

# Linguagens de Programação

Prof. Miguel Elias Mitre Campista

<http://www.gta.ufrj.br/~miguel>

## Parte II

Introdução à Programação em C++  
(Continuação)

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Relembrando da Última Aula...

- Sobrecarga de operadores
- Mais exemplos de programação orientada a objetos...

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

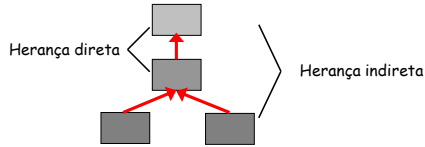
Prof. Miguel Campista

## 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

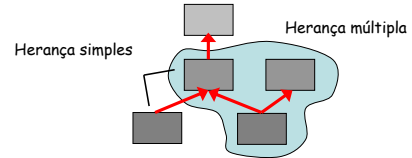


Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Hierarquia de Classes

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



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Três Tipos de Herança

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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## "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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Recomendações de Engenharia de Software

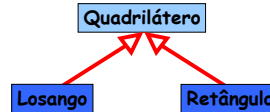
- Funções-membro de uma classe derivada
  - Não podem acessar diretamente os membros `private` da classe base
- Se uma classe derivada pudesse acessar os membros `private` de sua classe base
  - As classes que herdam dessa classe derivada também poderiam acessar os dados da classe base
    - Isso propagaria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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Classes Base e Derivadas

Classe básica	Classes derivadas
Aluno	AlunoDeGraduação, AlunoDePósGraduação
Forma	Círculo, Triângulo, Retângulo, Esfera, Cubo
Financiamento	FinanciamentoDeCarro, FinanciamentoDeReformaDeCasa, FinanciamentoDeCasaPrópria
Empregado	CorpoDocente, Funcionários
Conta	ContaCorrente, ContaPoupança

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

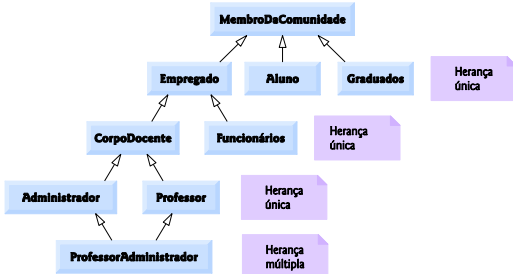
## Classes Base e Derivadas

- 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Classes Base e Derivadas



Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista



## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 12 - Exemplo 1
 * Arquivo CommissionCap12Ex1.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

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 12 - Exemplo 1
 * Arquivo CommissionCap12Ex1.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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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"
         << "firstName: " << employee.getFirstName ()
         << "\nlastName: " << employee.getLastName ()
         << "\nsocial security number: "
         << employee.getSocialSecurityNumber ()
         << "\ngross sales: " << employee.getGrossSales ()
         << "\ncommission rate: " << employee.getCommissionRate () << endl;

    employee.setGrossSales (8500);
    employee.setCommissionRate (1.1);

    cout << "Updated 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 BasePlus1
// Autor: Miguel Campista

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

using namespace std;

class CommissionEmployee {
public:
    CommissionEmployee(const string & f, const string & l,
                       const string & ssn, const double & s,
                       const double & r) : firstName(f), lastName(l),
                                           socialSecurityNumber(ssn),
                                           grossSales(s), commissionRate(r) {}

    void setFirstName(const string & f) { firstName = f; }
    void setLastName(const string & l) { lastName = l; }
    void setSocialSecurityNumber(const string & ssn) {
        socialSecurityNumber = ssn;
    }
    void setGrossSales(const double & s) { grossSales = s; }
    void setCommissionRate(const double & r) { commissionRate = r; }

    const string & getFirstName() const { return firstName; }
    const string & getLastName() const { return lastName; }
    const string & getSocialSecurityNumber() const { return socialSecurityNumber; }
    double getGrossSales() const { return grossSales; }
    double getCommissionRate() const { return commissionRate; }

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

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

int main() {
    CommissionEmployee emp("Sue Jones", "21-2222-2222", 5000.00, 0.06);
    cout << "Employee information obtained by get functions:\n";
    emp.getFirstName() << " ";
    emp.getLastName() << " ";
    emp.getSocialSecurityNumber() << " ";
    emp.getGrossSales() << " ";
    emp.getCommissionRate() << " ";
    cout << "\n";

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

    cout << "Employee's earnings: " << emp.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 commissionCap12Ex2.h
// Autor: Miguel Campista

#ifndef COMMISSIONCAP12EX2_H
#define COMMISSIONCAP12EX2_H

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

using namespace std;

class BasePlusCommissionEmployee {
public:
    BasePlusCommissionEmployee(const string & f, const string & l,
                               const string & ssn, const double & s,
                               const double & r, const double & bs) :
        CommissionEmployee(f, l, ssn, s, r), baseSalary(bs) {}

    void setFirstName(const string & f) { // Configura o nome
        firstName = f;
    }
    void setLastName(const string & l) {
        lastName = l;
    }
    void setSocialSecurityNumber(const string & ssn) {
        socialSecurityNumber = ssn;
    }
    void setGrossSales(const double & s) { // Configura a quant. de vendas brutas
        grossSales = s;
    }
    void setCommissionRate(const double & r) { // Conf. taxa de comissão
        commissionRate = r;
    }
    void setBaseSalary(const double & bs) { // Conf. o salário base
        baseSalary = bs;
    }

    double earnings() const { // Calcula os rendimentos
        return CommissionEmployee::earnings() + baseSalary;
    }

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

```

## Segundo Exemplo de Herança em C++

```

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

```

```

//
// Aula 12 - Exemplo 2
// Arquivo commissionCap12Ex2.cpp
// Autor: Miguel Campista

#include "commissionCap12Ex2.h"

BasePlusCommissionEmployee::BasePlusCommissionEmployee(const string & f,
                                                         const string & l, const string & ssn,
                                                         const double & s, const double & r,
                                                         const double & bs) :
    CommissionEmployee(f, l, ssn, s, r), baseSalary(bs) {}

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

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

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

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 ("Doe", "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"
    << "\tfirst name: " << employee.getFirstName ()
    << "\tlast name: " << employee.getLastName ()
    << "\tsocial security number: "
    << employee.getSocialSecurityNumber ()
    << "\tgross sales: " << employee.getGrossSales ()
    << "\ncommission rate: " << employee.getCommissionRate ()
    << "\nbase salary: " << employee.getBaseSalary () << endl;

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

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

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

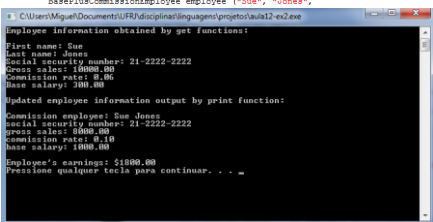
    return 0;
}
```

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

int main () {
    BasePlusCommissionEmployee employee ("Doe", "Jones",
    "21-2222-2222", 10000.06, 300);

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

    return 0;
}
```



Employee information obtained by get functions:  
 first name: Doe  
 last name: Jones  
 social security number: 21-2222-2222  
 gross sales: 10000.06  
 commission rate: 0.06  
 base salary: 300.00

Updated employee information output by print function:  
 Commission employee: Doe 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. . .

## 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 comissionCap12Ex3.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <iomanip>
#include "comissionCap12Ex1.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 salario 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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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; }

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

```

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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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);
}

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;
}

```

**Tenta acessar dados privados da classe base diretamente...**

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Terceiro Exemplo de Herança em C++

```

/*
 * Aula 12 - Exemplo 3
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "comissionCap12Ex3.h"

int main () {
    BasePlusCommissionEmployee emp ("Doe", "Jones",
        "21-2222-2222", 10000, 0.06, 300);

    // Configura o formato de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << "\nEmployee information obtained by get functions:\n";
    << "first name: " << emp.getFirstName ()
    << "\nlast name: " << emp.getLastName ()
    << "\nsocial security number: "
    << emp.getSocialSecurityNumber ()
    << "\ngross sales: " << emp.getGrossSales ()
    << "\ncommission rate: " << emp.getCommissionRate ()
    << "\nbase salary: " << emp.getBaseSalary () << endl;

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

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

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

    return 0;
}

```

## Terceiro Exemplo de Herança em C++

```

/*
 * Aula 12 - Exemplo 3
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "comissionCap12Ex3.h"

int main () {
    BasePlusCommissionEmployee emp ("Doe", "Jones",
        "21-2222-2222", 10000, 0.06, 300);

    // Configura o formato de saída de ponto flutuante
    cout << fixed << setprecision (2);

    cout << "\nEmployee information obtained by get functions:\n";
    << "first name: " << emp.getFirstName ()
    << "\nlast name: " << emp.getLastName ()
    << "\nsocial security number: "
    << emp.getSocialSecurityNumber ()
    << "\ngross sales: " << emp.getGrossSales ()
    << "\ncommission rate: " << emp.getCommissionRate ()
    << "\nbase salary: " << emp.getBaseSalary () << endl;

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

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

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

    return 0;
}

```



## Erro de Compilação

- Construtor da classe derivada chamar construtores de classes base com argumentos inconsistentes
  - Número ou tipo de parâmetros errados dos especificados nas definições dos construtores das classes base
- Em um construtor de classe derivada, inicializar os objetos-membro e invocar construtores de classes base explicitamente na lista de inicializadores de membro impede a inicialização duplicada de um construtor-padrão da classe base
  - Caso o construtor-padrão não exista, há erro de compilação

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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`

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```
/*
 * Aula 12 - Exemplo 4
 * Arquivo commissionCap12Ex4.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
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```
/*
 * Aula 12 - Exemplo 4
 * Arquivo commissionCap12Ex4.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;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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 <string>
#include "CommissionCap12Ex4.h"

using namespace std;

class BasePlusCommissionEmployee : public CommissionEmployee {
public:
    BasePlusCommissionEmployee (const string & f, const string & l,
        const string & ssn, 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

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quarto Exemplo de Herança em C++

```

/*
 * Aula 12 - Exemplo 4
 * Arquivo basepluscommissionCap12Ex4.cpp
 * Autor: Miguel Campista
 */
#include "basepluscommissionCap12Ex4.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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```

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

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",
        "21-2222-2222", 10000.00, .04, 300);

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

    cout << "Employee information obtained by get functions:\n";
    cout << "\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";
    cout << endl;
    employee.print ();

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

    return 0;
}

```

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```

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

int main () {
    BasePlusCommissionEmployee employee ("Sue", "Jones",
        "21-2222-2222", 10000.00, .04, 300);

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

    cout << "Employee information obtained by get functions:\n";
    cout << "\nfirst name: Sue\nlast name: Jones\nsocial security number: 21-2222-2222\nGross sales: 10000.00\nCommission rate: 0.04\nBase salary: 300.00\n";

    cout << "Updated employee information output by print function:\n";
    cout << "Commission employee: Sue Jones\nsocial security number: 21-2222-2222\nGross sales: 8000.00\nCommission rate: 0.10\nBase salary: 1000.00\n";

    cout << "Employee's earnings: 5100.00\n";

    return 0;
}

```

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```
/*
 * Aula 12 - Exemplo 6
 * Arquivo comissaoCap12Ex6.h
 * Autor: Miguel Campista
 */
#ifndef COMMISSION_H
#define COMMISSION_H

#include <iostream>
#include <string>
#include <omanip>

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
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```

/*
 * Aula 12 - Exemplo 8
 * Arquivo comissionCap12Ex8.cpp
 * Autor: Miguel Campista
 */
#include "comissionCap12Ex8.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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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 ();
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo de Herança em C++

```

/*
 * Aula 12 - Exemplo 9
 * Arquivo basePlusComissionCap12Ex8.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <iomanip>
#include "comissionCap12Ex8.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

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Quinto Exemplo de Herança em C++

```

/*
 * Aula 12 - Exemplo 8
 * Arquivo basePlusComissionCap12Ex8.cpp
 * Autor: Miguel Campista
 */
#include "basePlusComissionCap12Ex8.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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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...
  - Ocultar 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 ();
    ...
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Hierarquia de Herança Utilizando Dados private

- Incluir uma função-membro da classe base com uma assinatura diferente na classe derivada...
  - Ocultar 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!

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

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

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

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

    cout << "Employee information obtained by get functions:\n";
    cout << "First 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";
    cout << endl;
    employee.print ();

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

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

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

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

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

    cout << "Employee information obtained by get functions:\n";
    cout << "First name: Sue\nLast name: Jones\nSocial security number: 21-2222-2222\nGross sales: 8000.00\nCommission rate: 0.10\nBase salary: 1000.00\n";

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

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

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Construtores e Destrutores em Classes Derivadas

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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```
/*
 * 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
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

```

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

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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 : 0.02;
}

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 ();
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo de Herança em C++

```

/*
 * Aula 12 - Exemplo 6
 * Arquivo BasePlusComissionCap12Ex6.h
 * Autor: Miguel Campista
 */
#ifndef BASEPLUS_H
#define BASEPLUS_H

#include <iostream>
#include <string>
#include <omanip>
#include "comissionCap12Ex6.h"

using namespace std;

class BasePlusCommissionEmployee : public CommissionEmployee {
public:
    BasePlusCommissionEmployee (const string &t, const string &f,
const string &ssn, double s, double r, double b) :
        -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;
}

```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Sexto Exemplo de Herança em C++

```

/*
 * Aula 12 - Exemplo 6
 * Programa Principal
 * Autor: Miguel Campista
 */
#include "basepluscomissionCap12Ex6.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:\Users\Miguel> cd C:\Documents\NFRF\disciplinas\linguagens\projetos\aula12-ex6.exe
Constructor Commission Employee:
commission employee: Bob Lewis
social security number: 21-3333-3333
gross sales: 5000.00
commission rate: 0.04
base salary:

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

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

Constructor 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: Mark Sands
social security number: 21-8888-8888
gross sales: 8000.00
commission rate: 0.15
base salary:

Constructor 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
base salary: 2000.00

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
base salary: 800.00

C:\Users\Miguel> cd C:\Documents\NFRF\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**

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

### 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

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

### 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.

?

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista

### 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 getNome () const;
        int getAge () const;

private:
    string name;
    int age;
};

#endif

```

Linguagens de Programação – DEL-Poli/UFRJ Prof. Miguel Campista



## 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; }
```

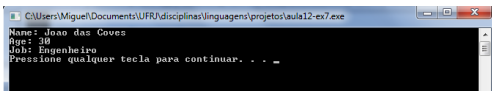
## Exemplo 1

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

int main() {
    PubCadastro cadi ("Joao das Covas", 30, "Engenheiro");
    cout << "Nome: " << cadi.getName ()
         << "idade: " << cadi.getAge ()
         << "job: " << cadi.getJob () << endl;

    return 0;
}
```

## Exemplo 1



```
C:\Users\Miguel\Documents\UFRJ\disciplinas\linguagens\projetos\aula12-ex7.exe
Name: Joao das Covas
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):
        Cadastro (string, int, int) {}
    int getId () const;
private:
    int id;
};
#endif
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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, i) {}

string PrivCadastro::getId () const { return id; }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Exemplo 1

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

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

    PrivCadastro cad2 ("José do Bonque", 40, 2);
    cout << "Name: " << cad2.getName ()
         << "\nAge: " << cad2.getAge ()
         << "\nId: " << cad2.getId () << endl;

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Exemplo 1

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

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

<< "\nId: " << cad2.getId () << endl;

return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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.

?

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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):
        getSenha () const;
private:
    string senha;
};

#endif
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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; }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Exemplo 2

```
/*
 * Aula 12 - Exemplo 7
 * Arquivo herdapublicacadastroCap12Ex7.h
 * Autor: Miguel Campista
 */
#ifndef PUBCADASTRO_H
#define PUBCADASTRO_H

#include <iostream>
#include <string>
#include "CadastroCap12Ex7.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
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## 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; }
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Exemplo 2

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

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

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Exemplo 2

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

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

    return 0;
}
```

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista

## Leitura Recomendada

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

Linguagens de Programação – DEL-Poli/UFRJ

Prof. Miguel Campista