



PCS3111

**Laboratório de Programação
Orientada a Objetos para
Engenharia Elétrica**

**Aula 8: Classe Abstrata e
Herança Múltipla**

Escola Politécnica da Universidade de São Paulo

Agenda

1. Classe abstrata
2. Herança múltipla
3. Atributos e métodos estáticos

Abstração

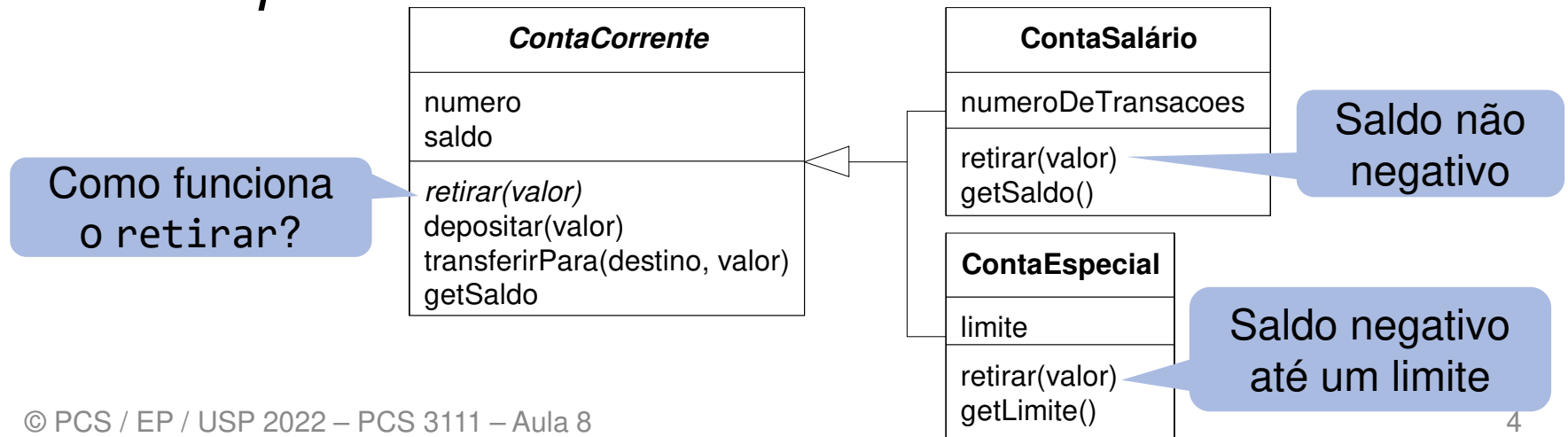
- Em uma hierarquia, às vezes é útil representar **conceitos abstratos**
 - Representa uma **abstração** de um ou mais conceitos
 - Não existe uma **instância** deste conceito
- *Exemplo:* ordem *Carnivora* da biologia



- Não existe um mamífero que seja dessa ordem mas que não pertença a uma de suas espécies

Classe abstrata

- Não pode ser **instanciada**
 - Serve de base para a criação de outras classes
 - Permite aproveitar os benefícios da herança
- *Pode* não estar **completamente implementada**
 - Define o comportamento, mas não sua implementação
 - **Métodos abstratos**
- *Exemplo*

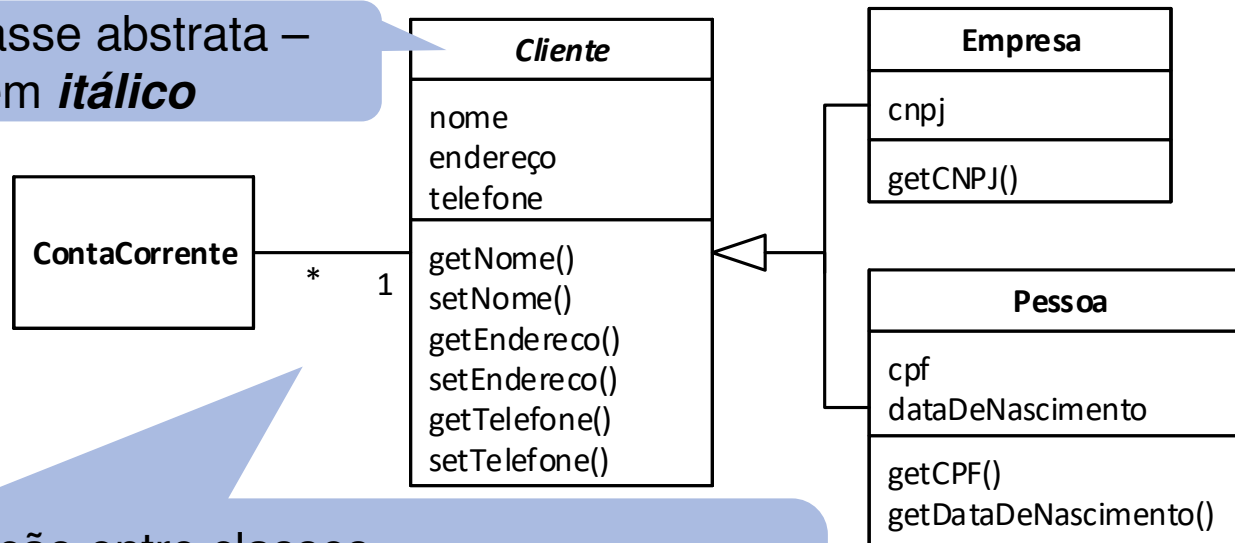


Alguns usos

1. Para ser uma **base** para herança

- Comportamento comum
- *Conceitualmente*, uma classe abstrata pode não ter métodos abstratos

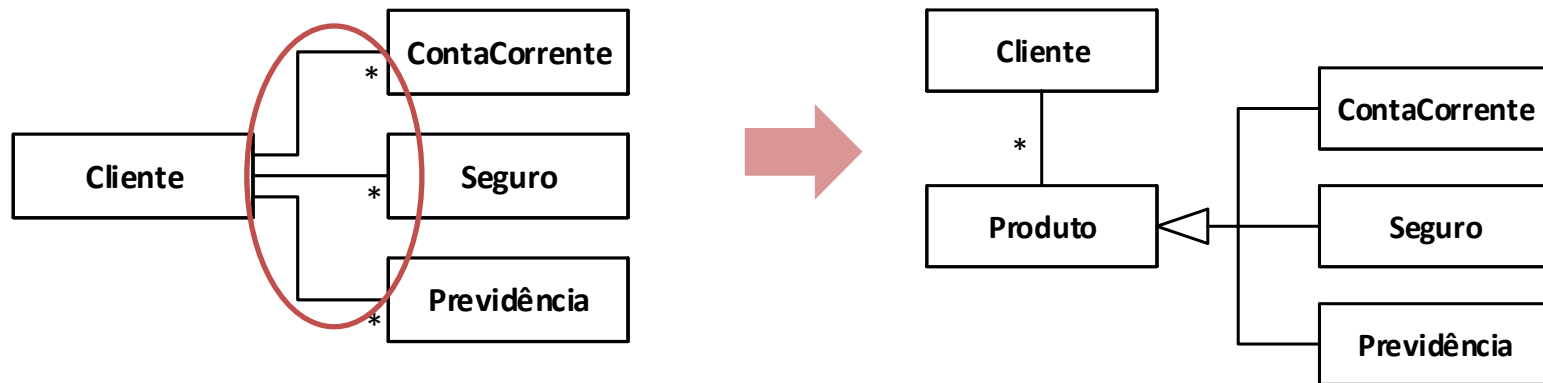
Cliente é a classe abstrata –
identificador em *itálico*



Associação entre classes
Um Cliente pode ter várias ContaCorrente (*)
Uma ContaCorrente é de um só Cliente (1)

Alguns usos

2. Para simplificar o relacionamento com classes que **tenham em comum** algum conceito geral



```
class Cliente {
public: ...
protected:
    ContaCorrente* contas[MAX];
    Seguro* seguros[MAX];
    Previdencia* previdencias[MAX];
};
```

```
class Cliente {
public: ...
protected:
    Produto* produtos[MAX];
};
```

Classe abstrata em C++

- Não há uma declaração **direta** de classe abstrata
 - Uma classe é abstrata quando **pelo menos um de seus métodos é abstrato**
 - C++: método ***puramente virtual***
 - Método definido como `virtual` e com “= 0” no final da declaração

```
...
4  class ContaCorrente {
5  public:
6      ContaCorrente (int numero);
7      virtual ~ContaCorrente();
8      virtual bool retirar (double valor) = 0;
9      virtual void depositar (double valor);
...
16 private:
17     int numero;
18     };
```

EX01

Método abstrato

Método concreto com
ligação dinâmica

Classe abstrata em C++

- Classe abstrata é uma classe normal
 - *Pode* ter uma superclasse
 - *Pode* ter atributos e métodos concretos
 - *Pode* ter um construtor

O método abstrato não deve ser colocado na implementação da classe abstrata

- C++ **não permite** criar uma classe abstrata em que todos os métodos são concretos
 - Mas é possível que ela tenha apenas métodos abstratos

Classe abstrata em C++

- Uma subclasse precisa implementar **todos** os métodos abstratos
 - Ou *ela também será abstrata*
 - Para implementar um método abstrato, ele deve ser definido na subclasse

...

```
6  class ContaEspecial : public ContaCorrente {
7  public:
8      ContaEspecial (int numero, double limite);
9      virtual ~ContaEspecial();
10     bool retirar (double valor);
11     virtual double getLimite();
12 private:
13     double limite;
14 };
```

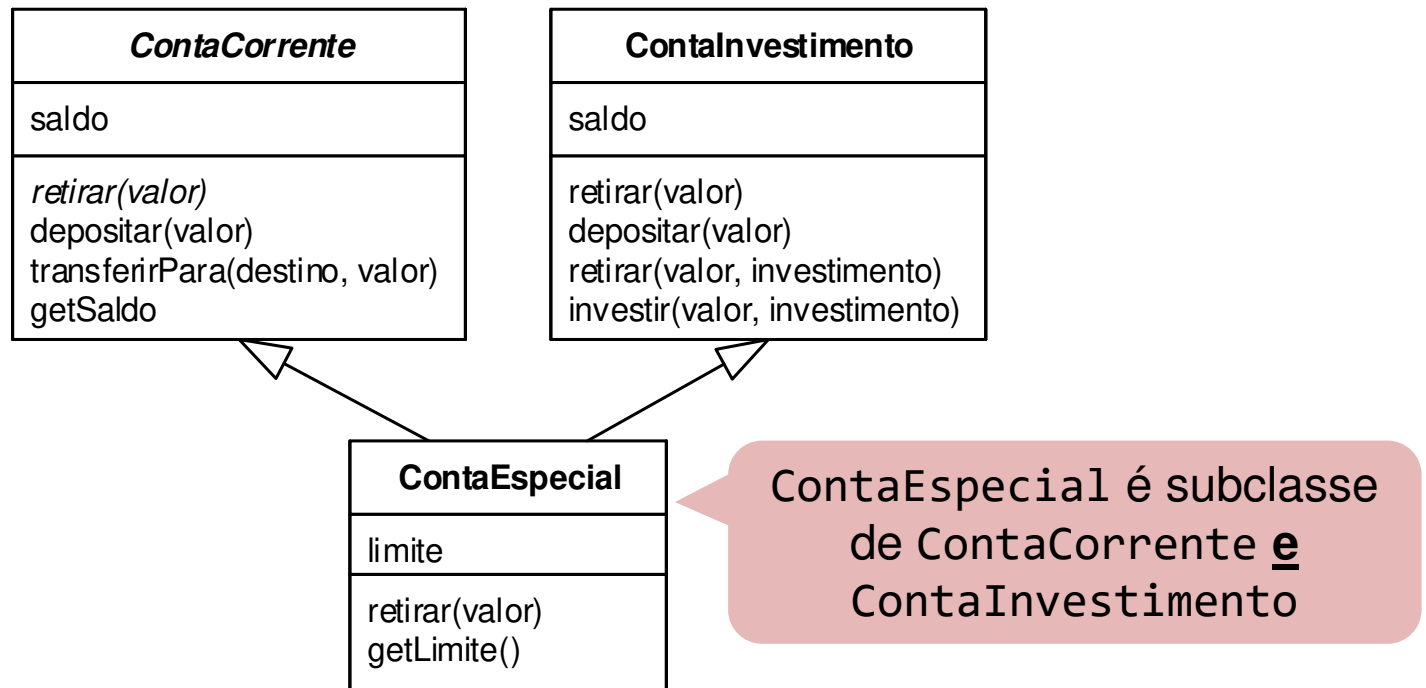
EX01

Método era abstrato na superclasse. Aqui é concreto!

Herança múltipla

Herança múltipla

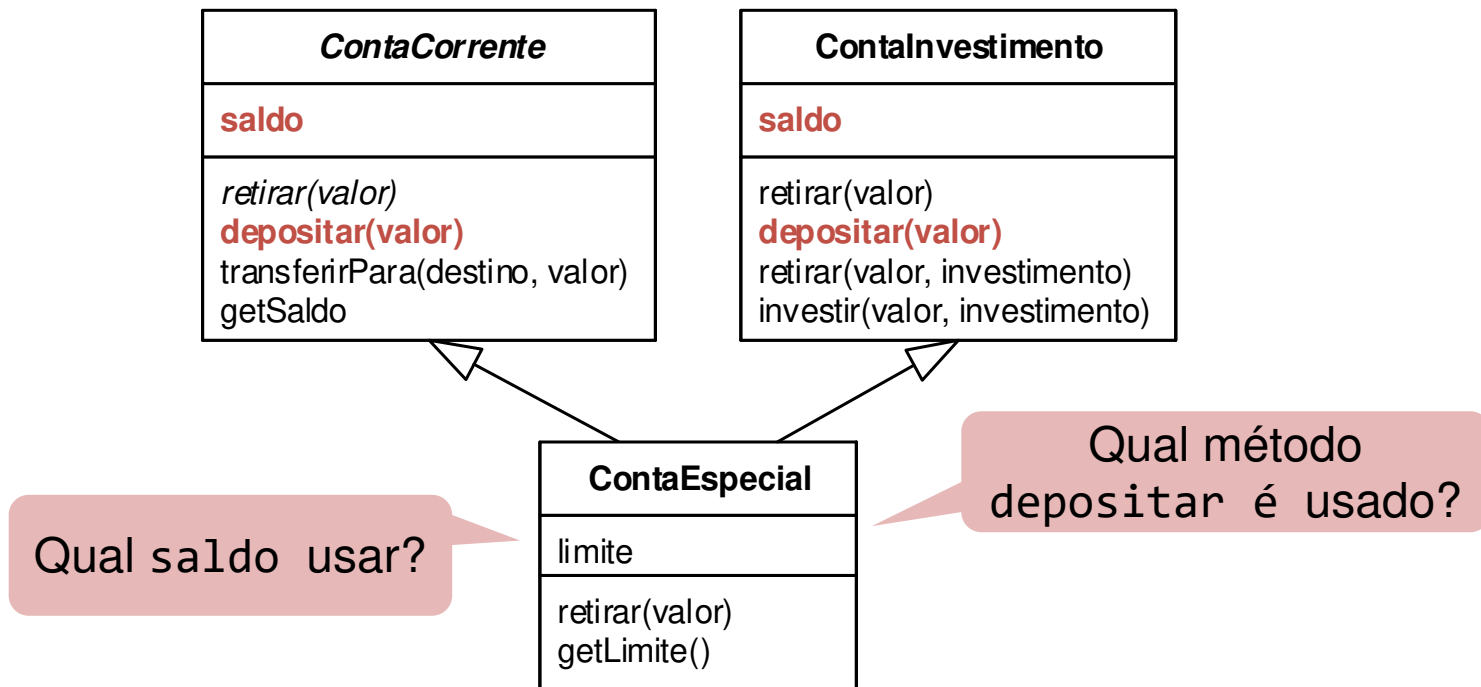
- Uma classe tem duas ou mais superclasses
 - *Exemplo:*



- *Qual é o problema disso?*

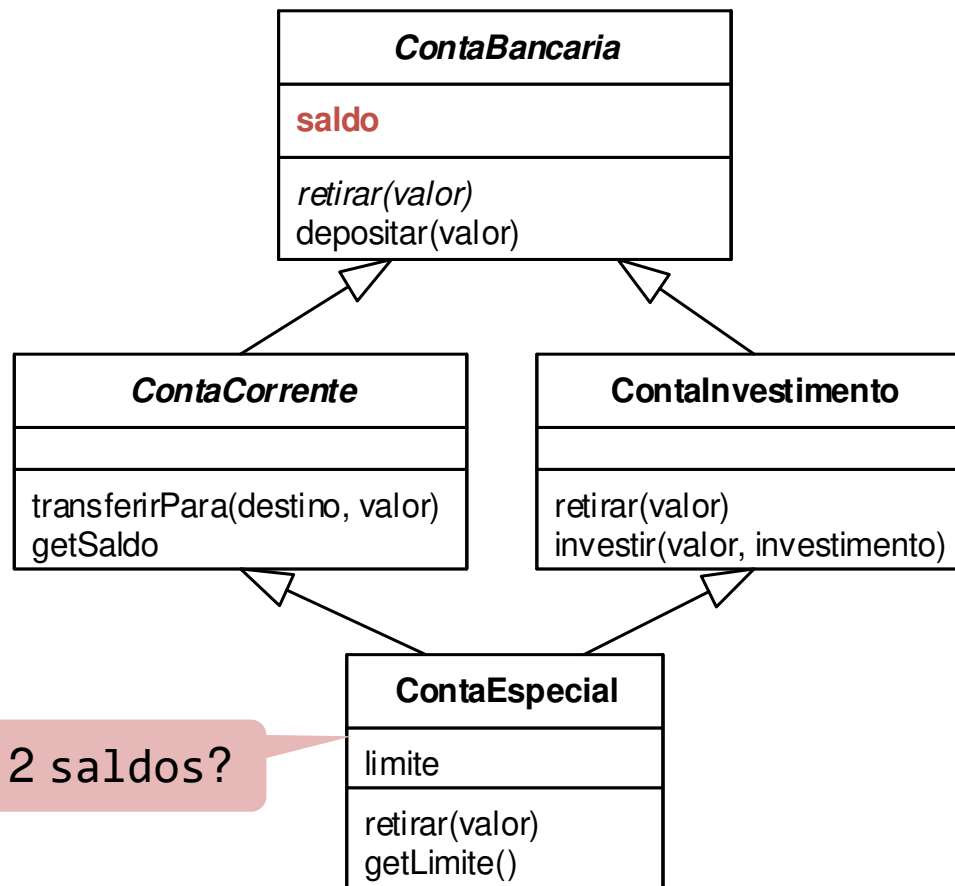
Problema: ambiguidade

- Se um **nome** estiver definido em mais de uma superclasse *qual nome usar*?
 - Exemplo:*



Problema: ancestral comum

- Quantos **atributos** a classe deve ter?
 - Problema do diamante (\diamond)



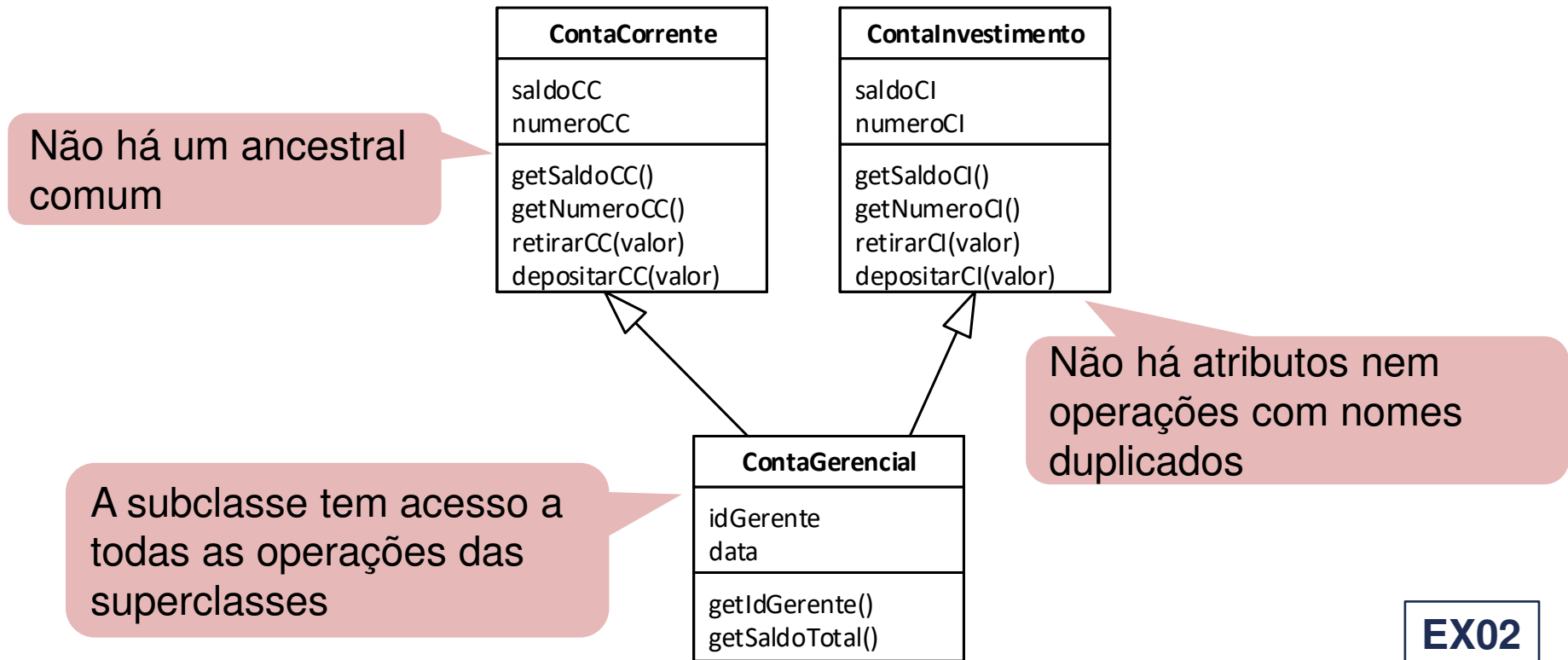
Tem 1 saldo ou 2 saldos?

Herança múltipla

- Algumas linguagens *não permitem* a herança múltipla
 - *Exemplo:* Java e C#
 - **O C++ permite!**
- Ainda assim é bom evitar a herança múltipla
 - *(Capítulo 13 do livro do Budd)*

Exemplo

- Considere o caso em que não ocorrem problemas de ambiguidade



EX02

Exemplo

■ Classes ContaCorrente e ContaInvestimento

```
8  class ContaCorrente {
9  public:
10     ContaCorrente(int numero);
11     virtual ~ContaCorrente();
12
13     bool retirarCC(double valor);
14     void depositarCC double valor);
15     double getSaldoCC();
16     int getNumeroCC();
17 protected:
18     double saldoCC;
19     int numeroCC;
20 };
```

EX02

```
9  class ContaInvestimento {
10 public:
11     ContaInvestimento(int numero);
12     virtual ~ContaInvestimento();
13
14     bool retirarCI(double valor);
15     void depositarCI(double valor);
16     double getSaldoCI();
17     int getNumeroCI();
18 protected:
19     double saldoCI;
20     int numeroCI;
21 };
```


Exemplo

- Em C++ deve-se separar por vírgula as superclasses

```
9  class ContaGerencial : public ContaCorrente, public ContaInvestimento {
10 public:
11     ContaGerencial (int idGerente, string data);
12     virtual ~ContaGerencial();
13     double getSaldoTotal();
14     int getIdGerente();
15
16 private:
17     int idGerente;
18     string data;
19 };
```

Separação entre os nomes das superclasses

EX02

Construtores

- O construtor das superclasses são *normais*
- A subclasse **precisa chamar todos** os construtores das superclasses
- *Exemplo:*

```
6  ContaGerencial::ContaGerencial (int idGerente, string data) :  
7      ContaCorrente (9990), ContaInvestimento (9991) {  
8      this->idGerente = idGerente;  
9      this->data = data;  
10 }
```

Chamada dos dois construtores,
separados por vírgula

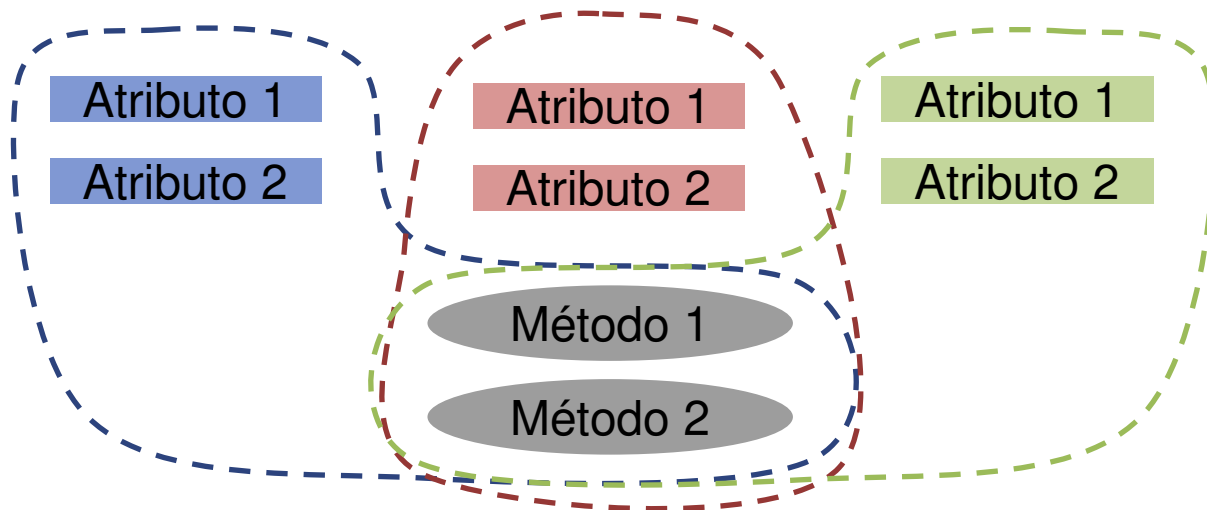
EX02

- Observação
 - Optou-se por um construtor bem simples, que define ele próprio os *números* que são atribuídos às superclasses

Atributos e métodos estáticos

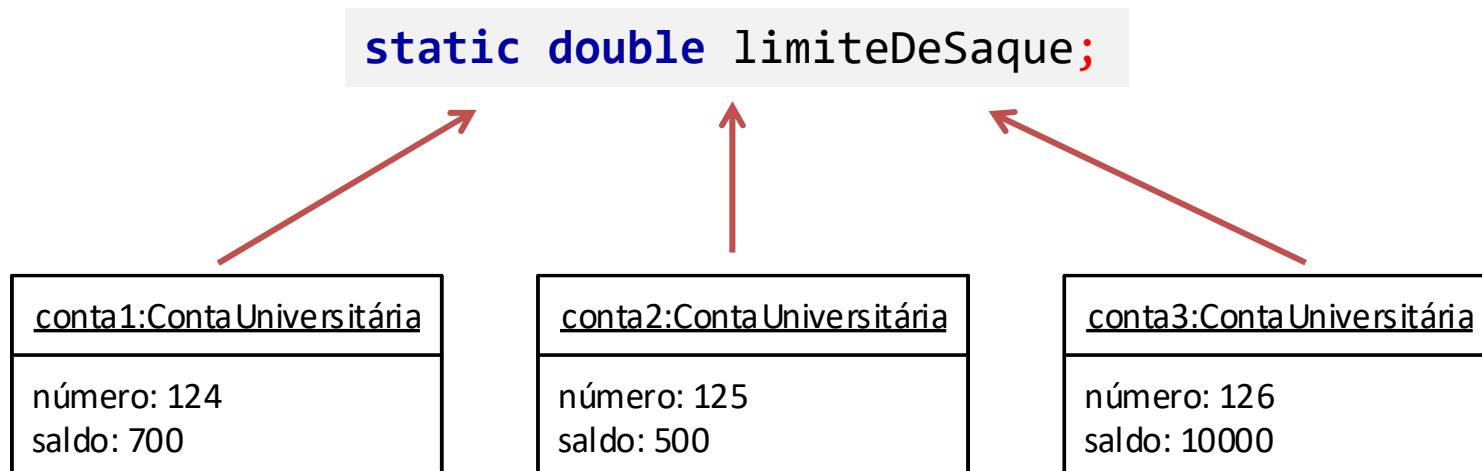
Como instâncias ocupam a memória

- Cada instância de uma classe tem um **valor específico** para cada atributo
 - São diferentes entre as instâncias
- O código dos métodos é **compartilhado**
 - É o mesmo para todas as instâncias
 - Usam os valores dos atributos de cada instância



Atributo estático

- Um atributo pode ser declarado como **estático**
 - Também chamado **atributo com escopo de classe**
 - Apenas **um valor** é criado para a classe
 - Não é definido em cada instância
 - O atributo existe mesmo que não haja instâncias
 - Ele existe por toda a execução do programa



Uso

- Permite compartilhar informações entre todas as instâncias de uma classe
- Exemplos de uso
 - Identificador sequencial
 - Contador de quantas instâncias a classe tem
 - Declaração de constantes (`static const`)
 - Informações comuns a todos os objetos

Atributo estático em C++

- Declaração: `static`

```
...
6  class ContaUniversitaria : public ContaCorrente {
7  public:
8      ContaUniversitaria (int numero);
9      virtual ~ContaUniversitaria();
10     bool retirar (double valor);
...
13  private:
14     static double limiteDeSaque;
15 };
```

Atributo estático

EX03

- Na implementação deve-se atribuir o valor inicial

```
1  #include "ContaUniversitaria.h"
2
3  double ContaUniversitaria::limiteDeSaque = 0;
...
```

EX03

Atributo estático em C++

■ Uso

- <nome da classe>::<nome do atributo>
 - Dentro da classe é possível omitir o <nome da classe>
 - (Mas pode ser confuso)
- *Exemplo*

```
...
12  bool ContaUniversitaria::retirar (double valor) {
13      if (valor <= ContaUniversitaria::limiteDeSaque
14          && saldo >= valor) {
15          saldo -= valor;
16          return true;
17      }
18
19      return false;
20  }
```

Acessando o limite

EX03

Método estático

- Método que não é executado no contexto de um objeto específico
 - Não é preciso ter uma instância para chamar o método
- Exemplos de uso
 - Manipular atributos estáticos
 - Métodos de apoio
 - *Exemplo:*
 - Arredondar um valor
 - Embaralhar um vetor
 - Imprimir um texto em um certo formato na saída

Método estático em C++

■ Declaração: static

```
...  
6  class ContaUniversitaria : public ContaCorrente {  
7  public:  
...  
11  static double getLimiteDeSaque();  
12  static void setLimiteDeSaque (double limite);  
13  private:  
14  static double limiteDeSaque;  
15  };
```

EX03

■ Implementação

Não deve-se colocar o static na implementação

```
...  
22  double ContaUniversitaria::getLimiteDeSaque() {  
23  return ContaUniversitaria::limiteDeSaque;  
24  }
```

EX03

Método estático em C++

- Chamada de método estático
 - Usando a instância
 - (Se confunde com método com escopo de objeto)
 - **Usando o nome da classe (recomendado)**

```
...  
7  ContaUniversitaria *u1 = new ContaUniversitaria(1);  
8  u1->depositar(100);  
9  
10 ContaUniversitaria *u2 = new ContaUniversitaria(1);  
11 u2->depositar(200);  
12  
13 ContaUniversitaria::setLimiteDeSaque(50);  
...  
24 u1->setLimiteDeSaque(100);
```

EX03

} Formas
possíveis

Bibliografia

- BUDD, T. **An Introduction to Object-Oriented Programming**. Addison-Wesley, 3rd ed. 2002.
 - Classe abstrata: Seção 8.5
 - Herança múltipla: Capítulo 13
- SAVITCH, W. **C++ Absoluto**. Pearson, 1st ed. 2003.
 - Atributos e métodos estáticos: Seção 7.2.