Universidade Federal do Ceará - Campus de Quixadá

Programação para Design

Herança

Prof. Anderson Lemos

Introdução

- Até agora vimos o conceito de objetos que possuem outros objetos em seu interior
  - A esse tipo de relacionamento, chamamos de **tem-um** (has-a)
  - Associação (agregação e composição)

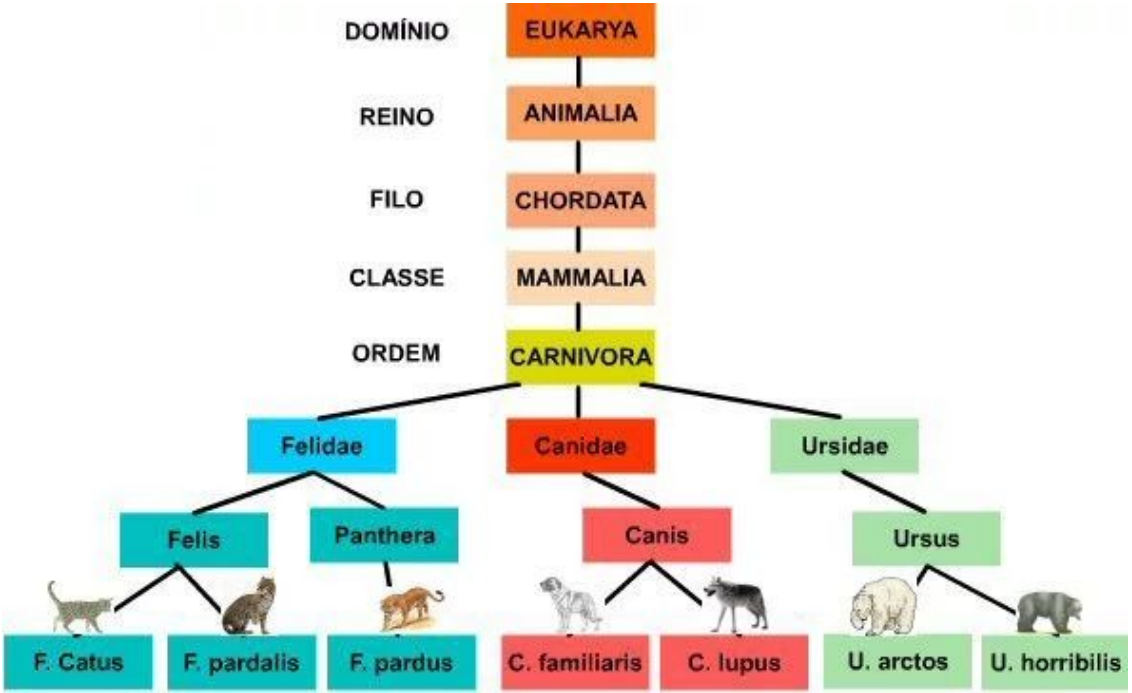
```
class Aluno {
    private disciplinas : Array<Disciplina>;
}
```

Herança

- Herança é um dos pilares da Programação Orientada a Objetos
  - Assim como abstração, encapsulamento e **polimorfismo**.
- A herança relaciona as classes semelhante a uma árvore genealógica ou uma taxonomia na biologia (Linnaeus).

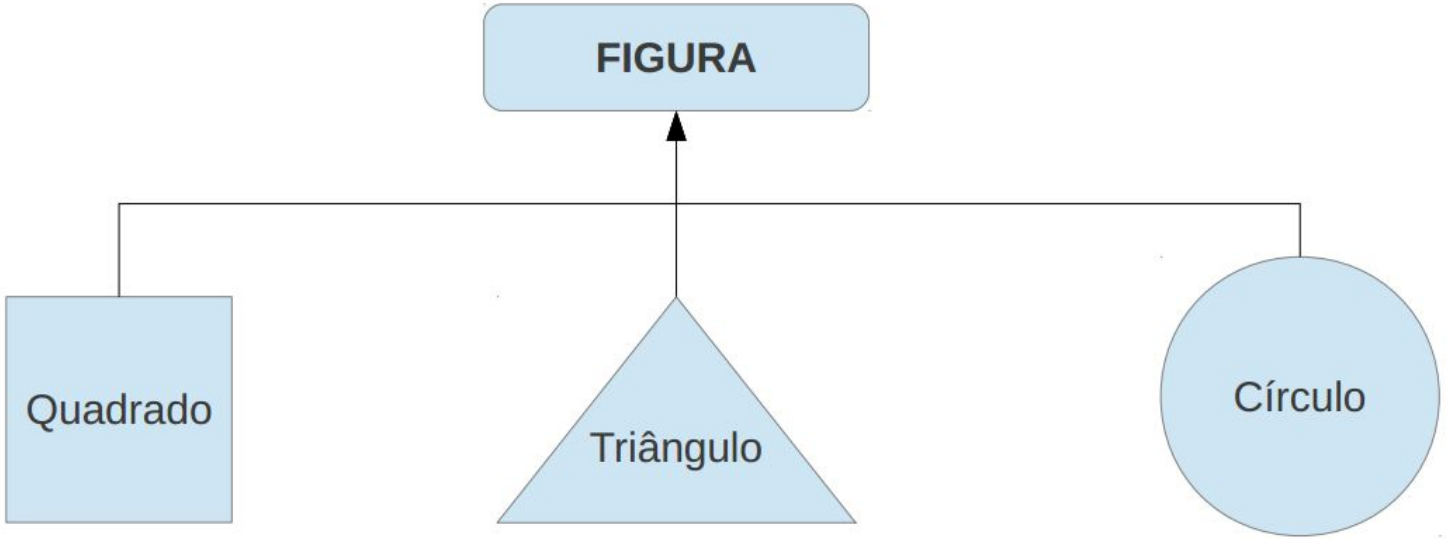
Herança

- Taxonomia



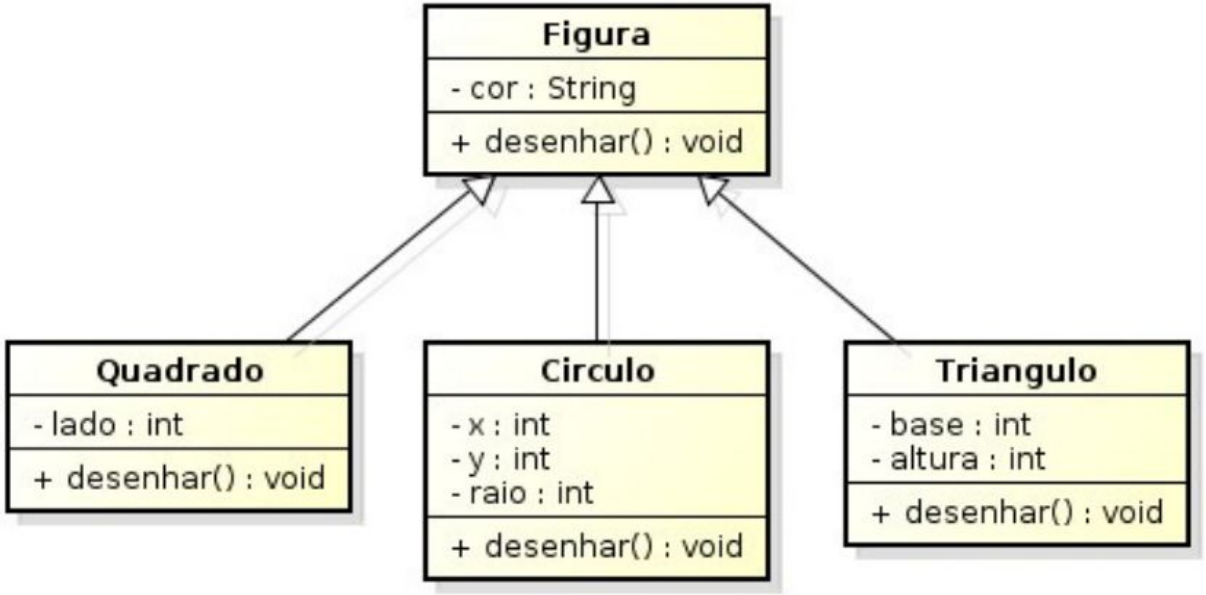
Herança

- A herança permite a uma classe “herdar” atributos e métodos de outra classe, reutilizando seu código.



Herança

- Em UML...



Herança

- Uma classe que herda de outra classe é chamada de classe “Filha”.
- No nosso exemplo, Figura é “Mãe” de Círculo, Quadrado e Triângulo.
- As classes Filhas também podem ser chamada de “subtipos” de Figura (“supertipo”).
- Esse relacionamento é chamado de “**é-um** (is-a)”.
- Usamos a palavra reservada extends (do inglês, estende)
  - Usamos `export class A extends B` , isto é, a classe **B** estende **A**, ou seja, **B** herda de **A**.

Herança

- Especificação
  - Capacidade das classes filhas de recriarem comportamentos específicos, restritos. Posição inferior da hierarquia.
- Generalização
  - É a representação genérica encontrada em uma classe Mãe. Quanto mais genérica, mais abrangente. Posição superior da hierarquia.

Herança

- Herança Simples
  - Quando uma classe herda apenas de uma única classe mãe.

- Herança Múltipla
  - Quando uma classe tem mais de uma classe mãe. Em TypeScript não existe herança múltipla, ao contrário de C++.

Herança

- Exercício


```
class Transporte {
    private capacidade : number;
}

class Terrestre extends Transporte {
    private numRodas : number;
}

class Automovel extends Terrestre {
    private cor : string;
    private numPortas : number;
    private placa : string;
}
```

Herança

- Exercício: Considerando o código das classes Transporte, Terrestre e Automóvel apresentados na Figura anterior, finalize a implementação delas, adicionando os métodos get() e set() para cada um de seus atributos. Em seguida, crie um arquivo Principal que cria um objeto da classe Automóvel, e chama todos os métodos set() e get() criados, inclusive das classes Transporte e Terrestre. Observe no seu exemplo, que é possível chamar todos os métodos get() e set() herdados pela classe Automóvel.

Herança

- Comando **super**
  - Palavra reservada super refere-se a classe ancestral(mãe) imediata da classe
  - Pode ser usada nos construtores para chamar os construtores das classes mãe em cascata.

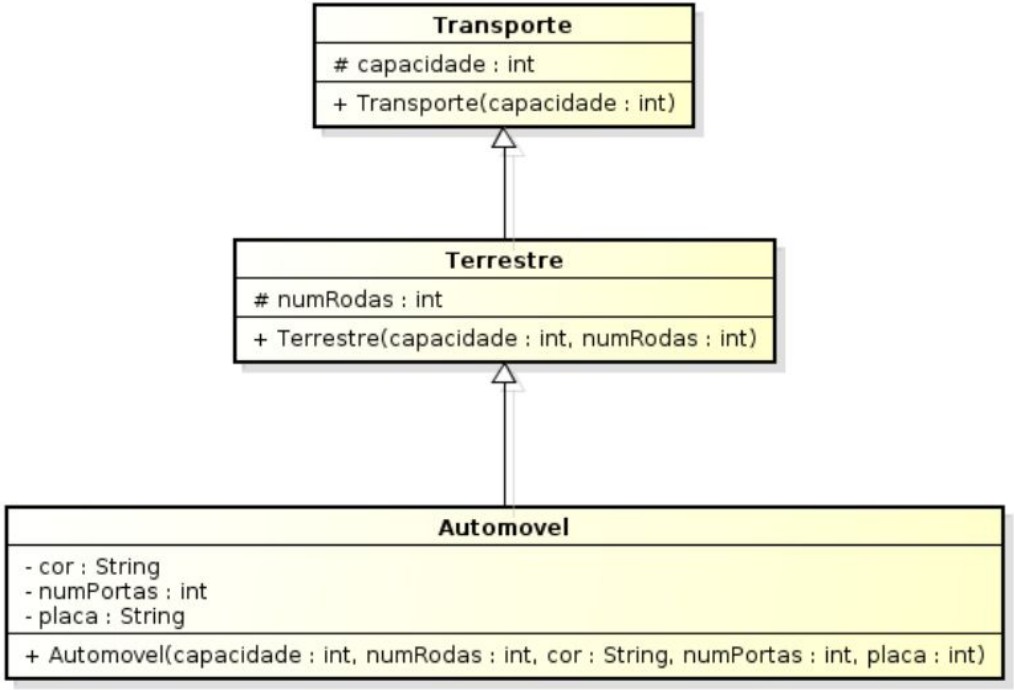
```
class Transporte{
    private capacidade : number;
    public constructor(capacidade : number){
        this.capacidade = capacidade;
    }
}
```

```
class Terrestre extends Transporte{
    private numRodas : number;
    public constructor(capacidade : number,
                       numRodas : number){
        super(capacidade);
        this.numRodas = numRodas;
    }
}
```

```
class Automovel extends Terrestre{
    private cor : string;
    private numPortas : number;
    private placa : string;
    public constructor(capacidade : number,
                       numRodas : number,
                       cor : string,
                       numPortas : number,
                       placa : string){
        super(capacidade, numRodas);
        this.cor = cor;
        this.numPortas = numPortas;
        this.placa = placa;
    }
}
```

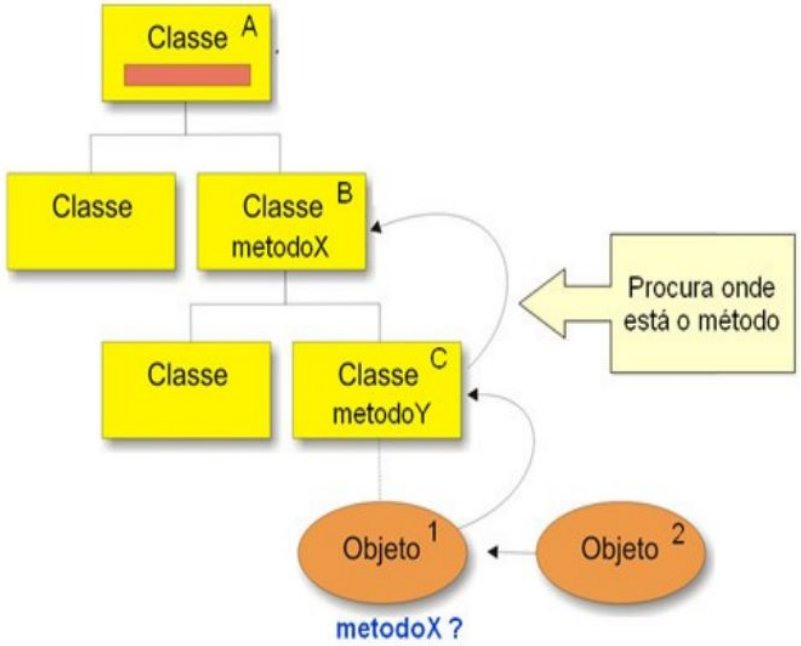
Herança

- Super e construtores



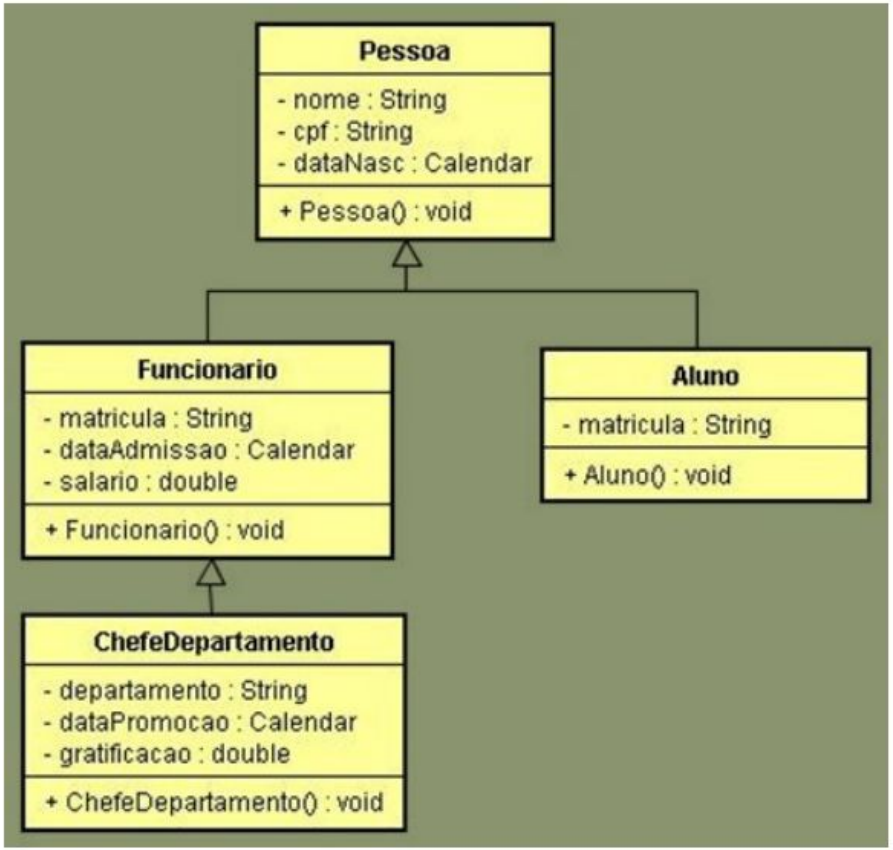
Herança

- Propagação de métodos



Herança

- Exercício:



Herança

- Exercício:
  - Acrescente aos construtores a lista de parâmetros necessária para instanciar o objeto. Por exemplo, a classe Pessoa deve ter nome, CPF e dataNasc. E essa lista é acumulativa, ou seja, o construtor da classe Funcionário deve ter a lista de seus atributos mais os atributos necessários para a classe Pessoa. Dica: não deixe de usar a palavra-chave super em cada um dos construtores para chamar o construtor da classe mãe, passando os atributos que são mantidos por ela e seus ancestrais.
  - Insira os seguintes métodos, para apresentar os valores dos atributos das classes, mostrarPessoa(), mostrarFuncionario(), mostrarChefe() e mostrarAluno(), respectivamente, às classes Pessoa, Funcionário, ChefeDepartamento e Aluno. Para imprimir os atributos, use o método console.log() em cada um dos métodos.
  - Crie um arquivo TestaTudo.ts, que instancia um objeto de cada uma das classe e exibe os valores dos atributos através de chamadas aos métodos mostrarPessoa(), mostrarFuncionario(), mostrarChefe() e mostrarAluno().

Perguntas?