Herança

BCC702-Programação de Computadores II

Emiliana Mara Lopes Simões
simoes.eml@gmail.com

Universidade Federal de Ouro Preto
abril 2010

Herança

- Uma característica importante da programação orientada a objetos é permitir a **criação de novas classes com base em uma classe já existente**
- O objetivo é proporcionar o **reuso de software** que:
  - Economiza tempo durante o desenvolvimento do programa;
  - Aumenta a probabilidade de um sistema ser eficientemente implementado;
  - Facilita a modificação do código;
  - Facilita a correção de erros.

Herança

- A herança é uma forma de reutilização de software em que o programador cria uma classe que *"**absorve**"* os dados e comportamentos de uma classe existente e os aprimora com novas capacidades

- Nomenclatura:

  - **Classe básica** (**superclasse**) classe já existente;

  - **Classe derivada** (**subclasse**) classe criada a partir da classe básica.

Exemplo: Hierarquia de Herança de **Pessoas** em uma Loja

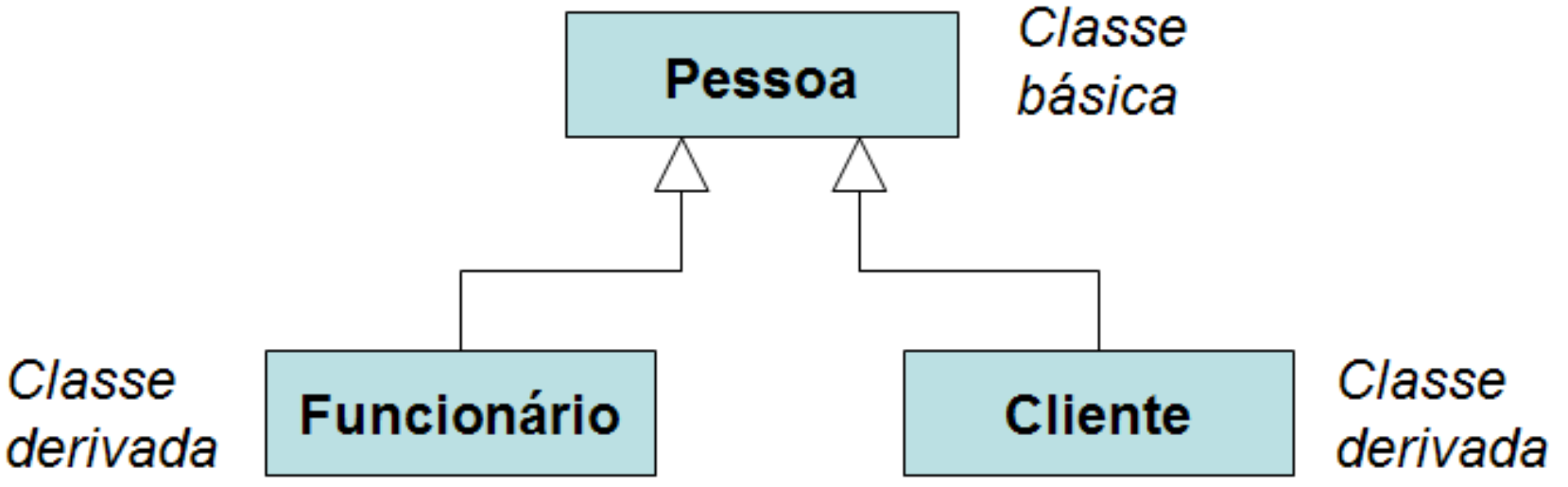
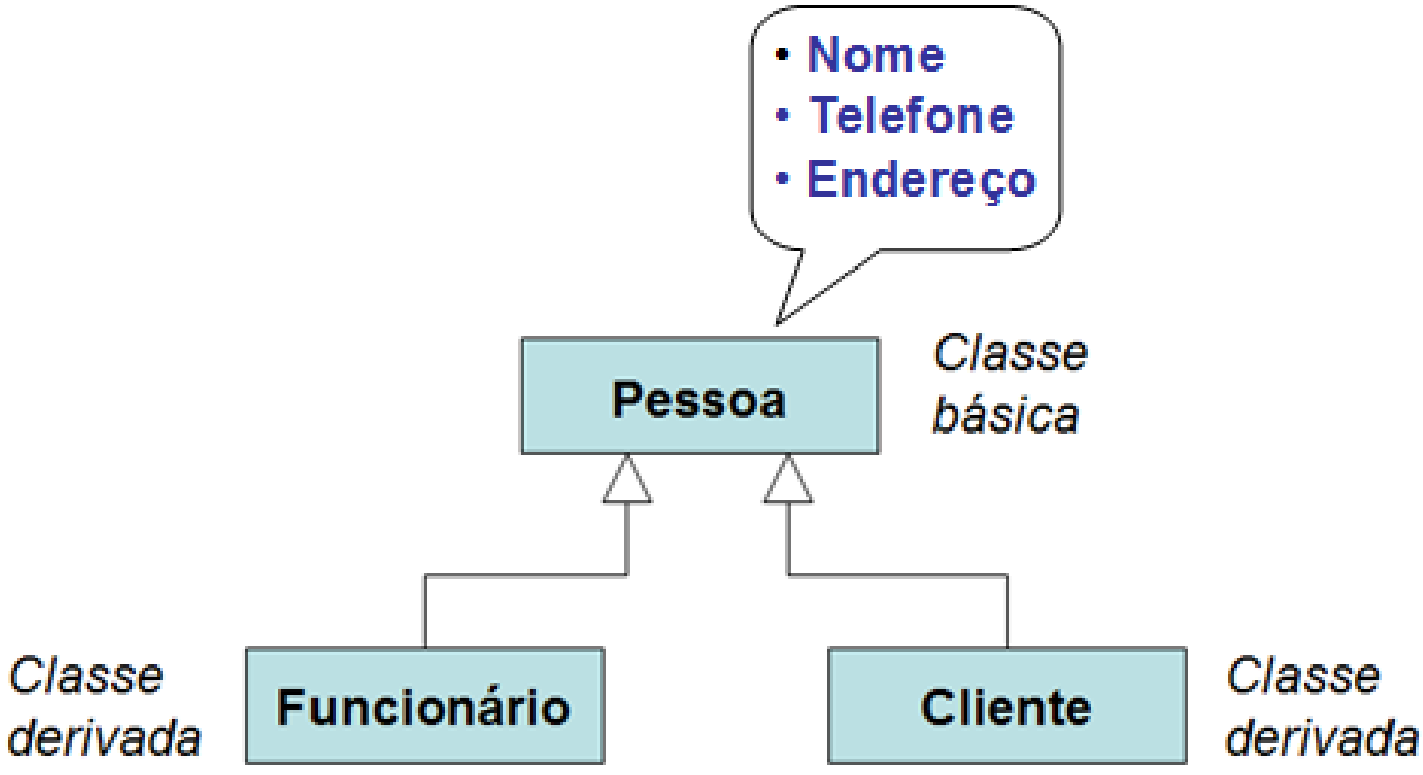
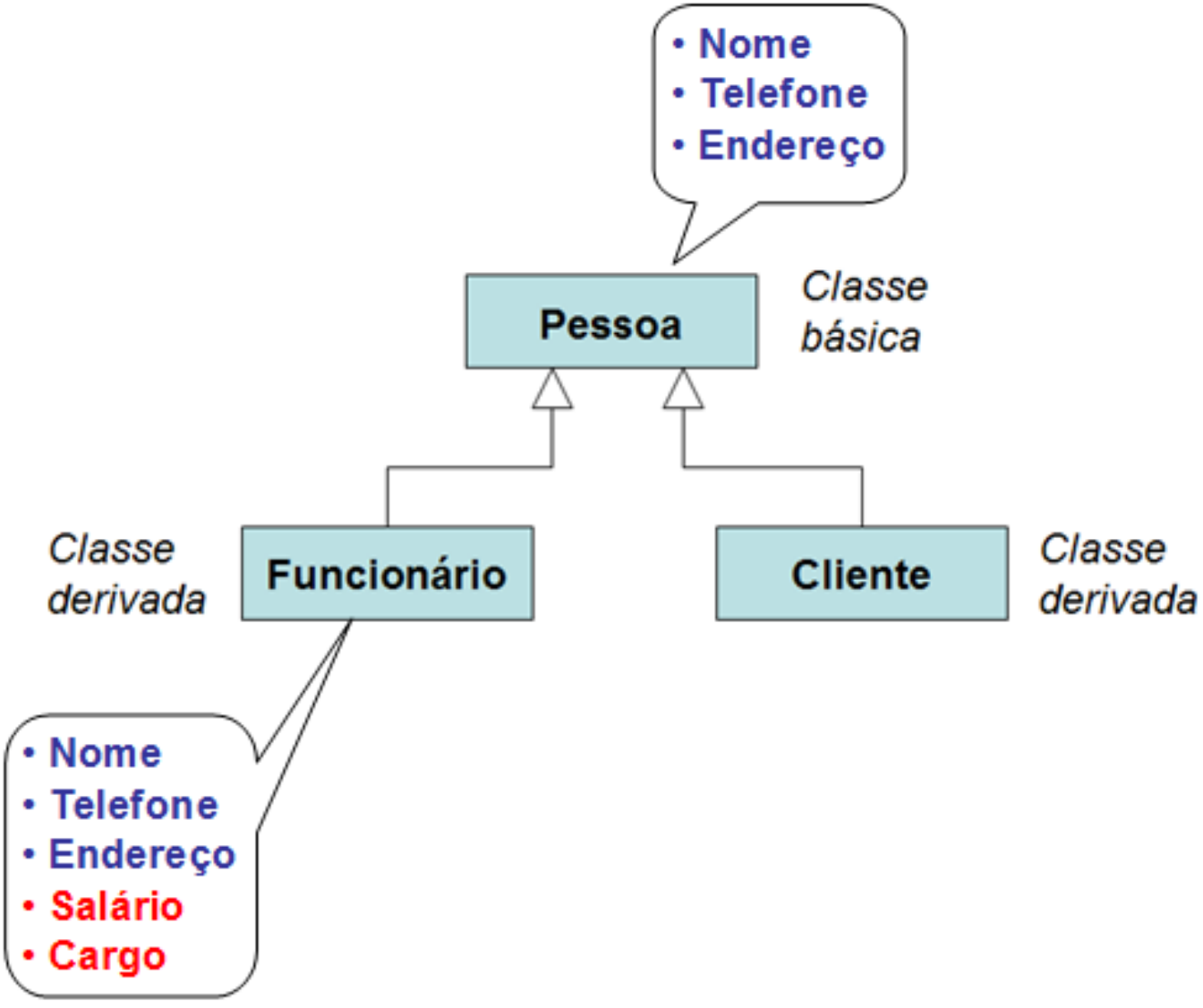


Diagrama de classe **UML (*Unified Modeling Language* - Linguagem de Modelagem Unificada)**

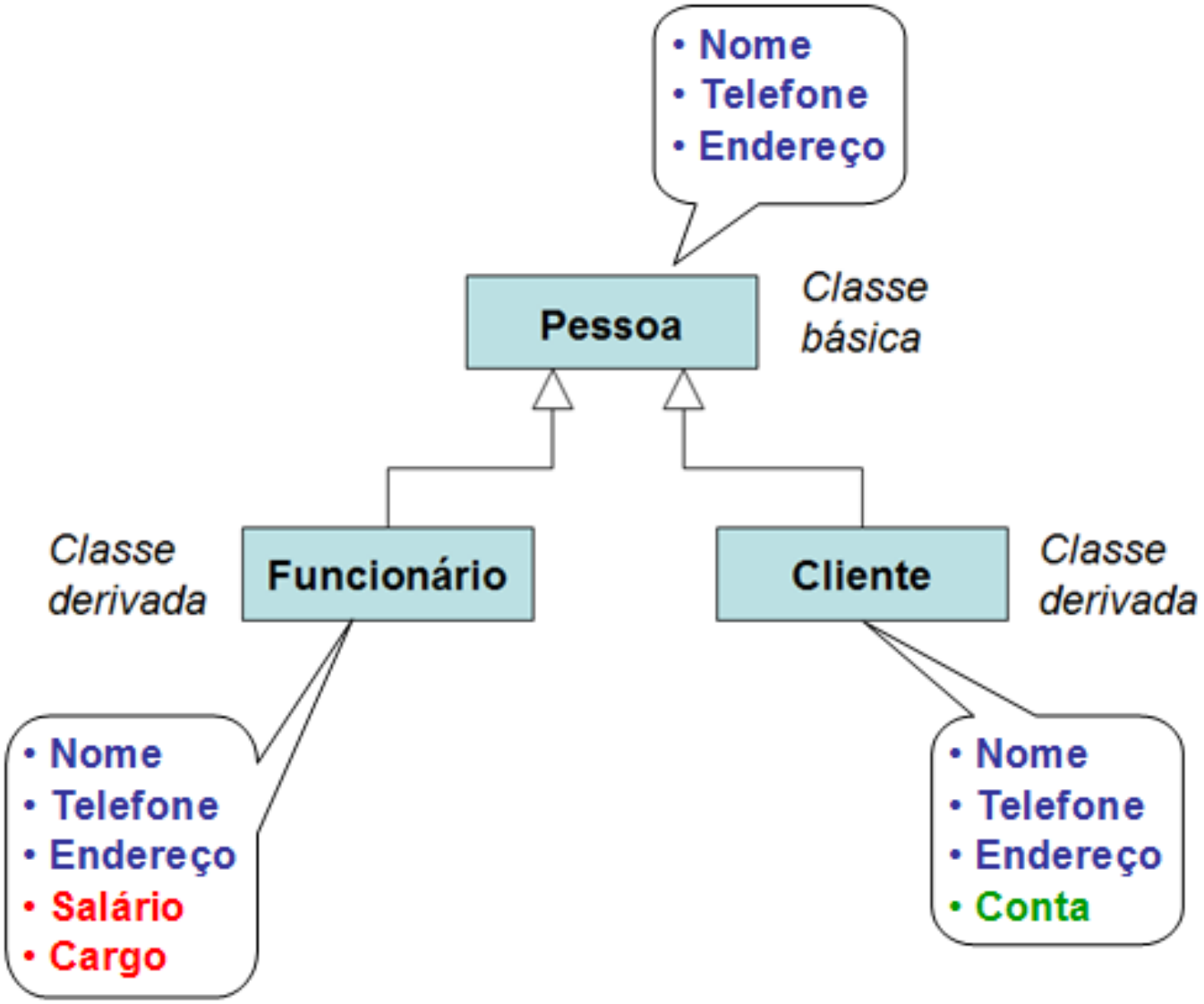
Exemplo: Hierarquia de Herança de **Pessoas** em uma Loja



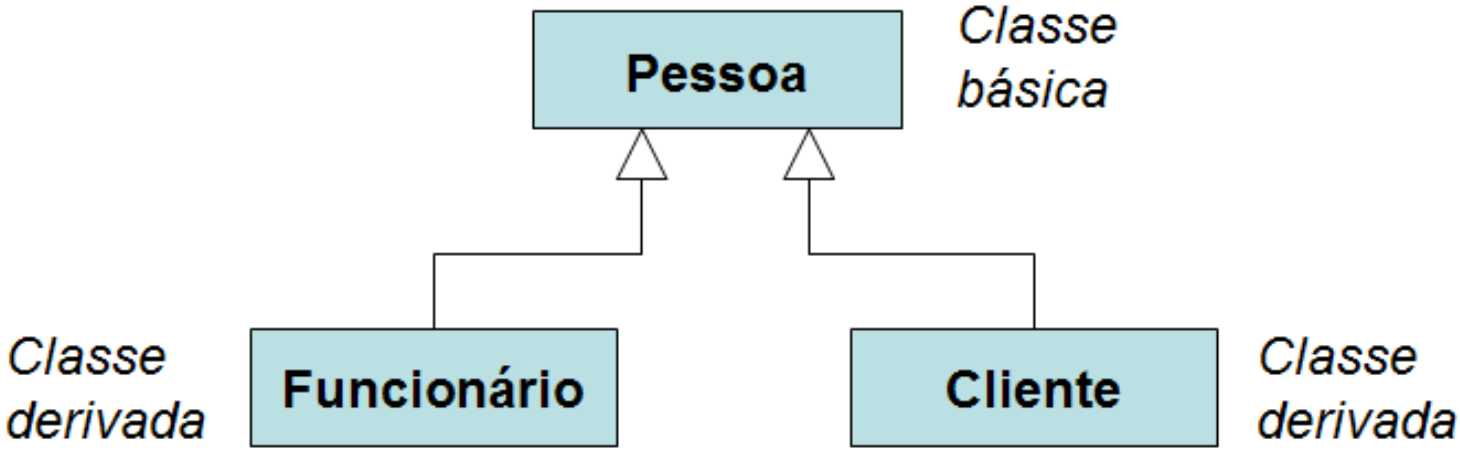
Exemplo: Hierarquia de Herança de **Pessoas** em uma Loja



Exemplo: Hierarquia de Herança de **Pessoas** em uma Loja



Exemplo: Hierarquia de Herança de **Pessoas** em uma Loja



- Relacionamento **é um**, um objeto de uma **classe derivada** também pode ser tratado como um objeto da **classe básica**.
- Assim, *"um Funcionário é uma Pessoa"* e *"um Cliente é uma Pessoa"*

Exemplo de Implementação de Herança

- Os professores de uma universidade dividem-se em 2 categorias:

Exemplo de Implementação de Herança

- Os professores de uma universidade dividem-se em 2 categorias:
  - Professores em dedicação exclusiva (DE);

Exemplo de Implementação de Herança

- Os professores de uma universidade dividem-se em 2 categorias:
  - Professores em dedicação exclusiva (DE);
  - Professores horistas.

Exemplo de Implementação de Herança

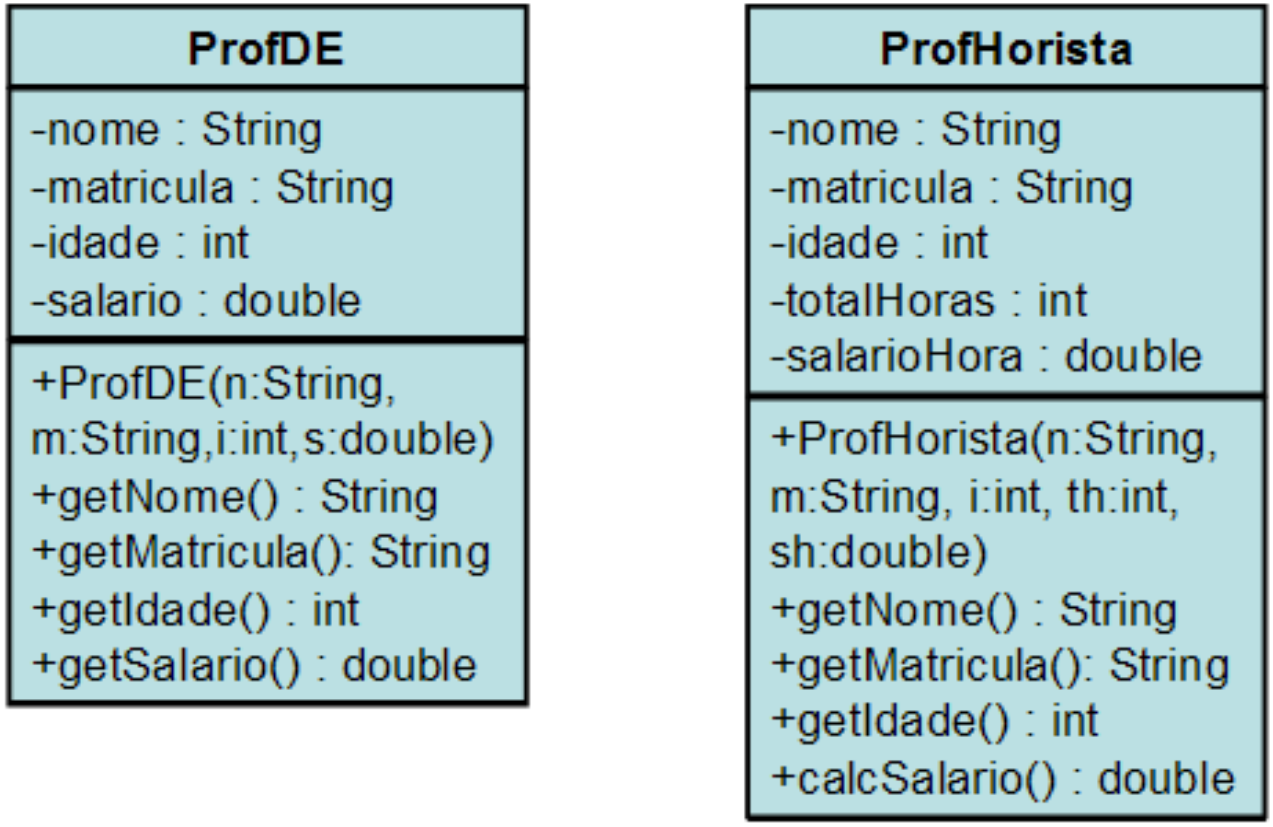
- Os professores de uma universidade dividem-se em 2 categorias:
  - Professores em dedicação exclusiva (DE);
  - Professores horistas.
- Professores em dedicação exclusiva possuem um salário fixo para 40 horas de atividade semanais;

Exemplo de Implementação de Herança

- Os professores de uma universidade dividem-se em 2 categorias:
  - Professores em dedicação exclusiva (DE);
  - Professores horistas.
- Professores em dedicação exclusiva possuem um salário fixo para 40 horas de atividade semanais;
- Professores horistas recebem um valor estipulado por hora.

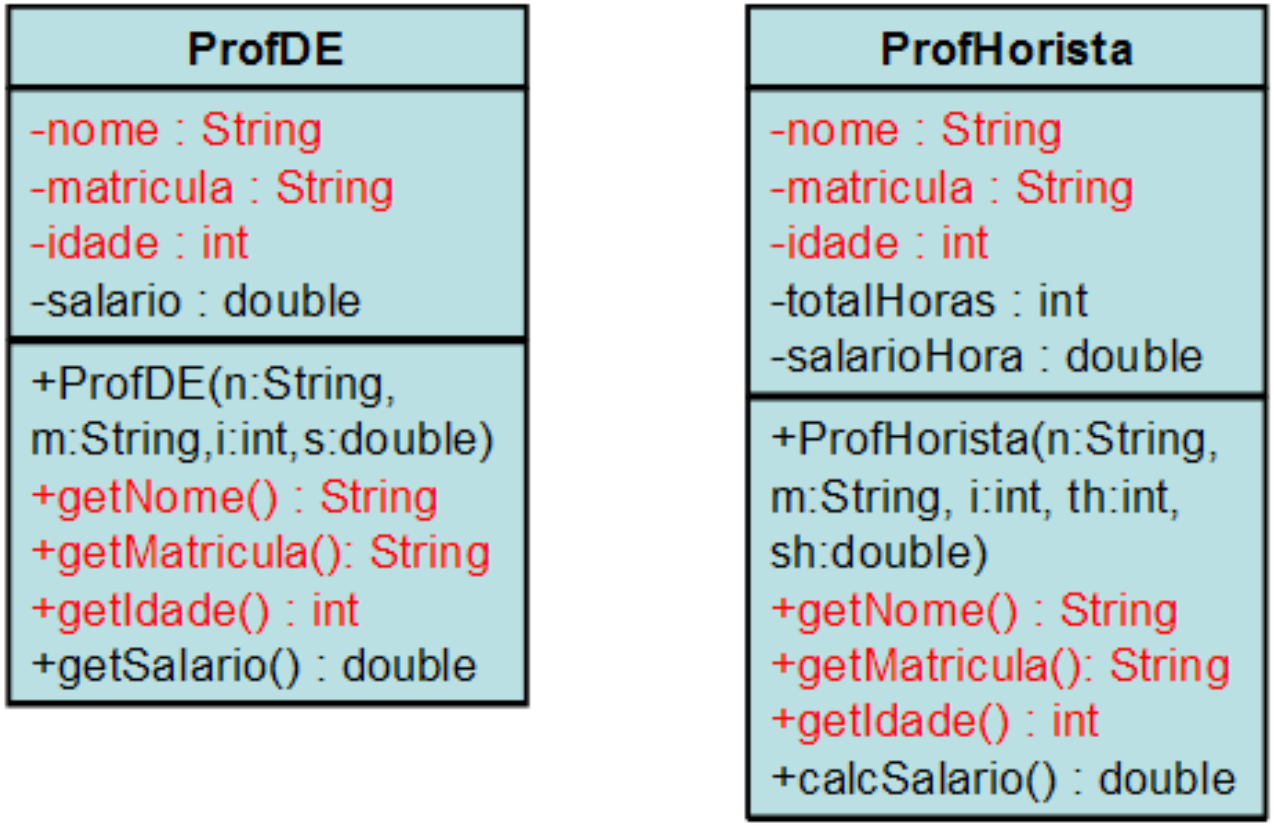
Exemplo de Implementação de Herança

- O problema pode ser resolvido através da seguinte modelagem de classes:



Exemplo de Implementação de Herança

- O problema pode ser resolvido através da seguinte modelagem de classes (**Observe as semelhanças!!!**):



Exemplo de Implementação de Herança

- Analisando esta primeira solução observa-se que:

Exemplo de Implementação de Herança

- Analisando esta primeira solução observa-se que:
  - Alguns atributos e funções são iguais nas classes **ProfDE** e **ProfHorista**

Exemplo de Implementação de Herança

- Analisando esta primeira solução observa-se que:
  - Alguns atributos e funções são iguais nas classes **ProfDE** e **ProfHorista**
  - E se precisarmos mudar a representação de algum atributo? Por exemplo, considerar o número de **matrícula** um **int** ao invés de um **string**?

Exemplo de Implementação de Herança

- Analisando esta primeira solução observa-se que:
  - Alguns atributos e funções são iguais nas classes **ProfDE** e **ProfHorista**
  - E se precisarmos mudar a representação de algum atributo? Por exemplo, considerar o número de **matrícula** um **int** ao invés de um **string**?
  - Será necessário alterar os **construtores** e os métodos ***getMatricula()*** nas duas classes. **RUIM PARA A PROGRAMAÇÃO!!!**

Exemplo de Implementação de Herança

- Analisando esta primeira solução observa-se que:
  - Alguns atributos e funções são iguais nas classes **ProfDE** e **ProfHorista**
  - E se precisarmos mudar a representação de algum atributo? Por exemplo, considerar o número de **matrícula** um **int** ao invés de um **string**?
  - Será necessário alterar os **construtores** e os métodos ***getMatricula()*** nas duas classes. **RUIM PARA A PROGRAMAÇÃO!!!**
  - Motivo: **Código redundante**

Exemplo de Implementação de Herança

- Analisando esta primeira solução observa-se que:
  - Alguns atributos e funções são iguais nas classes **ProfDE** e **ProfHorista**
  - E se precisarmos mudar a representação de algum atributo? Por exemplo, considerar o número de **matrícula** um **int** ao invés de um **string**?
  - Será necessário alterar os **construtores** e os métodos ***getMatricula()*** nas duas classes. **RUIM PARA A PROGRAMAÇÃO!!!**
  - Motivo: **Código redundante**
  - Como resolver? **Herança!**

Exemplo de Implementação de Herança

- Sendo assim:

Exemplo de Implementação de Herança

- Sendo assim:
  - Cria-se uma classe **Professor**, que contém os atributos e funções *comuns aos dois tipos de professores*;

Exemplo de Implementação de Herança

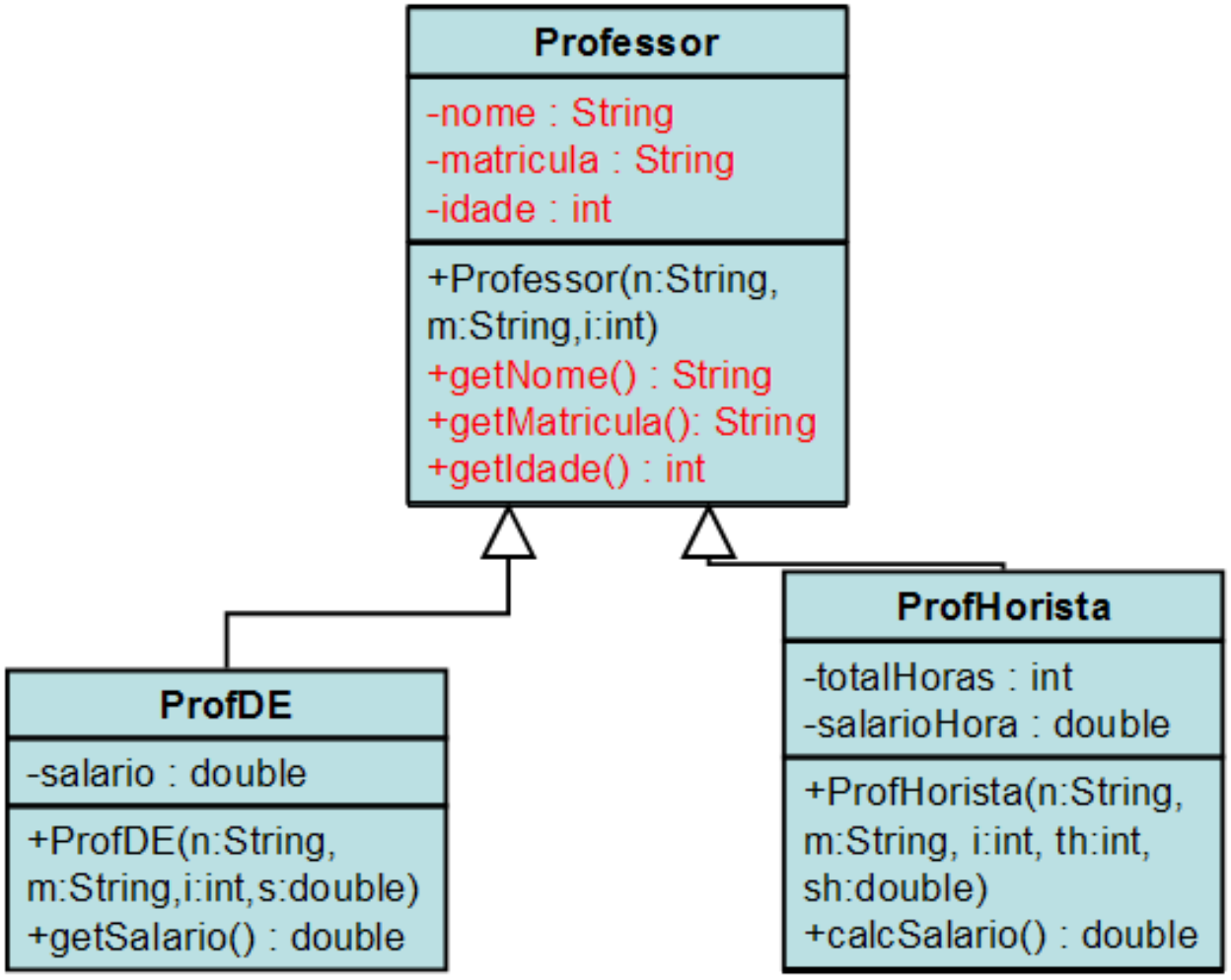
- Sendo assim:
  - Cria-se uma classe **Professor**, que contém os atributos e funções *comuns aos dois tipos de professores*;
  - A partir dela, *cria-se duas novas classes*, que representarão os professores horistas e dedicação exclusiva;

Exemplo de Implementação de Herança

- Sendo assim:
  - Cria-se uma classe **Professor**, que contém os atributos e funções *comuns aos dois tipos de professores*;
  - A partir dela, *cria-se duas novas classes*, que representarão os professores horistas e dedicação exclusiva;
  - Para isso, essas classes deverão **herdar** os atributos e funções declarados pela classe básica, **Professor**

Exemplo de Implementação de Herança

- Solução:



Exemplo de Implementação de Herança

Sintaxe em C++ para definição de uma classe derivada:

```
class nome_classe_derivada : acesso nome_classe_basica{

   //definição da classe

};
```

- *acesso* determina o tipo de herança que pode ser: **public**, **private** ou **protected**. (Serão explicados a seguir!)
- A **herança public** é a mais comumente usada

Exemplo: Definição da classe básica **Professor**

```
#include <string>
using namespace std;

class Professor{

   private:
      string nome;
      string matricula;
      int idade;

   public:
      Professor(string n, string m, int i);
      string getNome();
      string getMatricula();
      int getIdade();

};
```

Arquivo: **Professor.h**

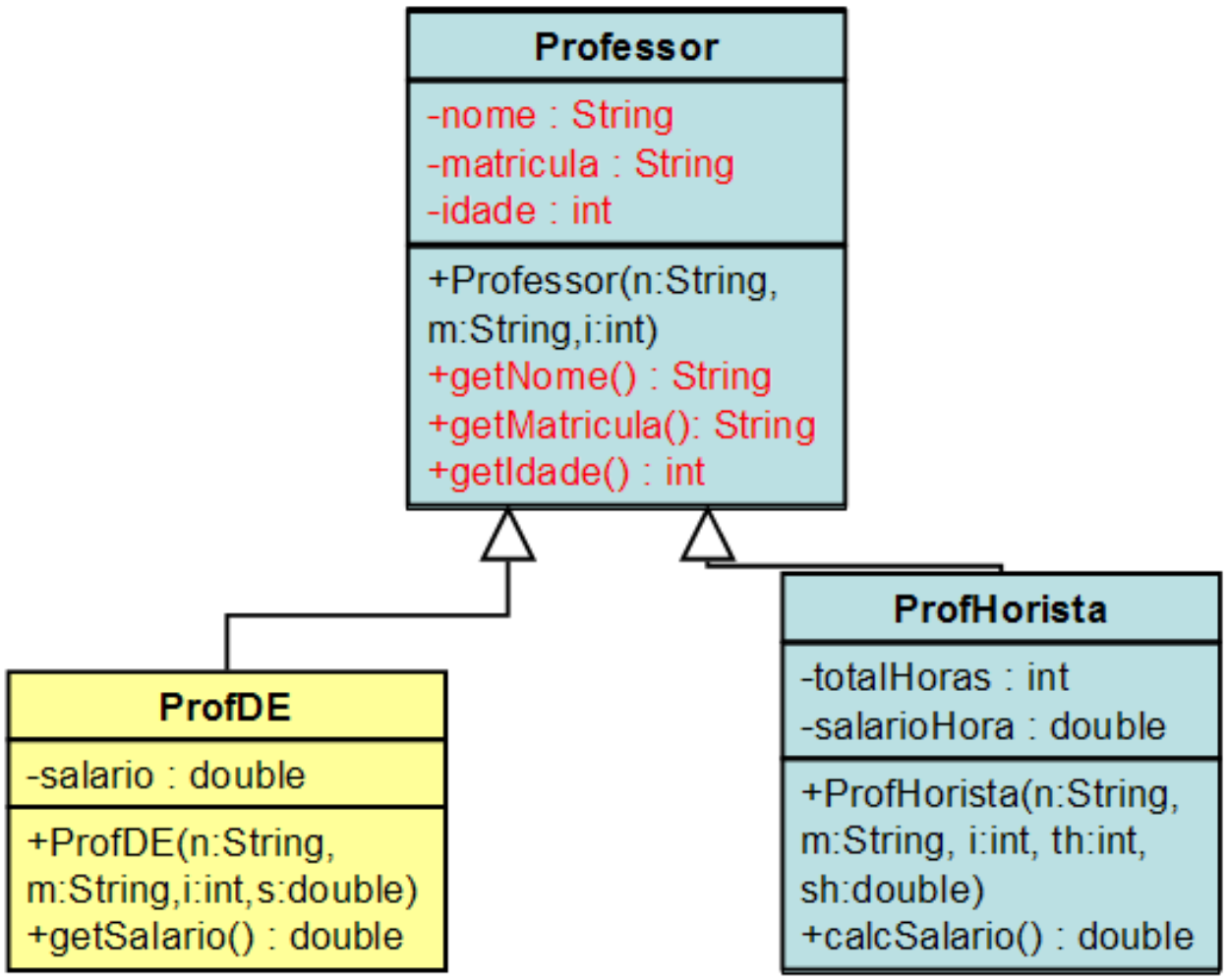
Exemplo: Definição da classe básica **Professor**

```
#include "Professor.h"
Professor::Professor(string n, string m, int i){
    nome = n;
    matricula = m;
    idade = i;
}

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

string Professor::getMatricula(){
    return matricula;
}

int Professor::getIdade(){
    return idade;
}
```

Arquivo: **Professor.cpp**

Exemplo: Definição da classe derivada **ProfDE**

- Solução:



Exemplo: Definição da classe derivada **ProfDE**

```
#include "Professor.h"

class ProfDE : public Professor{

   private:
       double salario;

   public:
      ProfDE(string n, string m, int i, double s);
      double getSalario();

};
```

Arquivo: **ProfDE.h**

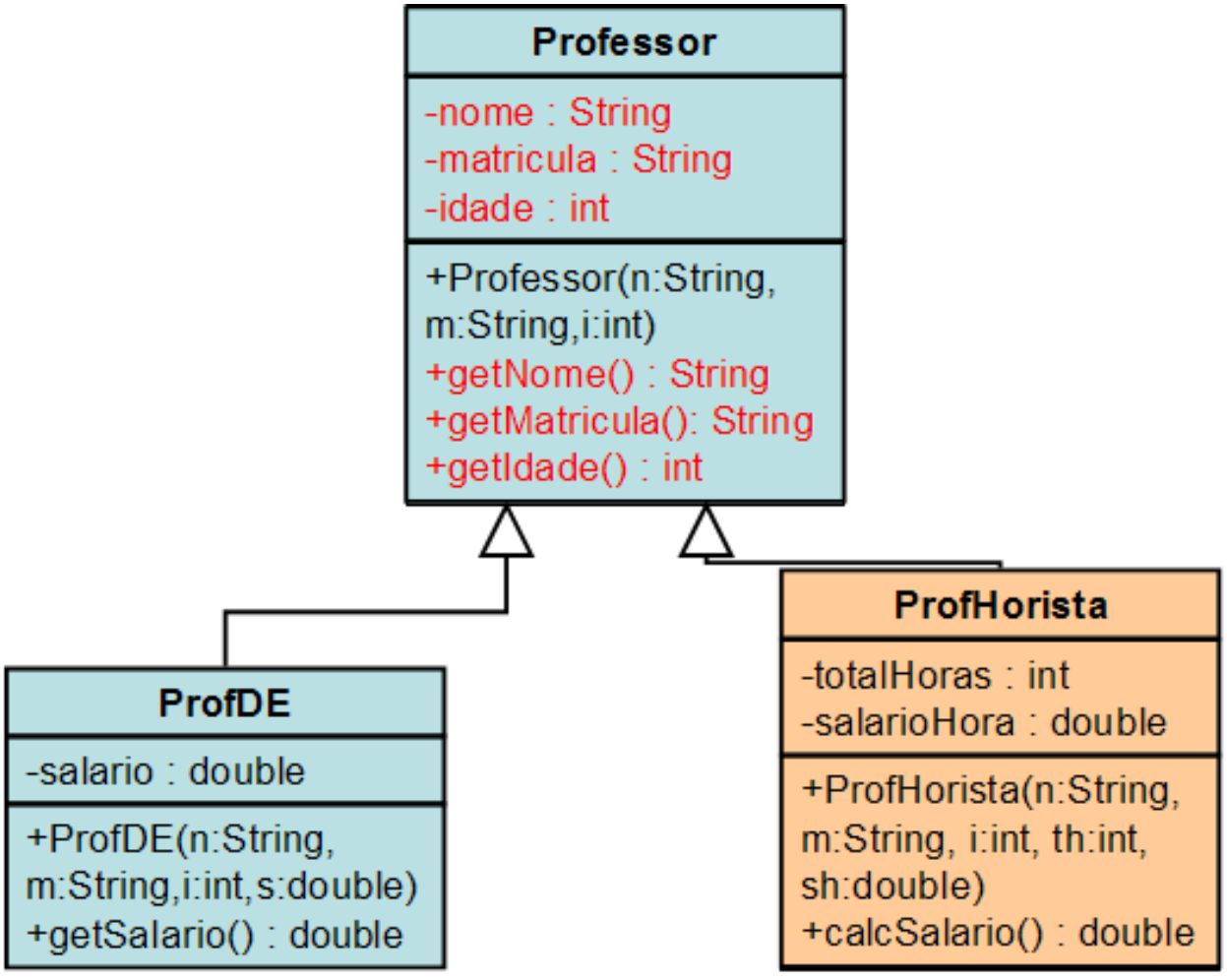
Exemplo: Definição da classe derivada **ProfDE**

```
#include "ProfDE.h"

ProfDE::ProfDE(string n, string m, int i, double s)
        : Professor(n, m, i){

   salario = s;

}

double ProfDE::getSalario(){
   return salario;
}
```

Arquivo: **ProfDE.cpp**

Exemplo: Definição da classe derivada **ProfHorista**

- Solução:



Exemplo: Definição da classe derivada **ProfHorista**

```
#include "Professor.h"

class ProfHorista : public Professor{

   private:
        int totalHoras;
        double salarioHora;

   public:
      ProfHorista(string n, string m, int i, int th, double sh);
      double calcSalario();

};
```

Arquivo: **ProfHorista.h**

Exemplo: Definição da classe derivada **ProfHorista**

```
#include “ProfHorista.h”

ProfHorista::ProfHorista(string n, string m, int i, int th, double sh)
        : Professor(n, m, i){

   totalHoras = th;
   salarioHora = sh;

}

double ProfHorista::calcSalario(){
   return totalHoras * salarioHora;
}
```

Arquivo: **ProfHorista.cpp**

Exemplo de Implementação de Herança: main

```
#include <iostream>
#include "ProfDE.h"
#include "ProfHorista.h"

int main(){

   Professor p1("Joao", "MG-1234", 45);
   cout << "Dados do professor: " << endl;
   cout << p1.getNome() << " " << p1.getMatricula()
        << " " << p1.getIdade() << endl;

   ProfDE pde1("Jose", "MG-3245", 40, 6000);
   cout << "Dados do professor dedicacao exclusiva: "
        << endl;
   cout << pde1.getNome() << " " << pde1.getMatricula()
        << " " << pde1.getIdade()
        << " " << pde1.getSalario()
        <<endl;
```

Exemplo de Implementação de Herança: main

```
    ...

    ProfHorista ph1(“Pedro”, “MG-3245”, 26, 20, 100);
    cout << “Dados do professor horista: ”
         << endl;
    cout << ph1.getNome() << “ ” << ph1.getMatricula()
         << “ ” << ph1.getIdade()
         << “ ” << ph1.getSalario()
         <<endl;

    system(“pause”);

}
```

Exemplo de Implementação de Herança

O EXEMPLO ANTERIOR POSSUI UM ERRO! VEJAMOS PORQUE..

“Empacotador” de pré-processador

- O **“Empacotador” de pré-processador** é utilizado em um arquivo de cabeçalhos (**.h**) para impedir que o código no cabeçalho seja incluído no mesmo arquivo de código-fonte mais de uma vez

- Assim, evita-se erros de **múltiplas definições** de uma mesma classe

- Utiliza as diretivas de pré-processador:
  #ifndef
  #define
  #endif

Exemplo: Modificação do arquivo **Professor.h**

```
#ifndef PROFESSOR_H
#define PROFESSOR_H

#include <string>
using namespace std;

class Professor{

   ...

};

#endif
```

Exemplo: Modificação do arquivo **Professor.h**

- Se o cabeçalho não foi incluído anteriormente em um arquivo, então:

Exemplo: Modificação do arquivo **Professor.h**

- Se o cabeçalho não foi incluído anteriormente em um arquivo, então:

  1. O nome PROFESSOR_H será definido pela diretiva #define

Exemplo: Modificação do arquivo **Professor.h**

- Se o cabeçalho não foi incluído anteriormente em um arquivo, então:

  1. O nome PROFESSOR_H será definido pela diretiva #define
  2. As instruções de arquivo de cabeçalho serão incluídas

Exemplo: Modificação do arquivo **Professor.h**

- Se o cabeçalho não foi incluído anteriormente em um arquivo, então:
  1. O nome PROFESSOR_H será definido pela diretiva #define
  2. As instruções de arquivo de cabeçalho serão incluídas
- Senão:

Exemplo: Modificação do arquivo **Professor.h**

- Se o cabeçalho não foi incluído anteriormente em um arquivo, então:
  1. O nome PROFESSOR_H será definido pela diretiva #define
  2. As instruções de arquivo de cabeçalho serão incluídas
- Senão:
  - PROFESSOR_H já está definido e o arquivo de cabeçalho não será novamente incluído

Resumo da acessibilidade do membro de classe básica em uma classe derivada

Especificador de acesso de membro de classe básica	**Tipos de herança**		
	Herança public	***Herança protected***	***Herança private***
public	**public** na classe derivada	**protected** na classe derivada	**private** na classe derivada
protected	**protected** na classe derivada	**protected** na classe derivada	**private** na classe derivada
private	**Oculta** na classe derivada	**Oculta** na classe derivada	**Oculta** na classe derivada

Observação: Com todas as formas de herança, os membros ***private*** de uma classe básica não são acessíveis diretamente de classes derivadas dessa classe, mas esses membros de classe básica private ainda são herdados.

Dúvidas?

Próxima Aula

Polimorfismo