.StartSe
<tech/academy>

# Composição

# Composição

- Além da herança, uma classe pode ser composta ou possuir objetos de outras classes através da **composição**
- Assim como a herança, a **composição** também permite **reuso** de código de uma outra classe

# Composição

- No mundo real, existem elementos complexos compostos de elementos mais simples
    - um avião é composto de (ou possui): turbinas, assentos, iluminação, etc
- Em POO, uma classe pode ser composta (possuir) um ou mais objetos de outras classes

# Composição

vetor que armazena os objetos da classe Produto

método para adicionar os objetos ao vetor

```
1  class Loja {
2      #nome;
3      #produtos = [];
4      constructor(nome){
5          this.#nome = nome;
6      }
7      adicionaProduto(produto) {
8          this.#produtos.push(produto);
9      }
10     exibeProdutos() {
11         console.log(`Lista de Produtos da Loja ${this.#nome}`)
12         for(const produto of this.#produtos) {
13             console.log(produto.getDados());
14         }
15     }
16 }
```

utilização do método getDados da classe Produto dentro da classe Loja

```
1  class Produto {
2      #tipo;
3      constructor(tipo) {
4          this.#tipo = tipo;
5      }
6      getDados() {
7          return this.#tipo;
8      }
9  }
```

A classe Loja é composta ou possui objetos da classe Produto

# Composição

```
1  const prod = new Produto('Generico');
2  const liv = new Livro('Livro','POO',50);
3  const liv2 = new Livro('Livro','JavaScript',120);
4
5  const minhaLoja = new Loja('MinhaLoja');
6  minhaLoja.adicionaProduto(prod);
7  minhaLoja.adicionaProduto(liv);
8  minhaLoja.adicionaProduto(liv2);
9  minhaLoja.exibeProdutos();
```

Saída no console:

```
Lista de Produtos da Loja MinhaLoja

        Tipo: Generico

        Tipo:
        Tipo: Livro
        Titulo: POO
        Num.Pág.: 50

        Tipo:
        Tipo: Livro
        Titulo: JavaScript
        Num.Pág.: 120
```

como Livro é uma subclasse da superclasse Produto seus objetos também foram adicionados ao objeto minhaLoja e seus dados foram exibidos através do método getDados utilizando o conceito de polimorfismo

.StartSe
<tech/academy>

# Composição

- Objetos de outras classes podem ser instanciados dentro de uma classe de interesse, permitindo o acesso aos métodos disponibilizados por este objeto (reuso de código)

# Composição

- No exemplo abaixo, a classe **CupomDesconto** é responsável por estabelecer os valores de cupons de desconto para clientes de acordo com o nível de gastos

```
class CupomDesconto {
    #desconto1 = 10;
    #desconto2 = 20;
    constructor() { }
    getDesconto(gastos) {
        if (gastos >= 1000 && gastos <= 3000) {
            console.log(`Você tem direito a ${this.#desconto1}% de descontos!`);
        } else if (gastos > 3000) {
            console.log(`Você tem direito a ${this.#desconto2}% de descontos!`);
        } else {
            console.log(`Infelizmente, você não tem descontos no momento!`);
        }
    }
}
```

# Composição

- A classe **Cliente** possui um cupom de desconto que através a instanciação do **objeto cupom** no construtor da classe a partir da classe **CupomDesconto**

```
1  class Cliente {
2      #nome;
3      #gastosAcumulados = 0;
4      #cupom;
5      constructor(nome) {
6          this.#nome = nome;
7          this.#cupom = new CupomDesconto();
8      }
9      getBeneficios() {
10         this.#cupom.getDesconto(this.#gastosAcumulados);
11     }
12     setGasto(valor) {
13         this.#gastosAcumulados += valor;
14     }
15 }
```

*uso do método getDesconto da classe CupomDesconto dentro da classe Cliente*

# Composição

- Após instanciar o **objeto c1** a partir da classe **Cliente**, é possível estabelecer valores gastos e verificar seus benefícios

```
1  const c1 = new Cliente('Ana');
2  c1.setGasto(500);
3  c1.getBeneficios();
4  c1.setGasto(500);
5  c1.getBeneficios();
```

```
Infelizmente, você não tem descontos no momento!
Você tem direito a 10% de descontos!
```

.StartSe
<tech/academy>

# Composição

- Como houve um acúmulo de gastos no valor igual a 1000, o método **getBeneficios** retornou o cupom de desconto de acordo com os critérios da classe **CupomDesconto**

```
1  const c1 = new Cliente('Ana');
2  c1.setGasto(500);
3  c1.getBeneficios();
4  c1.setGasto(500);
5  c1.getBeneficios();
```

```
Infelizmente, você não tem descontos no momento!
Você tem direito a 10% de descontos!
```

# Composição

- Dessa forma, os critérios e o valor dos descontos podem ser alterados sempre que necessário na classe **CupomDesconto** sem alterar o funcionamento do método **getBeneficios** da classe cliente

```
class CupomDesconto {
    #desconto1 = 10;
    #desconto2 = 20;
    constructor() { }
    getDesconto(gastos) {
        if (gastos >= 1000 && gastos <= 3000) {
            console.log(`Você tem direito a ${this.#desconto1}% de descontos!`);
        } else if (gastos > 3000) {
            console.log(`Você tem direito a ${this.#desconto2}% de descontos!`);
        } else {
            console.log(`Infelizmente, você não tem descontos no momento!`);
        }
    }
}
```