
LICENCIATURA EM ENGENHARIA INFORMÁTICA
2023/2024

# Programação Orientada a Objetos

## Trabalho Prático – Zonas de Habitação em C++

Bruno Oliveira - 2019136478
Fernando Pereira - 2020154532

# Índice

# Opções Tomadas

Foram criados ficheiros .h para os Aparelhos, Habitação, Interface, Processador, Propriedade,

Regras, Sensor e Zona.

Foram criados ficheiros .cpp para a Habitação, Interface e Zona.

Em todas as classes foram criadas o seu constructor e destrutor.

Na função validaComandos obtámos por utilizar strings de validação em vez de vectores, o que foi um erro pois deu mais trabalho e o Código ficou mais confuso.

# Implementação

## Interface

Foi criada com a intenção de gerir as janelas, os ficheiros e os comandos utilizados no trabalho, fazendo com que o Código estivesse mais organizado e visualmente apelativo.

```
class Interface{
public:
    Interface(Terminal &t);
    ~Interface();
    void validaComandos(string comando);
    bool leFicheiro(string fileName);
    void menuInterface(Terminal &t);

private:
    Terminal &t;
    Zona *z;
    Window comandos;
    Window mapa;
    Window out;
};
```

No ficheiro Interface.cpp foram implementados as 3 funções:

- validaComandos(string comando) – Valida os comandos pedidos e os seus parâmetros
- leFicheiro(string fileName) – Lê o ficheiro de texto com comandos
- menuInterface (Terminal &t) – Cria as 3 janelas pretendidas(Mapa, Input e Output)

## Habitação

Esta classe ainda não faz nada mas foi inicializada com o propósito de na meta 2 ser devidamente implementada e ajustada conforme pedido.

```
class Habitacao {
public:
    Habitacao(int numLinhas, int numColunas);
    ~Habitacao();

    void adicionarZona(Zona zona);
    void removerZona(string idZona);

    void listarZonas() const;

private:
    int numLinhas;
    int numColunas;

    bool posicaoValida(int linha, int coluna) const;
};
```

Esta classe vai servir de esqueleto para a criação das zonas.

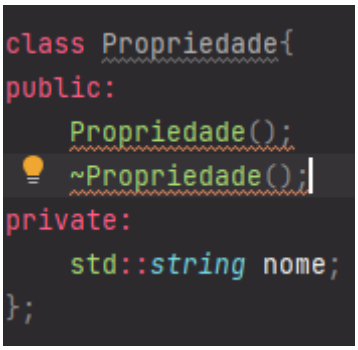
## Processador

Esta classe ainda não faz nada mas foi inicializada com o propósito de na meta 2 ser devidamente implementada e ajustada conforme pedido.

```
class Processador{
public:
    Processador();
    ~Processador();
    void adicionaRegra();
    void removerRegra();

private:
    std::string comando;
    int id;
};
```

## Propriedade

Esta classe ainda não faz nada mas foi inicializada com o propósito de na meta 2 ser devidamente implementada e ajustada conforme pedido.

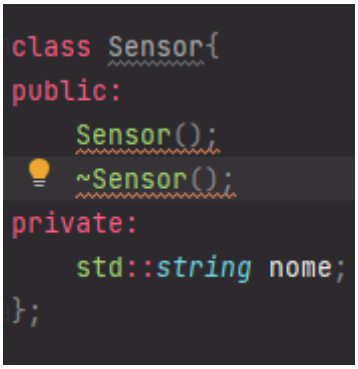



## Regras

Esta classe ainda não faz nada mas foi inicializada com o propósito de na meta 2 ser devidamente implementada e ajustada conforme pedido.


## Sensor

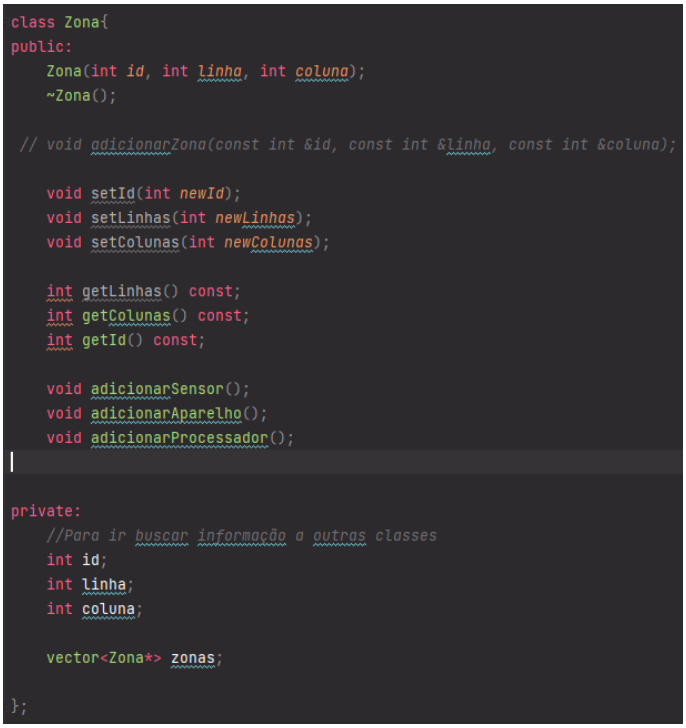
Esta classe ainda não faz nada mas foi inicializada com o propósito de na meta 2 ser devidamente implementada e ajustada conforme pedido.



## Zona

Esta classe foi criada e foi-lhe adicionado getters e settersnde linhas e colunas para a posterior implementação da sua criação e remoção.

Cada Zona tem o seu ID.



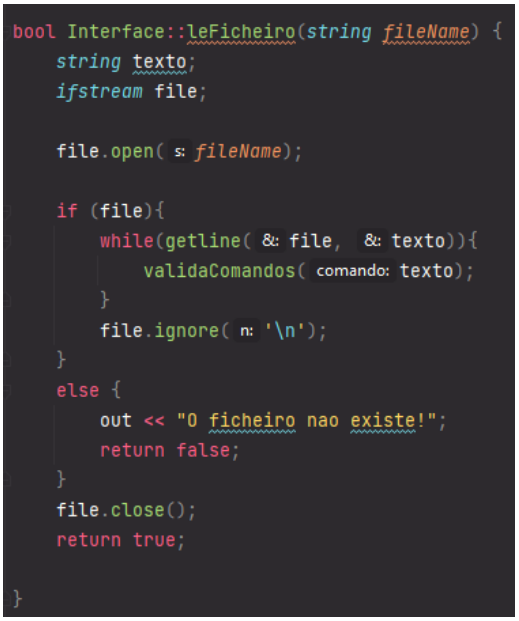
```
class Zona{
public:
    Zona(int id, int linha, int coluna);
    ~Zona();

 // void adicionarZona(const int &id, const int &linha, const int &coluna);

    void setId(int newId);
    void setLinhas(int newLinhas);
    void setColunas(int newColunas);

    int getLinhas() const;
    int getColunas() const;
    int getId() const;

    void adicionarSensor();
    void adicionarAparelho();
    void adicionarProcessador();


private:
    //Para ir buscar informação a outras classes
    int id;
    int linha;
    int coluna;

    vector<Zona*> zonas;

};
```

As funções a ser implementadas pela Zona:

- adicionarAparelho() – Adiciona um aparelho
- adicionarSensor() – Adiciona um Sensor
- AdicionaProcessador() – Adiciona um processador

## Leitura de comandos do ficheiro

Nesta função conseguimos ler o ficheiro txt (comandos.txt) e verificamos os comandos dentro do mesmo com recurso à função validaComandos.



```
bool Interface::leFicheiro(string fileName) {
    string texto;
    ifstream file;

    file.open( s: fileName);

    if (file){
        while(getline( &: file,  &: texto)){
            validaComandos( comando: texto);
        }
        file.ignore( n: '\n');
    }
    else {
        out << "O ficheiro nao existe!";
        return false;
    }
    file.close();
    return true;

}
```

# Conclusão

Com esta meta conseguimos implementar os cohecimentos adquiridos durante as aulas teóricas e práticas ao longo deste semestre.

Tivemos vários problemas com a biblioteca dada pelos professores com erros desnecessários e por vezes incompreensíveis.

Como grupo temos que trabalhar mais atempadamente para que consigamos na meta 2 realizar o trabalho com mais fluidez e coesão.

Na meta 2 iremos retificar erros encontrados ao longo desta primeira meta.