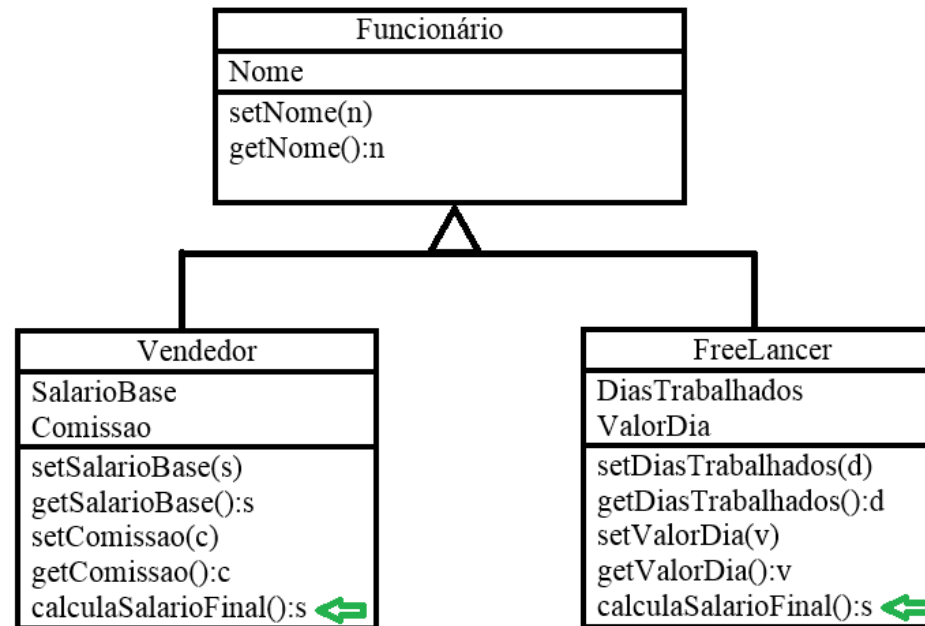


# Classes e objetos - Polimorfismo

Roberto Rocha

# Programação Orientada a Objetos

# Polimorfismo



O cálculo do salario final deverá ser feito da seguinte forma:

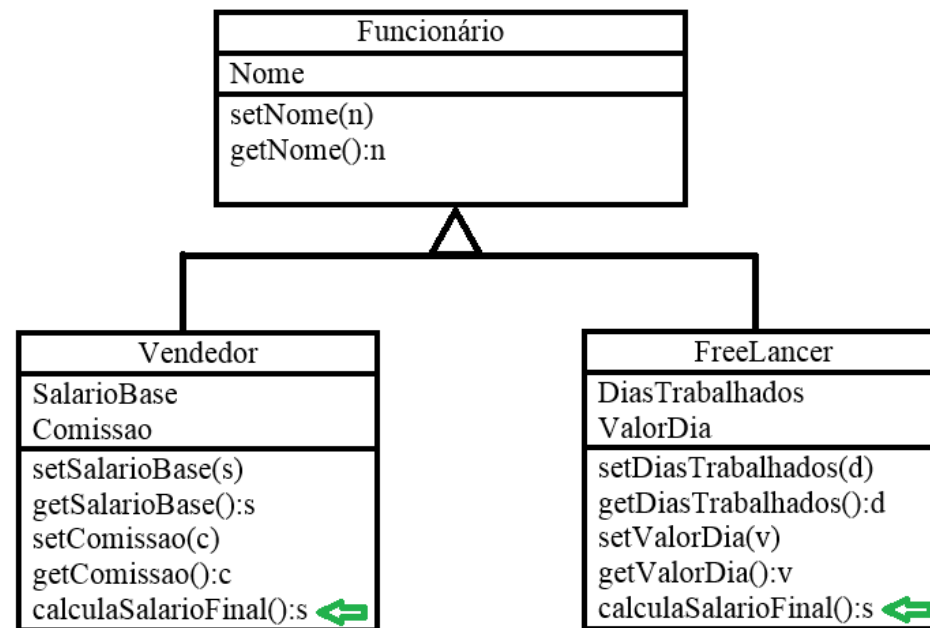
**Vendedor:** soma de SalarioBase + Comissão

**FreeLancer:** multiplica-se a quantidade de diastralbhados pelo valorDia

# Polimorfismo

A palavra **polimorfismo** quer dizer múltiplas formas.

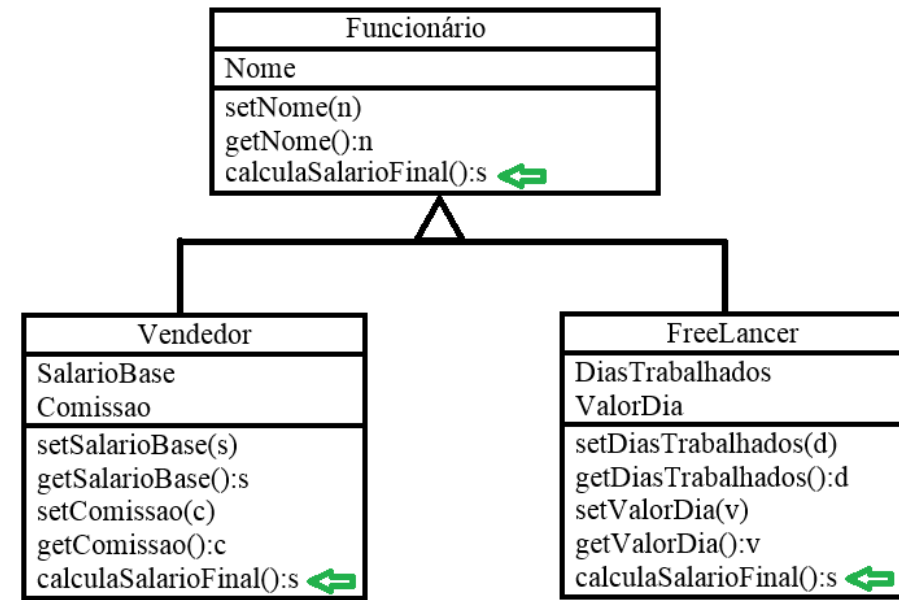
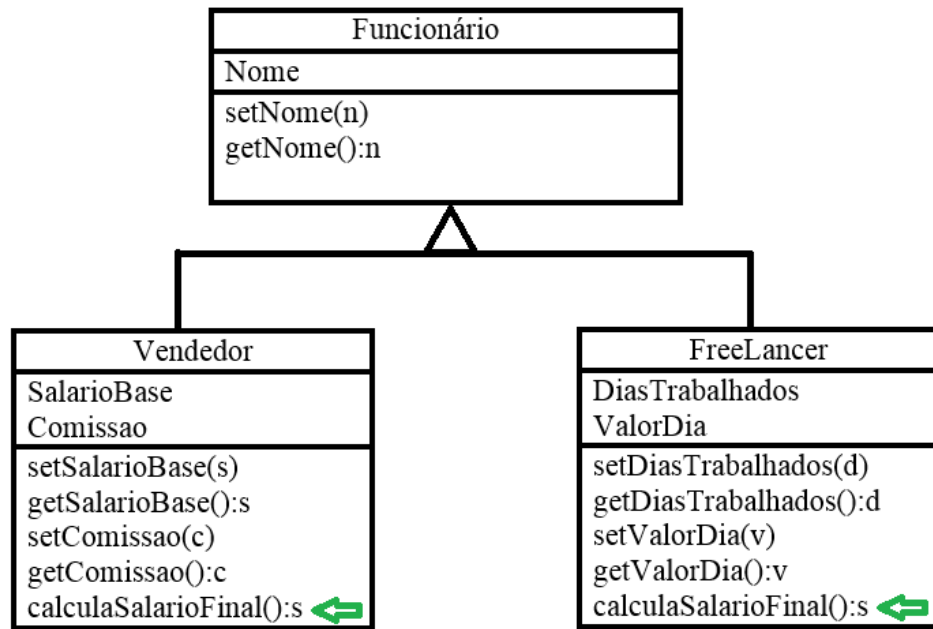
Na **Programação Orientada a Objetos**, polimorfismo se apresenta de diferentes maneiras.



# Polimorfismo

A palavra **polimorfismo** quer dizer múltiplas formas.

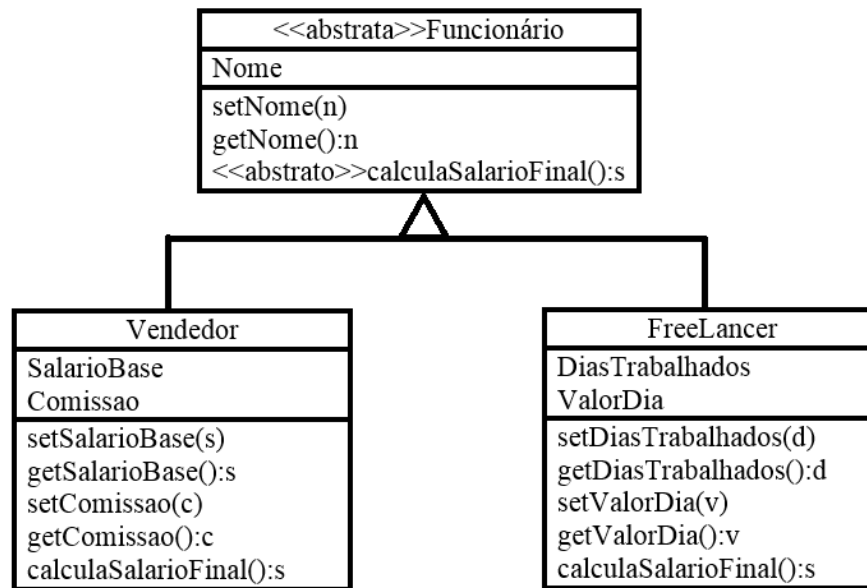
Na **Programação Orientada a Objetos**, polimorfismo se apresenta de diferentes maneiras.



# Polimorfismo

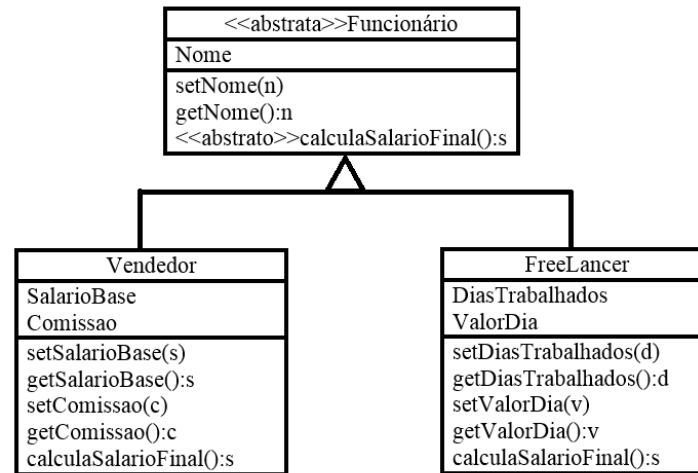
A palavra **polimorfismo** quer dizer múltiplas formas.

Na **Programação Orientada a Objetos**, polimorfismo se apresenta de diferentes maneiras.



**Polimorfismo por herança** permite que uma classe, em um nível mais genérico (denominada **classe mãe**), indique a necessidade de executar determinada operação.

# Polimorfismo

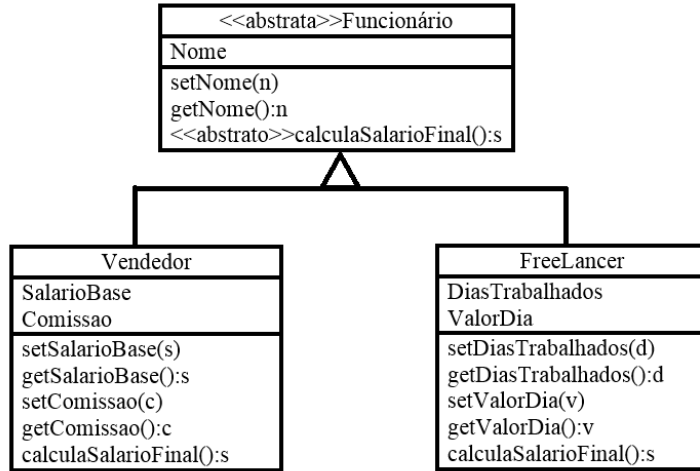


A **classe mãe** não tem conhecimento (ou dados) suficientes para realizar a ação. É chamada **classe abstrata**, não podendo ser instanciada, porém serve de **molde** para as **filhas** que venham a ser criadas

As **classes filhas** ficarão responsáveis por implementar a realização da ação.

As **classes filhas** são chamadas **concretas**, porque implementam o que é solicitado pela **classe mãe**.

# Polimorfismo



A classe funcionário é considerada **abstrata**, pois possui o método `calculaSalarioFinal` que deve ser implementado dependendo do tipo de funcionário.

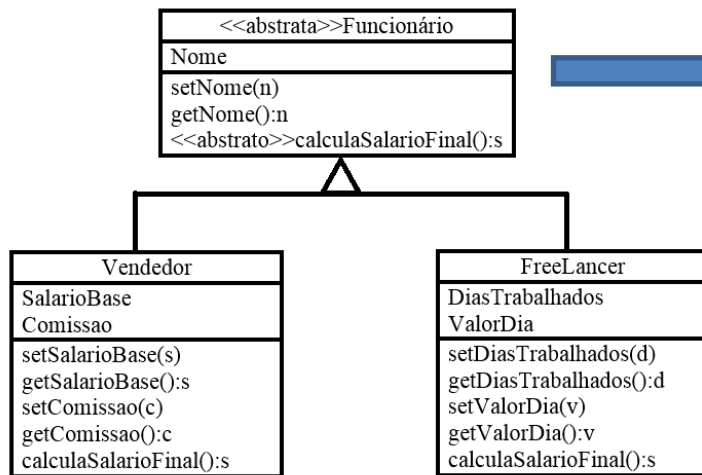
A classe **Funcionário** apenas indica que essa ação é necessária, deixando que suas subclasses resolvam de fato o problema.

As **classes filhas Vendedor e Freelancer** possuem conhecimento suficiente para realizar a ação de calcular o salário final.

A forma do cálculo muda de acordo com a especialização do funcionário. Isso quer dizer que a ação de calcular o salário final é **polimórfica**.



# Polimorfismo em C++

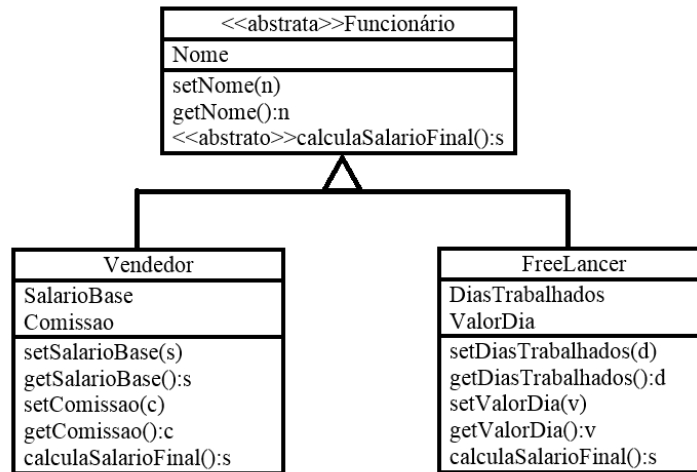


```
class Vendedor: public Funcionario
{
private:
    float salarioBase;
    float comissao;
public:
    Vendedor();
    float getSalarioBase();
    void setSalarioBase( float s);
    float getComissao();
    void setComissao( float c);
    float calculaSalarioFinal();
};
```

```
class FreeLancer: public Funcionario
{
private:
    int diasTrabalhados;
    float valorDia;
public:
    FreeLancer();
    int getDiasTrabalhados();
    void setDiasTrabalhados( int d);
    float getValorDia();
    void setValorDia( float v);
    float calculaSalarioFinal();
};
```

```
#include <iostream>
#include <string>
using namespace std;
class Funcionario
{
private:
    string nome;
public:
    Funcionario();
    string getNome();
    void setNome( string n);
    virtual float calculaSalarioFinal()
    {
        cout << "\ncalculo do salario ainda nao implementado \n";
        return 0;
    }
};
```

# Polimorfismo em C++

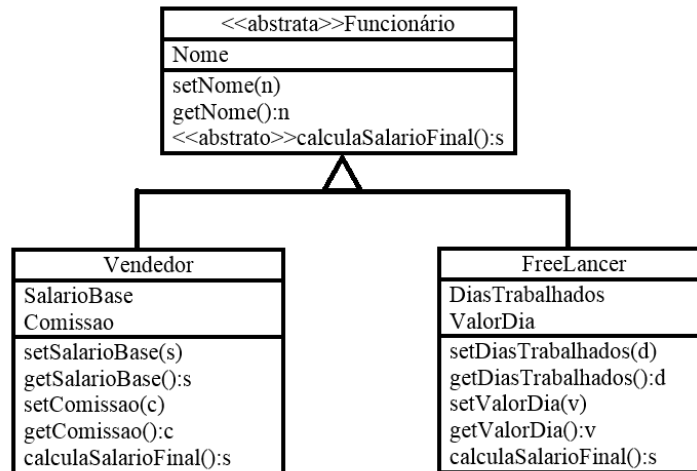


```
#include <iostream>
#include <string>
using namespace std;
class Funcionario
{
private:
    string nome;
public:
    Funcionario();
    string getNome();
    void setNome( string n);
    virtual float calculaSalarioFinal()
    {
        cout << "\ncalculo do salario ainda nao implementado \n";
        return 0;
    }
};
```

A blue arrow points to the `calculaSalarioFinal()` method in the `Funcionario` class, highlighting that this method is currently unimplemented in the base class.

```
Funcionario::Funcionario()
{
    cout << "\nCriando o objeto da classe Funcionario";
}
string Funcionario::getNome()
{
    return nome;
}
void Funcionario::setNome( string n)
{
    nome=n;
}
```

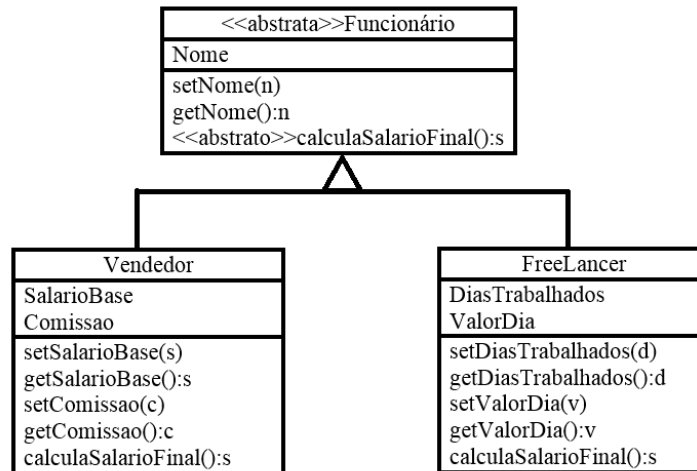
# Polimorfismo em C++



```
class Vendedor: public Funcionario
{
private:
    float salarioBase;
    float comissao;
public:
    Vendedor();
    float getSalarioBase();
    void setSalarioBase( float s);
    float getComissao();
    void setComissao( float c);
    float calculaSalarioFinal();
};
```

```
Vendedor::Vendedor()
{
    cout << "\nCriando o objeto da classe Vendedor \n";
}
float Vendedor::getSalarioBase()
{
    return salarioBase;
}
void Vendedor::setSalarioBase( float s)
{
    salarioBase = s;
}
float Vendedor::getComissao()
{
    return comissao;
}
void Vendedor::setComissao( float c)
{
    comissao = c;
}
float Vendedor::calculaSalarioFinal()
{
    return salarioBase+comissao;
}
```

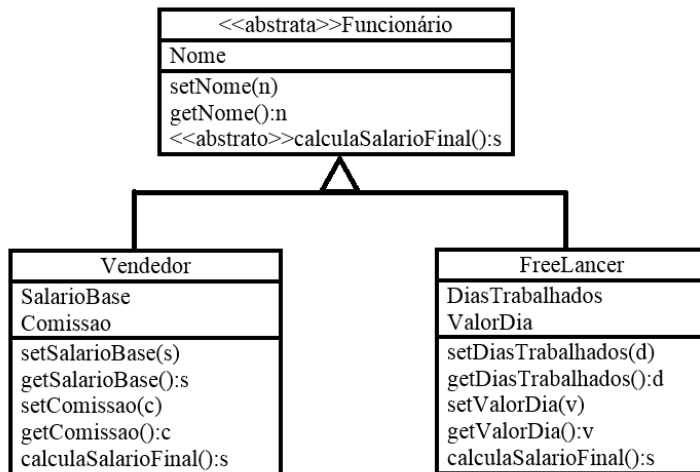
# Polimorfismo em C++



```
class Freelancer: public Funcionario
{
private:
    int diasTrabalhados;
    float valorDia;
public:
    Freelancer();
    int getDiasTrabalhados();
    void setDiasTrabalhados( int d);
    float getValorDia();
    void setValorDia( float v);
    float calculaSalarioFinal();
};
```

```
Freelancer::Freelancer()
{
    cout << "\nCriando o objeto da classe Freelancer \n";
}
int Freelancer::getDiasTrabalhados ()
{
    return diasTrabalhados;
}
void Freelancer::setDiasTrabalhados (int d)
{
    diasTrabalhados = d;
}
float Freelancer::getValorDia()
{
    return valorDia;
}
void Freelancer::setValorDia(float v)
{
    valorDia=v;
}
float Freelancer::calculaSalarioFinal()
{
    return diasTrabalhados * valorDia;
}
```

# Polimorfismo em C++



```
int main()
{
    Vendedor v;
    FreeLancer f;
    v.setNome("Ticio");
    v.setSalarioBase(3000);
    v.setComissao(15000);
    cout << "Salario de " << v.getNome() << " = " << obterSalarioDoFuncionario(&v) << "\n\n";
    f.setNome("Mevio");
    f.setDiasTrabalhados(20);
    f.setValorDia(1000);
    cout << "Salario de " << f.getNome() << " = " << obterSalarioDoFuncionario(&f) << "\n\n";
}
```

```
float obterSalarioDoFuncionario(Funcionario *funcionario)
{
    return funcionario->calculaSalarioFinal();
}
```

```
Criando o objeto da classe Funcionario
Criando o objeto da classe Vendedor

Criando o objeto da classe Funcionario
Criando o objeto da classe FreeLancer
Salario de Ticio = 18000
Salario de Mevio = 20000
```

# Polimorfismo em C++

```
float obterSalarioDoFuncionario(Funcionario *funcionario)
{
    return funcionario->calculaSalarioFinal();
}
```

A função obterSalarioDoFuncionario retorna um valor real (float) e recebe como parâmetro um ponteiro para a classe **Funcionario**.

```
int main()
{
    Vendedor v;
    Freelancer f;
    v.setNome("Ticio");
    v.setSalarioBase(3000);
    v.setComissao(15000);
    cout << "Salario de " << v.getNome() << " = " << obterSalarioDoFuncionario(&v) << "\n\n";
    f.setNome("Mevio");
    f.setDiasTrabalhados(20);
    f.setValorDia(1000);
    cout << "Salario de " << f.getNome() << " = " << obterSalarioDoFuncionario(&f) << "\n\n";
}
```

Na chamada da função obterSalarioDoFuncionario estamos passando o endereço da variável que segue o modelo ora de **Vendedor** (obterSalarioDoFuncionario(&v)) ora de **FreeLancer** (obterSalarioDoFuncionario(&f)) e não o de funcionários!

Lembre-se **Vendedor** e **FreeLancer** são descendentes de **Funcionario**

# Polimorfismo em C++

```
float obterSalarioDoFuncionario(Funcionario *funcionario)
{
    return funcionario->calculaSalarioFinal();
}
```

À primeira vista, pode-se pensar que isso poderia gerar algum tipo de erro (em tempo de compilação ou em tempo de execução).

O motivo de tudo funcionar corretamente é justamente o parâmetro esperado pelo método!

O parâmetro é um **ponteiro** para o tipo **Funcionário** e não uma variável do tipo Funcionário ( que nem poderia ser instanciada!!).

Assim, na chamada desse método, pode-se utilizar **qualquer tipo** de variável que esteja dentro do conjunto de **classes filhas de Funcionário**.

Uma vez que uma **classe filha** pode ser considerada um **tipo especial da classe mãe**. (Vendedor é um Funcionário e FreeLancer também é um Funcionário)

# Polimorfismo em C++

```
float obterSalarioDoFuncionario(Funcionario *funcionario)
{
    return funcionario->calculaSalarioFinal();
}
```

Quando a função for chamada a aplicação não precisa se preocupar com qual das versões do método **calculaSalarioFinal()** será executada, se a implementada na classe **Vendedor** ou a implementada na classe **FreeLancer**.

A decisão será tomada pelo tipo de referência passada!

É importante observar que, com essa estratégia, **não** há necessidade de uso de **estruturas condicionais** para descobrir qual o tipo de **funcionário** em questão para poder invocar a versão correta do método.

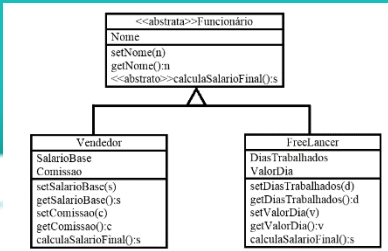
**Isso será feito no momento da execução!!!**

Isso facilita a **manutenção do código**.

No **polimorfismo de herança** o método é o mesmo, **calculaSalarioFinal()**, porém, dependendo do objeto que o acione, a implementação assume diferentes formas.



# Polimorfismo em C++ usando ponteiros para criar objetos



```
#include <iostream>
#include <string>
using namespace std;
class Funcionario
{
private:
    string nome;
public:
    Funcionario();
    string getNome();
    void setNome( string n);
    virtual float calculaSalarioFinal()
    {
        cout << " \ncalculo do salario ainda nao implementado \n";
        return 0;
    }
};

Funcionario::Funcionario() {
    cout << " \nCriando o objeto da classe Funcionario";
}

string Funcionario::getNome() {
    return nome;
}

void Funcionario::setNome( string n) {
    nome=n;
}
```

```
class Vendedor: public Funcionario
{
private:
    float salarioBase;
    float comissao;
public:
    Vendedor();
    float getSalarioBase();
    void setSalarioBase( float s);
    float getComissao();
    void setComissao( float c);
    float calculaSalarioFinal();
};
```

```
Vendedor::Vendedor() {
    cout << " \nCriando o objeto da classe Vendedor \n";
}

float Vendedor::getSalarioBase() {
    return salarioBase;
}

void Vendedor::setSalarioBase( float s) {
    salarioBase = s;
}

float Vendedor::getComissao() {
    return comissao;
}

void Vendedor::setComissao( float c) {
    comissao = c;
}

float Vendedor::calculaSalarioFinal() {
    return salarioBase+comissao;
}
```

```
class Freelancer: public Funcionario
{
private:
    int diasTrabalhados;
    float valorDia;
public:
    Freelancer();
    int getDiasTrabalhados();
    void setDiasTrabalhados( int d);
    float getValorDia();
    void setValorDia( float v);
    float calculaSalarioFinal();
};
```

```
Freelancer::Freelancer() {
    cout << " \nCriando o objeto da classe Freelancer \n";
}

int Freelancer::getDiasTrabalhados () {
    return diasTrabalhados;
}

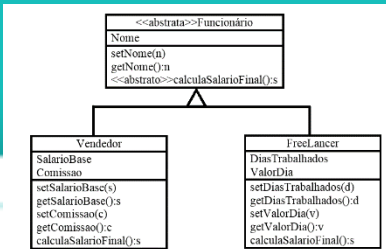
void Freelancer::setDiasTrabalhados ( int d) {
    diasTrabalhados = d;
}

float Freelancer::getValorDia() {
    return valorDia;
}

void Freelancer::setValorDia( float v) {
    valorDia=v;
}

float Freelancer::calculaSalarioFinal() {
    return diasTrabalhados * valorDia;
}
```

# Polimorfismo em C++ usando ponteiros para criar objetos



```
#include <iostream>
#include <string>
using namespace std;
class Funcionario
{
private:
    string nome;
public:
    Funcionario();
    string getNome();
    void setNome( string n);
    virtual float calculaSalarioFinal()
    {
        cout << " \ncalculo do salario ainda nao implementado \n";
        return 0;
    }
};

float obterSalarioDoFuncionario(Funcionario *funcionario)
{
    return funcionario->calculaSalarioFinal();
}
```

```
class Vendedor: public Funcionario
{
private:
    float salarioBase;
    float comissao;
public:
    Vendedor();
    float getSalarioBase();
    void setSalarioBase( float s);
    float getComissao();
    void setComissao( float c);
    float calculaSalarioFinal();
};
```

```
class Freelancer: public Funcionario
{
private:
    int diasTrabalhados;
    float valorDia;
public:
    Freelancer();
    int getDiasTrabalhados();
    void setDiasTrabalhados( int d);
    float getValorDia();
    void setValorDia( float v);
    float calculaSalarioFinal();
};
```

```
int main()
{
    Vendedor *v;
    Freelancer *f;
    v= new (Vendedor);
    v->setNome( "Ticio" );
    v->setSalarioBase(3000);
    v->setComissao(15000);
    cout << "Salario de "<< v->getNome() << " = " << obterSalarioDoFuncionario(v)<<"\n\n";
    f = new (Freelancer);
    f->setNome( "Mevio");
    f->setDiasTrabalhados(20);
    f->setValorDia(1000);
    cout << "Salario de "<< f->getNome() << " = " << obterSalarioDoFuncionario(f)<<"\n\n";
}
```

```
Criando o objeto da classe Funcionario
Criando o objeto da classe Vendedor
Salario de Ticio = 18000
```

```
Criando o objeto da classe Funcionario
Criando o objeto da classe Freelancer
Salario de Mevio = 20000
```

# Polimorfismo em C++ usando ponteiros para criar objetos

```
int main()
{
    Vendedor v;
    FreeLancer f;
    v.setNome("Ticio");
    v.setSalarioBase(3000);
    v.setComissao(15000);
    cout << "Salario de " << v.getNome() << " = " << obterSalarioDoFuncionario(&v) << "\n\n";
    f.setNome("Mevio");
    f.setDiasTrabalhados(20);
    f.setValorDia(1000);
    cout << "Salario de " << f.getNome() << " = " << obterSalarioDoFuncionario(&f) << "\n\n";
}
```

```
int main()
{
    Vendedor *v;
    FreeLancer *f;
    v = new Vendedor;
    v->setNome("Ticio");
    v->setSalarioBase(3000);
    v->setComissao(15000);
    cout << "Salario de " << v->getNome() << " = " << obterSalarioDoFuncionario(v) << "\n\n";
    f = new FreeLancer;
    f->setNome("Mevio");
    f->setDiasTrabalhados(20);
    f->setValorDia(1000);
    cout << "Salario de " << f->getNome() << " = " << obterSalarioDoFuncionario(f) << "\n\n";
}
```

```
Criando o objeto da classe Funcionario
Criando o objeto da classe Vendedor
```

```
Criando o objeto da classe Funcionario
Criando o objeto da classe FreeLancer
Salario de Ticio = 18000
```

```
Salario de Mevio = 20000
```

```
Criando o objeto da classe Funcionario
Criando o objeto da classe Vendedor
Salario de Ticio = 18000
```

```
Criando o objeto da classe Funcionario
Criando o objeto da classe FreeLancer
Salario de Mevio = 20000
```

# Polimorfismo em C++ usando ponteiros para criar objetos

A empresa xyz possui 5 funcionários que são vendedores ou freelancer.

## Funcionário

Nome: Tício

Vendedor

salário-base:  
1.000

Comissão: 5.000

## Funcionário

Nome: Maria

Freelancer

Dias  
trabalhados:15

Valor dia: 500

## Funcionário

Nome: Cássio

Freelancer

Dias  
trabalhados:20

Valor dia: 1000

## Funcionário

Nome: Mévio

Vendedor

salário-base:  
2.000

Comissão: 3.000

## Funcionário

Nome: Florisbela

Vendedor

salário-base:  
4.000

Comissão: 5.000

# Polimorfismo em C++ usando ponteiros para criar objetos

```
int main() {
    Funcionario *empregados[5];
    Vendedor *v;
    FreeLancer *f;
    int i,dT; float sB,c,vD; string nome; char tipo;
    for (i=0;i<5;i++) {
        cout << "Nome do Empregado:"; cin>>nome;
        cout << "Deseja cadastrar (<V>endedor ou <F>reelancer)?";
        tipo=getche();
        if (toupper(tipo)=='V') {
            cout << "\nSalario Base:"; cin >>sB;
            cout << "Comissão:"; cin >> c;
            v = new (Vendedor);
            v->setNome(nome); v->setSalarioBase(sB); v->setComissao(c);
            empregados[i]=v;
        }
        else {
            cout << "\nDias Trabalhados:"; cin >>dT;
            cout << "Valor do Dia:"; cin >> vD;
            f = new (FreeLancer);
            f->setNome(nome); f->setDiasTrabalhados(dT); f->setValorDia(vD);
            empregados[i]=f;
        }
    }
    // imprimindo o nome do funcionario e seu salario
    cout << "\n****Relação de Empregados ****\n-----\n";
    for (i=0;i<5;i++) {
        cout <<"Salario de "<< empregados[i]->getNome();
        cout << " = " << obterSalarioDoFuncionario(empregados[i]) << "\n";
    }
}
```

Estrutura de dados para  
representar a empresa  
xyz

Vetor de 5 posições de  
**Funcionários!**

empregados				
E1	E2	E3	E4	E5

# Polimorfismo em C++ usando ponteiros para criar objetos

Estrutura de dados para representar a empresa xyz

Vetor de 5 posições de **Funcionários**!

Após o cadastramento

empregados				
E1	E2	E3	E4	E5
Funcionário	Funcionário	Funcionário	Funcionário	Funcionário
Nome: Tício	Nome: Maria	Nome: Cássio	Nome: Mévio	Nome: Florisbela
Vendedor	Freelancer	Freelancer	Vendedor	Vendedor
salário-base: 1.000	Dias trabalhados:15	Dias trabalhados:20	salário-base: 2.000	salário-base: 4.000
Comissão: 5.000	Valor dia: 500	Valor dia: 1000	Comissão: 3.000	Comissão: 5.000

```
****Relatõ de Empregados ***
-----
Salario de Ticio = 6000
Salario de Maria = 7500
Salario de Cassio = 20000
Salario de Mevio = 5000
Salario de Florisbela = 9000
```

# Polimorfismo em C++ usando ponteiros para criar objetos

```
int main() {
    Funcionario *empregados[5];
    Vendedor *v;
    FreeLancer *f;
    int i,dT; float sB,c,vD; string nome; char tipo;
    for (i=0;i<5;i++) {
        cout << "Nome do Empregado:"; cin>>nome;
        cout << "Deseja cadastrar (<V>endedor ou <F>reelancer)?";
        tipo=getche();
        if (toupper(tipo)=='V') {
            cout << "\nSalario Base:"; cin >>sB;
            cout << "Comissão:"; cin >> c;
            v = new Vendedor();
            v->setNome(nome); v->setSalarioBase(sB); v->setComissao(c);
            empregados[i]=v;
        }
        else {
            cout << "\nDias Trabalhados:"; cin >>dT;
            cout << "Valor do Dia:"; cin >> vD;
            f = new FreeLancer();
            f->setNome(nome); f->setDiasTrabalhados(dT); f->setValorDia(vD);
            empregados[i]=f;
        }
    }
    // imprimindo o nome do funcionario e seu salario
    cout << "\n****Relação de Empregados ****\n";
    for (i=0;i<5;i++) {
        cout << "Salario de " << empregados[i]->getNome();
        cout << " = " << obterSalarioDoFuncionario(empregados[i]) << "\n";
    }
}
```

```
Nome do Empregado:Ticio
Deseja cadastrar <<U>endedor ou <F>reelancer)?v
Salario Base:1000
Comissão:5000

Criando o objeto da classe Funcionario
Criando o objeto da classe Vendedor
Nome do Empregado:Maria
Deseja cadastrar <<U>endedor ou <F>reelancer)?f
Dias Trabalhados:15
Valor do Dia:500

Criando o objeto da classe Funcionario
Criando o objeto da classe FreeLancer
Nome do Empregado:Cassio
Deseja cadastrar <<U>endedor ou <F>reelancer)?f
Dias Trabalhados:20
Valor do Dia:1000

Criando o objeto da classe Funcionario
Criando o objeto da classe FreeLancer
Nome do Empregado:Mevio
Deseja cadastrar <<U>endedor ou <F>reelancer)?v
Salario Base:2000
Comissão:3000

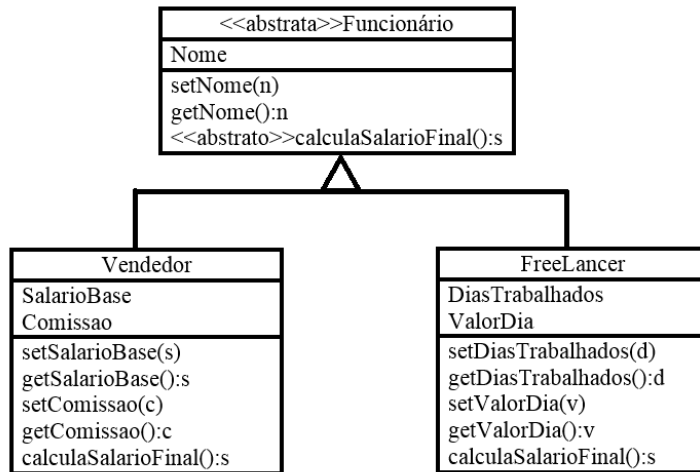
Criando o objeto da classe Funcionario
Criando o objeto da classe Vendedor
Nome do Empregado:Florishela
Deseja cadastrar <<U>endedor ou <F>reelancer)?v
Salario Base:4000
Comissão:5000

Criando o objeto da classe Funcionario
Criando o objeto da classe Vendedor

****Relação de Empregados ****
-----
Salario de Ticio = 6000
Salario de Maria = 7500
Salario de Cassio = 20000
Salario de Mevio = 5000
Salario de Florishela = 9000
```



# Polimorfismo em C++



## Exercício:

Criar os diagramas, atributos e métodos para as classes

Veiculo, Carro e Caminhão

Crie uma função utilizando polimorfismo calculando o frete para carros (valor bandeirada + km rodados x valor base) e para caminhão( km rodados x nr pneus x valor base)





**PUC Minas**  
**Virtual**