lerTipoAtributoParaImprimirFolhaSalarialFuncionario** ()
lerTipoAtributoParaImprimirFolhaSalarialFuncionario()

int **lerOpcaoParaImprimirFolhaSalarialEmpresa** ()
lerOpcaoParaImprimirFolhaSalarialEmpresa()

int **lerOpcaoParaBuscarFuncionario** ()
lerOpcaoParaBuscarFuncionario()

int **lerTipoInformacaoStrParaBuscarFuncionario** ()
lerTipoInformacaoStrParaBuscarFuncionario()

std::string **lerInformacaoStrParaBuscarFuncionario** (int tipoInformacao)
lerInformacaoStrParaBuscarFuncionario()

void **lerDataParaBuscarFuncionario** (int *dataInicial, int *dataFinal)
lerDataParaBuscarFuncionario()

int **validaMes** (std::string mes)
validaMes()

int **validaInteiro** (std::string texto)
validaInteiro()

Funções membros

Funcionario * Interface::lerAtributosFuncionario ()

lerAtributosFuncionario()

Lê os atributos necessários e instancia um funcionario.

Retorna

Retorna um ponteiro para o funcionário instanciado.

int Interface::lerCodigoParaExcluirFuncionario ()

lerCodigoParaExcluirFuncionario()

Lê o código de um funcionário para exclui-lo.

Retorna

Retorna um inteiro sendo ele o código lido.

int Interface::lerCodigoParaExibirFuncionario ()

lerCodigoParaExibirFuncionario()

Lê o código de um funcionário para exibi-lo.

Retorna

Retorna um inteiro sendo ele o código lido.

int Interface::lerCodigoParaImprimirFolhaSalarialFuncionario ()

lerCodigoParaImprimirFolhaSalarialFuncionario()

Lê um código de um funcionário para imprimir a folha salarial dele.

Retorna

Retorna um inteiro sendo ele o código lido.

int Interface::lerCodigoParaModificarFuncionario ()

lerCodigoParaModificarFuncionario()

Lê codigo do funcionário que vai ser solicitado uma modificação nos atributos.

Retorna

Retorna um inteiro sendo ele o código do funcionário lido.

void Interface::lerDataParaBuscarFuncionario (int * *dataInicial*, int * *dataFinal*)

lerDataParaBuscarFuncionario()

Lê a data inicial e final de um intervalo que vai ser usado para buscar funcionários nele.

Parâmetros

ponteiro	Aponta para o array que armazena a data inicial.
ponteiro	Aponta para o array que armazena a data final.

Retorna

void.

std::string Interface::lerInformacaoStrParaBuscarFuncionario (int *tipoInformacao*)

lerInformacaoStrParaBuscarFuncionario()

Lê a informação string que vai ser usado para buscar um funcionário.

Parâmetros

int	Tipo de informação string que vai ser lida

Retorna

Retorna uma string sendo ela a informação lida.

int Interface::lerMesParaCalcularFolhaSalarialEmpresa ()

lerMesParaCalcularFolhaSalarialEmpresa()

Lê um mês para calcular a folha salarial dos funcionários da empresa.

Retorna

Retorna um inteiro sendo ele o mês lido.

std::string Interface::lerNomeParaImprimirFolhaSalarialFuncionario ()

lerNomeParaImprimirFolhaSalarialFuncionario()

Lê um nome de um funcionário para imprimir a folha salarial dele.

Retorna

Retorna uma string sendo ela o nome lido.

void Interface::lerNovaDataParaModificarFuncionario (int * *data*)

lerNovaDataParaModificarFuncionario()

Lê uma data para modificar a data de ingresso de um funcionário e armazena em um array.

Parâmetros

Ponteiro	Aponta para a data que vai ser lida no metodo.

Retorna

void.

int Interface::lerNovoAtributoIntParaModificarFuncionario (int *opcao*)

lerNovoAtributoIntParaModificarFuncionario()

Lê um atributo int de acordo com a opção recebida para modificar um atributo int de um funcionário.

Parâmetros

int	Opção de atributo int a ser lido.

Retorna

Retorna o atributo int lido para modificação no funcionário.

std::string Interface::lerNovoAtributoStrParaModificarFuncionario (int *opcao*)

lerNovoAtributoStrParaModificarFuncionario()

Lê um atributo string de acordo com a opção recebida para modificar um atributo string de um funcionário.

Parâmetros

int	Opção de atributo string a ser lido.

Retorna

Retorna o atributo string lido para modificação no funcionário.

int Interface::lerOpcaoParaBuscarFuncionario ()

lerOpcaoParaBuscarFuncionario()

Lê a opção de buscar funcionário.

Retorna

Retorna um inteiro sendo ele a opção lida.

int Interface::lerOpcaoParaImprimirFolhaSalarialEmpresa ()

lerOpcaoParaImprimirFolhaSalarialEmpresa()

Lê a opção de impressão da folha salarial da empresa.

Retorna

Retorna um inteiro sendo ele a opção lida.

int Interface::lerOpcaoParaModificarFuncionario ()

lerOpcaoParaModificarFuncionario()

Lê a opção de modificação de atributo do funcionário a ser modificado.

Retorna

Retorna um inteiro sendo ele a opção de modificação.

int Interface::lerTipoAtributoParaImprimirFolhaSalarialFuncionario ()

lerTipoAtributoParaImprimirFolhaSalarialFuncionario()

Lê o tipo do atributo que vai ser usado para imprimir folha salarial de um funcionário, os atributos podem ser código ou nome.

Retorna

Retorna um inteiro sendo ele o tipo lido.

int Interface::lerTipoInformacaoStrParaBuscarFuncionario ()

lerTipoInformacaoStrParaBuscarFuncionario()

Lê o tipo da informação string que vai ser usado para buscar um funcionário.

Retorna

Retorna um inteiro sendo ele o tipo lido.

int Interface::lerTipoParaExibirFuncionarios ()

lerTipoParaExibirFuncionarios()

Lê o tipo de designação para exibir os funcionários do tipo.

Retorna

Retorna um inteiro sendo ele o tipo lido.

int Interface::menu ()

menu()

Exibe o menu do programa.

Retorna

Retorna um inteiro sendo opção escolhida do menu.

void Interface::menuTexto ()

menuTexto()

Texto exibido no metodo **menu()**.

Retorna

void.

void Interface::opcaoParaModificarFuncionarioTexto ()

opcaoParaModificarFuncionarioTexto()

Texto exibido no metodo **lerOpcaoParaModificarFuncionario()**.

Retorna

void.

int Interface::validaInteiro (std::string *texto*)

validaInteiro()

Valida uma string para verificar se ela armazena um inteiro.

Parâmetros

string	Contem o texto a ser verificado.

Retorna

Retorna um inteiro sendo ele o texto que foi validado na string.

int Interface::validaMes (std::string *mes*)

validaMes()

Valida se o mês recebido é valido.

Parâmetros

string	Contem o mês a ser validado.

Retorna

Retorna um inteiro sendo ele o mês que foi validado.

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Interface.h
Interface.cpp

Referência da Classe InvalidoArgumentoArquivoExcept

Função exception que invalida a alteração do presidente.

`#include <InvalidoArgumentoArquivoException.h>`

Membros Públicos

InvalidoArgumentoArquivoExcept (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Função exception que invalida a alteração do presidente.

Construtores e Destrutores

InvalidoArgumentoArquivoExcept::InvalidoArgumentoArquivoExcept (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * InvalidoArgumentoArquivoExcept::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

Atributos

char InvalidoArgumentoArquivoExcept::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

InvalidoArgumentoArquivoException.h
InvalidoArgumentoArquivoException.cpp

Referência da Classe OpcaoInvalidaException

Exception personalizado criado para reconhecer e informar ao usuário caso haja uma digitação incorreta ou uma opção inválida.

`#include <OpcaoInvalidaException.h>`

Membros Públicos

OpcaoInvalidaException (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Exception personalizado criado para reconhecer e informar ao usuário caso haja uma digitação incorreta ou uma opção inválida.

Construtores e Destrutores

OpcaoInvalidaException::OpcaoInvalidaException (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * OpcaoInvalidaException::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

55

Atributos

char OpcaoInvalidaException::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

OpcaoInvalidaException.h
OpcaoInvalidaException.cpp

Referência da Classe Operador

Operador é uma classe filha de Funcionário.
`#include <Operador.h>`

Membros Públicos

Operador ()
Construtor.

Operador (int codigo, std::string nome, std::string CPF, int idade, std::string CEP, int numeroResidencia, std::string telefone, int *data, int designacao)
Construtor.

void **calcularSalarioMensal** (int mes)
calcularSalarioMensal(int)

Outros membros herdados

Descrição detalhada

Operador é uma classe filha de Funcionário.

Construtores e Destrutores

Operador::Operador ()

Construtor.

Construtor padrão sem argumentos

Operador::Operador (int *codigo*, std::string *nome*, std::string *CPF*, int *idade*, std::string *CEP*, int *numeroResidencia*, std::string *telefone*, int * *data*, int *designacao*)

Construtor.

Inicializa os atributos do operador

Parâmetros

int-codigo	código do operador
string-nome	nome completo do operador
string-CPF	CPF do operador
int-idade	idade do operador
string-endereco	CEP da residência do operador

int-numeroResidencia	número da residência do operador
string-telefone	telefone do operador
int-data*	array de inteiros referentes a dia/mes/ano do ingresso do operador
int-designacao	designacao do operador (0)

Funções membros

void Operador::calcularSalarioMensal (int *mes*) `[virtual]`

calcularSalarioMensal(int)

Calcula o salário mensal do operador de acordo com o mês que recebe como parâmetro

Parâmetros

int-mes	mês que se deseja calcular o salário

Implementa **Funcionario** (*p.*).

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Operador.h
Operador.cpp

Referência da Classe Pessoa

`#include <Pessoa.h>`
Diagrama de hierarquia para Pessoa:

Membros Públicos

Pessoa ()
Construtor.

Pessoa (std::string nome, std::string telefone, std::string CPF, int idade, std::string CEP, int numeroResidencia)
Construtor.

virtual **~Pessoa** ()
Destrutor.

std::string **getNome** ()
Métodos Get.

std::string **getTelefone** ()
std::string **getCPF** ()
int **getIdade** ()
Endereco * getEndereco ()
void **setNome** (std::string nome)
Métodos Set.

void **setTelefone** (std::string telefone)
void **setCPF** (std::string CPF)
void **setIdade** (int idade)
void **setEndereco** (std::string endereco)
void **validaNome** (std::string nome)
validaNome(string)

int **validaIdade** (std::string idadeStr)
validaIdade(string)

Construtores e Destrutores

Pessoa::Pessoa ()

Construtor.
Construtor padrão sem argumentos

Pessoa::Pessoa (std::string *nome*, std::string *telefone*, std::string *CPF*, int *idade*, std::string *CEP*, int *numeroResidencia*)

Construtor.

Inicializa os atributos da pessoa

Parâmetros

string-nome	nome completo da pessoa
string-telefone	telefone da pessoa
string-CPF	CPF da pessoa
int-idade	idade da pessoa
string-endereco	CEP da residência da pessoa
int-numeroResidencia	número da residência da pessoa

Pessoa::~Pessoa () `[virtual]`

Destrutor.

Destrutor padrão

Funções membros

std::string Pessoa::getCPF ()

Endereco * Pessoa::getEndereco ()

int Pessoa::getIdade ()

std::string Pessoa::getNome ()

Métodos Get.

Parâmetros

getNome	retorna o nome da pessoa em forma de string
getTelefone	retorna o telefone da pessoa em forma de string
getCPF	retorna o CPF da pessoa em forma de string
getIdade	retorna a idade da pessoa em forma de int
getEndereco	retorna um ponteiro para o endereço desejado

std::string Pessoa::getTelefone ()

void Pessoa::setCPF (std::string *CPF*)

void Pessoa::setEndereco (std::string *endereco*)

void Pessoa::setIdade (int *idade*)

void Pessoa::setNome (std::string *nome*)

Métodos Set.

Parâmetros

setNome	instancia o nome da pessoa
setTelefone	instancia o telefone da pessoa
setCPF	instancia o CPF da pessoa
setIdade	instancia a idade da pessoa
setEndereco	instancia o endereço da pessoa

void Pessoa::setTelefone (std::string *telefone*)

int Pessoa::validaIdade (std::string *idadeStr*)

validaIdade(string)

Método para validar a idade digitada pelo usuário

Parâmetros

string-codigoStr	código digitado pelo usuário

void Pessoa::validaNome (std::string *nome*)

validaNome(string)

Método para validar o nome digitado pelo usuário

Parâmetros

string-codigoStr	código digitado pelo usuário

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Pessoa.h
Pessoa.cpp

Referência da Classe Presidente

Presidente é uma classe filha de Funcionário.
`#include <Presidente.h>`

Membros Públicos

Presidente ()
Construtor.

Presidente (int codigo, std::string nome, std::string CPF, int idade, std::string endereco, int numeroResidencia, std::string telefone, int *data, int designacao, std::string areaFormacao, std::string formacaoMax)
Construtor.

virtual std::string **getAreaFormacao** ()
Métodos Get.

virtual std::string **getFormacaoMax** ()
void **setAreaFormacao** (std::string areaFormacao)
Métodos Set.

void **setFormacaoMax** (std::string formacaoMax)
void **calcularSalarioMensal** (int mes)
calcularSalarioMensal(int)

Outros membros herdados

Descrição detalhada

Presidente é uma classe filha de Funcionário.

Construtores e Destrutores

Presidente::Presidente ()

Construtor.

Construtor padrão sem argumentos

Presidente::Presidente (int *codigo*, std::string *nome*, std::string *CPF*, int *idade*, std::string *endereco*, int *numeroResidencia*, std::string *telefone*, int * *data*, int *designacao*, std::string *areaFormacao*, std::string *formacaoMax*)

Construtor.

Inicializa os atributos do presidente

Parâmetros

int-codigo	código do presidente
string-nome	nome completo do presidente
string-CPF	CPF do presidente
int-idade	idade do presidente
string-endereco	CEP da residência do presidente
int-numeroResidencia	número da residência do presidente
string-telefone	telefone do presidente
int-data*	array de inteiros referentes a dia/mes/ano do ingresso do presidente
int-designacao	designacao do presidente (3)
string-areaFormacao	área de formação do presidente
string-formacaoMax	formação máxima do presidente

Funções membros

void Presidente::calcularSalarioMensal (int *mes*) `[virtual]`

calcularSalarioMensal(int)

Calcula o salário mensal do presidente de acordo com o mês que recebe como parâmetro

Parâmetros

int-mes	mês que se deseja calcular o salário

Implementa **Funcionario** (*p.*).

std::string Presidente::getAreaFormacao () `[virtual]`

Métodos Get.

Parâmetros

getAreaFormacao	retorna a área de formação como uma string
getFormacaoMax	retorna a formação máxima como uma string

std::string Presidente::getFormacaoMax () `[virtual]`

void Presidente::setAreaFormacao (std::string *areaFormacao*)

Métodos Set.

Parâmetros

setAreaFormacao	instancia a área de formação
setFormacaoMax	instancia a formação máxima

void Presidente::setFormacaoMax (std::string *formacaoMax*)

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

Presidente.h
Presidente.cpp

64

Referência da Classe TelefoneException

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar um código que já está em uso.

`#include <TelefoneException.h>`

Membros Públicos

TelefoneException (const char *e)
Construtor.

virtual const char * **what** ()
what( )

Atributos Protegidos

char **mensagem** [100]

Descrição detalhada

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar um código que já está em uso.

Construtores e Destrutores

TelefoneException::TelefoneException (const char * e)

Construtor.

Inicializa o atributo mensagem[] com a mensagem passada pela chamada do erro.

Parâmetros

char-e*	array de char referente a mensagem de erro

Funções membros

const char * TelefoneException::what () `[virtual]`

what( )

Método para a exibição da mensagem de erro

65

Atributos

char TelefoneException::mensagem[100] `[protected]`

A documentação para essa classe foi gerada a partir dos seguintes arquivos:

TelefoneException.h
TelefoneException.cpp

66

Referência da Classe TentativaAbrirArquivo

TentativaAbrirArquivo.

`#include <TentativaAbrirArquivo.h>`

Membros Públicos

TentativaAbrirArquivo (const std::string &nomeArquivo)

TentativaAbrirArquivo()

Descrição detalhada

TentativaAbrirArquivo.

Classe responsável pelo erro de abrir algum arquivo

Construtores e Destrutores

TentativaAbrirArquivo::TentativaAbrirArquivo (const std::string & *nomeArquivo*) `[inline]`

TentativaAbrirArquivo()

Responsável por enviar o throw sobre uma tentativa de abrir o arquivo envia a string para a classe mão de runtime_error

Parâmetros

sting-nomeArquivo	recebe o nome do arquivo que não conseguiu abrir.

Retorna

Nada

A documentação para essa classe foi gerada a partir do seguinte arquivo:

TentativaAbrirArquivo.h

67

Arquivos

Referência do Arquivo Arquivo.cpp

```
#include "Arquivo.h"
#include <iostream>
#include <string.h>
#include <sstream>
```


68

Referência do Arquivo Arquivo.h

```
#include <string>
#include <fstream>
#include "Funcionario.h"
#include "Gerente.h"
#include "Operador.h"
#include "Presidente.h"
#include "Diretor.h"
#include "HistoricoArquivo.h"
#include <vector>
#include "TentativaAbrirArquivo.h"
#include "InvalidoArgumentoArquivoException.h"
```

Componentes

class **Arquivo**

Definições e Macros

#define **QUANTIA_ARQUIVOS** 4
#define **TAMANHO_MAX** 20

Definições e macros

#define QUANTIA_ARQUIVOS 4

#define TAMANHO_MAX 20

Referência do Arquivo CadastrarFuncionarioException.cpp

```
#include "CadastrarFuncionarioException.h"
#include <string.h>
```


70

Referência do Arquivo CadastrarFuncionarioException.h

```
#include <exception>
```

Componentes

class **CadastrarFuncionarioException**

Exception personalidado usado para detectar e informar os erros que podem ocorrer durante o cadastramento de um funcionário.

71

Referência do Arquivo CEPException.cpp

```
#include "CEPException.h"
#include <string.h>
```


72

Referência do Arquivo CEPException.h

```
#include <exception>
```

Componentes

class **CEPException**

Exception personalizado criado para reconhecer e informar ao usuário possíveis erros durante o cadastro do CEP do funcionário.

73

Referência do Arquivo CPFException.cpp

```
#include "CPFException.h"
#include <string.h>
```


74

Referência do Arquivo CPFException.h

```
#include <exception>
```

Componentes

class **CPFException**

Exception personalizado criado para reconhecer e informar ao usuário possíveis erros durante o cadastro do CPF do funcionário.

75

Referência do Arquivo Data.cpp

```
#include "Data.h"
#include <iostream>
#include <chrono>
```


76

Referência do Arquivo Data.h

```
#include <string>
```

Componentes

class **Data**

77

Referência do Arquivo Diretor.cpp

```
#include "Diretor.h"
```


78

Referência do Arquivo Diretor.h

```
#include "Funcionario.h"
#include <string>
```

Componentes

class **Diretor**

Diretor *é uma classe filha de Funcionário.*

79

Referência do Arquivo Empresa.cpp

```
#include "Empresa.h"
```


80

Referência do Arquivo Empresa.h

```
#include <iostream>
#include <vector>
#include <string>
#include "Operador.h"
#include "Gerente.h"
#include "Diretor.h"
#include "Presidente.h"
#include "Data.h"
#include "FuncionarioJaCadastradoExcept.h"
#include "FuncionarioNaoEstaCadastradoExcept.h"
#include "Arquivo.h"
```

Componentes

class **Empresa**

Definições e Macros

#define **OPERADOR** 0
#define **GERENTE** 1
#define **DIRETOR** 2
#define **PRESIDENTE** 3
#define **BUSCAR_POR_NOME** 5
#define **BUSCAR_POR_ENDERECO** 6
#define **QTD_DE_TIPOS** 4

Definições e macros

#define BUSCAR_POR_ENDERECO 6

#define BUSCAR_POR_NOME 5

#define DIRETOR 2

#define GERENTE 1

#define OPERADOR 0

#define PRESIDENTE 3

#define QTD_DE_TIPOS 4

Referência do Arquivo Endereco.cpp

```
#include "Endereco.h"
#include "CEPException.h"
#include <fstream>
#include <string.h>
```


82

Referência do Arquivo Endereco.h

```
#include <iostream>
#include "CadastrarFuncionarioException.h"
```

Componentes

class **Endereco**

Classe criada para facilitar o amarzenamento de dados do endereço.

Referência do Arquivo FolhaSalarial.cpp

```
#include "FolhaSalarial.h"
```


84

Referência do Arquivo FolhaSalarial.h

```
#include <vector>
#include <iostream>
```

Componentes

class **FolhaSalarial**

Referência do Arquivo Funcionario.cpp

```
#include "Funcionario.h"
#include <cstdlib>
#include <ctime>
```


86

Referência do Arquivo Funcionario.h

```
#include "Endereco.h"
#include "Pessoa.h"
#include "FolhaSalarial.h"
#include <string>
#include "Data.h"
#include "CadastrarFuncionarioException.h"
```

Componentes

class **Funcionario**

Funcionario *é uma classe filha de pessoa.*

Referência do Arquivo FuncionarioJaCadastradoExcept.cpp

```
#include "FuncionarioJaCadastradoExcept.h"
#include <string.h>
```


88

Referência do Arquivo FuncionarioJaCadastradoExcept.h

```
#include <exception>
```

Componentes

class **FuncionarioJaCadastradoExcept**

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar um código que já está em uso.

89

Referência do Arquivo FuncionarioNaoEstaCadastradoExcept.cpp

```
#include "FuncionarioNaoEstaCadastradoExcept.h"
#include <string.h>
```


90

Referência do Arquivo FuncionarioNaoEstaCadastradoExcept.h

```
#include <exception>
```

Componentes

class **FuncionarioNaoEstaCadastradoExcept**

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar algum método que exija a existência de um funcionário.

91

Referência do Arquivo Gerente.cpp

```
#include "Gerente.h"
```


92

Referência do Arquivo Gerente.h

```
#include "Funcionario.h"
#include <string>
```

Componentes

class **Gerente**

Gerente *é uma classe filha de Funcionário.*

Referência do Arquivo HistoricoArquivo.cpp

```
#include "HistoricoArquivo.h"
#include <iostream>
```


94

Referência do Arquivo HistoricoArquivo.h

```
#include <fstream>
#include "Data.h"
#include <string>
#include "TentativaAbrirArquivo.h"
```

Componentes

class **HistoricoArquivo**

Definições e Macros

#define **QUANTIA_ARQUIVOS** 4

Definições e macros

#define QUANTIA_ARQUIVOS 4

Referência do Arquivo Interface.cpp

```
#include "Interface.h"
```


96

Referência do Arquivo Interface.h

```
#include <iostream>
#include "Operador.h"
#include "Gerente.h"
#include "Diretor.h"
#include "Presidente.h"
#include "TelefoneException.h"
#include "CPFException.h"
#include "CEPException.h"
#include "CadastrarFuncionarioException.h"
#include "OpcaoInvalidaException.h"
```

Componentes

class **Interface**

97

Referência do Arquivo InvalidoArgumentoArquivoException.cpp

```
#include "InvalidoArgumentoArquivoException.h"
#include <string.h>
```


98

Referência do Arquivo InvalidoArgumentoArquivoException.h

```
#include <exception>
```

Componentes

class **InvalidoArgumentoArquivoExcept**

Função exception que invalida a alteração do presidente.

99

Referência do Arquivo main.cpp

```
#include <iostream>
#include "Empresa.h"
#include "Operador.h"
#include "Gerente.h"
#include "Diretor.h"
#include "Presidente.h"
#include "Interface.h"
#include "FuncionarioJaCadastradoExcept.h"
#include "FuncionarioNaoEstaCadastradoExcept.h"
#include "OpcaoInvalidaException.h"
#include <locale.h>
```

Definições e Macros

#define **INT** 1
#define **STR** 2

Funções

int **main** ()

Definições e macros

#define INT 1

#define STR 2

Funções

int main ()

Referência do Arquivo OpcaoInvalidaException.cpp

```
#include "OpcaoInvalidaException.h"
#include <string.h>
```


101

Referência do Arquivo OpcaoInvalidaException.h

```
#include <exception>
```

Componentes

class **OpcaoInvalidaException**

Exception personalizado criado para reconhecer e informar ao usuário caso haja uma digitação incorreta ou uma opção inválida.

102

Referência do Arquivo Operador.cpp

```
#include "Operador.h"
```


103

Referência do Arquivo Operador.h

```
#include "Funcionario.h"
```

Componentes

class **Operador**
***Operador** é uma classe filha de Funcionário.*

104

Referência do Arquivo Pessoa.cpp

```
#include "Pessoa.h"
#include "TelefoneException.h"
#include "CPFException.h"
```


105

Referência do Arquivo Pessoa.h

```
#include "Endereco.h"
#include "CadastrarFuncionarioException.h"
```

Componentes

class **Pessoa**

106

Referência do Arquivo Presidente.cpp

```
#include "Presidente.h"
```


107

Referência do Arquivo Presidente.h

```
#include "Funcionario.h"
#include <string>
```

Componentes

class **Presidente**

Presidente *é uma classe filha de Funcionário.*

108

Referência do Arquivo TelefoneException.cpp

```
#include "TelefoneException.h"
#include <string.h>
```


109

Referência do Arquivo TelefoneException.h

```
#include <exception>
```

Componentes

class **TelefoneException**

Exception personalizado criado para reconhecer e informar ao usuário na hora de utilizar um código que já está em uso.

110

Referência do Arquivo TentativaAbrirArquivo.h

```
#include <stdexcept>
```

Componentes

class **TentativaAbrirArquivo**
TentativaAbrirArquivo.