Documentação

Sistema de Gerenciamento de Folha de Pagamento

Rayque Alencar de Melo
Matrícula: 20200015109

João Pessoa - PB
2022

Resumos sobre as classes

Nesse tópico são listadas as classes que contém no projeto junto com uma explicação breve do seu funcionamento.

Funcionarios - Classes que definem como os funcionários são feitos, era e uma classe abstrata, pois possui métodos para serem implementados nas classes filhas dela. Possui quatro classes filhas que herdam dela, sendo as seguintes: Operador, Gerente, Diretor, Presidente.

Operador - Tem todos os atributos da classe mãe Funcionarios, e implementando os métodos do tipo virtual puro que tem na classe mãe.

Gerente - Tem todos os atributos da classe mãe, e mais um atributos chamado “areaDeSupervisao”, que é um atributo criado somente para essa classe Gerente, implementando os métodos do tipo virtual puro que tem na classe mãe.

Diretor - Tem todos os atributos da classe mãe, e mais dois atributos chamados “areaDeSupervisao” e “areaDeFormacao”, são atributos criados para funcionamento da classe Diretor, implementando os métodos do tipo virtual puro que tem na classe mãe.

Presidente - Tem todos os atributos da classe mãe, e mais dois atributos chamados “areaDeFormacao” e “formacaoAcademicaMaxima”, são atributos criados para funcionamento da classe Presidente , implementando os métodos do tipo virtual puro que tem na classe mãe.

Essas 4 classes citadas acima herdam de Funcionarios, nela foi tentado fazer a ideia de polimorfismo.

Existem mais três classes que foram feitas para utilizar na classe mãe Funcionários, são elas Data, Endereco, FolhaSalarial. Elas foram Feita para fazer o tratamento e armazenamento de informação de um melhor maneira, a classe Data foi feita para fazendo o manipulamento das datas que faziam parte do cadastro de funcionários, juntamente com as outras duas, Endereco, para o tratamento e armazenamento de todas as informações de endereço do funcionário, já a FolhaSarial para ser feito o cálculo das folhas salariais dos funcionários. Com elas o funcionamento da class Funcionario foi melhorado.

Uma outra classe foi criado a Menu, nela é feita todo o funcionamento do programa sendo da parte mais básica com printar o menu inicial, a todas as outras funções do projeto com adicionar funcionário, editar funcionário, excluir funcionário, calcular folhas salarial, exibir folha salarial, entre outras funções.

Hierarquia das classes

Data.cpp
Data.h

Endereco.cpp
Data.h

FolhaSalarial.cpp
FolhaSalarial.h

Menu.cpp
Menu.h

Funcionarios.cpp
Funcionarios.h

    Operador.cpp
    Operador.h
    Gerente.cpp
    Gerente.h
    Diretor.cpp
    Diretor.h
    presidente.cpp
    presidente.h

Composição das Classes

Classe Data:

public:

```
//Construtores
Data();
Data(int dia, int mes, int ano);

//Metodos set
void setDia(int dia);
void setMes(int mes);
void setAno(int ano);

//Metodos get
int getDia();
int getMes();
int getAno();
```

```
    //Metodos para exibir a data
    void exibirData();

private:
    //Atributos
    int dia;
    int mes;
    int ano;
```

Classe Endereco:

```
public:
    //Construtores
    Endereco();
    Endereco(string cep, string numero);

    //metodo para pegar os dados do endereco com o cep
    void parseCEP();

    //metodos getters
    string getRua();
    string getBairro();
    string getCidade();
    string getEstado();
    string getCep();
    string getNumero();

    //metodos setters
    void setRua(string rua);
    void setBairro(string bairro);
    void setCidade(string cidade);
    void setEstado(string estado);
    void setCep(string cep);
    void setNumero(string numero);

    //metodo para imprimir os dados do endereco
    void exibirEnderecoCompleto();

private:
    //atributos
    string rua, bairro, cidade, estado, cep, numero;
```

classe FolhaSalarial:

public:

```
    //Construtore
    FolhaSalarial();

    //Metodos
    void calcularINSS(float salario);
    void calcularIRRF(float salario);
    void calcularSalarioLiquido(float salario);
    void calcularSalarioMensal(float salario);

    //lista de metodos set
    void setDiasTrabalhados(float diasTrabalhados);
    void setHorasExtras(float horasExtras);

    //lista de metodos get
    float getsalario();
    float getINSS();
    float getIRRF();
    float getSalarioLiquido();
    float getSalarioMesBrutoComHorasExtras();
    float getSalarioExtra();
    float getDiasTrabalhados();
    float getHorasExtras();
```

private:

```
    //atributos
    float salario, salarioLiquido , INSS , IRRF, salarioMesBrutoComHorasExtras, salarioExtra;
    float horasExtras, diasTrabalhados;
```

Classe Funcionarios:

public:

```
    //construtores
    Funcionarios();
     Funcionarios(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data
dataDeIngresso, string designacao, float salario);

    //metodos get
    int getNumeroDoFuncionario();
    string getNome();
    Endereco getEndereco();
    string getTelefone();
```

```
    Data getDataDeIngresso();
    string getDesignacao();
    float getSalario();
    FolhaSalarial getFolhaSalarial(int mes);

    //metodos set
    void setNumeroDoFuncionario(int);
    void setNome(string);
    void setEndereco(string cep, string numero);
    void setTelefone(string);
    void setDataDeIngresso(Data);
    void setDesignacao(string);
    void setSalario(float);

    //metodos
    void calcularFolhaSalarial();
    void setDiasTrabalhados(int diasTrabalhados, int mes);
    void setHorasExtras(int horasExtra, int mes);
    void calculaFolhaMensal(float Salario, int mes);
    void imprimirFolhaSalarial(int mes);

    //metodos para usar folha salarial
    void exibirFolhaAnual();
    void exibirFolhaMes(int mes);
    float getSalarioAnual();

    //metodos virtual puro
    virtual void LerAtributos(int numFuncionario) = 0;
    virtual void exibirAtributos() = 0;
    virtual void aumentoSalario() = 0;

    //metodos virtual
    virtual string getAreaDeFormacao();
    virtual string getAreaSupervisao();
    virtual string getFormacaoAcademicaMaxima();
    virtual void setAreaSupervisao(string areaSupervisao);
    virtual void setAreaDeFormacao(string areaDeFormacao);
    virtual void setFormacaoAcademicaMaxima(string FormacaoAcademicaMaxima);


private:

protected:
    //atributos
    int numeroDoFuncionario;
    string nome;
    Endereco endereco;
```

```
    string telefone;
    Data dataDeIngresso;
    string designacao;
    float salario;
    FolhaSalarial folhaSalarial[12];
```

Class Operador:

```
public:
    //construtores
    Operador();
      Operador(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data dataDeIngresso, string designacao, float salario);

    //metodos virutal puro herdados da class funcionario
    void LerAtributos(int numFuncionario);
    void exibirAtributos();
    void aumentoSalario();

    //metodos virtual herdados da class funcionario
    string getAreaDeFormacao();
    string getAreaSupervisao();
    string getFormacaoAcademicaMaxima();
    void setAreaSupervisao(string areaSupervisao);
    void setAreaDeFormacao(string areaDeFormacao);
    void setFormacaoAcademicaMaxima(string FormacaoAcademicaMaxima);

private:
```

Class Gerente:

```
public:
    Gerente();

    //construtor da classe Gerente para criar quando ler do arquivo
      Gerente(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data dataDeIngresso, string designacao, float salario, string getAreaDeSupervisao);

    //constructor da classe Gerente para quando criar funcionario manualmente
      Gerente(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data dataDeIngresso, string designacao, float salario);

    //metodos virutal puro herdados da class funcionario
    void LerAtributos(int numFuncionario);
```

```
    void exibirAtributos();
    void aumentoSalario();

    //metodos para o atributo da class gerente
    string getAreaDeSupervisao();
    void setAreaDeSupervisao(string areaDeSupervisao);

    //metodos herdados da class funcionario
    string getAreaDeFormacao();
    string getFormacaoAcademicaMaxima();
    void setAreaDeFormacao(string areaDeFormacao);
    void setFormacaoAcademicaMaxima(string FormacaoAcademicaMaxima);

private:

    //atributos da classe Gerente
    string areaDeSupervisao;
```

Class Diretor:

```
public:
    //Construtores
    Diretor();
        Diretor(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data dataDeIngresso, string designacao, float salario, string AreaDeSupervisao, string areaDeFormacao);
        Diretor(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data dataDeIngresso, string designacao, float salario);

    //Metodos virutais da classe Funcionarios
    void LerAtributos(int numFuncionario);
    void exibirAtributos();
    void aumentoSalario();

    //Metodos especificos da classe Diretor
    string getAreaDeSupervisao();
    void setAreaDeSupervisao(string areaDeSupervisao);
    string getAreaDeFormacao();
    void setAreaDeFormacao(string areaDeFormacao);

    string getFormacaoAcademicaMaxima();
```

```
    void setFormacaoAcademicaMaxima(string FormacaoAcademicaMaxima);

private:
    //Atributos especificos da classe Diretor
    string areaDeSupervisao;
    string areaDeFormacao;
```

Class Presidente:

```
public:
    //Construtores
    Presidente();
      Presidente(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data dataDeIngresso, string designacao, float salario, string areaDeFormacao, string formacaoAcademicaMaxima);
      Presidente(int numeroDoFuncionario, string nome, Endereco endereco, string telefone, Data dataDeIngresso, string designacao, float salario);

    //Metodos virutais da classe Funcionarios
    void LerAtributos(int numFuncionario);
    void exibirAtributos();
    void aumentoSalario();

    //Metodos especificos da classe Presidente
    string getAreaDeFormacao();
    void setAreaDeFormacao(string areaDeFormacao);
    string getFormacaoAcademicaMaxima();
    void setFormacaoAcademicaMaxima(string formacaoAcademicaMaxima);

    string getAreaSupervisao();
    void setAreaSupervisao(string areaSupervisao);

private:
    //Atributos especificos da classe Presidente
    string areaDeFormacao;
    string formacaoAcademicaMaxima;
```

Class Menu:

public:

```
    Menu();
    void menuInicial();
    void adicionarFuncionario();
    void exibirFuncionario(int indice);
    void exibirListaDeFuncionarios();
    void procurarFuncionario();
    void editarFuncionario();
    void excluirFuncionario();
    void exibirFuncionariosPorTipo(string tipo);

    void concederAumento();
    void calcularFolhaSalarial();
    void setDiasAleatorios(int mes, int diasAleatorios);
    void setHorasAleatorias(int mes, int horasAleatorias);
    void imprimeFolhaSalarialFuncionario();
    void exibirFolhaDaEmpresa();

    int getFuncionarioPeloNumeroDoFuncionario(int numeroDoFuncionario);
    int getFuncionarioPorNome(string nome);

    void salvarArquivoFuncionario();
    void lerArquivoFuncionario();
    void salvarArquivoFuncionario(int);
```

private:

```
    //atributos
    bool folhaSalarialCalculada[12];
    vector<Funcionarios*> vFuncionarios;
```

Fazer uma breve explicação de como funciona a class menu, já que ela é que faz toda a implementação do projeto.

Começando com os Atributos dela temos um Vector<Funcionarios*> vFuncionarios, o vector foi

feito com um ponteiro de funcionários, já que a classe Funcionários e abstrata não podemos chamar instanciar ela diretamente.

Codigo abaixo mostrando como foi feito a criação dos funcionários:

```
Funcionarios *funcionario;

    switch (opcao){

    case 1:
        funcionario = new Operador(numeroDoFuncionario, nome, endereco, telefone, dataDeIngresso, "Operador", salario);
        funcionario->LerAtributos(numeroDoFuncionario);
        break;

    case 2:
        funcionario = new Gerente(numeroDoFuncionario, nome, endereco, telefone, dataDeIngresso, "Gerente", salario);
        funcionario->LerAtributos(numeroDoFuncionario);
        break;

    case 3:
        funcionario = new Diretor(numeroDoFuncionario, nome, endereco, telefone, dataDeIngresso, "Diretor" ,salario);
        funcionario->LerAtributos(numeroDoFuncionario);
        break;

    case 4:
        funcionario = new Presidente(numeroDoFuncionario, nome, endereco, telefone, dataDeIngresso, "Presidente" ,salario);
        funcionario->LerAtributos(numeroDoFuncionario);
        break;
    }


vFuncionarios.push_back(funcionario);
```

O código foi todo comentado, podendo ser utilizado para obter mais informações sobre como está ocorrendo o funcionamento do projeto.

Diagrama das Classes(ULM)