

# Diagrama de classes

---

EDUARDO HABIB BECHELANE MAIA

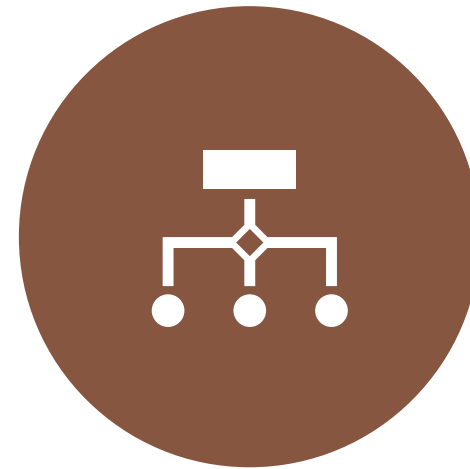
[HABIB@CEFETMG.BR](mailto:HABIB@CEFETMG.BR)

# Introdução

---



INTRODUÇÃO – DIAGRAMA  
DE CLASSES



ELEMENTOS DO DIAGRAMA  
DE CLASSES

# Introdução

---

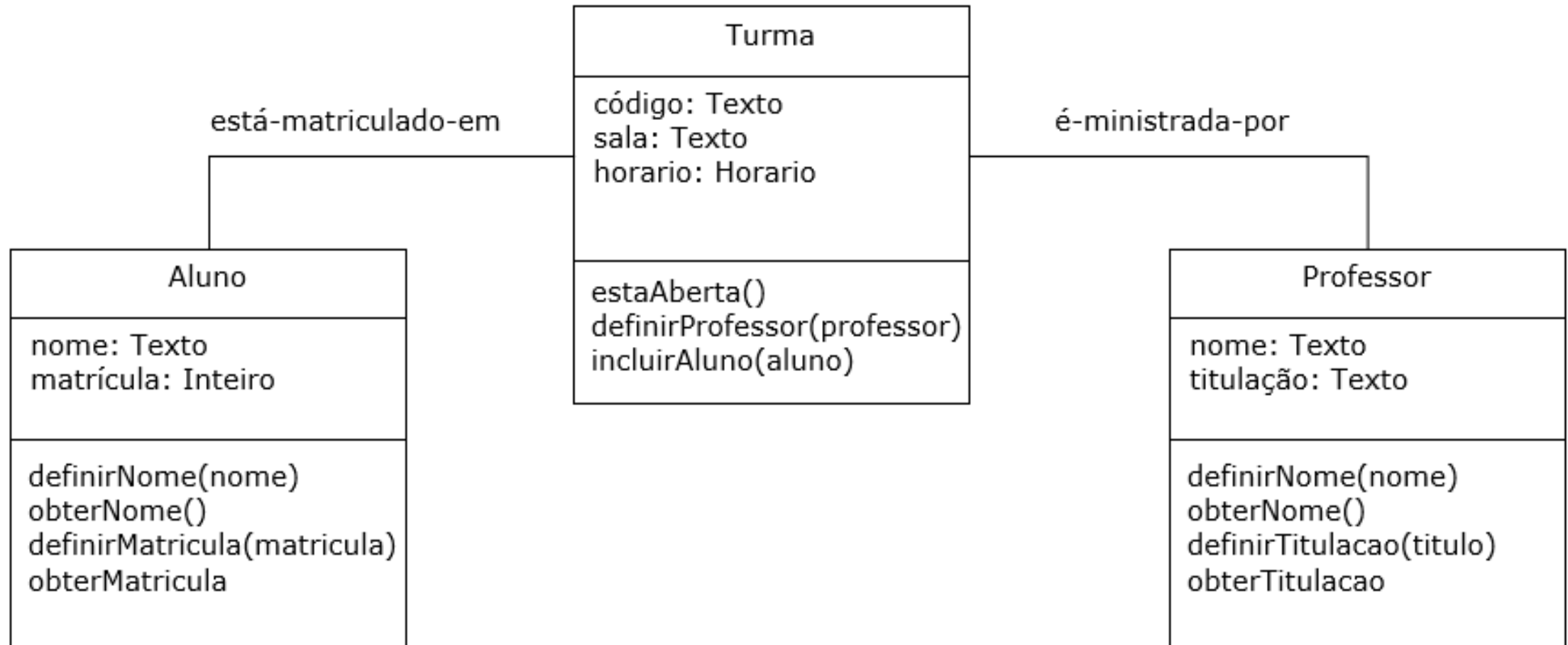


Mostra um conjunto de classes e seus relacionamentos.



É o diagrama central da modelagem orientada a objetos.

# Introdução



# Elementos dos diagramas de classes

---

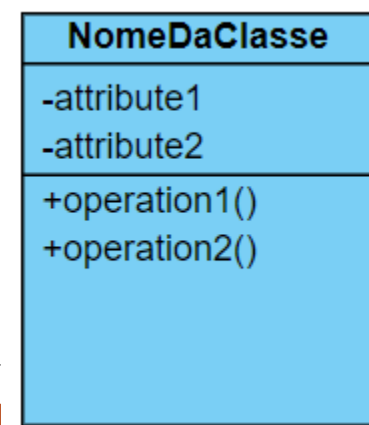


Classes

Relacionamentos

- Associação
  - Agregação
  - Composição
- Generalização
- Dependência

# Classes



Graficamente, as classes são representadas por retângulos incluindo nome, atributos e métodos.

Devem receber nomes de acordo com o vocabulário do domínio do problema.

É comum adotar um padrão para nomeá-las

- **Ex:** todos os nomes de classes serão substantivos singulares com a primeira letra maiúscula

# Classes

---

## Atributos

- Representam o conjunto de características (estado) dos objetos daquela classe
- Visibilidade:
  - + público: visível em qualquer classe de qualquer pacote
  - # protegido: visível para classes do mesmo pacote
  - privado: visível somente para classe

## Exemplo:

+ nome : String

# Elementos dos diagramas de classes

---

Classes



Relacionamentos

- Associação
  - Agregação
  - Composição
- Generalização
- Dependência



# Relacionamentos

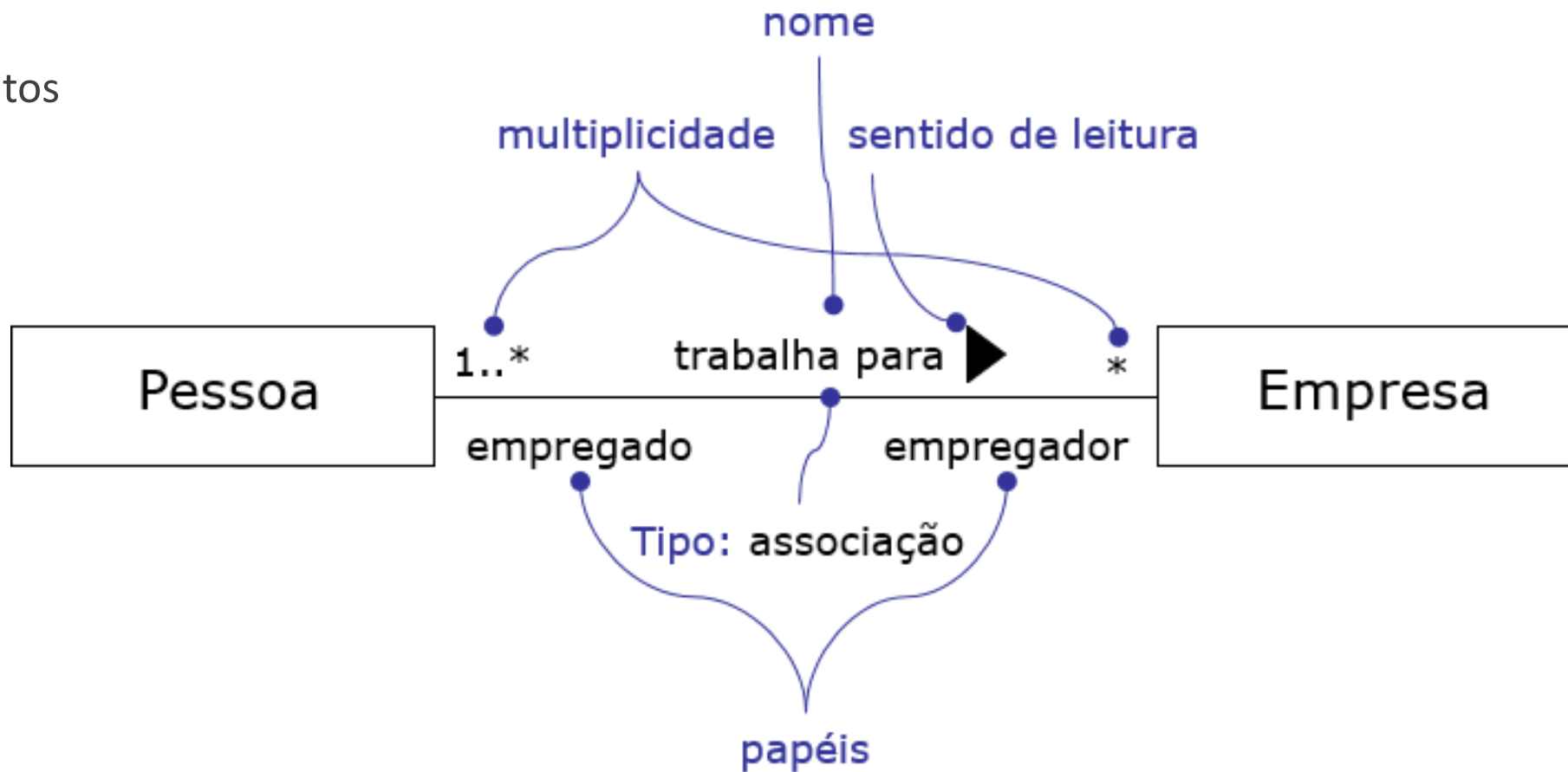
---

## Os relacionamentos possuem:

- Nome: descrição dada ao relacionamento (faz, tem, possui,...)
- Sentido de leitura
- Navegabilidade: indicada por uma seta no fim do relacionamento
- Multiplicidade: 0..1, 0..\*, 1, 1..\*, 2, 3..7
- Tipo: associação (agregação, composição), generalização e dependência
- Papéis: desempenhados por classes em um relacionamento

# Diagrama de classes

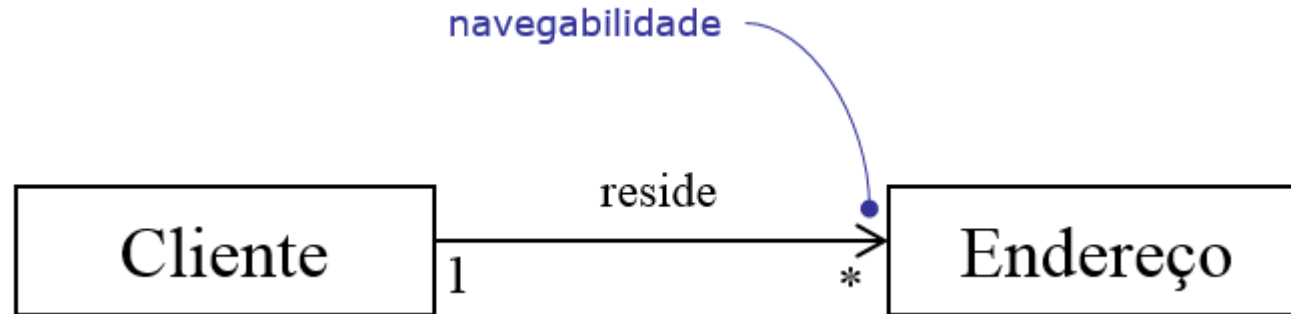
Relacionamentos



# Diagrama de classes

---

## Relacionamentos




- O cliente sabe quais são seus endereços, mas o endereço não sabe a quais clientes pertence

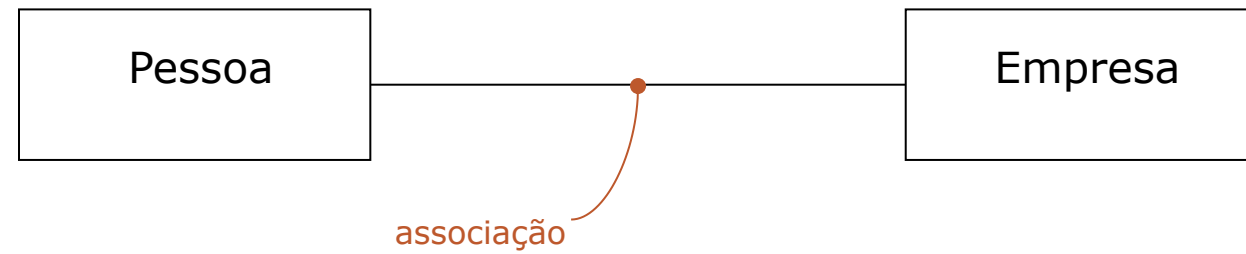
# Elementos dos diagramas de classes

---

Classes

Relacionamentos

- 
- Associação
  - Agregação
  - Composição
  - Generalização
  - Dependência



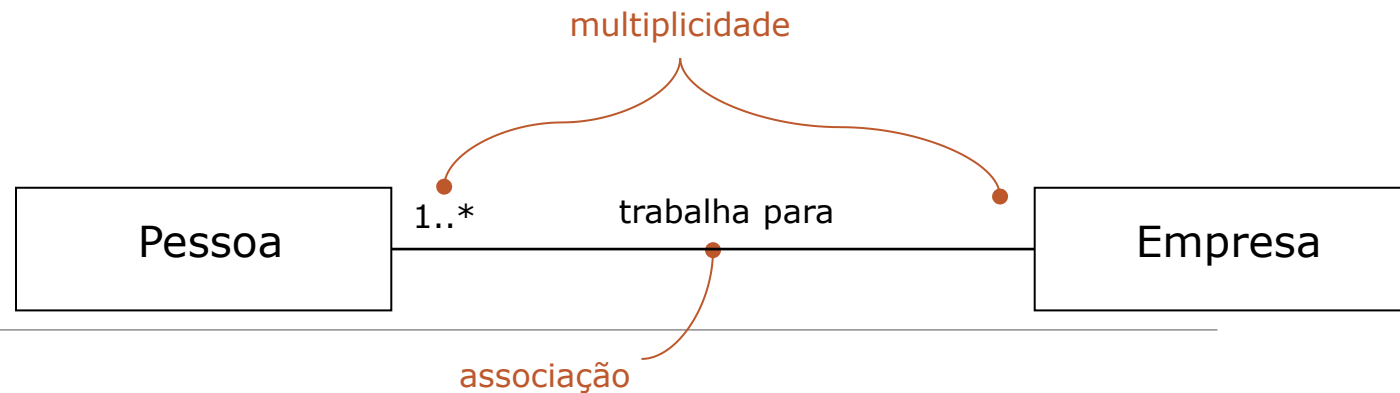
# Associação

---

Uma **associação** é um relacionamento estrutural que indica que os objetos de uma classe estão vinculados a objetos de outra classe.

Uma associação é representada por uma linha sólida conectando duas classes.

# Associação



## Indicadores de multiplicidade:

- 1 Exatamente um
- 1..\* Um ou mais
- 0..\* Zero ou mais (muitos)
- \* Zero ou mais (muitos)
- 0..1 Zero ou um
- m..n Faixa de valores (por exemplo: 4..7)

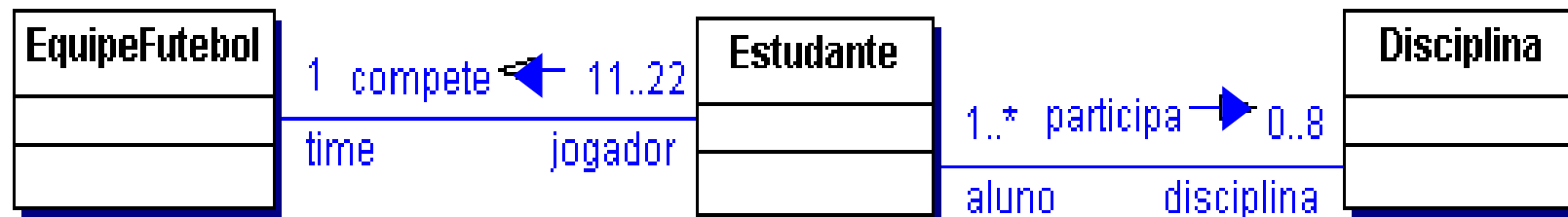
# Associação

---

- Exemplo:
- Um **Estudante** pode ser:
  - um **aluno** de uma Disciplina e
  - um **jogador** da Equipe de Futebol

Cada Disciplina deve ser cursada por no mínimo 1 aluno

Um aluno pode cursar de 0 até 8 disciplinas



# Elementos dos diagramas de classes

---

Classes

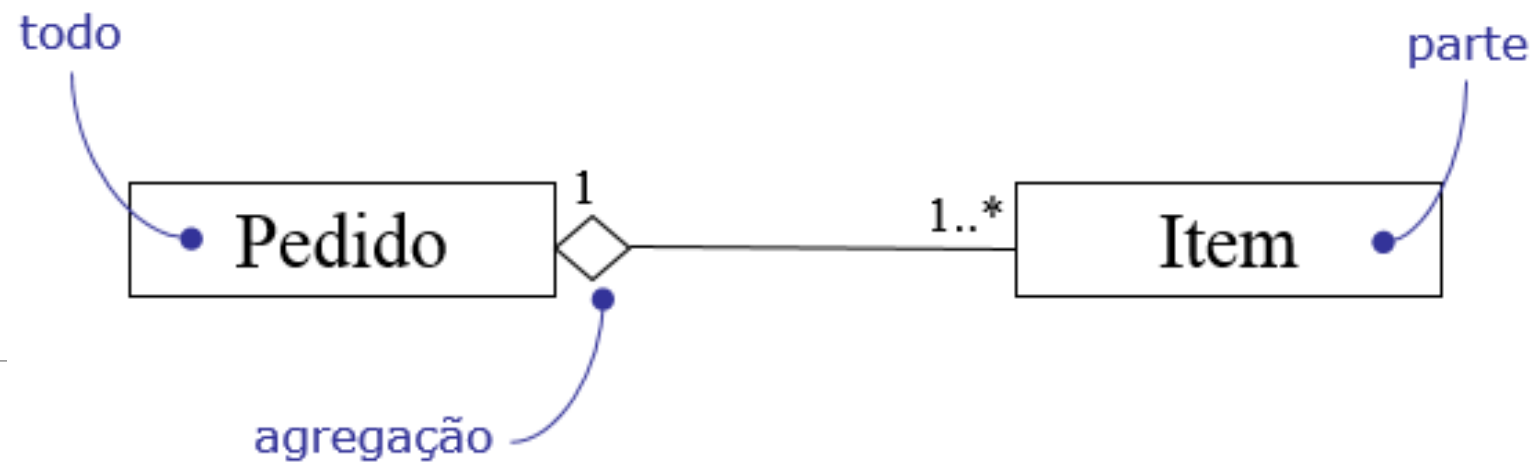
Relacionamentos

- Associação
- Agregação
- Composição
- Generalização
- Dependência





# Agregação



É um tipo especial de associação

Utilizada para indicar “todo-parte”

- um objeto “parte” continua existindo sem o todo

# Agregação

---

## Mais exemplos

- Carro e rodas
- Time e jogadores
- Computador e componentes (memória, placa de vídeo, placa mãe)
- Biblioteca e livros
- Bicicleta e peças
- Etc.

# Elementos dos diagramas de classes

---

Classes

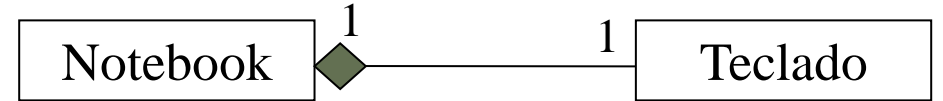
Relacionamentos

- Associação
- Agregação
- Composição
- Generalização
- Dependência



# Composição

---



## Os objetos:

- Pertencem a um único todo
- Têm o tempo de vida coincidindo entre as partes e o objeto.
- Se o objeto deixar de existir, suas partes também deixam de existir

# Composição

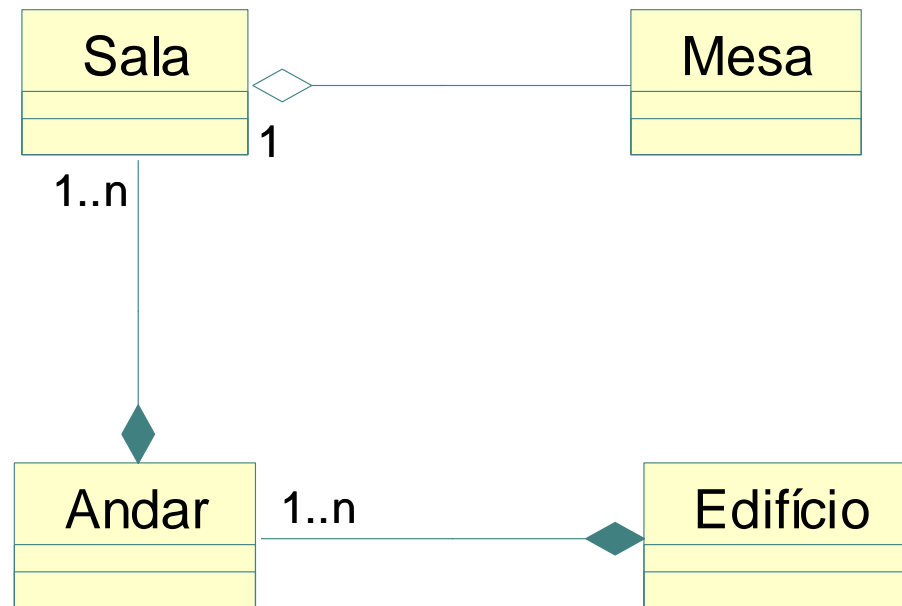
---

Mais exemplos:

- Corpo humano e órgãos
- Avião, asa, fuselagem
- Universidade e departamentos
- Etc.

# Elementos – Diagrama de Classes

Agregação X Composição



# Elementos dos diagramas de classes

---

Classes

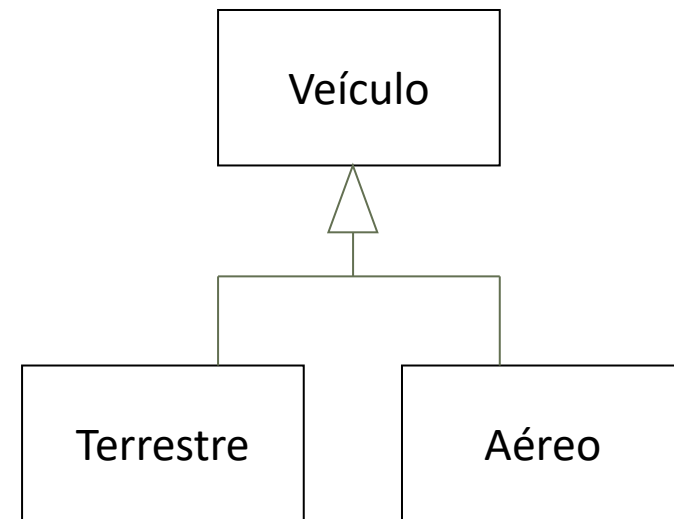
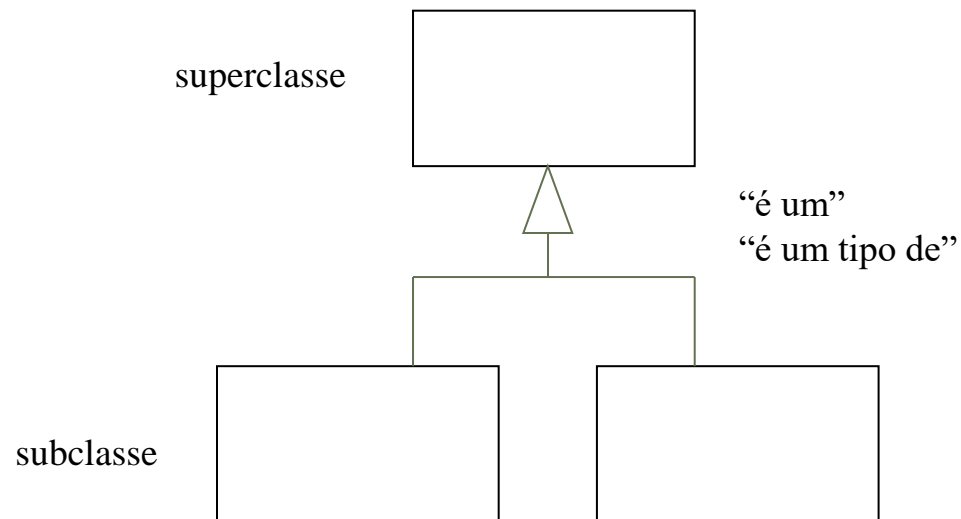
Relacionamentos

- Associação
  - Agregação
  - Composição
- Generalização
- Dependência



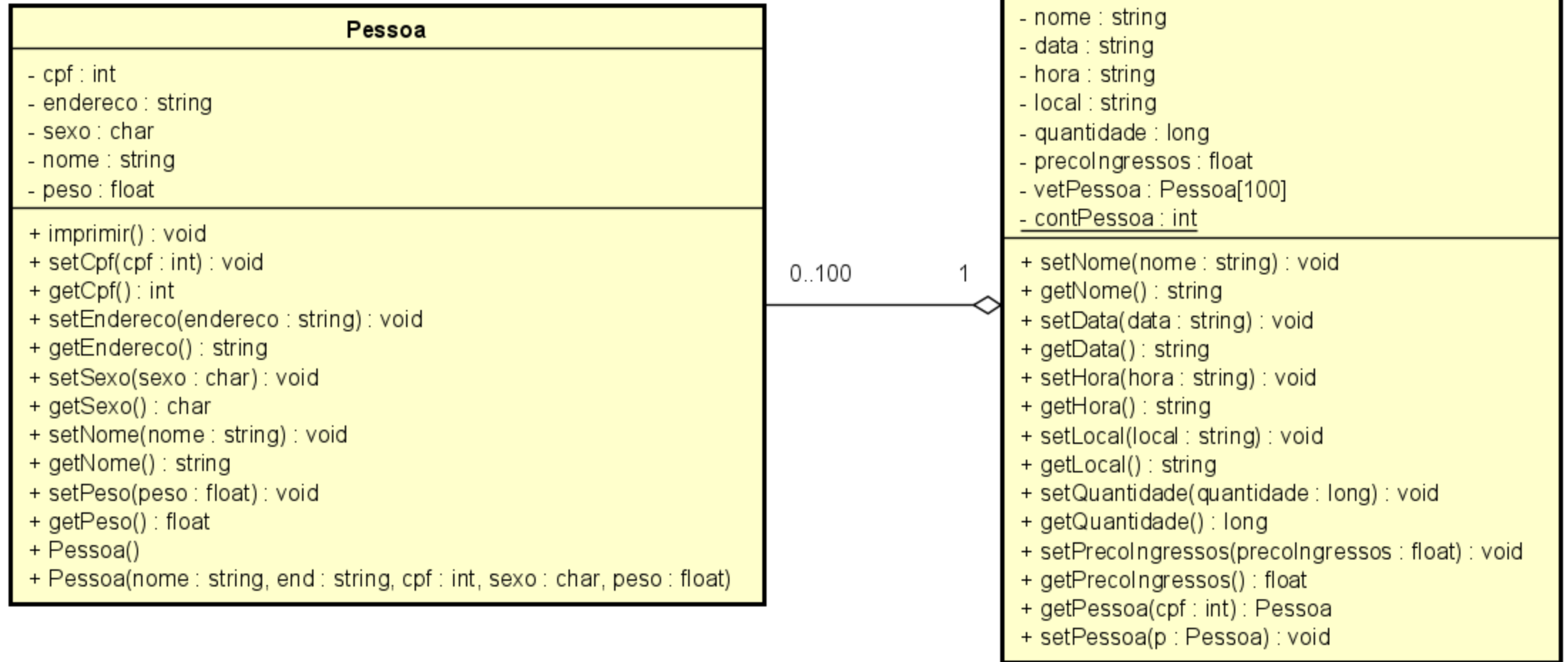
# Generalização

É um relacionamento entre itens gerais (superclasses) e itens mais específicos (subclasses)





# Exemplo



# Código em Java

---

# Pessoa.java

```
public class Pessoa {

    private int cpf;
    private String endereco;
    private char sexo;
    private String nome;
    private float peso;

    public Pessoa() {
    }

    public Pessoa(String nome, String endereco, int cpf,
char sexo, float peso) {
        this.nome = nome;
        this.endereco = endereco;
        this.cpf = cpf;
        this.sexo = sexo;
        this.peso = peso;
    }

    public void imprimir() {
        System.out.println("\nnome = " + nome);
        System.out.println("cpf = " + cpf);
        System.out.println("endereco = " + endereco);
        System.out.println("sexo = " + sexo);
        System.out.println("peso = " + peso);
    }

    // Implementar Getters e Setters

}
```

# Evento.java

```
public class Evento {

    private static int contPessoa = 0;
    private String nome;
    private String data;
    private String hora;
    private String local;
    private long quantidade;
    private float precoIngressos;
    private Pessoa[] vetPessoa = new
Pessoa[100];

    public Evento() {
        for (int i = 0; i < contPessoa; i++) {
            vetPessoa[i] = new Pessoa();
            vetPessoa[i].setCpf(0);
        }
    }

    public Pessoa getPessoa(int cpf) {
        Pessoa p1;
        Pessoa p2 = new Pessoa();
        p2.setCpf(0);

        for (int i = 0; i < contPessoa; i++) {
            p1 = vetPessoa[i];
```

```
            if (p1.getCpf() == cpf) {
                return p1;
            }
        }
        return p2;
    }

    public void setPessoa(Pessoa p) {
        if (contPessoa < 100) {
            this.vetPessoa[contPessoa] = p;
            contPessoa++;
        } else {
            System.out.println("O evento já está
cheio");
        }
    }

    // Implementar Getters e Setters
}
```

# Main.java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int opcao = 0;
        String nome, endereco;
        int cpf;
        char sexo;
        float peso;
        Evento e1 = new Evento();
        Pessoa p1 = new Pessoa();

        do {
            System.out.println("\n\n\n Programa de venda de ingressos para eventos");
            System.out.println("O que você deseja fazer?");
            System.out.println("Digite 1 para cadastrar pessoa");
            System.out.println("Digite 2 para mostrar dados da pessoa cadastrada");
            System.out.println("Digite 3 para listar a quantidade de pessoas no evento");
            System.out.println("Digite 0 para sair");

            opcao = scanner.nextInt();

            switch (opcao) {
                case 1:
                    scanner.nextLine();
                    System.out.println("Digite o nome: ");
                    nome = scanner.nextLine();
                    System.out.println("Digite o cpf: ");
                    cpf = scanner.nextInt();
                    scanner.nextLine();
                    System.out.println("Digite o Endereco: ");
                    endereco = scanner.nextLine();
                    System.out.println("Digite o sexo: ");
```

```
                    sexo = scanner.next().charAt(0);
                    System.out.println("Digite o peso: ");
                    peso = scanner.nextFloat();

                    p1 = new Pessoa(nome, endereco, cpf, sexo, peso);
                    e1.setPessoa(p1);
                    break;
                case 2:
                    System.out.println("\nDigite o cpf da pessoa que deseja procurar: ");
                    cpf = scanner.nextInt();
                    Pessoa y = e1.getPessoa(cpf);
                    if (y.getCpf() != 0) {
                        y.imprimir();
                    } else {
                        System.out.println("\nPessoa não existe");
                    }
                    break;
                case 3:
                    System.out.println("O numero de pessoas no evento é: " + Evento.getContPessoa());
                    break;
                case 0:
                    System.out.println("\nSaindo do programa");
                    break;
                default:
                    System.out.println("\nOpção inválida!");
                    break;
            }
        } while (opcao != 0);
        scanner.close();
    }
}
```

# Código em c++

---

# Pessoa.hpp

```
1  #ifndef __PESSOA_HPP // se o header não está definido
2  #define __PESSOA_HPP // define o header
3  #include <string>
4
5  using namespace std;
6
7  class Pessoa
8  {
9      private:
10         int cpf;
11         string endereco;
12         char sexo;
13         string nome;
14         float peso;
15     public:
16         void setNome(string nome);
17         string getNome();
```

```
18
19         void setPeso(float peso);
20         float getPeso();
21
22         void setCPF(int cpf);
23         int getCPF();
24
25         void setSexo(char sexo);
26         char getSexo();
27
28         void setEndereco(string endereco);
29         string getEndereco();
30
31         void imprimir();
32     };
33
34 #endif
```

# Pessoa.cpp

```
1  #include <iostream> //biblioteca
2  #include "Pessoa.hpp"
3
4  using namespace std;
5
6  void Pessoa::setNome(string nome)
7  {
8      this->nome = nome;
9  }
10 string Pessoa::getNome()
11 {
12     return this->nome;
13 }
14
15 void Pessoa::setPeso(float p)
16 {
17     this->peso = p;
18 }
19 float Pessoa::getPeso()
20 {
21     return this->peso;
22 }
```

```
23
24 void Pessoa::setCPF(int cpf)
25 {
26     this->cpf = cpf;
27 }
28 int Pessoa::getCPF()
29 {
30     return this->cpf;
31 }
32
33 void Pessoa::setSexo(char sexo)
34 {
35     this->sexo = sexo;
36 }
37 char Pessoa::getSexo()
38 {
39     return this->sexo;
40 }
41
```



# Pessoa.cpp - Continuação

---

```
42 void Pessoa::setEndereco(string endereco)
43 {
44     this->endereco = endereco;
45 }
46 string Pessoa::getEndereco()
47 {
48     return this->endereco;
49 }
50
51 void Pessoa::imprimir()
52 { //Imprime os dados da Pessoa
53     cout << "-----Pessoa-----" << endl;
54     cout << "Nome: " << this->nome << endl;
55     cout << "sexo: " << this->sexo << endl;
56     cout << "Peso: " << this->peso << endl;
57     cout << "Endereco: " << this->endereco << endl;
58     cout << "CPF: " << this->cpf << endl;
59 }
```

# Evento.hpp

```
1  #include <iostream>
2  #include "Pessoa.hpp"
3
4  using namespace std;
5
6  class Evento {
7  private:
8      string nomeEvento;
9      string dataEvento;
10     string horaEvento;
11     string localEvento;
12     long quantidadeIngressos;
13     float precoIngressos;
14     Pessoa vetPessoa[100];
15
16 public:
17     Evento();
18     static int contPessoa;
19
```

```
20     string getNomeEvento();
21     void setNomeEvento(string nomeEvento);
22
23     string getDataEvento();
24     void setDataEvento(string dataEvento);
25
26     string getHoraEvento();
27     void setHoraEvento(string horaEvento);
28
29     long getQuantidadeIngressos();
30     void setQuantidadeIngressos(long quantidadeIngressos);
31
32     float getPrecoIngressos();
33     void setPrecoIngressos(float precoIngressos);
34
35     Pessoa getPessoa(int cpf);
36     void setPessoa(Pessoa pessoa);
37 };
```

# Evento.cpp

```
1  #include "Evento.hpp"
2
3  Evento::Evento()
4  {
5      for (int i = 0; i < 100; i++)
6          vetPessoa[i].setCPF(0);
7  }
8
9  int Evento::contPessoa = 0;
10
11  string Evento::getNomeEvento()
12  {
13      return nomeEvento;
14  }
15
16  void Evento::setNomeEvento(string nomeEvento)
17  {
18      this->nomeEvento = nomeEvento;
19  }
20
```

```
21  string Evento::getDataEvento()
22  {
23      return this->dataEvento;
24  }
25
26  void Evento::setDataEvento(string dataEvento)
27  {
28      this->dataEvento = dataEvento;
29  }
30
31  string Evento::getHoraEvento()
32  {
33      return this->horaEvento;
34  }
35
36  void Evento::setHoraEvento(string horaEvento)
37  {
38      this->horaEvento = horaEvento;
39  }
40
```

# Evento.cpp - Continuação

```
41 long Evento::getQuantidadeIngressos()  
42 {  
43     return this->quantidadeIngressos;  
44 }  
45  
46 void Evento::setQuantidadeIngressos(long quantidadeIngressos)  
47 {  
48     this->quantidadeIngressos = quantidadeIngressos;  
49 }  
50  
51 float Evento::getPrecoIngressos()  
52 {  
53     return this->precoIngressos;  
54 }  
55  
56 void Evento::setPrecoIngressos(float precoIngressos)  
57 {  
58     this->precoIngressos = precoIngressos;  
59 }  
60
```

```
61 Pessoa Evento::getPessoa(int cpf)  
62 {  
63     Pessoa x;  
64     Pessoa y;  
65     y.setCPF(0);  
66     for (int i = 0; i < contPessoa; i++)  
67     {  
68         x = vetPessoa[i];  
69         if (x.getCPF() == cpf)  
70         {  
71             return x;  
72         }  
73     }  
74  
75     return y;  
76 }  
77
```

# Evento.cpp - Continuação

---

```
77
78 void Evento::setPessoa(Pessoa pessoa)
79 {
80     if (contPessoa < 100)
81     {
82         this->vetPessoa[contPessoa] = pessoa;
83         contPessoa++;
84     }
85     else
86     {
87         cout << "limite de pessoas cadastradas foi atingido.";
88     }
89 }
90
```

# Main.cpp

```
1  #include <iostream>
2  #include "Pessoa.hpp"
3  #include "Evento.hpp"
4
5  int main()
6  {
7      int opcao = 0;
8      string nome, endereco;
9      long cpf;
10     char sexo;
11     float peso;
12     Evento e1;
13     Pessoa p1, y;
14
15     do
16     {
17         cout << "\n\nPrograma de venda de Ingressos para Eventos:" << endl;
18         cout << "O que você deseja fazer" << endl;
19         cout << "Digite 1 para cadastrar Pessoa" << endl;
20         cout << "Digite 2 para imprimir pessoas cadastradas" << endl;
21         cout << "Digite 0 para sair" << endl;
22         cin >> opcao;
23         switch (opcao)
24         {
25             case 1:
26                 cin.ignore();
27                 cout << "Digite o nome:";
28                 getline(cin, nome);
29                 cout << "Digite o CPF:";
30                 cin >> cpf;
```

```
31         cout << "Digite o endereco:";
32         cin.ignore();
33         getline(cin, endereco);
34         cout << "Digite o Sexo:";
35         cin >> sexo;
36         cout << "Digite o Peso:";
37         cin >> peso;
38         p1.setNome(nome);
39         p1.setCPF(cpf);
40         p1.setEndereco(endereco);
41         p1.setSexo(sexo);
42         p1.setPeso(peso);
43         e1.setPessoa(p1);
44         break;
45     case 2:
46         cout << "Digite o cpf da pessoa que deseja imprimir os dados:";
47         cin >> cpf;
48         y = e1.getPessoa(cpf);
49         if (y.getCPF() != 0)
50             y.imprimir();
51         break;
52     case 0: cout << "Saindo do Programa" << endl;
53         break;
54     default:
55         cout << "Opção Inválida" << endl;
56         break;
57     }
58 } while (opcao != 0);
59 return 0;
60 }
```