

Orientação a Objetos 1

Associações

Prof. Dr. Vinícius Camargo Andrade

vcandrade@utfpr.edu.br

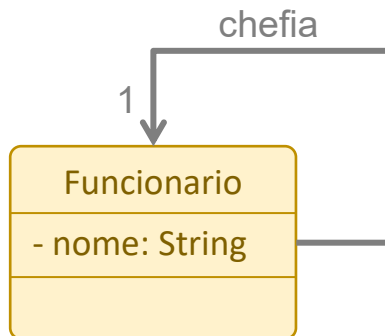
Departamento Acadêmico de Informática
Universidade Tecnológica Federal do Paraná

Associação Unária/Reflexiva

Associações

Associação Unária ou Reflexiva

Ocorre quando existe um relacionamento de um objeto de uma classe com objetos da *mesma classe*.



```
public class Funcionario {

    private String nome;
    private Funcionario funcionario;

    public Funcionario() {

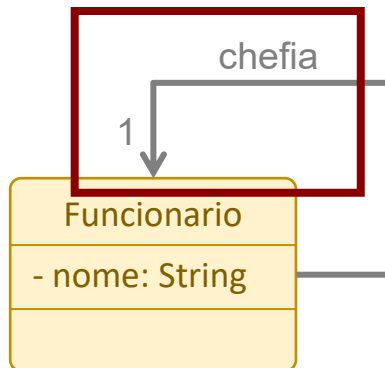
    }

    // métodos da classe

}
```

Associação Unária ou Reflexiva

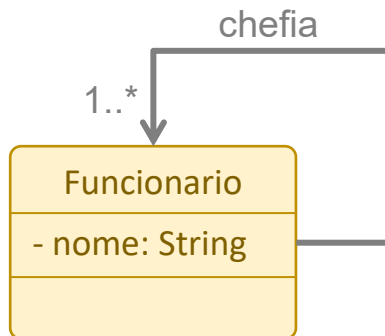
Ocorre quando existe um relacionamento de um objeto de uma classe com objetos da *mesma classe*.



```
public class Funcionario {  
    private String nome;  
    private Funcionario funcionario;  
    public Funcionario() {  
        }  
    // métodos da classe  
}
```

Associação Unária ou Reflexiva

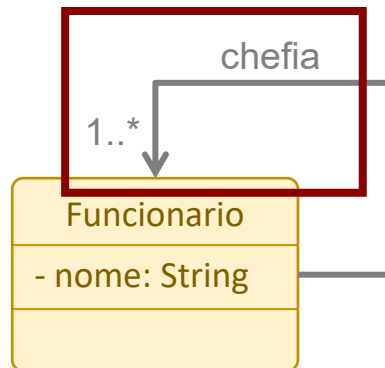
E se um Funcionário puder chefiar de 0 a N Funcionários?



```
public class Funcionario {  
    private String nome;  
    private List <Funcionario> subordinados;  
    public Funcionario() {  
    }  
}
```

Associação Unária ou Reflexiva

E se um Funcionário puder chefiar de 0 a N Funcionários?



```
public class Funcionario {  
    private String nome;  
    private List <Funcionario> subordinados;  
    public Funcionario() {  
    }  
}
```

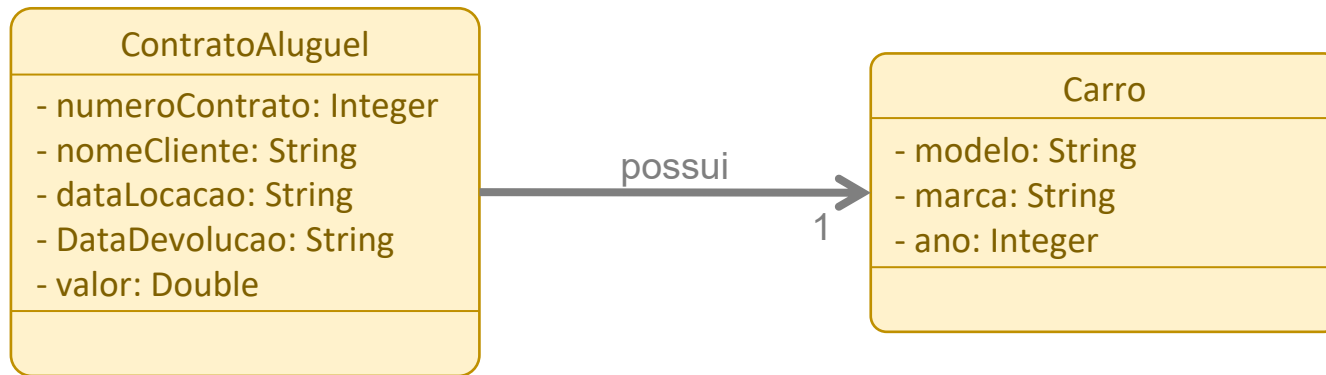
Associação Binária

Associações

Associação Binária

*Ocorre quando são identificados relacionamento entre objetos de **duas classes distintas**.*

Associação Binária



```
public class ContratoAluguel {

    private int numeroContrato;
    private String nomeCliente;
    private String dataLocacao;
    private String dataDevolucao;
    private double valor;
    private Carro carro;

    public int getNumeroContrato() {
        return numeroContrato;
    }

    public void setNumeroContrato(int numeroContrato) {
        this.numeroContrato = numeroContrato;
    }

}
```

```
public class Carro {

    private String modelo;
    private String marca;
    private int ano;

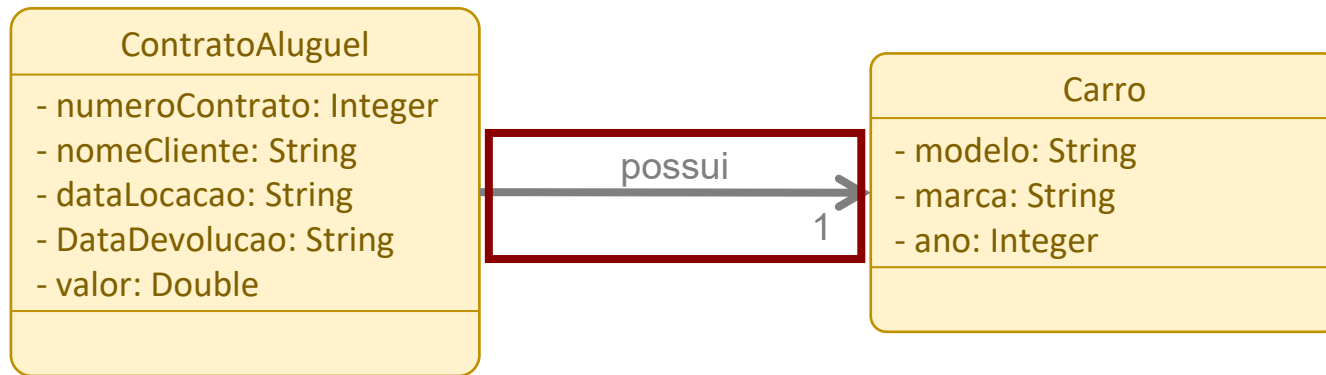
    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public int getAno() {

}
```

Associação Binária



```
public class ContratoAluguel {

    private int numeroContrato;
    private String nomeCliente;
    private String dataLocacao;
    private String dataDevolucao;
    private double valor;
    private Carro carro;

    public int getNumeroContrato() {
        return numeroContrato;
    }

    public void setNumeroContrato(int numeroContrato) {
        this.numeroContrato = numeroContrato;
    }
}
```

```
public class Carro {

    private String modelo;
    private String marca;
    private int ano;

    public String getModelo() {
        return modelo;
    }

    public void setModelo(String modelo) {
        this.modelo = modelo;
    }

    public int getAno() {
    }
```

Agregação

Associações

Agregação

*É um tipo de associação em que se tenta demonstrar que as informações de um objeto **precisam ser complementadas** pelas informações contidas em um ou mais objetos de outra classe.*

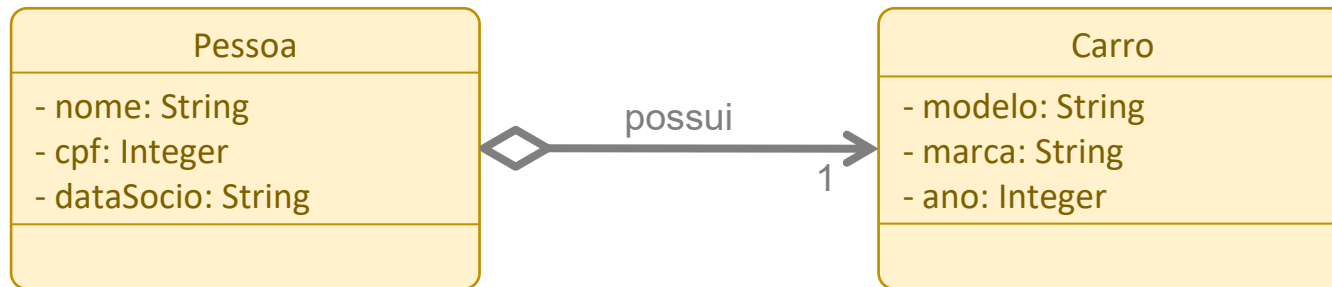
Agregação

*Este tipo de associação tenta demonstrar um relação **todo/parte** entre os objetos associados.*

Agregação

*Isso significa que a **parte** de um tipo *A* está contida em um tipo *B*, quando esse tem relação de agregação entre eles, porém, essa mesma parte ***A não existe somente para compor B***, essa parte pode agregar outros tipos.*

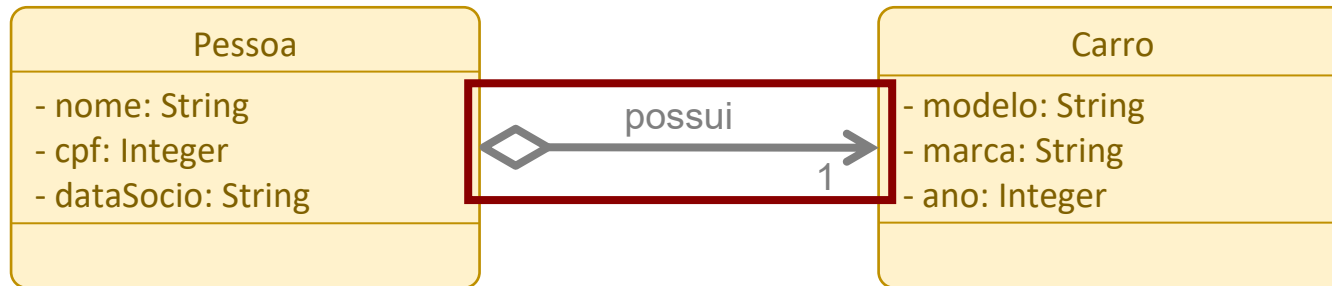
Agregação – Exemplo 1



```
public class Pessoa {  
    private String nome;  
    private int cpf;  
    private int cnh;  
    private Carro carro;  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
}
```

```
public class Carro {  
    private String modelo;  
    private String marca;  
    private int ano;  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
}
```

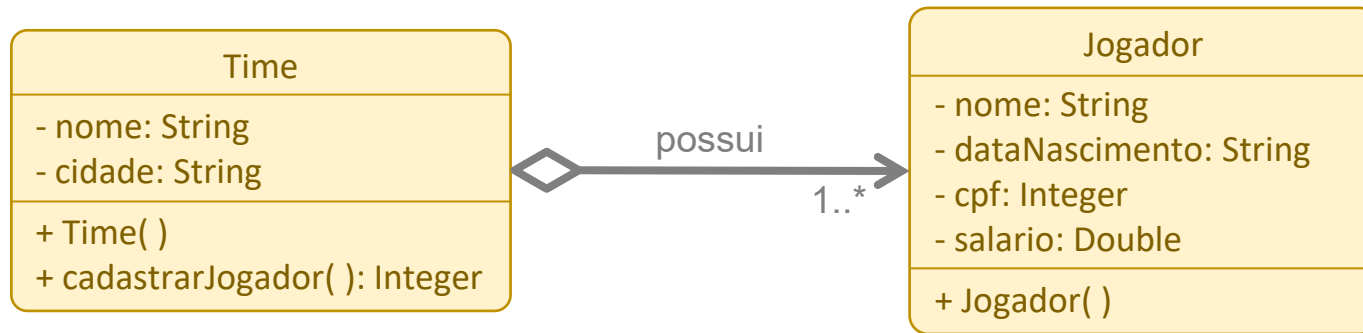
Agregação – Exemplo 1



```
public class Pessoa {  
    private String nome;  
    private int cpf;  
    private int cnh;  
    private Carro carro;  
    public String getNome() {  
        return nome;  
    }  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
}
```

```
public class Carro {  
    private String modelo;  
    private String marca;  
    private int ano;  
    public String getModelo() {  
        return modelo;  
    }  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
}
```

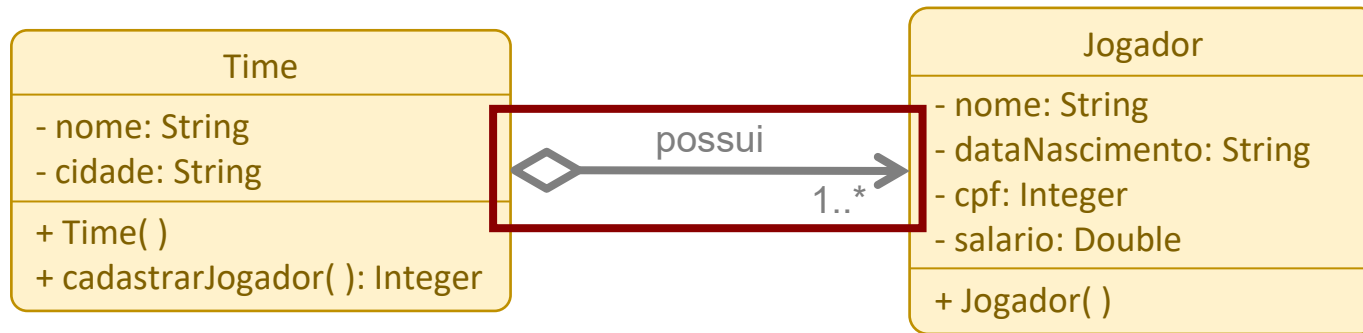

Agregação – Exemplo 2



```
public class Time {  
  
    private String nome;  
    private String cidade;  
    private List<Jogador> jogadores;  
  
    public Time() {  
    }  
  
    public int cadastrarJogador() {  
        return 0;  
    }  
}
```

```
public class Jogador {  
  
    private String nome;  
    private String dataNascimento;  
    private int cpf;  
    private double salario;  
  
    public Jogador() {  
    }  
  
    public String getNome() {  
        return nome;  
    }  
}
```

Agregação – Exemplo 2



```
public class Time {  
    private String nome;  
    private String cidade;  
    private List<Jogador> jogadores;  
    public Time() {  
    }  
    public int cadastrarJogador() {  
        return 0;  
    }  
}
```

```
public class Jogador {  
    private String nome;  
    private String dataNascimento;  
    private int cpf;  
    private double salario;  
    public Jogador() {  
    }  
    public String getNome() {  
        return nome;  
    }  
}
```

Composição

Associações

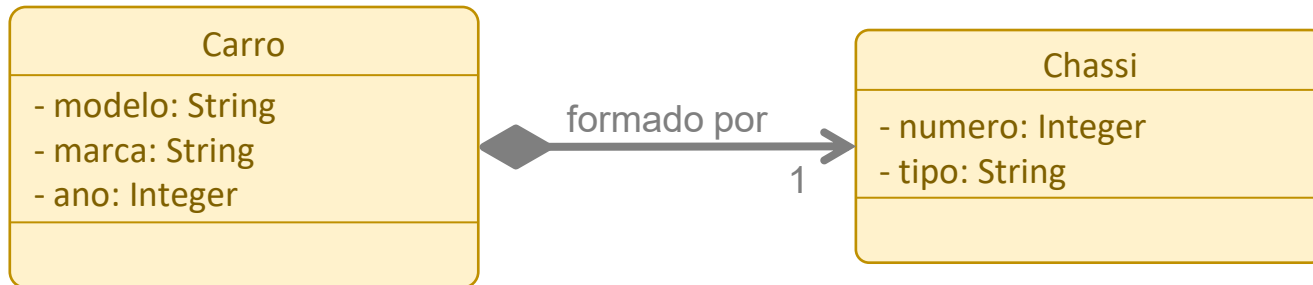
Composição

*Associação do tipo composição é uma variação da agregação, porém é apresentado um **vínculo mais forte entre os objeto-todo e os objetos-parte**, procurando demonstrar que os objetos-parte tem de estar associados a um único objeto-todo.*

Composição

*Em uma **composição** os objetos-parte não podem ser destruídos por um objeto diferente do objeto-todo ao qual estão relacionados.*

Composição – Exemplo 1



```
public class Carro {  
  
    private String modelo;  
    private String marca;  
    private int ano;  
    private Chassi chassi;  
  
    public String getModelo() {  
        return modelo;  
    }  
  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
}
```

```
public class Chassi {  
  
    private int numero;  
    private String tipo;  
  
    public Chassi() {  
    }  
  
    public int getNumero() {  
        return numero;  
    }  
  
    public void setNumero(int numero) {  
        this.numero = numero;  
    }  
}
```

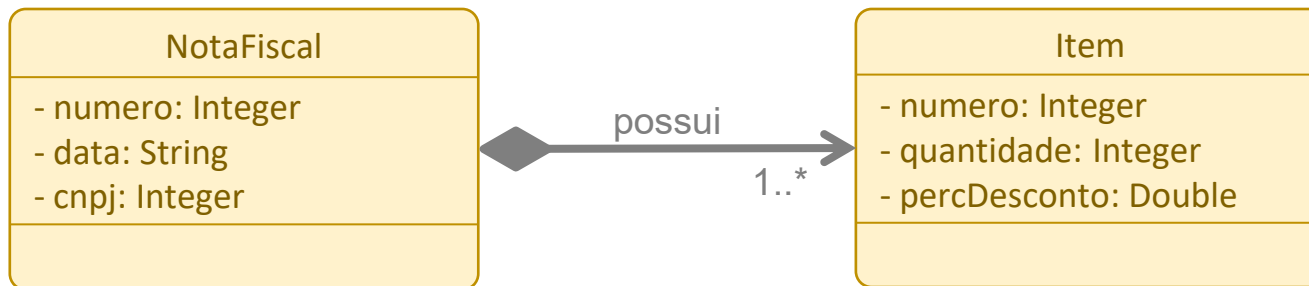
Composição – Exemplo 1



```
public class Carro {  
    private String modelo;  
    private String marca;  
    private int ano;  
    private Chassi chassi;  
    public String getModelo() {  
        return modelo;  
    }  
    public void setModelo(String modelo) {  
        this.modelo = modelo;  
    }  
}
```

```
public class Chassi {  
    private int numero;  
    private String tipo;  
    public Chassi() {  
    }  
    public int getNumero() {  
        return numero;  
    }  
    public void setNumero(int numero) {  
        this.numero = numero;  
    }  
}
```

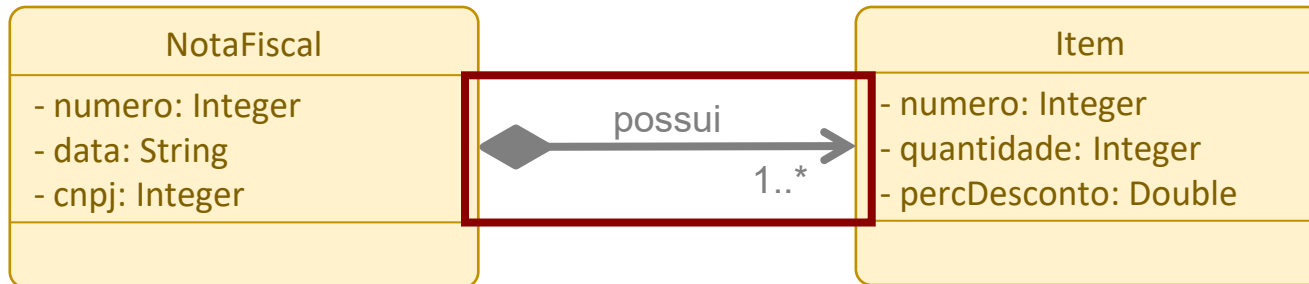
Composição – Exemplo 2



```
public class NotaFiscal {  
  
    private int numero;  
    private String data;  
    private int cnpj;  
    private List<Item> itens;  
  
    public NotaFiscal() {  
  
    }  
  
    public int getNumero() {  
        return numero;  
    }  
}
```

```
public class Item {  
  
    private int numero;  
    private int quantidade;  
    private double percDesconto;  
  
    public Item() {  
  
    }  
  
    public int getNumero() {  
        return numero;  
    }  
}
```


Composição – Exemplo 2



```
public class NotaFiscal {  
    private int numero;  
    private String data;  
    private int cnpj;  
    private List<Item> itens;  
    public NotaFiscal() {  
    }  
    public int getNumero() {  
        return numero;  
    }  
}
```

```
public class Item {  
    private int numero;  
    private int quantidade;  
    private double percDesconto;  
    public Item() {  
    }  
    public int getNumero() {  
        return numero;  
    }  
}
```

Associação, Agregação e Composição

Comparação

Associação, Agregação e Composição

Considere um cenário em que tem-se duas classes: “A” e “B” e precisa definir qual é o relacionamento entre elas.

- 1. Se a classe “A” for excluída, precisa excluir também a classe “B”?*
 - *Sim* = relacionamento de composição;
 - *Não* = Pergunta 2.
- 2. A classe “B” tem alguma utilidade sozinha?*
 - *Sim* = associação comum;
 - *Não* = relacionamento de agregação.

Classe Associativa

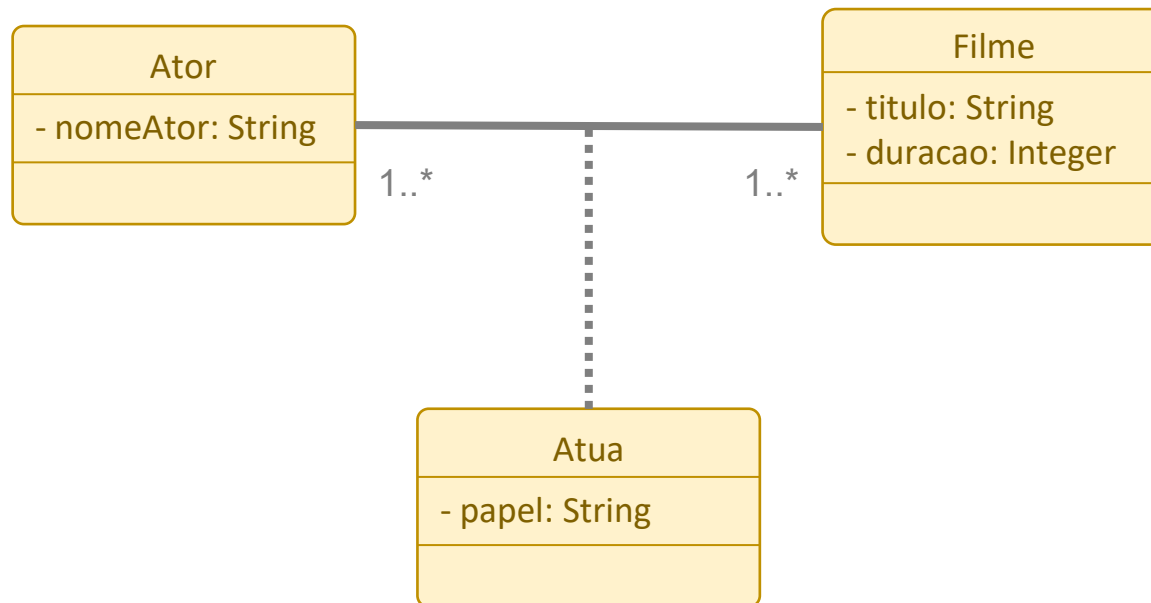
Classe Associativa

São classes produzidas quando da ocorrência de associações que tenham multiplicidade muito () em todas as suas extremidades.*

Classe Associativa

As classes associativas são necessárias nos casos em que existem atributos relacionados à associação que não podem ser armazenados por nenhuma das classes envolvidas.

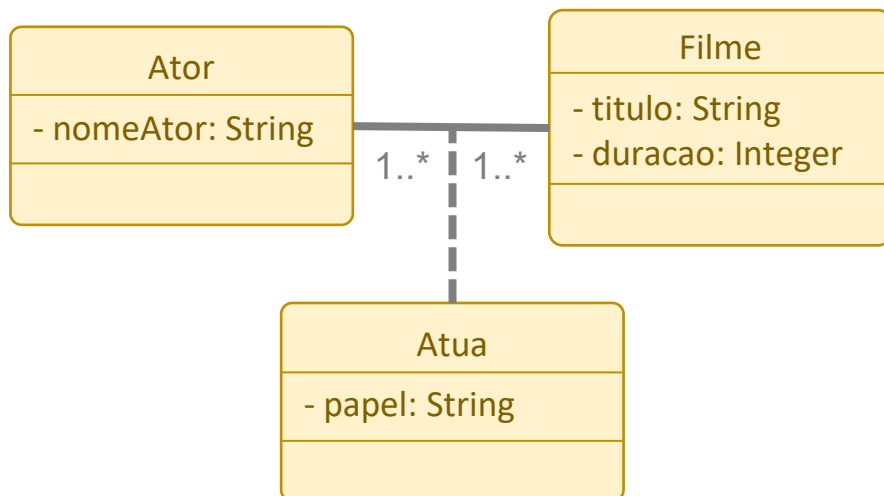
Classe Associativa



Classe Associativa

```
public class Ator {  
  
    private String nomeAtor;  
  
    public String getNomeAtor() {  
        return nomeAtor;  
    }  
  
    public void setNomeAtor(String nomeAtor) {  
        this.nomeAtor = nomeAtor;  
    }  
}
```

```
public class Filme {  
  
    private String tituloFilme;  
    private int duracaoFilme; // em minutos  
  
    public String getTituloFilme() {  
        return tituloFilme;  
    }  
  
    public void setTituloFilme(String tituloFilme) {  
        this.tituloFilme = tituloFilme;  
    }  
}
```



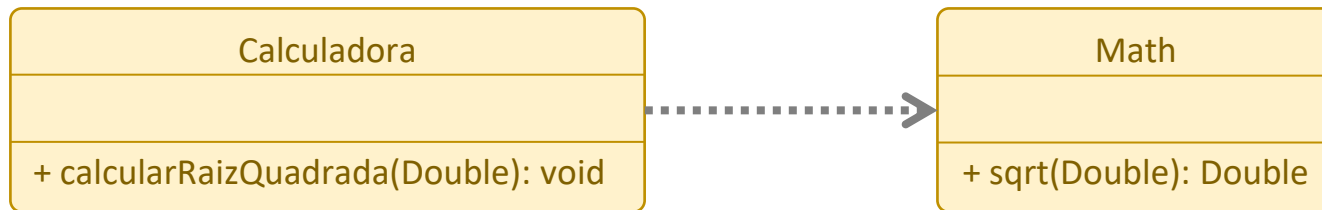
```
public class Atua {  
  
    private Ator ator;  
    private Filme filme;  
    private String papel;  
  
    public Ator getAtor() {  
        return ator;  
    }  
  
    public void setAtor(Ator ator) {  
        this.ator = ator;  
    }  
}
```


Dependência

Dependência

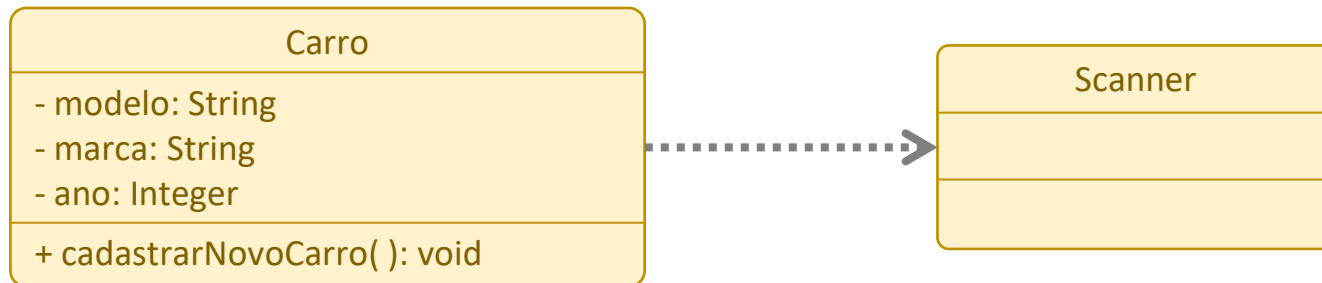
*Indica o grau de dependência entre um classe à outra.
Uma alteração realizada na classe independente, afeta
diretamente a classe dependente.*

Dependência



```
public class Calculadora {  
    public void calcularRaizQuadrada(double x) {  
        double resultado = Math.sqrt(x);  
        System.out.println("Raiz Quadrada = " + resultado);  
    }  
}
```

Dependência



```
public class Carro {  
  
    private String modelo;  
    private String marca;  
    private int ano;  
  
    public void cadastrarNovoCarro() {  
  
        Scanner input = new Scanner(System.in);  
  
        // solicitar informações para o usuário.  
    }  
  
    public String getModelo() {  
        return modelo;  
    }  
}
```

Realização

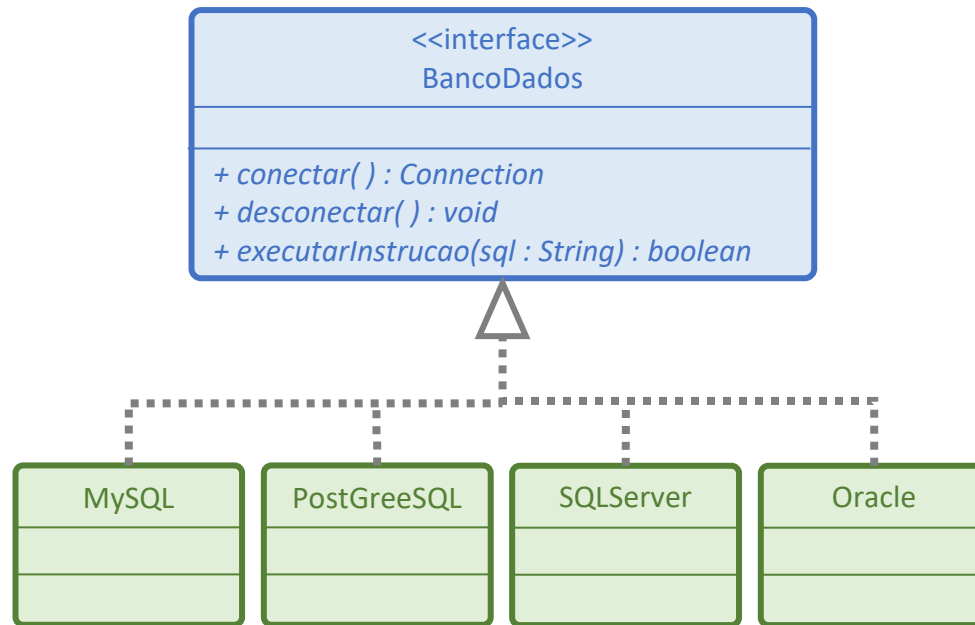
Realização

O conceito de realização é utilizado para identificar classes responsáveis por executar funções para outras classes.

Realização





Este tipo de relacionamento herda o comportamento de uma classe, mas não sua estrutura.

Realização

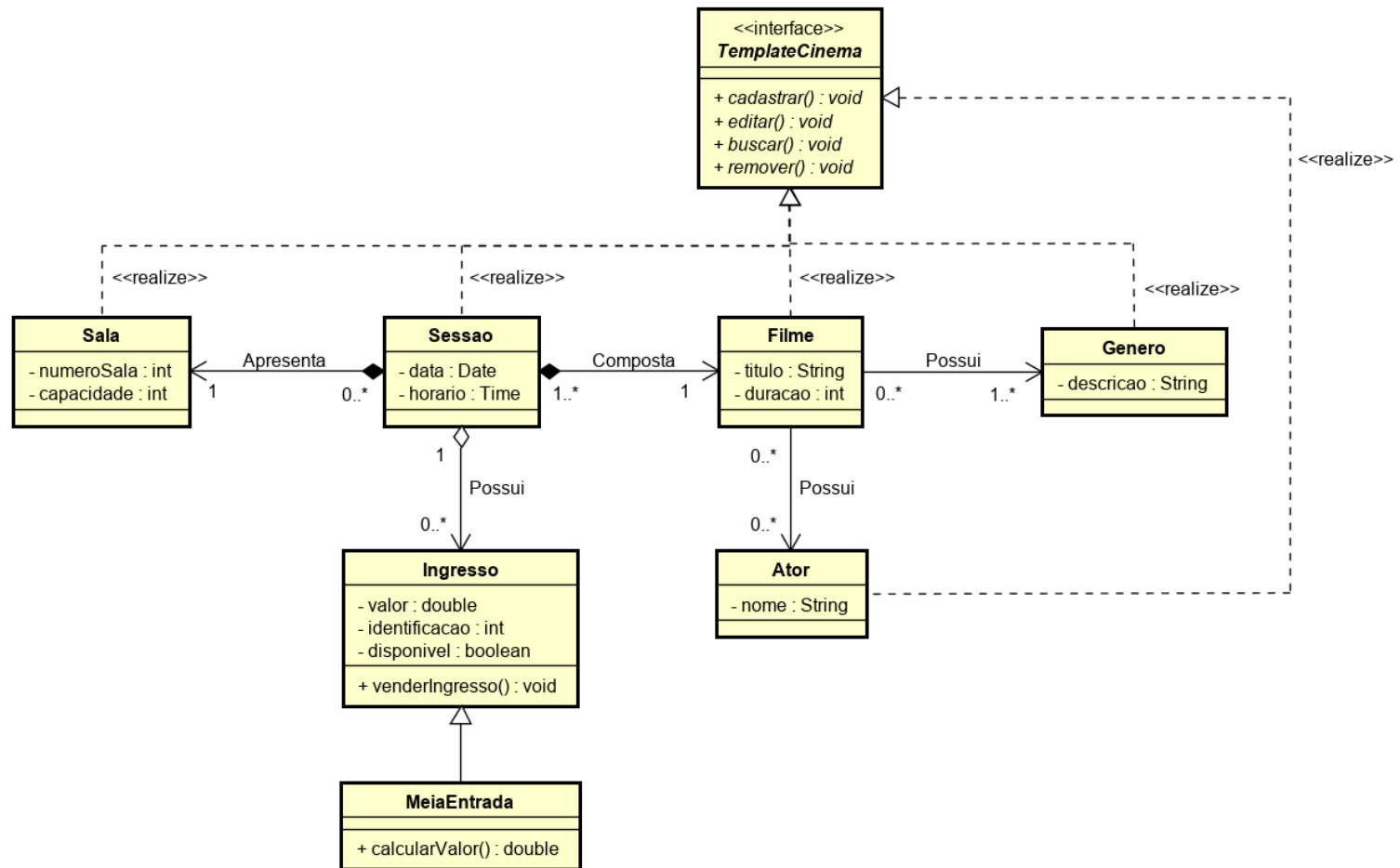


Resumo

Resumo de Notações

	Associação
	Agregação
	Composição
	Classe Associativa
	Dependência
	Realização

Exemplo



Exercício 1

Exercício 1

Implemente o modelo do slide seguinte. Os atributos devem ser inicializados pelo método construtor da respectiva classe. O método imprimirDados() deve imprimir todos os dados de Pessoa e Endereço.

Exercício 1

