Aula 5

Programação II

Prof. Sandino Jardim

CC-UFMT-CUA

Interface pública vs. Implementação

A interface pública de um objeto contém os membros visíveis a outros objetos.

- Declarados com o modificador de acesso `public`.

A implementação de um objeto contém os membros privados do objeto.

- Visíveis apenas dentro dos objetos onde eles foram declarados.
- Declarados com o modificador de acesso `private`.

A distinção entre interface pública e implementação é chamada **encapsulamento**.

Encapsulamento

Separa os aspectos externos de um objeto dos detalhes internos de implementação do objeto;

Evita que pequenas mudanças possam ter grandes efeitos colaterais

Permite que a implementação possa ser modificada sem afetar as aplicações que usam o objeto

- Ex: melhorias de desempenho, correção de erros, alteração no processo, etc.

Encapsulamento

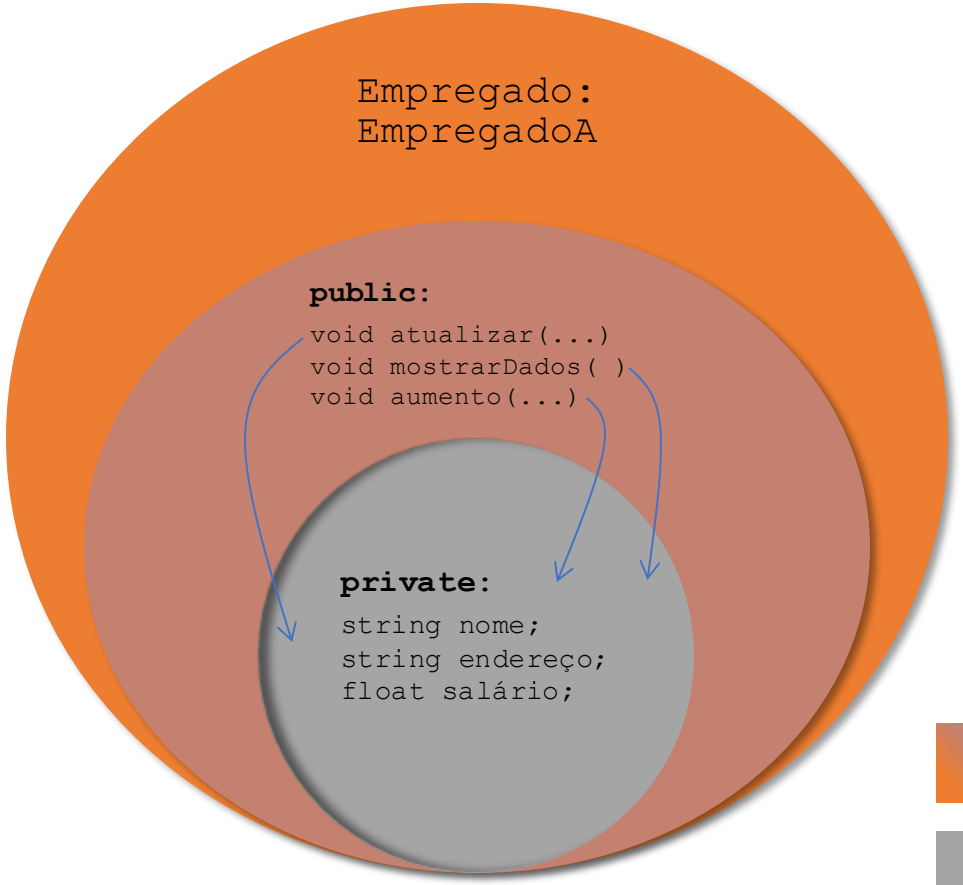
Em geral, numa classe:

- Atributos fazem parte da implementação – `private`
- Métodos fazem parte da interface pública – `public`

Objetivo:

- Atributos manipulados apenas através de seus métodos
- Métodos implementam as regras que mantém os dados íntegros

Encapsulamento

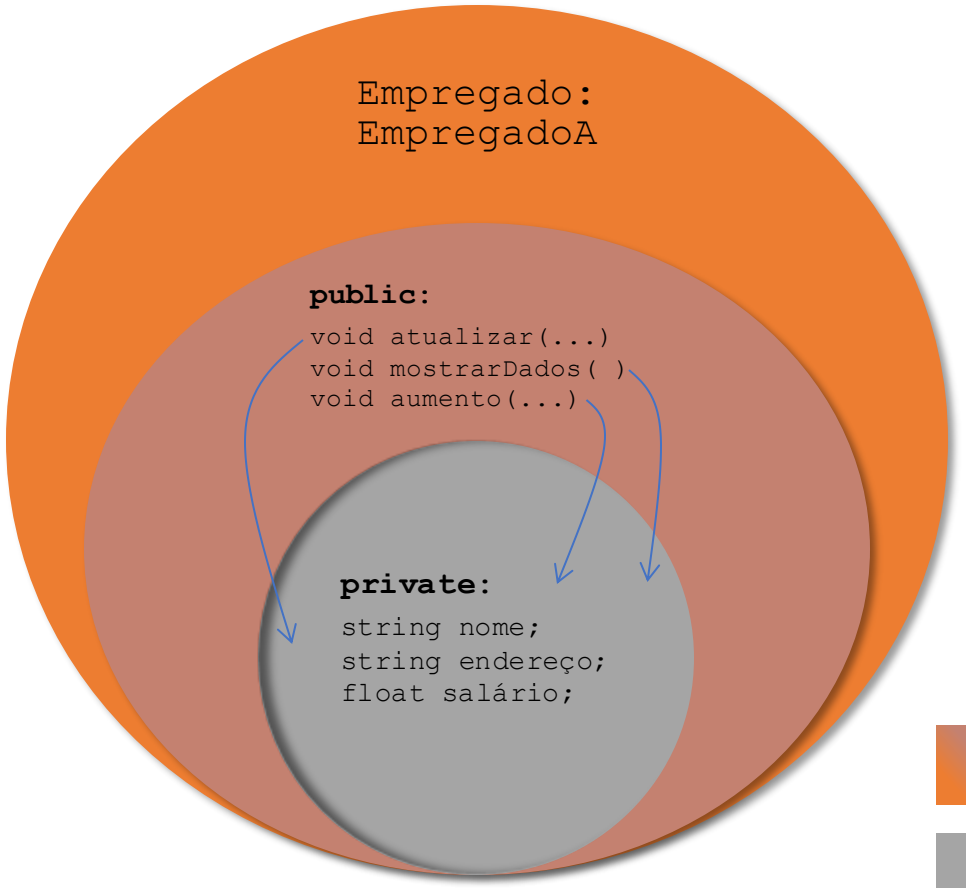


Empregado
nome:String endereço:String salário:float
void atualizar(nome, end, sal) void mostrarDados( ) void aumento(float pct)

Interface Pública

Parte privada

Encapsulamento




```
E1.nome = "João";
E1.endereco = "Rua 2";
E1.salario = 2500.00;
```



```
E1.atualiza("João","Rua 2",2500.00);
```

Interface Pública

Parte privada

Encapsulamento

- Em geral, métodos de uma classe são dos seguintes tipos:
  - Seletores (`get`)
  - Modificadores (`set`)
  - Construtores

Métodos Seletores

- Permitem obter os valores guardados nas variáveis de instância
- Geralmente possuem argumento vazio e limitam-se a devolver o valor de um atributo
- Habitualmente designados pela palavra **get** seguida do nome do atributo

```
class Triangulo{
private:
    float base;
    float altura;
public:
    //Construtores
    Triangulo() = default;
    //Triangulo() : base(1.0), altura(1.0) {};
    Triangulo(float b, float a) : base(b), altura(a){};
    //Métodos seletores
    float getBase(){return base;}
    float getAltura(){return altura;}
    //Métodos modificadores
    void setBase(float b){base = b;}
    void setAltura(float a){altura = a;}
};
```

Métodos Modificadores

- Permitem alterar os valores das variáveis de instância
- Geralmente têm como argumento o novo valor a atribuir
- Não devolvem qualquer valor
- Habitualmente designados pela palavra **set** seguido do nome do atributo

```
class Triangulo{
private:
    float base;
    float altura;
public:
    //Construtores
    Triangulo() = default;
    //Triangulo() : base(1.0), altura(1.0) {};
    Triangulo(float b, float a) : base(b), altura(a){};
    //Métodos seletores
    float getBase(){return base;}
    float getAltura(){return altura;}
    //Métodos modificadores
    void setBase(float b){base = b;}
    void setAltura(float a){altura = a;}
};
```

Construtores

- Evocados quando se pretende criar uma nova instância da classe
- Possuem declaração especial:
  - Mesmo nome da classe
  - Não fazem menção a valores de retorno
- Podem exigir parâmetros, usados para inicialização
- Classes podem possuir mais de um construtor

```
class Triangulo{
private:
    float base;
    float altura;
public:
    //Construtores
    Triangulo() = default;
    //Triangulo() : base(1.0), altura(1.0) {};
    Triangulo(float b, float a) : base(b), altura(a){};
    //Métodos seletores
    float getBase(){return base;}
    float getAltura(){return altura;}
    //Métodos modificadores
    void setBase(float b){base = b;}
    void setAltura(float a){altura = a;}
};
```

Destruidores

- Função membro de uma classe que deleta um objeto
- Automaticamente invocados quando um objeto é destruído
  - Quando o escopo de utilização é encerrado (função, programa, etc).
- Não possui argumentos nem retorno, nem mesmo void
- Definidos por padrão nas classes quando não declarados
- Necessário quando classe manipula dinamicamente a memória

```
class Arquivo {
    private:
        std::string nomeArquivo;
        std::ofstream arquivo;


    public:
        // Construtor
        Arquivo(const std::string &nome) : nomeArquivo(nome) {
        // Abre o arquivo no modo de escrita
        arquivo.open(nomeArquivo, std::ios::out);
        if (!arquivo.is_open()) {
            std::cerr << "Erro ao abrir o arquivo: " << nomeArquivo << std::endl;
        } else {
            std::cout << "Arquivo " << nomeArquivo << " aberto com sucesso."
    << ;
        }
    }

        // Destrutor
        ~Arquivo() {
        if (arquivo.is_open()) {
            arquivo.close();
            std::cout << "Arquivo " << nomeArquivo << " fechado com sucesso."
    <<
        }
    }
};
```

Exercício de Exemplo

Conta
- numero:int - nome:String - saldo:float = 0.0
+ Conta(numero:int, nome:String) + Conta(numero:int, nome:String,saldo:float) + getNum( ):int + getNome( ):String + getSaldo( ):float + depositar(qtde:float):float + sacar(qtde:float):float + imprime( ):void