




Swift POO



Polimorfismo



Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura) mas comportamentos distintos, especializados para cada classe.

The background features a large, faint, light-gray diamond-shaped pattern composed of smaller squares. In the top-right and bottom-left corners, there are clusters of colorful triangles in red, dark blue, and light gray, arranged in a geometric, star-like pattern.

Vamos codar!

Classe Gato



Atributos

- Nome
- Cor

Ações

- Miar() -> String
- ComerPeixe()

Classe Gato

```
class Gato {  
    var nome: String  
    var cor: String  
    var cor: String
```

```
    init(nome: String, cor: String) {  
        self.nome = nome  
        self.cor = cor  
    }
```

```
    func miar() -> String {  
        return "Miaaaaaau"  
    }
```

```
    func comerPeixe() {  
        print("Devorando um peixe")  
    }  
}
```

Classe Vaca



Atributos

- nome
- cor
- litrosDeLeiteProduzidosPorDia

Ações

- mugir() -> String
- comerCapim()

Classe Vaca

```
class Vaca {  
    var nome: String  
    var cor: String  
    var litrosDeLeitePorDia: Int
```

```
    init(nome: String, cor: String, litrosDeLeitePorDia: Int) {  
        self.nome = nome  
        self.cor = cor  
        self.litrosDeLeitePorDia = litrosDeLeitePorDia  
    }
```

```
    func mugir() -> String {  
        return "Muu"  
    }
```

```
    func comerCapim() {  
        print("Hmmm que capim gostoso!")  
    }
```

```
}
```


Nós já aprendemos Herança

```
class Animal {  
    var nome: String  
    var cor: String
```

```
    init(nome: String, cor: String) {  
        self.nome = nome  
        self.cor = cor  
    }  
}
```

Nós já aprendemos Herança

```
class Animal {  
    var nome: String  
    var cor: String
```

```
    init(nome: String, cor: String) {  
        self.nome = nome  
        self.cor = cor  
    }  
}
```

**Péra! Se minha vaca e meu come alguma coisa
e também emitem um som, eu não poderia
herdar isso também?**



Classe Animal

```
func emitirSom() -> String {  
    return "0 animal está emitindo um som"  
}
```

```
func comer() {  
    print("0 animal está comendo")  
}
```

Nossos objetos permanecem os mesmos

```
var mimosa: Vaca = Vaca(nome: "Mimosa", cor: "Marron",  
litrosDeLeitePorDia: 3)  
var adamastor: Gato = Gato(nome: "Adamastor", cor:  
"Preto", soltaPelo: true)
```

Vaca é um animal e Gato também

```
var mimosa: Animal = Vaca(nome: "Mimosa", cor: "Marron",  
litrosDeLeitePorDia: 3)  
var adamastor: Animal = Gato(nome: "Adamastor", cor:  
"Preto", soltaPelo: true)
```

Vaca é um animal e Gato também

```
var mimosa: Animal = Vaca(nome: "Mimosa", cor: "Marron",  
litrosDeLeitePorDia: 3)  
var adamastor: Animal = Gato(nome: "Adamastor", cor:  
"Preto", soltaPelo: true)
```


Exercício Funcionários

Definir uma classe **Funcionário** com **nome**, **salário** e **cpf**, depois defina as classes **Programador** que tem **plataforma de trabalho(ex: Android, Web, iOS)**, e uma classe **Designer** que tem **ferramenta preferida(ex: Photoshop, Sketch, Gimp)**. Ambos tem um bonus anual baseado no salário, Programador recebe 20% e Designer recebe 15%, faça um método que calcule o bonus anual e mostre o valor do bonus na tela. Use polimorfismo para resolver este problema.

Exercício - Veículos

Definir uma classe **Veículo** com **cor**, **preço** e **quantidade de passageiros**, depois defina as classes **Carro** que tem **tamanho das rodas**, e **Avião** que tem **piloto e companhia aérea**. queremos descobrir a quantidade de combustível que cada um dos veículos gasta por viagem, defina um método que recebe como parâmetro a distancia em **Km** e calcule a quantidade de combustível.

Carro: **tamanho das rodas * quantidade de passageiros * distancia**

Avião **quantidade de passageiros * distancia**