

# Programação Orientada a Objetos para Redes de Computadores

**Prof. Miguel Elias Mitre Campista**

`http://www.gta.ufrj.br/~miguel`

# PARTE 2

## Programação em C++ - Herança

# Herança

- Criação de uma nova classe de uma classe existente
  - Absorve os **dados** e os **comportamentos** da classe existente
  - **Aprimora** os dados e os comportamentos com novas capacidades

# Herança

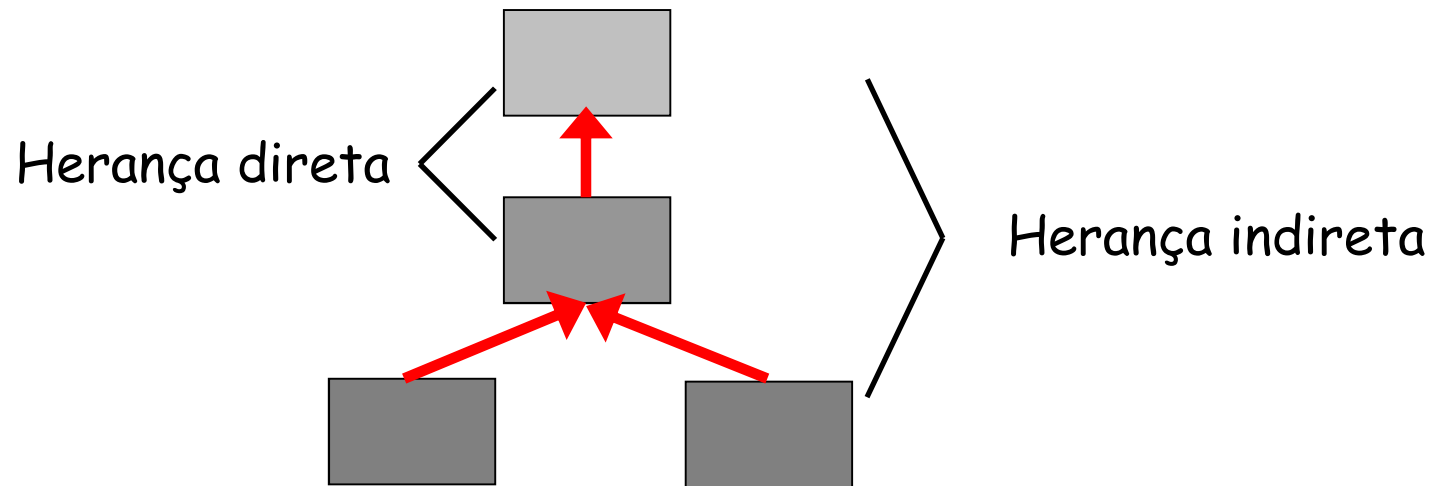
- Classe derivada herda da classe base
  - Grupo mais especializado de objetos
  - Comportamentos herdados da classe base
    - Os quais podem ser personalizados
  - E outros comportamentos
- Reuso de software
  - Facilita implementação e utiliza código já depurado

# Hierarquia de Classes

- Classe original
  - Chamada de classe base
- Classe nova
  - Chamada de classe derivada
- Outras linguagens dão nomes diferentes
  - Ex.: Java chama de superclasse e subclasse

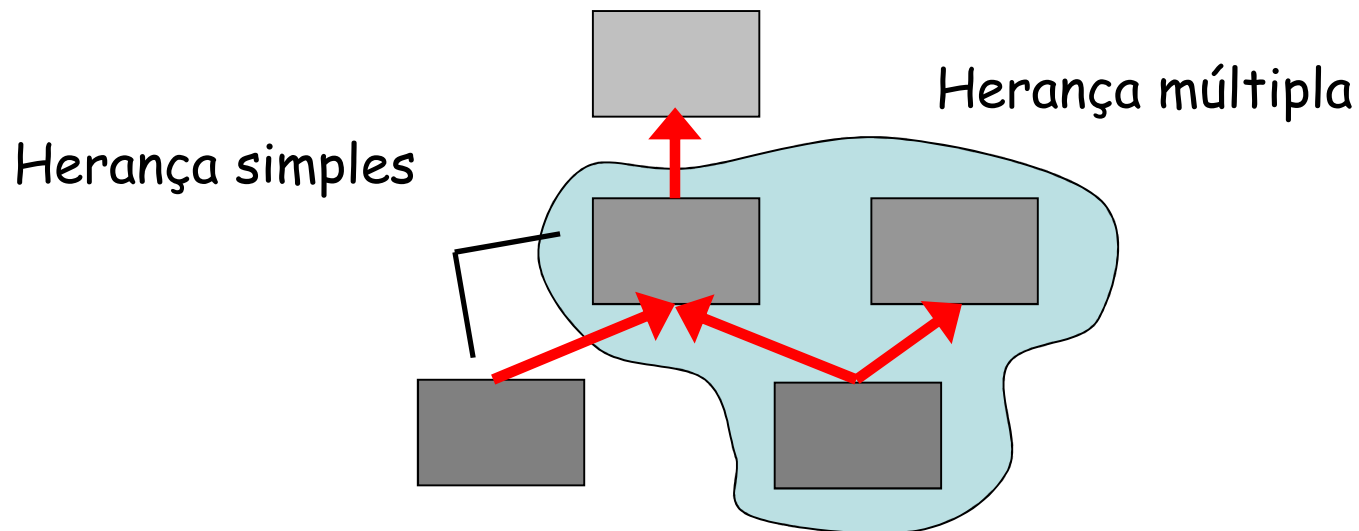
# Hierarquia de Classes

- Classe base direta
  - É herdada explicitamente (de um nível acima da hierarquia)
- Classe base indireta
  - É herdada de dois ou mais níveis da hierarquia



# Hierarquia de Classes

- Herança simples
  - Herda de uma classe base
- Herança múltipla
  - Herda de múltiplas classes base
    - Classes base possivelmente não relacionadas



# Três Tipos de Herança

- **public**

- Todo objeto da classe derivada é também um objeto da classe base
  - Os objetos da classe base não são objetos das classes derivadas
    - Ex.: Todos os carros são veículos, mas nem todos os veículos são carros
- É possível acessar membros não-private da classe base
  - Para acessar membros **private** da classe base
    - A classe derivada deve usar funções-membro não-private herdadas



# Três Tipos de Herança

- **private**
  - Uma alternativa à composição
- **protected**
  - Esse tipo de herança é raramente utilizado

# Abstração

- Os programadores concentram-se em...
  - Aspectos comuns entre objetos no sistema
- Dessa forma, classes base podem ser construídas apenas de maneira mais abstrata possível
  - Os detalhes podem ser deixados para as classes derivadas implementarem

# "É um" versus "Tem um"

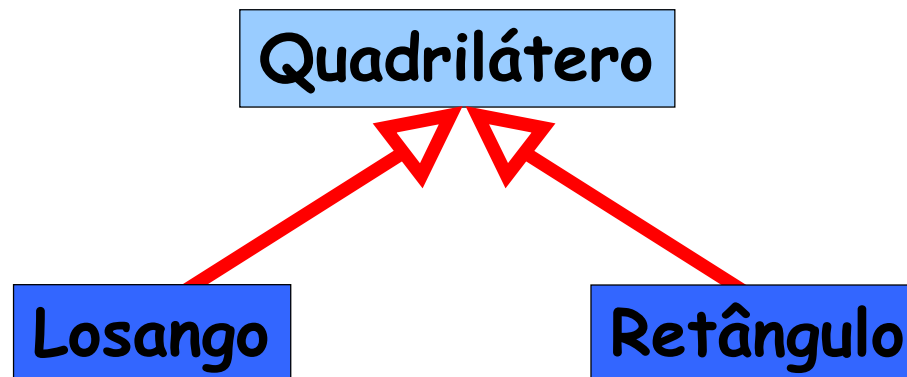
- "É um" → Herança
  - O objeto da classe derivada pode ser tratado como um objeto da classe base
    - Ex.: O carro *é um* veículo
      - » As propriedades/comportamentos de veículos também se aplicam a um carro
- "Tem um" → Composição
  - O objeto contém um ou mais objetos de outras classes como membros
    - Ex.: O carro *tem (uma)* direção

# Recomendações de Engenharia de *Software*

- Funções-membro de uma classe derivada
  - Não podem acessar diretamente os membros `private` da classe base
- Se uma classe derivada pudesse acessar os membros `private` de sua classe base
  - As classes que herdaram dessa classe derivada também poderiam acessar os dados da classe base
    - Isso propagaria acesso a variáveis privadas e os benefícios da ocultação de informações seriam perdidos

# Classes Base e Derivadas

- O objeto de uma classe "é um" objeto de outra classe
  - Ex.: O retângulo é um quadrilátero
    - A classe **Retângulo** herda da classe **Quadrilátero**
      - **Quadrilátero** é a classe base
      - **Retângulo** e **Losango** são as classes derivadas



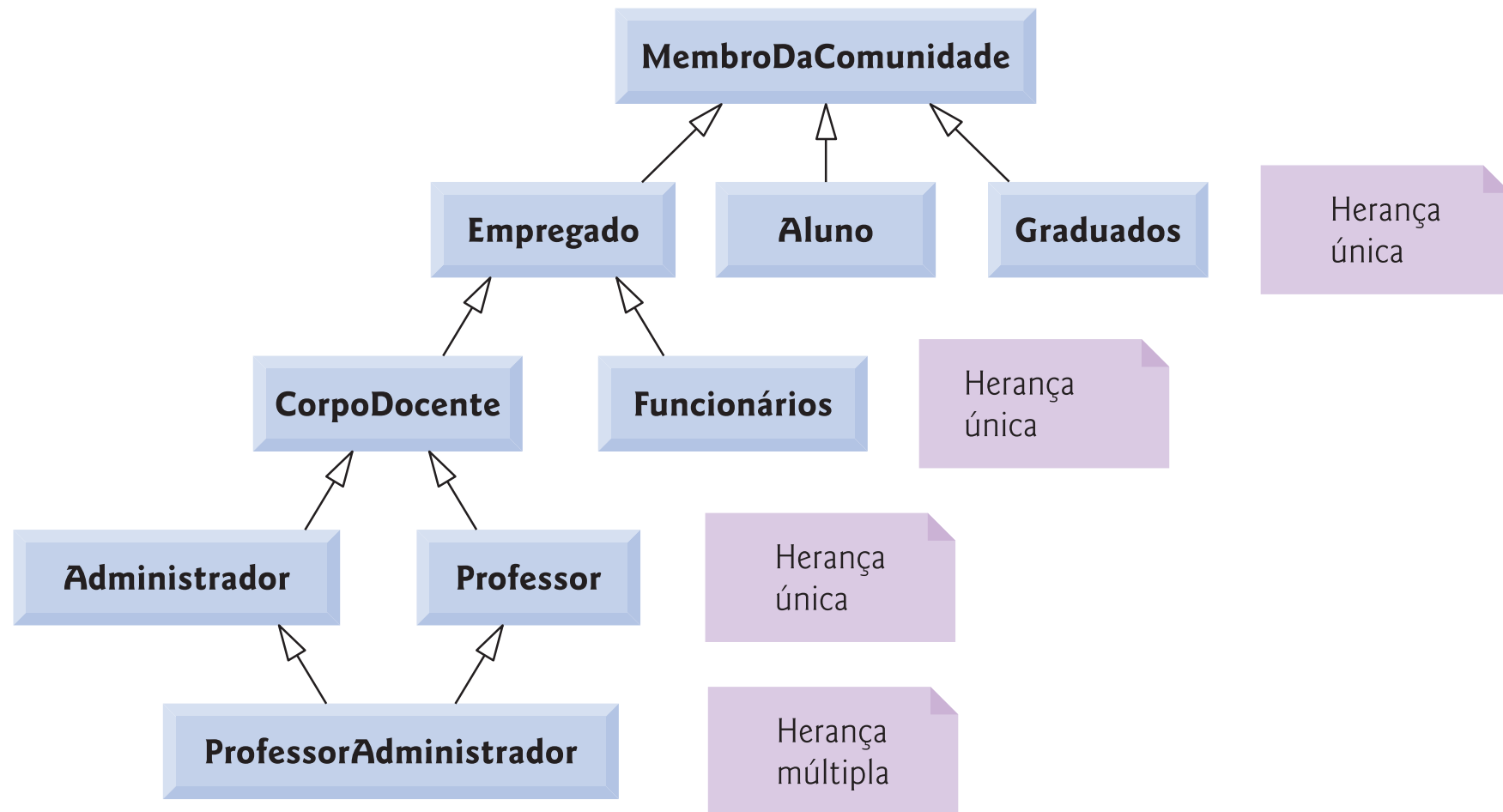
# Classes Base e Derivadas

- A classe base em geral representa um conjunto maior de objetos que as classes derivadas
  - Ex.:
    - Classe base: **Vehicle**
      - Inclui carros, caminhões, barcos, bicicletas etc.
    - Classe derivada: **Car**
      - Um subconjunto menor e mais específico de veículos

# Classes Base e Derivadas

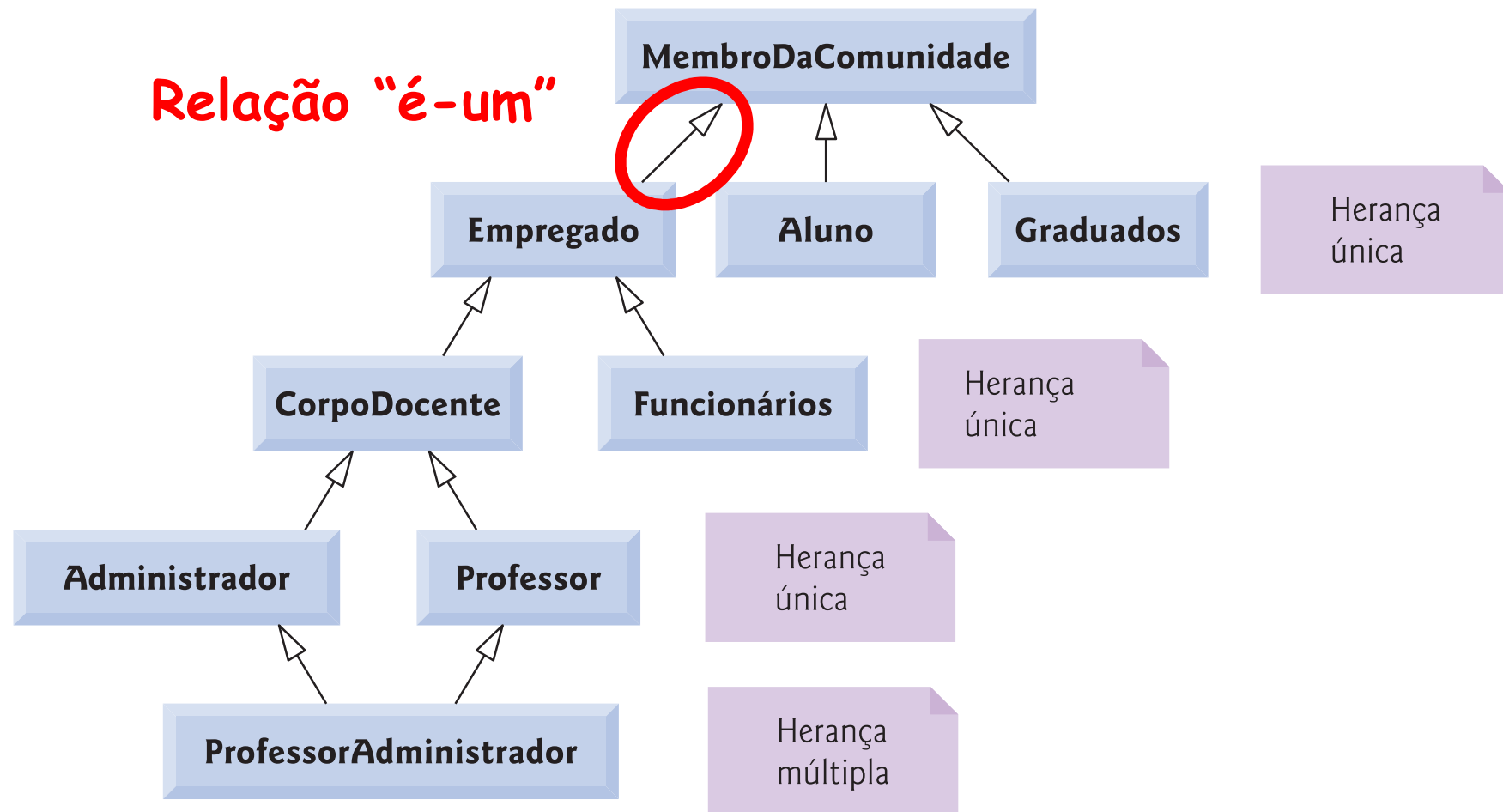
- Hierarquia de herança
  - Relacionamentos de herança:
    - Estrutura hierárquica do tipo árvore
  - Cada classe torna-se
    - Uma classe base
      - Fornece dados/comportamentos a outras classes
    - OU**
    - Uma classe derivada
      - Herda dados/comportamentos de outras classes

# Classes Base e Derivadas





# Classes Base e Derivadas



# Classes Base e Derivadas

- Herança `public`
  - É especificada com:
    - `class TwoDimensionalShape : public Shape`
      - A classe `TwoDimensionalShape` herda da classe `Shape`
  - Membros `private` da classe base
    - Não podem ser acessados diretamente
    - Ainda assim são herdados
      - Manipulados por meio das funções-membro `public` herdadas

# Classes Base e Derivadas

- Herança `public`
  - Membros `public` e `protected` da classe base
    - São herdados com o mesmo acesso do membro original
  - Funções `friend`
    - Não são herdadas

# Membros protected

- **Acesso protected**
  - Nível intermediário de proteção entre `public` e `private`
  - Os membros `protected` podem ser acessados por:
    - Membros da própria classe base
    - Funções `friend` da própria classe base
    - Membros da classe derivada
    - Funções `friend` da classe derivada

# Membros protected

- Membros da classe derivada
  - Podem acessar membros `public` e `protected` da classe base
    - Podem simplesmente usar o nome dos membros
  - Membros da classe base redefinidos nas classes derivadas
    - Podem ser acessados por meio do nome da classe base e do operador binário de resolução de escopo (`::`)
      - Ex.: `Base::membro_de_dados`

# Relacionamento entre Classes

## Bases e Derivadas

- Hierarquia de herança

- Ex.:

- `CommissionEmployee/BasePlusCommissionEmployee`

- **CommissionEmployee**

- Nome, sobrenome, SSN, taxa de comissão, quantidade de vendas brutas

- **BasePlusCommissionEmployee**

- Nome, sobrenome, SSN, taxa de comissão, quantidade de vendas brutas

- **E adicionalmente: salário-base**

# Criando e Utilizando uma Classe `CommissionEmployee`

- Classe `CommissionEmployee`
  - Arquivo de cabeçalho `CommissionEmployee`
    - *Especifica serviços públicos*
      - *Construtor*
      - *Funções `get` e `set`*
      - *Funções-membro `earnings` e `print`*
  - Arquivo de código-fonte `CommissionEmployee`
    - *Especifica definições das funções-membro*

```
/*
 * Aula 12 - Exemplo 1
 * Arquivo comissionCap12Ex1.h
 * Autor: Miguel Campista
 */
#ifndef COMMISSION_H
#define COMMISSION_H

#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

class CommissionEmployee {
public:
    CommissionEmployee (const string &, const string &,
                        const string &, double = 0.0, double = 0.0);
    void setFirstName (const string &); // Configura o nome
    string getFirstName () const; // Retorna o nome

    void setLastName (const string &);
    string getLastName () const;

    void setSocialSecurityNumber (const string &);
    string getSocialSecurityNumber () const;

    void setGrossSales (double); // Configura a quant. de vendas brutas
    double getGrossSales () const; // Retorna a quant. de vendas brutas

    void setCommissionRate (double); // Conf. taxa de comissão
    double getCommissionRate () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
};
```



# Primeiro Exemplo de Herança em C++

```
private:  
    string firstName, lastName;  
    string socialSecurityNumber;  
    double grossSales;  
    double commissionRate;  
};  
  
#endif
```

```

/*
 * Aula 12 - Exemplo 1
 * Arquivo comissionCap12Ex1.cpp
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex1.h"

CommissionEmployee::CommissionEmployee (const string &first, const string &last,
                                       const string &ssn, double sales, double rate) {
    firstName = first;
    lastName = last;
    socialSecurityNumber = ssn;
    setGrossSales (sales);
    setCommissionRate (rate);
}

void CommissionEmployee::setFirstName (const string &first) {
    firstName = first;
}

string CommissionEmployee::getFirstName () const { return firstName; }

void CommissionEmployee::setLastName (const string &last) {
    lastName = last;
}

string CommissionEmployee::getLastName () const { return lastName; }

void CommissionEmployee::setSocialSecurityNumber (const string &ssn) {
    socialSecurityNumber = ssn;
}

string CommissionEmployee::getSocialSecurityNumber () const {
    return socialSecurityNumber;
}

```

# Primeiro Exemplo de Herança em C++

```
void CommissionEmployee::setGrossSales (double sales) {
    grossSales = (sales < 0.0) ? 0.0 : sales;
}

double CommissionEmployee::getGrossSales () const { return grossSales; }

void CommissionEmployee::setCommissionRate (double rate) {
    commissionRate = (rate > 0.0 && rate < 1.0) ? rate : 0.0;
}

double CommissionEmployee::getCommissionRate () const { return commissionRate; }

double CommissionEmployee::earnings () const {
    return grossSales*commissionRate;
}

void CommissionEmployee::print () const {
    cout << "Commission employee: " << firstName << ' ' << lastName
        << "\nsocial security number: " << socialSecurityNumber
        << "\ngross sales: " << grossSales
        << "\ncommission rate: " << commissionRate;
}
```

# Primeiro Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 1
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex1.h"

int main () {
    CommissionEmployee employee ("Sue", "Jones", "21-2222-2222", 10000, .06);

    // Configura a formatação de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << "Employee information obtained by get functions:\n"
        << "\nFirst name: " << employee.getFirstName ()
        << "\nLast name: " << employee.getLastName ()
        << "\nSocial Security Number: "
        << employee.getSocialSecurityNumber ()
        << "\nGross Sales: " << employee.getGrossSales ()
        << "\nCommissionRate: " << employee.getCommissionRate () << endl;

    employee.setGrossSales (8000);
    employee.setCommissionRate (.1);

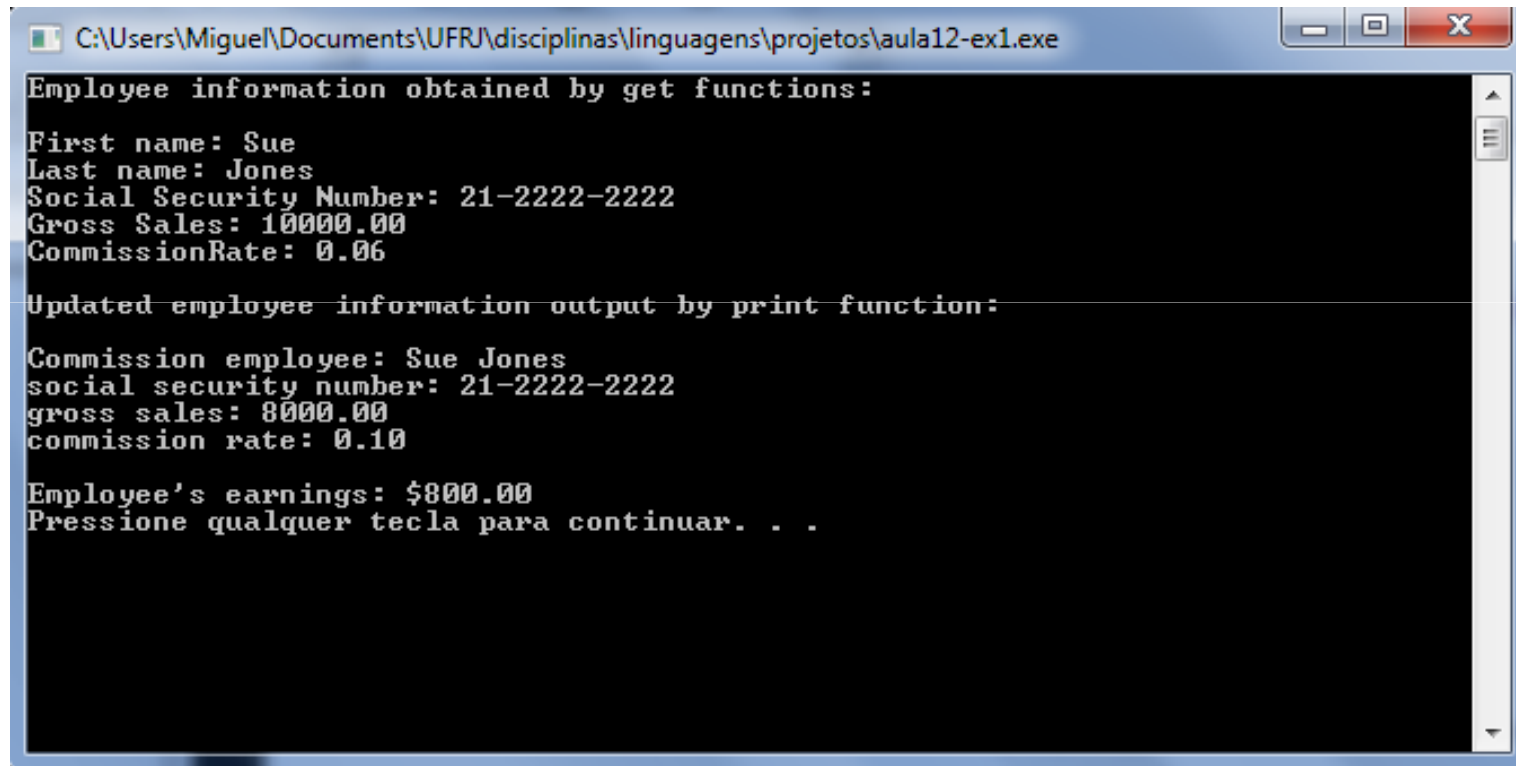
    cout << "\nUpdated employee information output by print function:\n"
        << endl;
    employee.print ();

    // Exibe os rendimentos do empregado
    cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

    return 0;
}
```

# Primeiro Exemplo de Herança em C++

```
/*  
 * Aula 12 - Exemplo 1  
 * Programa Principal  
 * Autor: Miguel Campista
```



```
C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos\aula12-ex1.exe  
Employee information obtained by get functions:  
First name: Sue  
Last name: Jones  
Social Security Number: 21-2222-2222  
Gross Sales: 10000.00  
CommissionRate: 0.06  
Updated employee information output by print function:  
Commission employee: Sue Jones  
social security number: 21-2222-2222  
gross sales: 8000.00  
commission rate: 0.10  
Employee's earnings: $800.00  
Pressione qualquer tecla para continuar. . .
```

```
// Exibe os rendimentos do empregado  
cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;  
  
return 0;  
}
```

# Criação SEM Herança da Classe BasePlusCommissionEmployee

- Classe BasePlusCommissionEmployee
  - Grande parte do código é semelhante a CommissionEmployee
    - Membros de dados `private`
    - Métodos `public`
    - Construtor
  - Adições
    - Membro de dados `private baseSalary`
    - Métodos `setBaseSalary` e `getBaseSalary`

# Criação SEM Herança da Classe BasePlusCommissionEmployee

- Classe BasePlusCommissionEmployee
  - Grande parte do código é semelhante a CommissionEmployee
    - Membros de dados `private`
    - Métodos `public`
    - Construtor
  - Adições
    - Membro de dados `private baseSalary`
    - Métodos `setBaseSalary` e `getBaseSalary`

**Como criar a classe BasePlusCommissionEmployee sem utilizar herança?**

```
/*
 * Aula 12 - Exemplo 2
 * Arquivo comissionCap12Ex2.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

class BasePlusCommissionEmployee {
public:
    BasePlusCommissionEmployee (const string &, const string &,
                                const string &, double = 0.0, double = 0.0, double = 0.0);
    void setFirstName (const string &); // Configura o nome
    string getFirstName () const; // Retorna o nome

    void setLastName (const string &);
    string getLastName () const;

    void setSocialSecurityNumber (const string &);
    string getSocialSecurityNumber () const;

    void setGrossSales (double); // Configura a quant. de vendas brutas
    double getGrossSales () const; // Retorna a quant. de vendas brutas

    void setCommissionRate (double); // Conf. taxa de comissão
    double getCommissionRate () const;

    void setBaseSalary (double); // Conf. o salário base
    double getBaseSalary () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
```



# Segundo Exemplo de Herança em C++

```
private:  
    string firstName, lastName;  
    string socialSecurityNumber;  
    double grossSales;  
    double commissionRate;  
    double baseSalary;  
};  
  
#endif
```

```
/*
 * Aula 12 - Exemplo 2
 * Arquivo comissionCap12Ex2.cpp
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex2.h"

BasePlusCommissionEmployee::BasePlusCommissionEmployee (const string &first,
                                                         const string &last, const string &ssn,
                                                         double sales, double rate, double salary) {
    firstName = first;
    lastName = last;
    socialSecurityNumber = ssn;
    setGrossSales (sales);
    setCommissionRate (rate);
    setBaseSalary (salary);
}

void BasePlusCommissionEmployee::setFirstName (const string &first) {
    firstName = first;
}

string BasePlusCommissionEmployee::getFirstName () const { return firstName; }

void BasePlusCommissionEmployee::setLastName (const string &last) {
    lastName = last;
}

string BasePlusCommissionEmployee::getLastName () const { return lastName; }

void BasePlusCommissionEmployee::setSocialSecurityNumber (const string &ssn) {
    socialSecurityNumber = ssn;
}

string BasePlusCommissionEmployee::getSocialSecurityNumber () const {
    return socialSecurityNumber;
}
```

# Segundo Exemplo de Herança em C++

```
void BasePlusCommissionEmployee::setGrossSales (double sales) {
    grossSales = (sales < 0.0) ? 0.0 : sales;
}

double BasePlusCommissionEmployee::getGrossSales () const { return grossSales; }

void BasePlusCommissionEmployee::setCommissionRate (double rate) {
    commissionRate = (rate > 0.0 && rate < 1.0) ? rate : 0.0;
}

double BasePlusCommissionEmployee::getCommissionRate () const {
    return commissionRate;
}

void BasePlusCommissionEmployee::setBaseSalary (double salary) {
    baseSalary = (salary < 0.0) ? 0.0 : salary;
}

double BasePlusCommissionEmployee::getBaseSalary () const { return baseSalary; }

double BasePlusCommissionEmployee::earnings () const {
    return baseSalary + (grossSales*commissionRate);
}

void BasePlusCommissionEmployee::print () const {
    cout << "Commission employee: " << firstName << ' ' << lastName
        << "\nsocial security number: " << socialSecurityNumber
        << "\ngross sales: " << grossSales
        << "\ncommission rate: " << commissionRate
        << "\nbase salary: " << baseSalary;
}
```

```

/*
 * Aula 12 - Exemplo 2
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex2.h"

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",
                                         "21-2222-2222", 10000, .06, 300);

    // Configura a formatação de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << "Employee information obtained by get functions:\n"
         << "\nFirst name: " << employee.getFirstName ()
         << "\nLast name: " << employee.getLastName ()
         << "\nSocial security number: "
         << employee.getSocialSecurityNumber ()
         << "\nGross sales: " << employee.getGrossSales ()
         << "\nCommission rate: " << employee.getCommissionRate ()
         << "\nBase salary: " << employee.getBaseSalary () << endl;

    employee.setBaseSalary (1000);
    employee.setGrossSales (8000);
    employee.setCommissionRate (.1);

    cout << "\nUpdated employee information output by print function:\n"
         << endl;
    employee.print ();

    // Exibe os rendimentos do empregado
    cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

    return 0;
}

```

```

/*
 * Aula 12 - Exemplo 2
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex2.h"

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",

```

```

Employee information obtained by get functions:
First name: Sue
Last name: Jones
Social security number: 21-2222-2222
Gross sales: 10000.00
Commission rate: 0.06
Base salary: 300.00

Updated employee information output by print function:
Commission employee: Sue Jones
social security number: 21-2222-2222
gross sales: 8000.00
commission rate: 0.10
base salary: 1000.00

Employee's earnings: $1800.00
Pressione qualquer tecla para continuar. . . _

```

```

    employee.print ();

```

```

    // Exibe os rendimentos do empregado
    cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

```

```

    return 0;

```

```

}

```

# Vantagens do Uso de Herança

- Copiar e colar código de uma classe para a outra pode espalhar erros por múltiplos arquivos de código-fonte
  - Para evitar a duplicação de código (e possivelmente erros), utilize a herança, em vez do método "copiar e colar"
    - Em situações em que você quer que uma classe "absorva" os membros de dados e as funções-membro de outra classe

# Vantagens do Uso de Herança

- Com a herança, os membros de dados e as funções-membro comuns a todas as classes na hierarquia são declarados em uma classe base
  - Quando esses recursos comuns requerem mudanças, as alterações são feitas somente na classe base
    - *As classes derivadas herdam as alterações*
    - *Sem a herança, as alterações precisariam ser feitas em todos os arquivos de código-fonte que contêm uma cópia do código em questão*

# Criação de uma Hierarquia de Herança

- Classe `BasePlusCommissionEmployee`
  - Derivada da classe `CommissionEmployee`
    - É uma `CommissionEmployee`
    - Herda todos os membros `public`
  - O construtor nunca é herdado
    - Usa a sintaxe inicializadora da classe base para inicializar membros de dados
  - Adiciona o membro de dados `baseSalary`



# Terceiro Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 3
 * Arquivo commissionCap12Ex3.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <iomanip>
#include "commissionCap12Ex1.h"

using namespace std;

class BasePlusCommissionEmployee : public CommissionEmployee {
public:
    BasePlusCommissionEmployee (const string &, const string &,
                                const string &, double = 0.0, double = 0.0, double = 0.0);

    void setBaseSalary (double); // Conf. o salário base
    double getBaseSalary () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
private:
    double baseSalary;
};

#endif
```

# Terceiro Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 3
 * Arquivo comissionCap12Ex3.cpp
 * Autor: Miguel Campista
 */
#include "comissionCap12Ex3.h"

BasePlusCommissionEmployee::BasePlusCommissionEmployee (const string &first,
    const string &last, const string &ssn,
    double sales, double rate, double salary) :
    CommissionEmployee (first, last, ssn, sales, rate) {
    setBaseSalary (salary);
}

void BasePlusCommissionEmployee::setBaseSalary (double salary) {
    baseSalary = (salary < 0.0) ? 0.0 : salary;
}

double BasePlusCommissionEmployee::getBaseSalary () const { return baseSalary; }

double BasePlusCommissionEmployee::earnings () const {
    return baseSalary + (grossSales*commissionRate);
}

void BasePlusCommissionEmployee::print () const {
    cout << "Commission employee: " << firstName << ' ' << lastName
        << "\nsocial security number: " << socialSecurityNumber
        << "\ngross sales: " << grossSales
        << "\ncommission rate: " << commissionRate
        << "\nbase salary: " << baseSalary;
}
}
```

# Terceiro Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 3
 * Arquivo comissionCap12Ex3.cpp
 * Autor: Miguel Campista
 */
#include "comissionCap12Ex3.h"

BasePlusCommissionEmployee::BasePlusCommissionEmployee (const string &first,
    const string &last, const string &ssn,
    double sales, double rate, double salary) :
    CommissionEmployee (first, last, ssn, sales, rate) {
    setBaseSalary (salary);
}

void BasePlusCommissionEmployee::setBaseSalary (double salary) {
    baseSalary = (salary < 0.0) ? 0.0 : salary;
}

double BasePlusCommissionEmployee::getBaseSalary () const { return baseSalary; }
```

**Sintaxe para inicialização da classe base que utiliza inicialização de membro para passar argumentos para o construtor da classe base**

```
<< "\nsocial security number: " << socialSecurityNumber
<< "\ngross sales: " << grossSales
<< "\ncommission rate: " << commissionRate
<< "\nbase salary: " << baseSalary;
}
```

# Terceiro Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 3
 * Arquivo comissionCap12Ex3.cpp
 * Autor: Miguel Campista
 */
#include "comissionCap12Ex3.h"

BasePlusCommissionEmployee::BasePlusCommissionEmployee (const string &first,
    const string &last, const string &ssn,
    double sales, double rate, double salary) :
    CommissionEmployee (first, last, ssn, sales, rate) {
    setBaseSalary (salary);
}

```

Tenta acessar dados privados da classe base diretamente...

```
double BasePlusCommissionEmployee::earnings () const {
    return baseSalary + (grossSales*commissionRate);
}

```

```
void BasePlusCommissionEmployee::print () const {
    cout << "Commission employee: " << firstName << ' ' << lastName
        << "\nsocial security number: " << socialSecurityNumber
        << "\ngross sales: " << grossSales
        << "\ncommission rate: " << commissionRate
        << "\nbase salary: " << baseSalary;
}

```

# Terceiro Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 3
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex3.h"

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",
                                         "21-2222-2222", 10000, .06, 300);

    // Configura a formatação de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << "Employee information obtained by get functions:\n"
         << "\nFirst name: " << employee.getFirstName ()
         << "\nLast name: " << employee.getLastName ()
         << "\nSocial security number: "
         << employee.getSocialSecurityNumber ()
         << "\nGross sales: " << employee.getGrossSales ()
         << "\nCommission rate: " << employee.getCommissionRate ()
         << "\nBase salary: " << employee.getBaseSalary () << endl;

    employee.setBaseSalary (1000);
    employee.setGrossSales (8000);
    employee.setCommissionRate (.1);

    cout << "\nUpdated employee information output by print function:\n"
         << endl;
    employee.print ();

    // Exibe os rendimentos do empregado
    cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

    return 0;
}
```

# Terceiro Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 3
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex3.h"

i 39      C:\Users\Miguel\Documents\UFRJ\... In member function `double BasePlusCommissionEmployee::earnings() const':
22      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::grossSales' is private
39      C:\Users\Miguel\Documents\UFRJ\... within this context
22      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::grossSales' is private
40      C:\Users\Miguel\Documents\UFRJ\... within this context
22      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::commissionRate' is private
40      C:\Users\Miguel\Documents\UFRJ\... within this context
22      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::commissionRate' is private
37      C:\Users\Miguel\Documents\UFRJ\... within this context
26      C:\Users\Miguel\Documents\UFRJ\... In member function `void BasePlusCommissionEmployee::print() const':
37      C:\Users\Miguel\Documents\UFRJ\... `std::string CommissionEmployee::firstName' is private
26      C:\Users\Miguel\Documents\UFRJ\... within this context
37      C:\Users\Miguel\Documents\UFRJ\... `std::string CommissionEmployee::firstName' is private
26      C:\Users\Miguel\Documents\UFRJ\... within this context
37      C:\Users\Miguel\Documents\UFRJ\... `std::string CommissionEmployee::lastName' is private
26      C:\Users\Miguel\Documents\UFRJ\... within this context
37      C:\Users\Miguel\Documents\UFRJ\... `std::string CommissionEmployee::lastName' is private
26      C:\Users\Miguel\Documents\UFRJ\... within this context
38      C:\Users\Miguel\Documents\UFRJ\... `std::string CommissionEmployee::socialSecurityNumber' is private
27      C:\Users\Miguel\Documents\UFRJ\... within this context
38      C:\Users\Miguel\Documents\UFRJ\... `std::string CommissionEmployee::socialSecurityNumber' is private
27      C:\Users\Miguel\Documents\UFRJ\... within this context
39      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::grossSales' is private
28      C:\Users\Miguel\Documents\UFRJ\... within this context
39      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::grossSales' is private
28      C:\Users\Miguel\Documents\UFRJ\... within this context
40      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::commissionRate' is private
29      C:\Users\Miguel\Documents\UFRJ\... within this context
40      C:\Users\Miguel\Documents\UFRJ\... `double CommissionEmployee::commissionRate' is private
29      C:\Users\Miguel\Documents\UFRJ\... within this context
      C:\Users\Miguel\Documents\UFRJ\... [Build Error] exe: **** [commissionCap12Ex3.o] Error 1

      cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

      return 0;
}
```

# Criação de uma Hierarquia de Herança

- Inclusão do arquivo de cabeçalho da classe base
  - O arquivo de cabeçalho da classe base deve ser incluído no arquivo de cabeçalho da classe derivada por três motivos:
    - O compilador deve saber que a classe base existe
    - O compilador deve conhecer o tamanho dos membros de dados herdados
    - O compilador deve garantir que os membros da classe herdada sejam utilizados apropriadamente

# Criação de uma Hierarquia de Herança

- Uso de dados `protected`
  - Permite que a classe `BasePlusCommissionEmployee` acesse diretamente os membros de dados da classe base
  - Os membros `protected` da classe base são herdados por todas as suas classes derivadas
- Boa prática de programação...
  - Em primeiro lugar, declare os membros `public`, em segundo, os membros `protected` e, por último, os membros `private`



```

/*
 * Aula 12 - Exemplo 4
 * Arquivo comissionCap12Ex4.h
 * Autor: Miguel Campista
 */
#ifndef COMMISSION_H
#define COMMISSION_H

#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

class CommissionEmployee {
public:
    CommissionEmployee (const string &, const string &,
                        const string &, double = 0.0, double = 0.0);
    void setFirstName (const string &); // Configura o nome
    string getFirstName () const; // Retorna o nome

    void setLastName (const string &);
    string getLastName () const;

    void setSocialSecurityNumber (const string &);
    string getSocialSecurityNumber () const;

    void setGrossSales (double); // Configura a quant. de vendas brutas
    double getGrossSales () const; // Retorna a quant. de vendas brutas

    void setCommissionRate (double); // Conf. taxa de comissão
    double getCommissionRate () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;

```

# Quarto Exemplo de Herança em C++

```
protected:  
    string firstName, lastName;  
    string socialSecurityNumber;  
    double grossSales;  
    double commissionRate;  
};  
  
#endif
```

```

/*
 * Aula 12 - Exemplo 4
 * Arquivo comissionCap12Ex4.cpp
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex4.h"

CommissionEmployee::CommissionEmployee (const string &first, const string &last,
                                       const string &ssn, double sales, double rate) {
    firstName = first;
    lastName = last;
    socialSecurityNumber = ssn;
    setGrossSales (sales);
    setCommissionRate (rate);
}

void CommissionEmployee::setFirstName (const string &first) {
    firstName = first;
}

string CommissionEmployee::getFirstName () const { return firstName; }

void CommissionEmployee::setLastName (const string &last) {
    lastName = last;
}

string CommissionEmployee::getLastName () const { return lastName; }

void CommissionEmployee::setSocialSecurityNumber (const string &ssn) {
    socialSecurityNumber = ssn;
}

string CommissionEmployee::getSocialSecurityNumber () const {
    return socialSecurityNumber;
}

```

# Quarto Exemplo de Herança em C++

```
void CommissionEmployee::setGrossSales (double sales) {
    grossSales = (sales < 0.0) ? 0.0 : sales;
}

double CommissionEmployee::getGrossSales () const { return grossSales; }

void CommissionEmployee::setCommissionRate (double rate) {
    commissionRate = (rate > 0.0 && rate < 1.0) ? rate : 0.0;
}

double CommissionEmployee::getCommissionRate () const { return commissionRate; }

double CommissionEmployee::earnings () const {
    return grossSales*commissionRate;
}

void CommissionEmployee::print () const {
    cout << "Commission employee: " << firstName << ' ' << lastName
        << "\nsocial security number: " << socialSecurityNumber
        << "\ngross sales: " << grossSales
        << "\ncommission rate: " << commissionRate;
}
```

# Quarto Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 4
 * Arquivo basepluscommissionCap12Ex4.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <iomanip>
#include "commissionCap12Ex4.h"

using namespace std;

class BasePlusCommissionEmployee : public CommissionEmployee {
public:
    BasePlusCommissionEmployee (const string &, const string &,
                                const string &, double = 0.0, double = 0.0, double = 0.0);

    void setBaseSalary (double); // Conf. o salário base
    double getBaseSalary () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
private:
    double baseSalary;
};

#endif
```

# Quarto Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 4
 * Arquivo basepluscomissionCap12Ex4.cpp
 * Autor: Miguel Campista
 */
#include "basepluscomissionCap12Ex4.h"

BasePlusCommissionEmployee::BasePlusCommissionEmployee (const string &first,
    const string &last, const string &ssn,
    double sales, double rate, double salary) :
    CommissionEmployee (first, last, ssn, sales, rate) {
    setBaseSalary (salary);
}

void BasePlusCommissionEmployee::setBaseSalary (double salary) {
    baseSalary = (salary < 0.0) ? 0.0 : salary;
}

double BasePlusCommissionEmployee::getBaseSalary () const { return baseSalary; }

double BasePlusCommissionEmployee::earnings () const {
    return baseSalary + (grossSales*commissionRate);
}

void BasePlusCommissionEmployee::print () const {
    cout << "Commission employee: " << firstName << ' ' << lastName
    << "\nsocial security number: " << socialSecurityNumber
    << "\ngross sales: " << grossSales
    << "\ncommission rate: " << commissionRate
    << "\nbase salary: " << baseSalary;
}
```

```

/*
 * Aula 12 - Exemplo 4
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "basepluscommissionCap12Ex4.h"

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",
                                         "21-2222-2222", 10000, .06, 300);

    // Configura a formatação de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << "Employee information obtained by get functions:\n"
         << "\nFirst name: " << employee.getFirstName ()
         << "\nLast name: " << employee.getLastName ()
         << "\nSocial security number: "
         << employee.getSocialSecurityNumber ()
         << "\nGross sales: " << employee.getGrossSales ()
         << "\nCommission rate: " << employee.getCommissionRate ()
         << "\nBase salary: " << employee.getBaseSalary () << endl;

    employee.setBaseSalary (1000);
    employee.setGrossSales (8000);
    employee.setCommissionRate (.1);

    cout << "\nUpdated employee information output by print function:\n"
         << endl;
    employee.print ();

    // Exibe os rendimentos do empregado
    cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

    return 0;
}

```

```
/*
 * Aula 12 - Exemplo 4
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "basepluscommissionCap12Ex4.h"

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",
```

```
C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos\aula12-ex4.exe
Employee information obtained by get functions:
First name: Sue
Last name: Jones
Social security number: 21-2222-2222
Gross sales: 10000.00
Commission rate: 0.06
Base salary: 300.00

Updated employee information output by print function:

Commission employee: Sue Jones
social security number: 21-2222-2222
gross sales: 8000.00
commission rate: 0.10
base salary: 1000.00

Employee's earnings: $1800.00
Pressione qualquer tecla para continuar. . .
```

```
employee.print ();
```

```
// Exibe os rendimentos do empregado
```

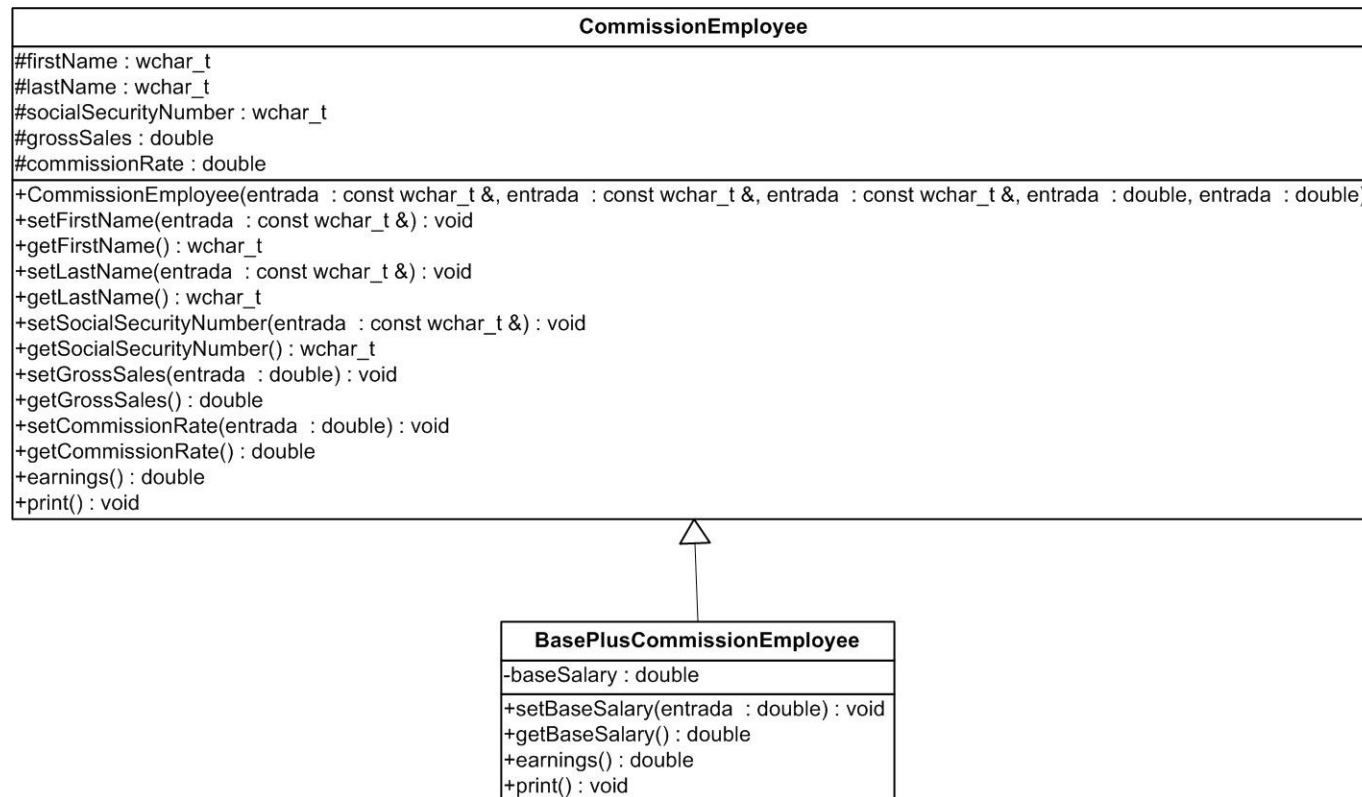
```
cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;
```

```
return 0;
```

```
}
```



# Quarto Exemplo de Herança em C++



# Uso de Membros de Dados `protected`

- Vantagens
  - A classe derivada pode modificar valores diretamente
    - Evita a sobrecarga de chamada de método *set/get*
      - Aumenta ligeiramente o desempenho

# Uso de Membros de Dados protected

- Desvantagens
  - Não há verificação de validação
    - A classe derivada pode atribuir valores inválidos
  - Depende da implementação
    - As funções da classe derivada são provavelmente mais dependentes da implementação da classe base
    - Alterações na implementação da classe base podem provocar alterações na classe derivada
      - O software é dependente

# Uso de Membros de Dados `protected`

- É apropriado utilizar o especificador `protected` quando uma classe base tiver de fornecer um serviço (uma função-membro) **apenas a suas classes derivadas (e friends)**
  - Não a outros clientes
- Declarar membros de dados de classe base como `private` (em vez de declará-los `protected`) permite aos programadores alterarem a classe base
  - Sem alterar as implementações de classe derivada

# Uso de Membros de Dados `protected`

- Deve-se evitar incluir membros de dados `protected` em uma classe base
  - Em vez disso, funções-membro não-`private` devem ser usadas para acessar membros de dados `private`, assegurando que o objeto mantenha um estado consistente

# Hierarquia de Herança Utilizando Dados `private`

- Reexamine a hierarquia
  - Use a melhor prática de engenharia de software
    - Declare os membros de dados como `private`
    - Forneça as funções `get` e `set` `public`
    - Use o método `get` para obter os valores dos membros de dados

```
/*
 * Aula 12 - Exemplo 5
 * Arquivo comissionCap12Ex5.h
 * Autor: Miguel Campista
 */
#ifndef COMMISSION_H
#define COMMISSION_H

#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

class CommissionEmployee {
public:
    CommissionEmployee (const string &, const string &,
                       const string &, double = 0.0, double = 0.0);
    void setFirstName (const string &); // Configura o nome
    string getFirstName () const; // Retorna o nome

    void setLastName (const string &);
    string getLastName () const;

    void setSocialSecurityNumber (const string &);
    string getSocialSecurityNumber () const;

    void setGrossSales (double); // Configura a quant. de vendas brutas
    double getGrossSales () const; // Retorna a quant. de vendas brutas

    void setCommissionRate (double); // Conf. taxa de comissão
    double getCommissionRate () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
```

# Quinto Exemplo de Herança em C++

```
private:  
    string firstName, lastName;  
    string socialSecurityNumber;  
    double grossSales;  
    double commissionRate;  
};  
  
#endif
```



```

/*
 * Aula 12 - Exemplo 5
 * Arquivo comissionCap12Ex5.cpp
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex5.h"

CommissionEmployee::CommissionEmployee (const string &first, const string &last,
        const string &ssn, double sales, double rate) :
        firstName (first), lastName (last), socialSecurityNumber (ssn) {
    setGrossSales (sales);
    setCommissionRate (rate);
}

void CommissionEmployee::setFirstName (const string &first) {
    firstName = first;
}

string CommissionEmployee::getFirstName () const { return firstName; }

void CommissionEmployee::setLastName (const string &last) {
    lastName = last;
}

string CommissionEmployee::getLastName () const { return lastName; }

void CommissionEmployee::setSocialSecurityNumber (const string &ssn) {
    socialSecurityNumber = ssn;
}

string CommissionEmployee::getSocialSecurityNumber () const {
    return socialSecurityNumber;
}

```

# Quinto Exemplo de Herança em C++

```
void CommissionEmployee::setGrossSales (double sales) {
    grossSales = (sales < 0.0) ? 0.0 : sales;
}

double CommissionEmployee::getGrossSales () const { return grossSales; }

void CommissionEmployee::setCommissionRate (double rate) {
    commissionRate = (rate > 0.0 && rate < 1.0) ? rate : 0.0;
}

double CommissionEmployee::getCommissionRate () const { return commissionRate; }

double CommissionEmployee::earnings () const {
    return getGrossSales () * getCommissionRate ();
}

void CommissionEmployee::print () const {
    cout << "commission employee: " << getFirstName ()
         << ' ' << getLastName ()
         << "\nsocial security number: " << getSocialSecurityNumber ()
         << "\ngross sales: " << getGrossSales ()
         << "\ncommission rate: " << getCommissionRate ();
}
```

# Hierarquia de Herança

## Utilizando Dados `private`

- Utilizar uma função-membro para acessar o valor de um membro de dados é um pouco mais lento que acessar os dados diretamente
  - Entretanto, os atuais compiladores são projetados para realizar otimizações implicitamente (como colocar `inline` as chamadas de funções-membro `get` e `set`)
    - Em decorrência disso, os programadores devem escrever código que obedeça aos princípios apropriados de engenharia de software e deixar questões de otimização para o compilador

# Quinto Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 5
 * Arquivo basepluscommissionCap12Ex5.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <iomanip>
#include "commissionCap12Ex5.h"

using namespace std;

class BasePlusCommissionEmployee : public CommissionEmployee {
public:
    BasePlusCommissionEmployee (const string &, const string &,
                                const string &, double = 0.0, double = 0.0, double = 0.0);

    void setBaseSalary (double); // Conf. o salário base
    double getBaseSalary () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
private:
    double baseSalary;
};

#endif
```

# Quinto Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 5
 * Arquivo basepluscommissionCap12Ex5.cpp
 * Autor: Miguel Campista
 */
#include "basepluscommissionCap12Ex5.h"

BasePlusCommissionEmployee::BasePlusCommissionEmployee (const string &first,
    const string &last, const string &ssn,
    double sales, double rate, double salary) :
    CommissionEmployee (first, last, ssn, sales, rate) {
    setBaseSalary (salary);
}

void BasePlusCommissionEmployee::setBaseSalary (double salary) {
    baseSalary = (salary < 0.0) ? 0.0 : salary;
}

double BasePlusCommissionEmployee::getBaseSalary () const { return baseSalary; }

double BasePlusCommissionEmployee::earnings () const {
    return baseSalary + CommissionEmployee::earnings ();
}

void BasePlusCommissionEmployee::print () const {
    cout << "Base salary" << endl;

    CommissionEmployee::print ();

    cout << "\nbase salary: " << baseSalary;
}

```

# Hierarquia de Herança Utilizando Dados `private`

- Quando uma função-membro de classe base é redefinida por uma classe derivada, a versão da classe derivada frequentemente chama a versão da classe base para fazer o trabalho adicional
  - Não utilizar o operador `::` prefixado com o nome da classe base provoca recursão infinita

```
BasePlusCommissionEmployee::print () const {  
    cout << "Base salary" << endl;  
    print (); // Recursão infinita!  
    cout << "\nbase salary: " << baseSalary;  
}
```

# Hierarquia de Herança Utilizando Dados `private`

- Incluir uma função-membro da classe base com uma assinatura diferente na classe derivada...
  - Oculta a função da classe base
    - Tentativas de chamar a versão da classe base pela interface `public` de um objeto da classe derivada provocam erros de compilação

```
class Base {  
    public:  
        ...  
    void print ();  
        ...  
};
```

```
class Derivada :  
    public Base {  
    public:  
        ...  
    void print (int);  
        ...  
};
```

```
int main () {  
    Derivada d;  
    d.print ();  
        ...  
}
```

# Hierarquia de Herança Utilizando Dados `private`

- Incluir uma função-membro da classe base com uma assinatura diferente na classe derivada...
  - Oculta a função da classe base
    - Tentativas de chamar a versão da classe base pela interface `public` de um objeto da classe derivada provocam erros de compilação

```
class Base {  
    public:  
        ...  
    void print ();  
        ...  
};
```

```
class Derivada :  
    public Base {  
    public:  
        ...  
    void print (int);  
        ...  
};
```

```
int main () {  
    Derivada d;  
    d.print ();  
}
```

**Erro de compilação!**



```

/*
 * Aula 12 - Exemplo 5
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "basepluscommissionCap12Ex5.h"

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",
                                         "21-2222-2222", 10000, .06, 300);

    // Configura a formatação de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << "Employee information obtained by get functions:\n"
         << "\nFirst name: " << employee.getFirstName ()
         << "\nLast name: " << employee.getLastName ()
         << "\nSocial security number: "
         << employee.getSocialSecurityNumber ()
         << "\nGross sales: " << employee.getGrossSales ()
         << "\nCommission rate: " << employee.getCommissionRate ()
         << "\nBase salary: " << employee.getBaseSalary () << endl;

    employee.setBaseSalary (1000);
    employee.setGrossSales (8000);
    employee.setCommissionRate (.1);

    cout << "\nUpdated employee information output by print function:\n"
         << endl;
    employee.print ();

    // Exibe os rendimentos do empregado
    cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

    return 0;
}

```

```

/*
 * Aula 12 - Exemplo 5
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "basepluscommissionCap12Ex5.h"

```

```

Employee information obtained by get functions:
First name: Sue
Last name: Jones
Social security number: 21-2222-2222
Gross sales: 10000.00
Commission rate: 0.06
Base salary: 300.00

Updated employee information output by print function:
Base salary
commission employee: Sue Jones
social security number: 21-2222-2222
gross sales: 8000.00
commission rate: 0.10
base salary: 1000.00

Employee's earnings: $1800.00
Pressione qualquer tecla para continuar. . . _

```

```

        << endl;
        employee.print ();

        // Exibe os rendimentos do empregado
        cout << "\n\nEmployee's earnings: $" << employee.earnings () << endl;

        return 0;
    }

```

# Construtores e Destrutores em Classes Derivadas

- Instanciando um objeto de classe derivada
  - Cadeia de chamadas de construtor
    - O construtor de classe derivada invoca o construtor de classe básica
      - Implicitamente ou explicitamente

# Construtores e Destrutores em Classes Derivadas

- Instanciando um objeto de classe derivada
  - Cadeia de chamadas de construtor
    - Base da hierarquia de herança
      - Último construtor chamado na cadeia, mas primeiro a terminar a execução
      - Ex.: Hierarquia  
`CommissionEmployee/BasePlusCommissionEmployee`
        - » `CommissionEmployee` é o construtor chamado por último e o primeiro a terminar a execução
  - Inicializando membros de dados
    - Cada construtor de classe base inicializa os respectivos membros de dados herdados pela classe derivada

# Construtores e Destrutores em Classes Derivadas

- Quando um programa cria um objeto de classe derivada
  - O construtor da classe derivada chama imediatamente o construtor da classe base
  - O corpo do construtor da classe base executa
  - Em seguida, os inicializadores de membro da classe derivada executam
  - Por fim, o corpo do construtor da classe derivada executa
    - Esse processo coloca a hierarquia em cascata se ela contiver mais de dois níveis

# Construtores e Destrutores em Classes Derivadas

- Destruindo um objeto de classe derivada
  - Cadeia de chamadas de destrutor
    - Ordem inversa da cadeia de construtor
    - Primeiro, o destrutor de um objeto de classe derivada é chamado
    - O destrutor da classe derivada só invoca o destrutor da classe base seguinte no nível superior da hierarquia após o término de sua tarefa
    - Continua até que o destrutor da classe base no topo da hierarquia seja chamado
      - Depois do destrutor da classe base no topo, o objeto é removido da memória

# Construtores e Destrutores em Classes Derivadas

- Construtores, destrutores e operadores sobrecarregados de atribuição definidos na classe base
  - Não são herdados pelas classes derivadas!

# Construtores e Destrutores em Classes Derivadas

- Criação de um objeto de uma classe derivada em que tanto a classe base quanto a classe derivada contenham objetos de outras classes
  - Construtores para os objetos-membro da classe base executam primeiro, em seguida o construtor da classe base, os construtores para os objetos-membro da classe derivada e o construtor da classe derivada executam
  - Os destrutores dos objetos da classe derivada são chamados na ordem inversa de seus construtores correspondentes



```
/*
 * Aula 12 - Exemplo 6
 * Arquivo comissionCap12Ex6.h
 * Autor: Miguel Campista
 */
#ifndef COMMISSION_H
#define COMMISSION_H

#include <iostream>
#include <string>
#include <iomanip>

using namespace std;

class CommissionEmployee {
public:
    CommissionEmployee (const string &, const string &,
                       const string &, double = 0.0, double = 0.0);
    ~CommissionEmployee (); // destrutor

    void setFirstName (const string &); // Configura o nome
    string getFirstName () const; // Retorna o nome

    void setLastName (const string &);
    string getLastName () const;

    void setSocialSecurityNumber (const string &);
    string getSocialSecurityNumber () const;

    void setGrossSales (double); // Configura a quant. de vendas brutas
    double getGrossSales () const; // Retorna a quant. de vendas brutas

    void setCommissionRate (double); // Conf. taxa de comissão
    double getCommissionRate () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
```

# Sexto Exemplo de Herança em C++

```
private:  
    string firstName, lastName;  
    string socialSecurityNumber;  
    double grossSales;  
    double commissionRate;  
};  
  
#endif
```

```

/*
 * Aula 12 - Exemplo 6
 * Arquivo comissionCap12Ex6.cpp
 * Autor: Miguel Campista
 */
#include "commissionCap12Ex6.h"

CommissionEmployee::CommissionEmployee (const string &first, const string &last,
    const string &ssn, double sales, double rate) :
    firstName (first), lastName (last), socialSecurityNumber (ssn) {
    setGrossSales (sales);
    setCommissionRate (rate);

    cout << "Construtor Commission Employee: " << endl;
    print ();
    cout << "\n\n";
}

CommissionEmployee::~~CommissionEmployee () {
    cout << "Destructor Commission Employee: " << endl;
    print ();
    cout << "\n\n";
}

void CommissionEmployee::setFirstName (const string &first) {
    firstName = first;
}

string CommissionEmployee::getFirstName () const { return firstName; }

void CommissionEmployee::setLastName (const string &last) {
    lastName = last;
}

string CommissionEmployee::getLastName () const { return lastName; }

```

# Sexto Exemplo de Herança em C++

```
void CommissionEmployee::setSocialSecurityNumber (const string &ssn) {
    socialSecurityNumber = ssn;
}

string CommissionEmployee::getSocialSecurityNumber () const {
    return socialSecurityNumber;
}

void CommissionEmployee::setGrossSales (double sales) {
    grossSales = (sales < 0.0) ? 0.0 : sales;
}

double CommissionEmployee::getGrossSales () const { return grossSales; }

void CommissionEmployee::setCommissionRate (double rate) {
    commissionRate = (rate > 0.0 && rate < 1.0) ? rate : 0.0;
}

double CommissionEmployee::getCommissionRate () const { return commissionRate; }

double CommissionEmployee::earnings () const {
    return getGrossSales () * getCommissionRate ();
}

void CommissionEmployee::print () const {
    cout << "commission employee: " << getFirstName ()
         << ' ' << getLastName ()
         << "\nsocial security number: " << getSocialSecurityNumber ()
         << "\ngross sales: " << getGrossSales ()
         << "\ncommission rate: " << getCommissionRate ();
}
```

# Sexto Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 6
 * Arquivo basepluscommissionCap12Ex6.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <iomanip>
#include "commissionCap12Ex6.h"

using namespace std;

class BasePlusCommissionEmployee : public CommissionEmployee {
public:
    BasePlusCommissionEmployee (const string &, const string &,
                                const string &, double = 0.0, double = 0.0, double = 0.0);
    ~BasePlusCommissionEmployee (); // destrutor

    void setBaseSalary (double); // Conf. o salário base
    double getBaseSalary () const;

    double earnings () const; // Calcula os rendimentos
    void print () const;
private:
    double baseSalary;
};

#endif
```

# Sexto Exemplo de Herança em C++

```
/*  
 * Aula 12 - Exemplo 6  
 * Arquivo basepluscomissionCap12Ex6.cpp  
 * Autor: Miguel Campista  
 */  
#include "basepluscomissionCap12Ex6.h"  
  
BasePlusCommissionEmployee::BasePlusCommissionEmployee (const string &first,  
    const string &last, const string &ssn,  
    double sales, double rate, double salary) :  
    CommissionEmployee (first, last, ssn, sales, rate) {  
    setBaseSalary (salary);  
  
    cout << "Construtor Base Plus Commission Employee: " << endl;  
    print ();  
    cout << "\n\n";  
}  
  
BasePlusCommissionEmployee::~~BasePlusCommissionEmployee () {  
    cout << "Destructor Base Plus Commission Employee: " << endl;  
    print ();  
    cout << "\n\n";  
}  
  
void BasePlusCommissionEmployee::setBaseSalary (double salary) {  
    baseSalary = (salary < 0.0) ? 0.0 : salary;  
}  
  
double BasePlusCommissionEmployee::getBaseSalary () const { return baseSalary; }  
  
double BasePlusCommissionEmployee::earnings () const {  
    return baseSalary + CommissionEmployee::earnings ();  
}
```

# Sexto Exemplo de Herança em C++

```
void BasePlusCommissionEmployee::print () const {
    cout << "Base salary" << endl;

    CommissionEmployee::print ();

    cout << "\nbase salary: " << baseSalary;
}
```

# Sexto Exemplo de Herança em C++

```
/*
 * Aula 12 - Exemplo 6
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "basepluscommissionCap12Ex6.h"

int main () {
    // Configura a formatação de saída de ponto flutuante
    cout << fixed << setprecision (2);

    { // Inicia um escopo
        CommissionEmployee employee1 ("Bob", "Lewis",
                                       "21-3333-3333", 5000, .04);
    }

    cout << endl;
    BasePlusCommissionEmployee employee2 ("Lisa", "Jones",
                                           "21-5555-5555", 2000, .06, 800);

    cout << endl;
    BasePlusCommissionEmployee employee3 ("Mark", "Sands",
                                           "21-8888-8888", 8000, .15, 2000);

    // Configura a formatação de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << endl;

    return 0;
}
```



C:\Windows\system32\cmd.exe - aula12-ex6.exe

C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos>aula12-ex6.exe

Construtor Commission Employee:  
commission employee: Bob Lewis  
social security number: 21-3333-3333  
gross sales: 5000.00  
commission rate: 0.04

Destrutor Commission Employee:  
commission employee: Bob Lewis  
social security number: 21-3333-3333  
gross sales: 5000.00  
commission rate: 0.04

Construtor Commission Employee:  
commission employee: Lisa Jones  
social security number: 21-5555-5555  
gross sales: 2000.00  
commission rate: 0.06

Construtor Base Plus Commission Employee:  
Base salary  
commission employee: Lisa Jones  
social security number: 21-5555-5555  
gross sales: 2000.00  
commission rate: 0.06  
base salary: 800.00

Construtor Commission Employee:  
commission employee: Mark Sands  
social security number: 21-8888-8888  
gross sales: 8000.00  
commission rate: 0.15

Construtor Base Plus Commission Employee:  
Base salary  
commission employee: Mark Sands  
social security number: 21-8888-8888  
gross sales: 8000.00  
commission rate: 0.15  
base salary: 2000.00

Pressione qualquer tecla para continuar. . .

# Sexto Exemplo de Herança em C++

```
/*  
 * Aula 12 - Exemplo 6  
 * Programa Principal
```

```
C:\Windows\system32\cmd.exe  
Pressione qualquer tecla para continuar. . .  
Destructor Base Plus Commission Employee:  
Base salary  
commission employee: Mark Sands  
social security number: 21-8888-8888  
gross sales: 8000.00  
commission rate: 0.15  
base salary: 2000.00  
  
Destructor Commission Employee:  
commission employee: Mark Sands  
social security number: 21-8888-8888  
gross sales: 8000.00  
commission rate: 0.15  
  
Destructor Base Plus Commission Employee:  
Base salary  
commission employee: Lisa Jones  
social security number: 21-5555-5555  
gross sales: 2000.00  
commission rate: 0.06  
base salary: 800.00  
  
Destructor Commission Employee:  
commission employee: Lisa Jones  
social security number: 21-5555-5555  
gross sales: 2000.00  
commission rate: 0.06  
  
C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos>
```

```
return 0;
```

```
}
```

# Herança `public`, `protected` e `private`

- Herança `public`
  - Membros `public` da classe base
    - Se tornam membros `public` da classe derivada
  - Membros `protected` da classe base
    - Se tornam membros `protected` da classe derivada
  - Membros `private` da classe base
    - Não podem ser acessados

# Herança `public`, `protected` e `private`

- Herança `protected` (não é um relacionamento "é um")
  - Membros `public` e `protected` da classe base
    - Se tornam membros `protected` da classe derivada
- Herança `private` (não é um relacionamento "é um")
  - Membros `public` e `protected` da classe base
    - Se tornam membros `private` da classe derivada

# Exemplo 1

- Escreva um programa que implemente a classe Cadastro que possui nome e idade como atributo e oferece como métodos públicos funções do tipo "get" para obter os valores desses atributos. Implemente ainda a classe PubCadastro que herda os métodos e atributos da classe Cadastro e ainda adiciona o atributo trabalho e um método público para acessar o novo atributo.



# Exemplo 1

```
/*
 * Aula 12 - Exemplo 7
 * Arquivo cadastroCap12Ex7.h
 * Autor: Miguel Campista
 */
#ifndef CADASTRO_H
#define CADASTRO_H

#include <iostream>
#include <string>

using namespace std;

class Cadastro {
public:
    Cadastro (string, int);
    string getName () const;
    int getAge () const;

private:
    string name;
    int age;
};

#endif
```

# Exemplo 1

```
/*  
 * Aula 12 - Exemplo 7  
 * Arquivo cadastroCap12Ex7.cpp  
 * Autor: Miguel Campista  
 */  
#include "cadastroCap12Ex7.h"  
  
Cadastro::Cadastro (string n, int a) : name (n), age (a) {}  
  
string Cadastro::getName () const { return name; }  
  
int Cadastro::getAge () const { return age; }
```

# Exemplo 1

```
/*
 * Aula 12 - Exemplo 7
 * Arquivo pubcadastroCap12Ex7.h
 * Autor: Miguel Campista
 */
#ifndef PUBCADASTRO_H
#define PUBCADASTRO_H

#include <iostream>
#include <string>
#include "cadastroCap12Ex7.h"

using namespace std;

class PubCadastro: public Cadastro {
public:
    PubCadastro (string, int, string);
    string getJob () const;
private:
    string job;
};
#endif
```



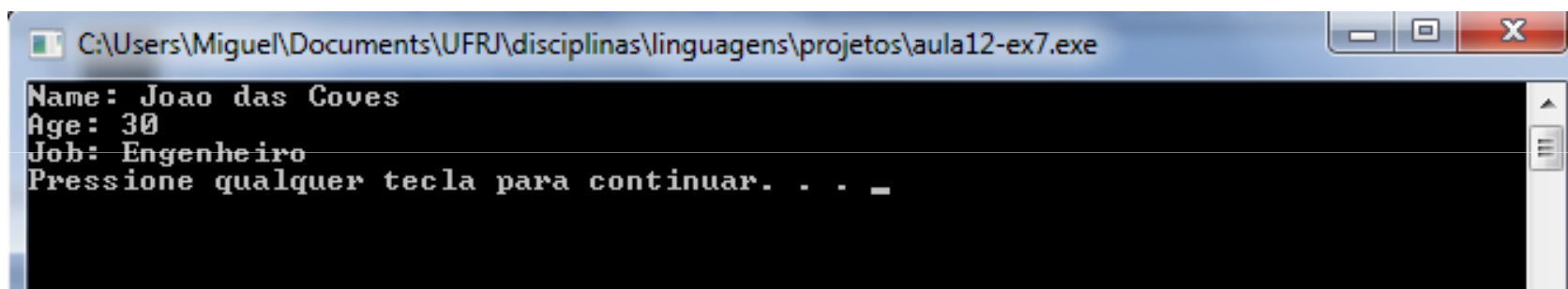
# Exemplo 1

```
/*  
 * Aula 12 - Exemplo 7  
 * Arquivo pubcadastroCap12Ex7.cpp  
 * Autor: Miguel Campista  
 */  
#include "pubcadastroCap12Ex7.h"  
  
PubCadastro::PubCadastro (string n, int a, string j):  
    Cadastro (n, a), job (j) {}  
  
string PubCadastro::getJob () const { return job; }
```

# Exemplo 1

```
/*  
 * Aula 12 - Exemplo 7  
 * Programa Principal  
 * Autor: Miguel Campista  
 */  
#include "pubcadastroCap12Ex7.h"  
  
int main() {  
    PubCadastro cad1 ("Joao das Coves", 30, "Engenheiro");  
    cout << "Name: " << cad1.getName ()  
        << "\nAge: " << cad1.getAge ()  
        << "\nJob: " << cad1.getJob () << endl;  
  
    return 0;  
}
```

# Exemplo 1



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos\aula12-ex7.exe. The window contains the following text:

```
Name: Joao das Goves  
Age: 30  
Job: Engenheiro  
Pressione qualquer tecla para continuar. . . _
```

# Exemplo 1

- E se fosse criada uma classe que herdasse em modo `private`



# Exemplo 1

```
/*
 * Aula 12 - Exemplo 7
 * Arquivo privcadastroCap12Ex7.h
 * Autor: Miguel Campista
 */
#ifndef PRIVCADASTRO_H
#define PRIVCADASTRO_H

#include <iostream>
#include <string>
#include "cadastroCap12Ex7.h"

using namespace std;

class PrivCadastro: private Cadastro {
public:
    PrivCadastro (string, int, int);
    int getId () const;
private:
    int id;
};
#endif
```

# Exemplo 1

```
/*  
 * Aula 12 - Exemplo 7  
 * Arquivo privcadastroCap12Ex7.cpp  
 * Autor: Miguel Campista  
 */  
#include "privcadastroCap12Ex7.h"  
  
PrivCadastro::PrivCadastro (string n, int a, int i):  
    Cadastro (n, a), id (i) {}  
  
string PrivCadastro::getId () const { return id; }
```

# Exemplo 1

```
/*  
 * Aula 12 - Exemplo 7  
 * Programa Principal  
 * Autor: Miguel Campista  
 */  
#include "pubcadastroCap12Ex7.h"  
#include "privcadastroCap12Ex7.h"  
  
int main() {  
    PubCadastro cad1 ("Joao das Coves", 30, "Engenheiro");  
    cout << "Name: " << cad1.getName ()  
         << "\nAge: " << cad1.getAge ()  
         << "\nJob: " << cad1.getJob () << endl;  
  
    PrivCadastro cad2 ("Jose do Bosque", 40, 2);  
    cout << "Name: " << cad2.getName ()  
         << "\nAge: " << cad2.getAge ()  
         << "\nId: " << cad2.getId () << endl;  
  
    return 0;  
}
```

# Exemplo 1

```
/*  
 * Aula 12 - Exemplo 7  
 * Programa Principal  
 * Autor: Miguel Campista  
 */  
#include "pubcadastroCap12Ex7.h"  
#include "privcadastroCap12Ex7.h"
```

Line	File	Message
	Z:\disciplinas\linguagens\projetos\ca...	In function 'int main()':
18	Z:\disciplinas\linguagens\projetos\ca...	'std::string Cadastro::getName() const' is inaccessible
16	Z:\disciplinas\linguagens\projetos\au...	within this context
16	Z:\disciplinas\linguagens\projetos\au...	'Cadastro' is not an accessible base of 'PrivCadastro'
19	Z:\disciplinas\linguagens\projetos\ca...	'int Cadastro::getAge() const' is inaccessible
17	Z:\disciplinas\linguagens\projetos\au...	within this context
17	Z:\disciplinas\linguagens\projetos\au...	'Cadastro' is not an accessible base of 'PrivCadastro'
	Z:\disciplinas\linguagens\projetos\M...	[Build Error] [aula12-ex7.o] Error 1

```
<< "\nId: " << cadz.getId () << endl;
```

```
return 0;
```

```
}
```



# Exemplo 2

- Escreva um programa que implemente a classe Cadastro que possui nome e idade como atributo e oferece como métodos públicos funções do tipo "get" para obter os valores desses atributos. Implemente ainda a classe Senha que possui o atributo senha e uma função "get". Por fim, implemente a classe PubCadastro que herda os métodos e atributos da classe Cadastro e Senha e ainda adiciona o atributo trabalho e um método público para acessar o novo atributo.



# Exemplo 2

```
/*  
 * Aula 12 - Exemplo 7  
 * Arquivo senhaCap12Ex7.h  
 * Autor: Miguel Campista  
 */  
#ifndef SENHA_H  
#define SENHA_H  
  
#include <iostream>  
#include <string>  
  
using namespace std;  
  
class Senha {  
    public:  
        Senha (string);  
        string getSenha () const;  
  
    private:  
        string senha;  
};  
  
#endif
```

# Exemplo 2

```
/*  
 * Aula 12 - Exemplo 7  
 * Arquivo senhaCap12Ex7.cpp  
 * Autor: Miguel Campista  
 */  
#include "senhaCap12Ex7.h"  
  
Senha::Senha (string s) : senha (s) {}  
  
string Senha::getSenha () const { return senha; }
```

# Exemplo 2

```
/*
 * Aula 12 - Exemplo 7
 * Arquivo herdapublicacadaastroCap12Ex7.h
 * Autor: Miguel Campista
 */
#ifndef PUBCADASTRO_H
#define PUBCADASTRO_H

#include <iostream>
#include <string>
#include "cadaastroCap12Ex7.h"
#include "senhaCap12Ex7.h"

using namespace std;

class PubCadastro : public Cadastro, public Senha {
public:
    PubCadastro (string, int, string, string);
    string getJob () const;
private:
    string job;
};

#endif
```

# Exemplo 2

```
/*  
 * Aula 12 - Exemplo 7  
 * Arquivo herdapublicacadastroCap12Ex7.cpp  
 * Autor: Miguel Campista  
 */  
#include "herdapublicacadastroCap12Ex7.h"  
  
PubCadastro::PubCadastro (string n, int a, string s, string j)  
                        : Cadastro (n, a), Senha (s), job (j) {  
}  
  
string PubCadastro::getJob () const { return job; }
```

# Exemplo 2

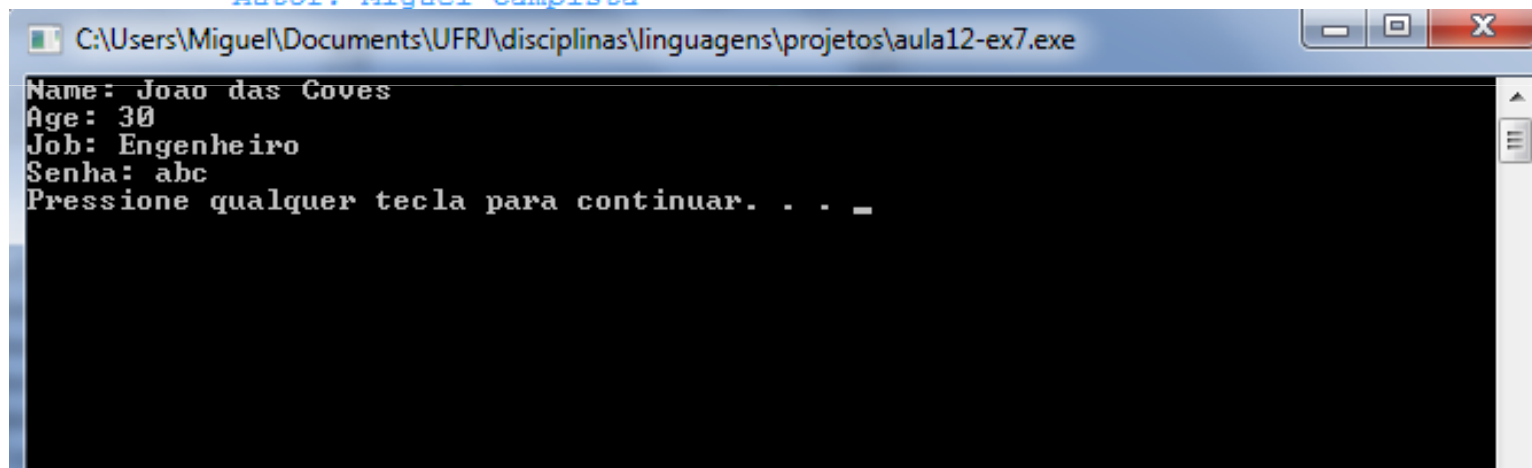
```
/*
 * Aula 12 - Exemplo 7
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "herdapublicacadastroCap12Ex7.h"

int main () {
    PubCadastro cad1 ("Joao das Covas", 30, "abc", "Engenheiro");
    cout << "Name: " << cad1.getName ()
         << "\nAge: " << cad1.getAge ()
         << "\nJob: " << cad1.getJob ()
         << "\nSenha: " << cad1.getSenha () << endl;

    return 0;
}
```

# Exemplo 2

```
/*  
 * Aula 12 - Exemplo 7  
 * Programa Principal  
 * Autor: Miguel Campista
```



The screenshot shows a Windows command prompt window titled "C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos\aula12-ex7.exe". The window contains the following text:

```
Name: Joao das Goves  
Age: 30  
Job: Engenheiro  
Senha: abc  
Pressione qualquer tecla para continuar. . . _
```

# Leitura Recomendada

- Capítulos 12 do livro
  - Deitel, "*C++ How to Program*", 5th edition, Editora Prentice Hall, 2005