.StartSe
<tech/academy>

Conceitos e Pilares da POO

Conceito de Classe em POO

- A classe funciona como um template ou modelo que define os atributos e os métodos de cada objeto, como por exemplo:
    - Uma classe livro pode definir as características e os métodos de um livro
        - a partir dessa definição, vários objetos do tipo livro podem ser criados (instanciados)
            - cada objeto representa um livro em particular

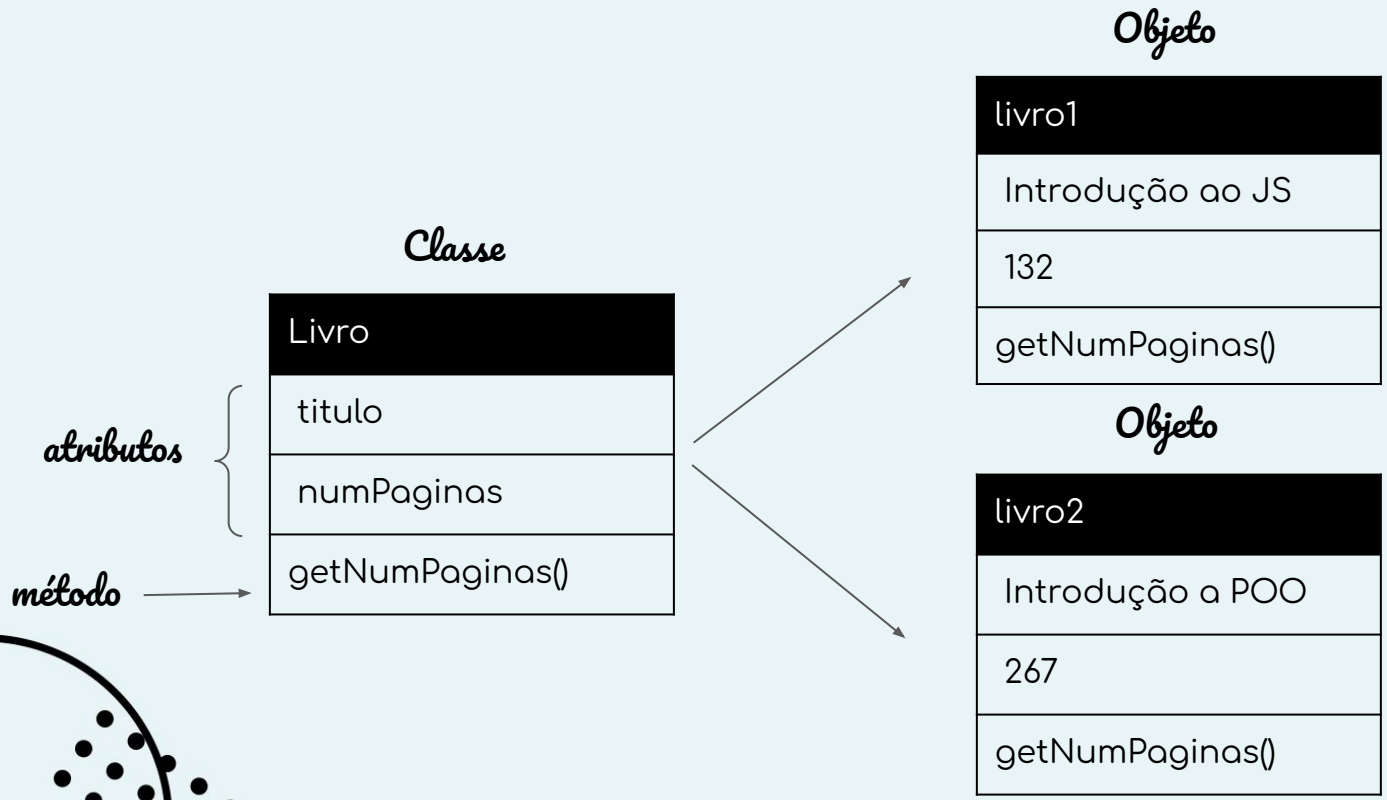
Conceito de Objeto em POO

- Um objeto é uma **abstração** de elementos do mundo real que possuem características próprias como por exemplo:
  - pessoa (nome, cpf, endereço, telefone, etc)
  - filme (título, elenco, ano, idioma, etc)
  - carro (marca, modelo, ano, potência, etc)

Conceito de Objeto em POO

- Em POO cada **objeto** possui:
  - **atributos** (características como nome, modelo, título, código, etc)
  - **métodos** (funções que são associadas aos objetos - determinam o comportamento do objeto)

Conceito de Classe e Objeto em POO



.StartSe
<tech/academy>

Instância de uma Classe

- Como vários objetos podem ser criados a partir de uma classe, dizemo que:
    - cada **objeto** é uma **instância** de uma **classe**
    - o ato de **criar** um objeto é conhecido como **instanciar** um objeto

Pilares da POO em JavaScript

- Encapsulamento
- Herança
- Polimorfismo

Encapsulamento

- O encapsulamento é uma forma de **esconder** ou **ocultar** os dados de uma classe
    - permite proteger o acesso aos atributos (propriedades) de um objeto
    - ao invés de acessar um atributo diretamente, por exemplo, é necessário utilizar um método para atribuir ou recuperar algum dado

Herança

- É uma forma de **aproveitar atributos e métodos** de uma classe em outra classe derivada
    - neste caso, uma classe recebe por **herança** atributos e métodos de uma outra classe
        - a classe que passa a herança para uma outra classe é chamada de **superclasse**
        - a classe que recebe a herança é chamada de **subclasse**

Herança

- A **superclasse** reúne os atributos e métodos **comuns** a todas as **subclasses**
- As **subclasses** reúnem os atributos e métodos **específicos** para cada contexto
    - Exemplo na próxima página

Herança

```
class Produto {
    #tipo;
    constructor(tipo) {
        this.#tipo = tipo;
    }
    getDados() {
        return this.#tipo;
    }
}
```

```
class Livro extends Produto {
    #titulo;
    #numPag;
    constructor(tipo,titulo,numPag) {
        super(tipo);
        this.#titulo = titulo;
        this.#numPag = numPag;
    }
    getDados() {
        return(`
        Tipo: ${super.getDados()}
        Titulo: ${this.#titulo}
        Num.Pág.: ${this.numPag}
        `);
    }
}
```

```
const prod = new Produto('Generico');
console.log(prod.getDados());
const liv = new Livro('Livro','POO',50);
console.log(liv.getDados());
```

.StartSe
<tech/academy>

Polimorfismo

- Quando uma subclasse e uma superclasse possuem métodos com o mesmo nome, porém com códigos que produzem resultados diferentes