Mapeando relacionamento entre classese noTypeScript

Prof. Dr. Anderson Rodrigues

Link com material:

https://drive.google.com/drive/folders/1TKG_7kWpPFoUzhwSsniYzIYR_1I2AFwi?usp=sharing

Tipos de relacionamento entre classes

- **ASSOCIAÇÃO**:
  - É o relacionamento mais comum entre duas classes, onde uma classe utiliza os serviços de outra classe temporariamente. Pode ser uma associação unidirecional ou bidirecional.

- **AGREGAÇÃO**
  - É um tipo de associação onde uma classe é composta por outras classes, mas as classes associadas podem existir independentemente da classe principal. Por exemplo, *Uma biblioteca pode ter vários livros em seu acervo. Os livros podem ser adicionados, removidos e emprestados, independentemente da biblioteca em que estão.*

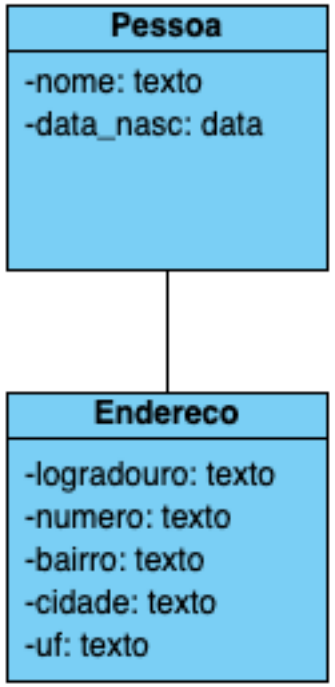
- **COMPOSIÇÃO**:
  - É semelhante à agregação, mas com uma diferença crucial: a classe principal é responsável pela criação e destruição das classes associadas. Por exemplo, Uma casa é composta por diferentes cômodos, como quartos, sala, cozinha, etc. Se a casa for destruída, os cômodos também serão.

- **HERANÇA**:
  - É um relacionamento onde uma classe (subclasse ou classe filha) herda atributos e métodos de outra classe (superclasse ou classe pai). A subclasse pode estender o comportamento da superclasse adicionando novos métodos ou substituindo métodos existentes.

Senac FACULDADE

Mapeamento para o TypeScript

- **ASSOCIAÇÃO**:

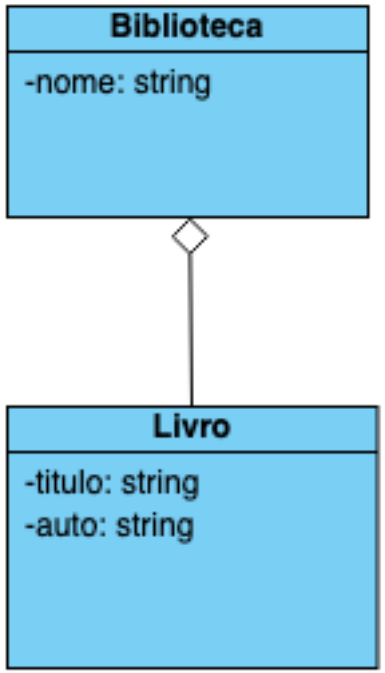



```
class Pessoa {

    private _nome: string;
    private _data_nasc: Date;
    private _endereco: Endereco;

    constructor(
        nome: string,
        data_nasc: Date,
        endereco: Endereco) {
            this._nome = nome;
            this._data_nasc = data_nasc;
            this._endereco = endereco;
    }
}

class Endereco {
    // Atributos e métodos do endereço
    private _logradouro: string;
    private _numero: string;
    private _bairro: string;
    private _cidade: string;
    private _uf: string;
}
```

Senac FACULDADE

Mapeamento para o TypeScript

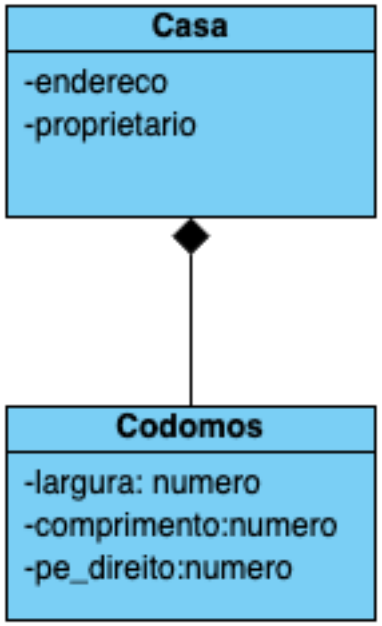
- **AGREGAÇÃO**:




```
class Biblioteca {

  livros: Livro[];

  constructor(livros: Livro[]) {
    this.livros = livros;
  }

}


class Livro {

  // Atributos e métodos do livro
  private _titulo: string;
  private _autor: string;

  constructor(titulo: string, autor: string) {
    this._titulo = titulo;
    this._autor = autor;
  }

}
```

Mapeamento para o TypeScript

- **COMPOSIÇÃO**:

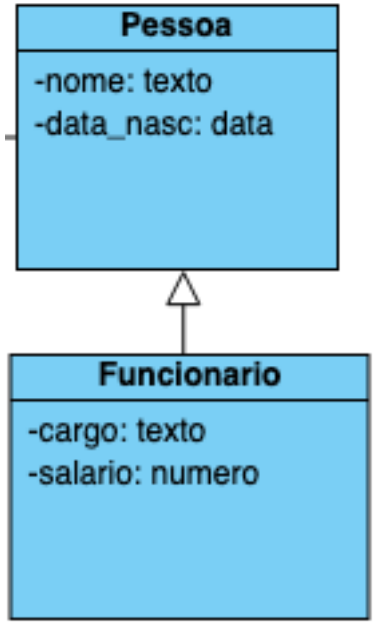



```
class Casa {
  private _quartos: Comodo[];
  private _sala: Comodo;
  private _cozinha: Comodo;

  constructor(
        quartos:Comodo[],
        sala:Comodo,
        cozinha:Comodo) {
              this._quartos = quartos;
              this._sala = sala;
              this._cozinha = cozinha;
        }
  }

class Comodo {
  // Atributos e métodos do cômodo
  private _largura: number;
  private _comprimento: number;
  private _pe_direito: number;

  constructor(
        largura:number,
        comprimento: number,
        pe_direito: number) {
              this._largura = largura;
              this._comprimento = comprimento;
              this._pe_direito = pe_direito;
  }
}
```

Senac FACULDADE

Mapeamento para o TypeScript

- **HERANÇA**:




```typescript
class Pessoa {
  private _nome: string;
  private _data_nasc: Date;
  private _endereco: Endereco;

  constructor(
     nome: string,
     data_nasc: Date,
     endereco: Endereco) {
        this._nome = nome;
        this._data_nasc = data_nasc;
        this._endereco = endereco;
  }
}

class Funcionario extends Pessoa {
  private _cargo: string;
  private _salario: number;

  constructor(
    nome: string,
    data_nasc: Date,
    endereco: Endereco,
    cargo: string,
    salario: number) {
        super(nome, data_nasc, endereco);
        this._cargo = cargo;
        this._salario = salario;
    }
}
```

Senac FACULDADE