Programação Orientada a Objetos (POO) - Aula 12

Professor: Jerffeson Gomes Dutra

Herança

É um conceito usado para organizar e estruturar dados de forma hierárquica, onde as características de uma classe são compartilhadas por uma ou mais subclasses.

Isso é especialmente útil quando há uma relação de "é um" entre diferentes tipos de entidades.

Por exemplo:

- Gato é um Animal;
- Cachorro é um Animal;

- Mago é um Personagem;
- Cavaleiro é um Personagem;

Exemplo em Código

```
class Animal {
    nome: string
}

class Gato extends Animal {
}

class Cachorro extends Animal {
}
```

```
let gato = new Animal();

let cachorro = new Animal();


gato.nome = "Mingual";
chachorro.nome = "Zeus";


console.log(gato.nome)
console.log(cachorro.nome)
```

Polimorfismo

Se refere à capacidade de diferentes objetos responderem de maneira diferente a uma mesma mensagem ou ação.

Em outras palavras, o polimorfismo permite que:

- Uma mesmo método sejam tratados de maneira diferentes;
- Um objeto de uma subclasse pode ser usada no lugar de sua superclasse;

Exemplo em Código

```
class Animal {
    nome: string;

    emitirSom(): void {
        console.log("O animal emite
um som.");
    }
}

class Gato extends Animal {
}

class Cachorro extends Animal {
}
```

```
// Exemplo de uso
let meuGato = new Gato();
meuGato.nome = "Whiskers";
meuGato.emitirSom();

// Exemplo de uso
let meuCachorro = new Cachorro();
meuCachorro.nome = "Rex";
meuCachorro.emitirSom();



// Saída
"O animal emite um som."
"O animal emite um som."
```

Polimorfismo - 1º Caso

Ele se baseia na ideia de que objetos de classes diferentes podem ser tratados de maneira uniforme, desde que compartilhem uma mesma herança.

Exemplo em Código

```
class Animal {
    nome: string;

    emitirSom(): void {
        console.log("O animal emite
um som.");
    }
}

class Gato extends Animal {
    emitirSom(): void {
        console.log("Miau! Miau!");
    }
}
class Cachorro extends Animal {
    emitirSom(): void {
        console.log("Au! Au!");
    }
}
```

```
// Exemplo de uso
let meuGato = new Gato();
meuGato.nome = "Whiskers";
meuGato.emitirSom();

// Exemplo de uso
let meuCachorro = new Cachorro();
meuCachorro.nome = "Rex";
meuCachorro.emitirSom();



// Saída
"Miau! Miau!"
"Au! Au!"
```

Polimorfismo - 1 Caso - Sobrescrita de método

Quando uma subclasse fornece uma implementação específica de um método que já foi definido em uma de suas superclasses.

Em outras palavras, a subclasse substitui a implementação do método da superclasse por uma nova implementação.

Polimorfismo - 2º Caso

Este tipo de polimorfismo ocorre quando uma subclasse pode ser usada no lugar de sua superclasse. Isso significa que um objeto de uma classe derivada pode ser tratado como um objeto da classe base.

Ou seja, nossos objetos do tipo Gato e Cachorro podem ser tratados como um animal.

Exemplo em Código

```
class Animal {
    nome: string
}

class Gato extends Animal {
}

class Cachorro extends Animal {
}
```

```
let gato = new Animal();

let cachorro = new Animal();


gato.nome = "Mingual";
chachorro.nome = "Zeus";


console.log(gato.nome)
console.log(cachorro.nome)
```

Polimorfismo - 2º Caso

```
class PetShop {

    darBanho(animal: Animal): void {

        console.log("Dando banho em " +
animal.nome);
    }
}
```

```
let meuGato = new Gato();
meuGato.nome = "Whiskers";

// Exemplo de uso
let meuCachorro = new Cachorro();
meuCachorro.nome = "Rex";

let petShopNatal = new PetShop();

petShopNatal.darBanho(meuGato);
petShopNatal.darBanho(meuCachorro);
```

Instanceof

Operador `instanceof`: Em linguagens orientadas a objetos, como TypeScript, você pode usar o operador `instanceof` para verificar se um objeto é uma instância de uma determinada classe.

Exemplo em Código

```
class Animal {}

class Gato extends Animal {}

class Cachorro extends Animal {}

// Função para verificar o tipo do animal
function verificarTipo(animal: Animal): string {
    if (animal instanceof Gato) {
        return 'Gato';
    } else if (animal instanceof Cachorro) {
        return 'Cachorro';
    } else {
        return 'Desconhecido';
    }
}
```

```
// Exemplo de uso
const meuGato = new Gato();
const meuCachorro = new Cachorro();

console.log(verificarTipo(meuGato)); //
Saída: Gato
console.log(verificarTipo(meuCachorro));
// Saída: Cachorro
```

Exercício 01

1. Cria a classes Personagem, Mago e Guerreiro (essas herdam da classe Personagem).
2. A classe Personagem tem:
   a. Os atributos nome, HP, MP e Vigor.
   b. O método *atacar()*, nesse método imprima o texto “O personagem atacou”.
   c. Crie o método *imprmirInfo()* e imprima todas as informações do personagem.
3. Na classe Guerreiro, sobrescreva o método *atacar(),* onde ao atacar, o vigor do personagem é reduzido em 10pts.
4. Na classe Mago, sobrescreva o método *atacar(),* onde ao atacar, o MP do mago é reduzido em 15pts.
5. Crie um objeto de cada classe e chame os métodos *atacar()* e *iimprimirInfo()* várias vezes e veja o comportamento de cada um.

Exercício 02

1. Crie as classes chamada Inventário, Item, Pocao e Equipamento;
2. As classes Pocao e Equipamento herdam da classe Item;
3. A classe Item tem os atributos nome e quantidade;
4. A classe Inventário tem como atributo um array de Item;
5. A classe Inventário tem:
    a. Um método chamado adicionarItem(item: Item) que adicionar o item ao inventário;
    b. um método que lista os itens do inventário;

6. Criei vários objeto dos tipo Pocao e Equipamento e adicione no inventário.