# Programação Orientada a Objetos

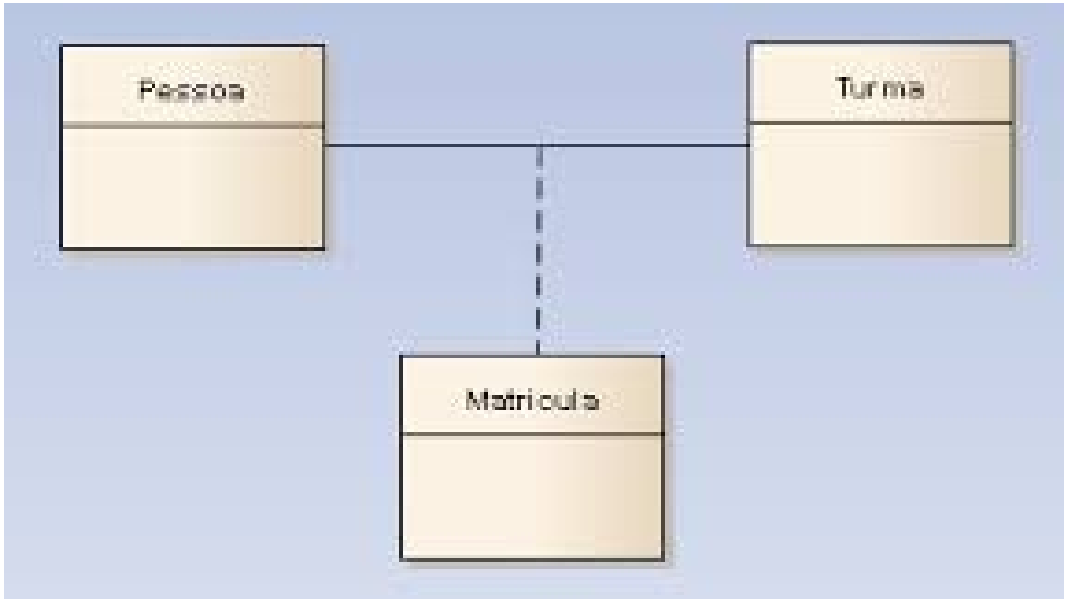
Ana Claudia Bastos Loureiro Monção

anaclaudia@inf.ufg.br

# Programação Orientada a Objetos

RELACIONAMENTO ENTRE CLASSES

(Associações)



# Programação Orientada a Objetos

## Relacionamento entre Classes

Em um sistema **Orientado** a **Objetos**, as classes não trabalham sozinhas, existem relacionamentos e comunicações entre elas.

O tipo do relacionamento e a forma de  comunicação entre as classes definem responsabilidades. Existem 3 tipos :

- Associação, agregação e composição;
- Herança (generalizações e especializações);
- Dependências

# Programação Orientada a Objetos

## Relacionamento entre Classes

Uma classe pode ter atributos que são instâncias de uma outra classe, ou dela mesma (associação).

Uma classe pode herdar atributos e métodos de uma outra classe (herança).

Uma classe pode fazer referência a outra classe utilizando seus métodos e atributos (dependência).

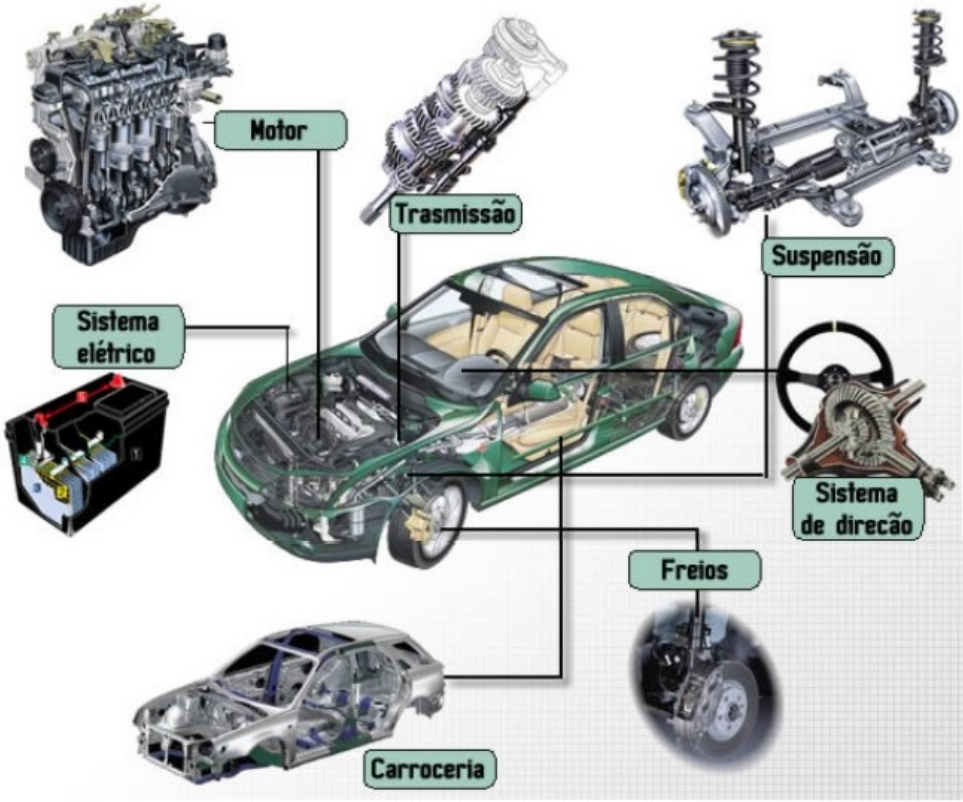
# Programação Orientada a Objetos

## Relacionamento entre Classes

Um automóvel, por exemplo, é formado por peças que apresentam relações muito definidas.

Possui: motor, rodas, banco, freio, Carroceria, volante e diversas outras engrenagens.

Esses elementos, associados ou conectados de maneira correta, permitem a existência e o correto funcionamento do automóvel.



# Programação Orientada a Objetos

## Relacionamento entre Classes

São alguns exemplos de relacionamento entre as peças que compõe um Automóvel:

• **Associação**: Pneus, Rodas, Amortecedores e Freios formam a Suspensão do Automóvel.

*Composição e Agregação*: tipos específicos de associação

• **Generalização**: Filtro de Ar, Filtro de Óleo e Filtro de Combustível são tipos específicos de Filtro.

• **Dependência**: Motor depende de Bomba de Gasolina para bombear a gasolina para o seu interior.

# Programação Orientada a Objetos

## Associação

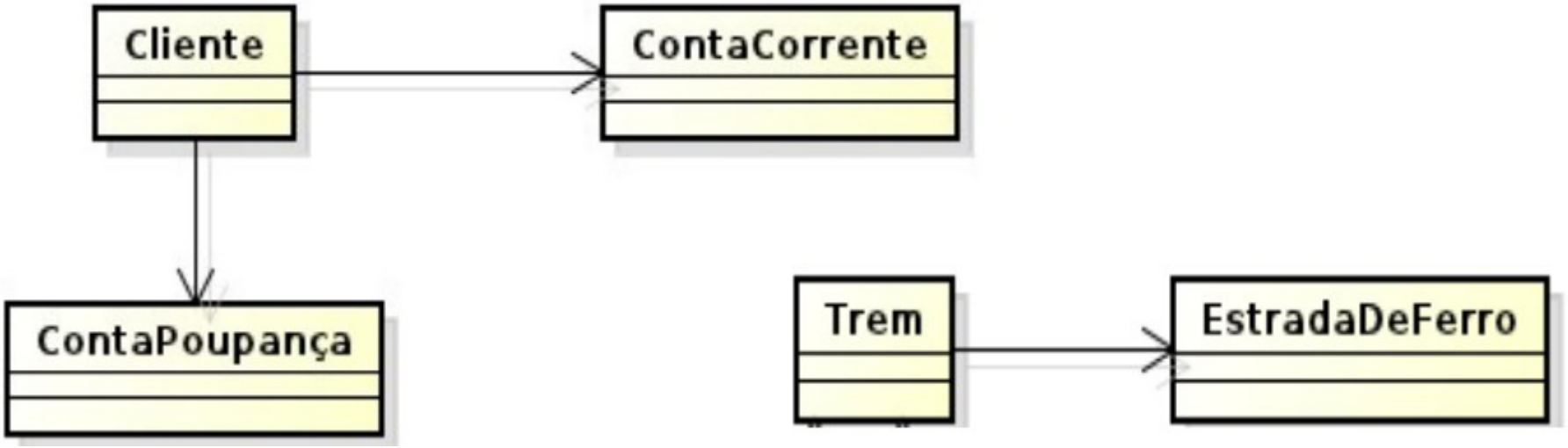
Uma associação indica algum relacionamento significativo e de interesse entre objetos.

É representado por uma linha conectando os dois objetos.

- Pode existir uma seta no fim da linha, apontando para o objeto que está sendo usado.
- A associação pode também receber um nome e uma multiplicidade.

# Programação Orientada a Objetos

## Associação




- Um cliente possui uma conta corrente e uma conta poupança
- Um trem usa uma estrada de ferro

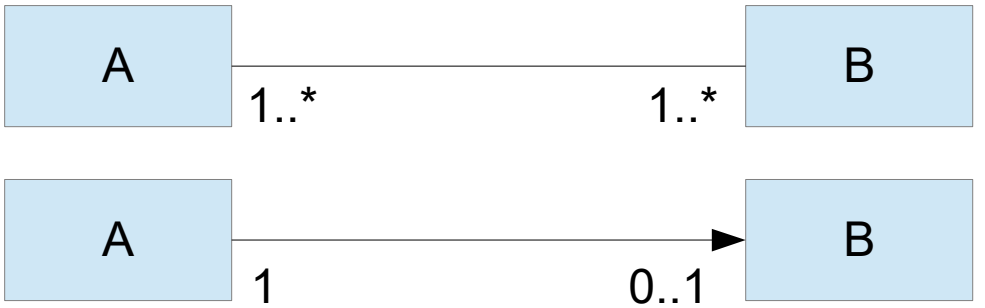
# Programação Orientada a Objetos

## Multiplicidade nas Associações

Define quantas instâncias de uma classe A podem estar relacionadas a uma instância de um classe B

Valores de multiplicidade:

- 0 : zero
- 0..1 : zero ou 1
- 1 : um
- 1..* : um ou mais
- 0..* ou * : zero ou mais (muitos)



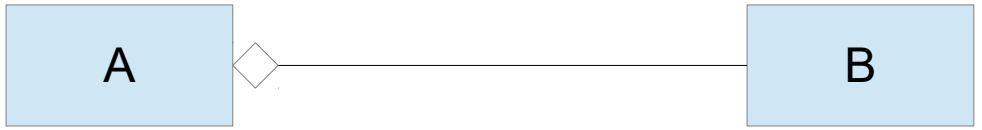
# Programação Orientada a Objetos

## Agregação

Um tipo de associação onde a relação tem o sentido de: **“é parte de”**

Relaciona um objeto (o todo) com sua(as) parte(s)

- A parte só é criada quando o todo é criado
- É representado por um losango vazio junto ao objeto representando o todo



# Programação Orientada a Objetos

## Composição

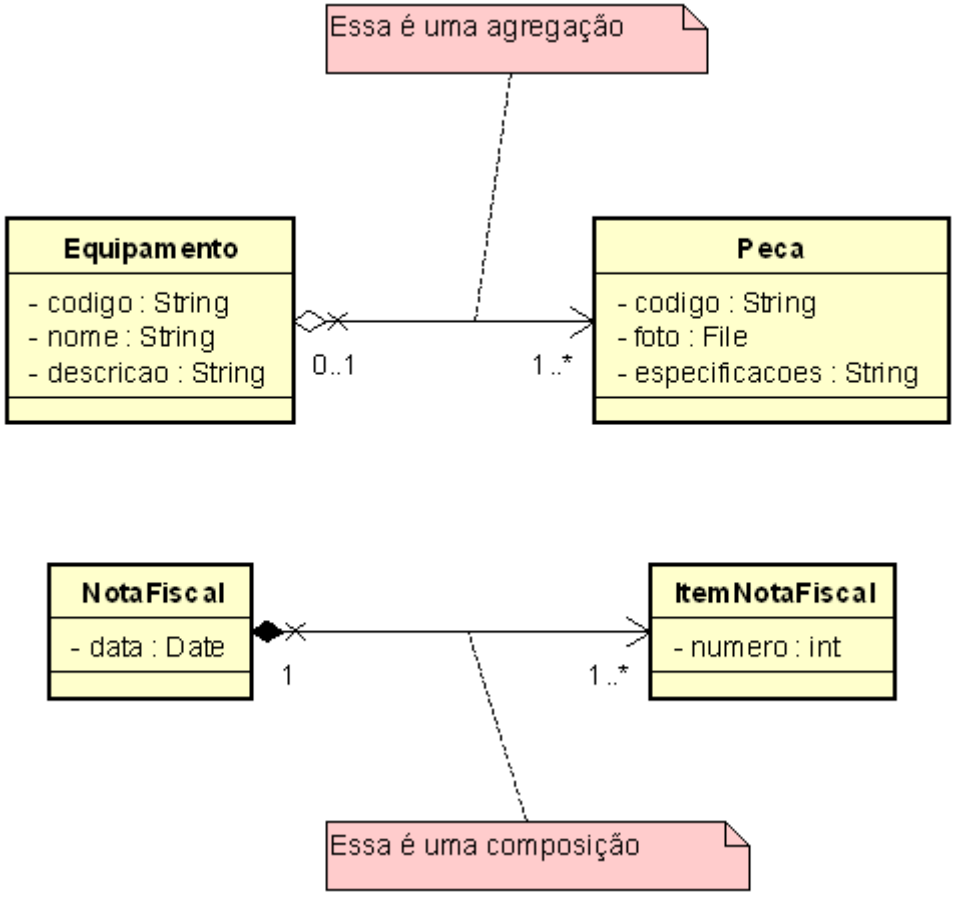
A composição é uma agregação mais forte

- um objeto **“é parte essencial”** de outro
- Na composição, o objeto composto não existe sem os seus componentes

- É representado por um losango preenchido em preto
- Obs.: Em geral, é mais comum usar agregação mesmo quando o relacionamento é mais forte

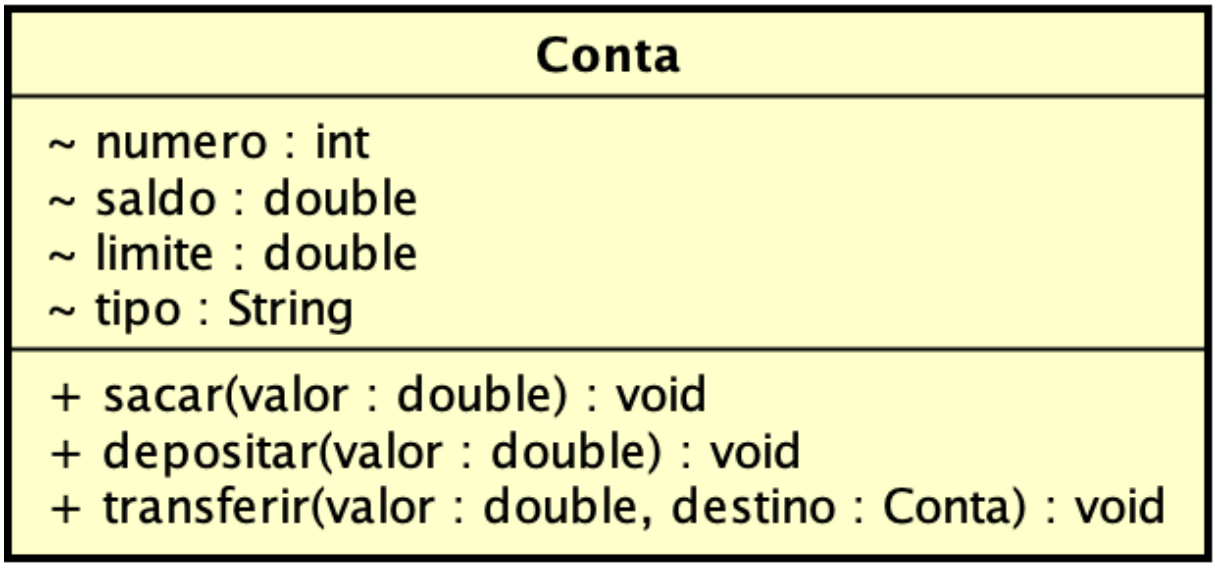
# Programação Orientada a Objetos

## Agregação X Composição



# Programação Orientada a Objetos

## Modelo da Conta em UML



# Programação Orientada a Objetos

## Acrescentando Atributos à Conta

**Informações necessárias para abrir uma conta:**

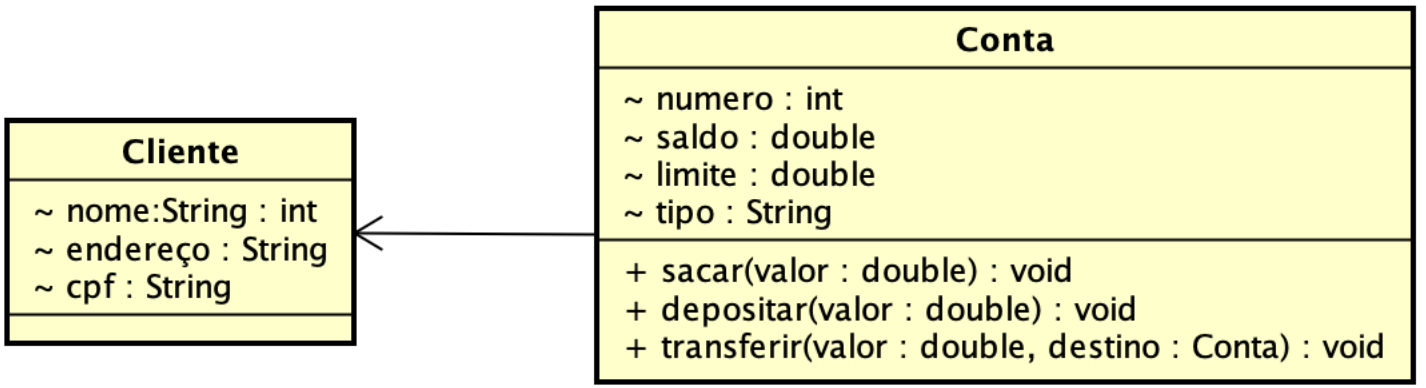
- **Nome**
- **Endereço**
- **CPF**
- Data de Nascimento
- Telefone
- Etc..

São atributos de Cliente e não de Conta

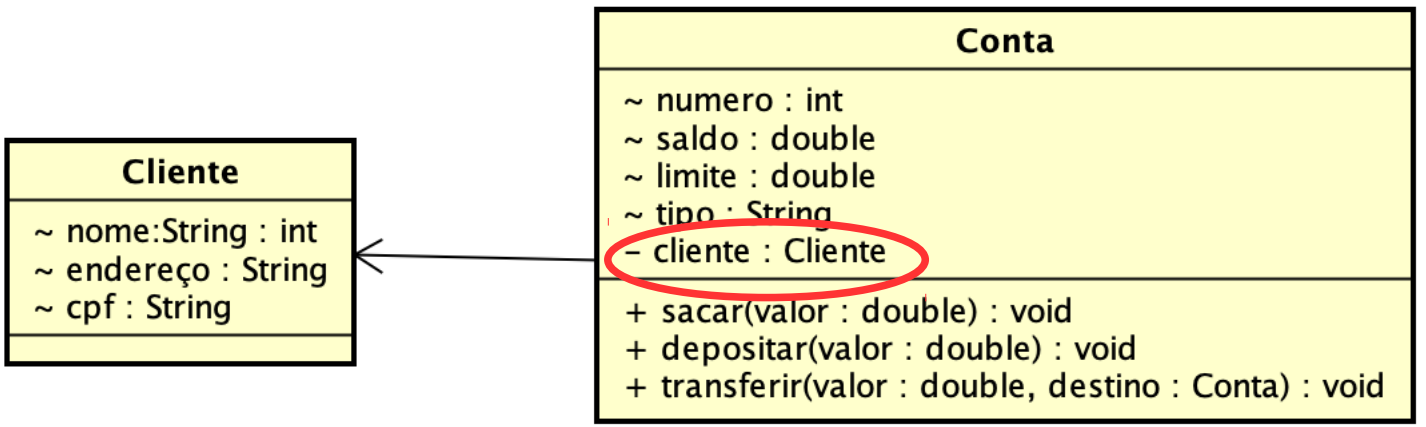
Um Cliente pode ter mais de uma Conta

# Programação Orientada a Objetos

## Modelo de Conta associada a um Cliente em UML

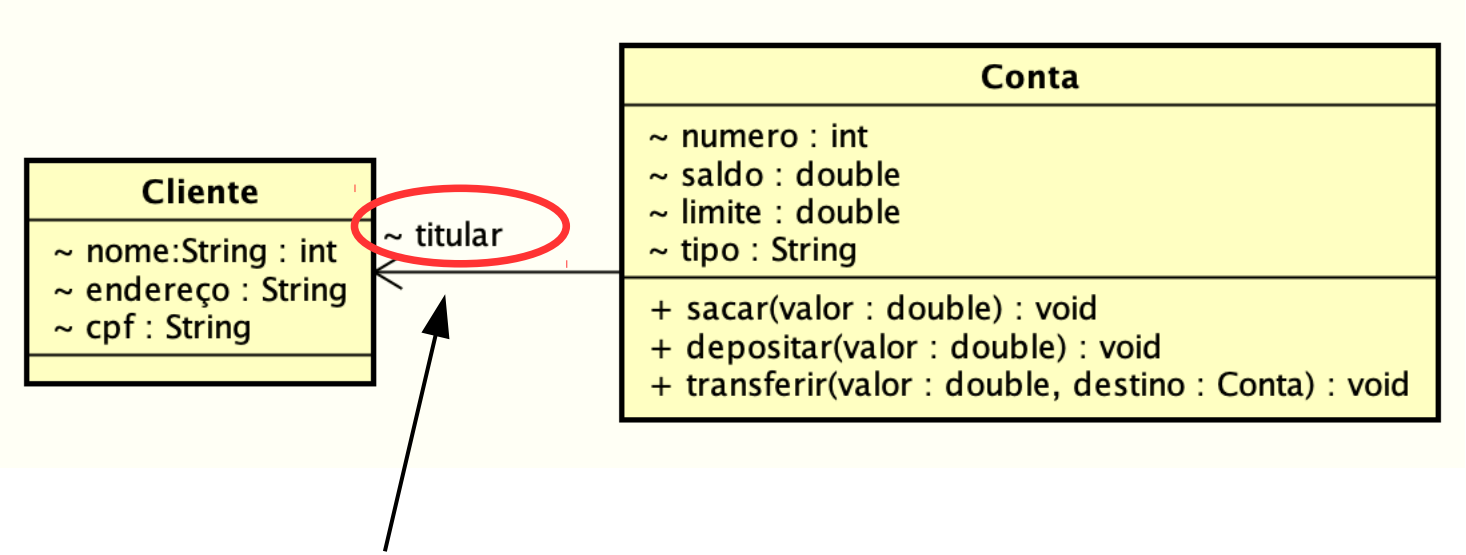



**o mesmo que (não precisa representar o atributo no diagrama):**



# Programação Orientada a Objetos

## Modelo de Conta associada a um Cliente em UML




Representação muito utilizada quando nome do atributo diferente do nome da classe

# Programação Orientada a Objetos

## Associando Classes

```
class Cliente {
    String nome;
    String endereco;
    String cpf;   ...
}
class Conta {
    int numero;
    double saldo;
    double limite;
    String tipo;
    Cliente titular;
}
```

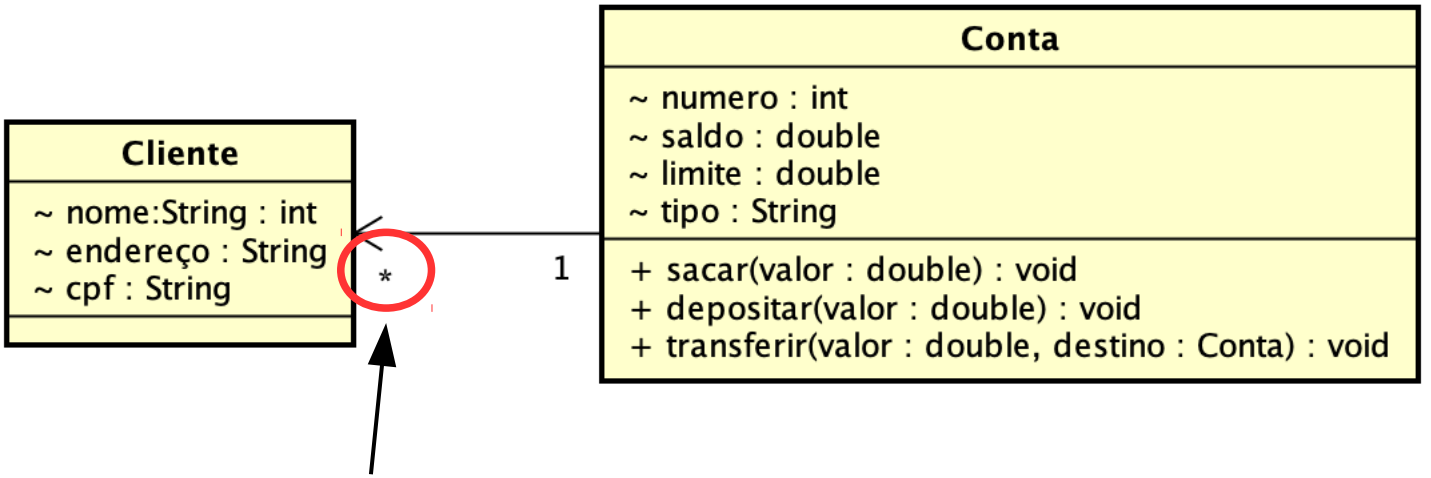
```
class Programa {
    public void main (String[] args) {
        Conta minhaConta = new Conta();

        Cliente c = new Cliente();
        minhaConta.titular = c;

        minhaConta.titular.nome = “Ana”;

    }
}
```

**minhaConta** agora tem uma referência para o Cliente correspondente a c

Forma de Acessar o Cliente

# Programação Orientada a Objetos

## Conta associada a mais de um Cliente em UML




Uma conta pode ter vários Clientes

# Programação Orientada a Objetos

## Associando Classes

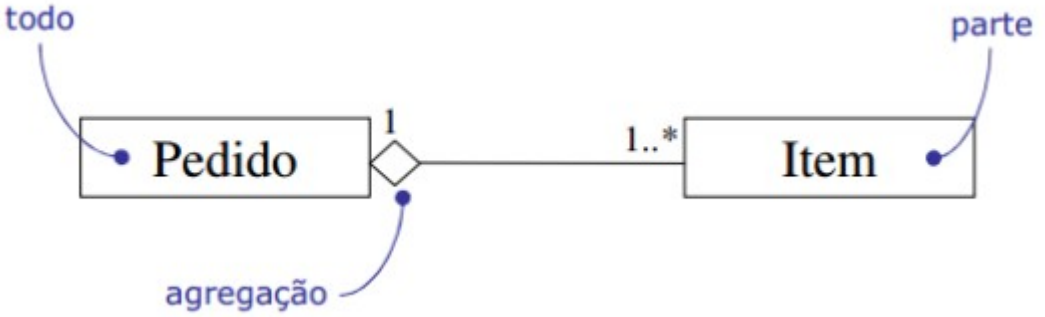
### (Vários clientes em uma conta)

```
class Cliente {
    String nome;
    String endereco;
    String cpf;   ...
}
class Conta {
    int numero;
    double saldo;
    double limite;
    String tipo;
    Cliente clientes[ ] = new Cliente[3];
}
```

```
class Programa {
    public void main (String[] args) {
        Conta minhaConta = new Conta();
        Cliente c = new Cliente();
        minhaConta.clientes[0] = c;
        minhaConta.clientes[0].nome = “Ana”;
        for (int i=1;i<3;i++) {
            minhaConta.clientes[i] = new Cliente();
            minhaConta.clientes[i].nome = “Dependente” + (i+1);
        }
    }
}
```

# Programação Orientada a Objetos

## Agregação




```
public class Pedido {
   int codigo;
   ArrayList <Item> itens = new ArrayList();

   Pedido(int codigo) {
      this.codigo = codigo;
   }
}

public class Item {
   int codigo;
   String descricao;

   Item(int codigo, String descricao) {
      this.codigo = codigo;
      this.descricao = descricao;
   }
}
```

```
public class Principal {

    public static void main(String[] args) {

      Pedido pedido = new Pedido(20);

      pedido.itens.add(new Item(1,"Pão"));

      pedido.itens.add(new Item(2,"Leite"));

      pedido.itens.add(new Item(3,"Manteiga"));

    }

}
```

# Programação Orientada a Objetos

## Composição


```
public class Window {
   ArrayList <Frame> frames = new ArrayList();

   Window(String titulo, String menu, String conteudo){

      frames.add(new Frame(titulo));
      frames.add(new Frame(menu));
      frames.add(new Frame(conteudo));
   }

}

public class Frame {
   String descricao;

   Frame(String descricao) {
      this.descricao = descricao;
   }
}
```

```
public class Principal {

    public static void main(String[] args) {

       Window window = new Window("Titulo","Menu
      Lateral”,"Conteudo");

    }

}
```

# Programação Orientada a Objetos

## Dependência

Um objeto utiliza recursos (atributo ou métodos de outro objeto) para realizar suas operações.

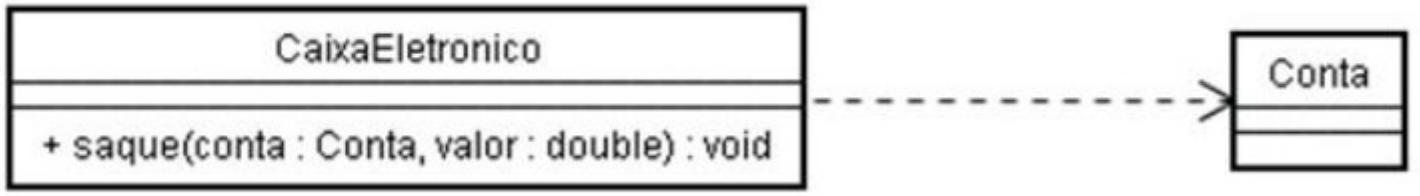
Uma modificação em um objeto fornecedor pode afetar o comportamento de outros objetos.

- É representado por uma linha tracejada do cliente (A) para o fornecedor (B)

# Programação Orientada a Objetos

## Dependência

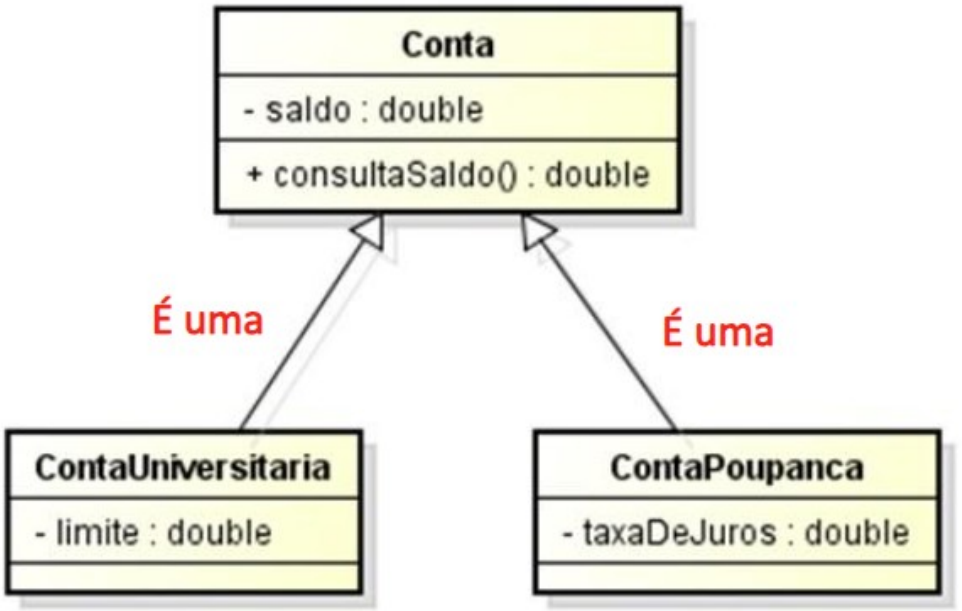
- Existe uma dependência de CaixaEletronico (cliente) com Conta (fornecedor).
- O método saque depende de um objeto da classe Conta.



# Programação Orientada a Objetos

## Herança (Generalização e Especialização)

Uma relação que tem o sentido de: **“é um”** ou **“é um tipo de”**




**Próximas aulas**

# Programação Orientada a Objetos

## Exercício

- Crie um modelo para representar Músicas e Compositores associando as duas classes. Os cadastros devem manter o nome da música, tipo, ano, nome do compositor e a sua nacionalidade.
- Crie um modelo para representar Alunos que estão matriculados em um curso de um departamento. O cadastro deve manter o nome do aluno, a matrícula, o ano de ingresso, o curso, com seu nome, sigla e duração, e o nome e sigla do departamento.
- Crie um modelo para representar Clientes com seu nome, telefone e endereço sendo que um endereço contempla a rua, numero, complemento, cep, bairro, cidade e estado
- Crie um modelo para representar um empregado que trabalha em uma empresa sendo que para a empresa é preciso manter o cnpj, razão social e o seu endereço que é formado pela rua, número, complemento, cep, bairro, cidade e estado