Polimorfismo
aprendendo a prever o futuro

Profª Lucília Ribeiro

A única constante é a mudança!

Pilares da POO

01

ENCAPSULAMENTO

Construir componentes de software independentes

02

HERANÇA

Reutilizar e estender esses componentes

03

POLIMORFISMO

Permitir escrever softwares à prova do futuro ☺

Software à prova do futuro

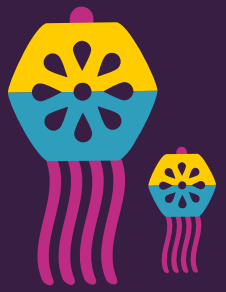
- Se adapta aos requisitos futuros sem alteração
- Permite que você faça alterações e adicione novos recurs facilmente
- Um software de sucesso NÃO é estático

01

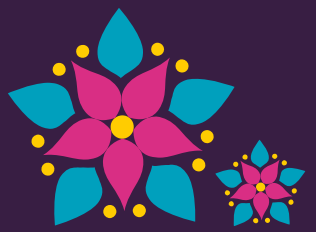
Polimorfismo

A nocaute final

Dois Socos e O Nocaute



Sem o ENCAPSULAMENTO e a HERANÇA, não é possível o POLIMORFISMO



Sem o POLIMORFISMO, a POO não seria EFICAZ

O que é POLIMORFISMO?

Significa MUITAS FORMAS.
Permite que um único nome de método represente um código diferente, selecionado por algum mecanismo automático

Um NOME pode assumir muitas FORMAS e representar muitos COMPORTAMENTOS diferentes

```java
//POLIMORFISMO COM HERANÇA
// Classe base
class Animal {
    public void fazerSom() {
        System.out.println("Som de animal");
    }
}

// Classes derivadas
class Cachorro extends Animal {
    public void fazerSom() {
        System.out.println("AU AU AU");
    }
}

class Gato extends Animal {
    public void fazerSom() {
        System.out.println("MIAU MIAU");
    }
}

public class TesteAnimal {
    public static void main(String[] args) {
        Animal cachorro = new Cachorro();
        Animal gato = new Gato();

        cachorro.fazerSom(); // Chama o latido
        gato.fazerSom(); // Chama o miado
    }
}
```

"O Polimorfismo é o distúrbio das muitas personalidades do mundo do software, pois um único nome pode expressar muitos comportamentos diferentes "

—Antony Sintes

ABRIR

Uma porta

Uma Caixa

Uma Janela

Uma Conta Bancária

```java
class ObjetoPersonalidade {
    public String falar() {
        return "Eu sou um Objeto";
    }
}

class ObjetoPessimista extends ObjetoPersonalidade {
    public String falar() {
        return "O copo esta meio vazio";
    }
}

class ObjetoOtimista extends ObjetoPersonalidade {
    public String falar() {
        return "O copo esta meio cheio";
    }
}

class ObjetoIntrovertido extends ObjetoPersonalidade {
    public String falar() {
        return "Oi...";
    }
}

class ObjetoExtrovertido extends ObjetoPersonalidade {
    public String falar() {
        return "E ai, bla, bla, bla, voce sabia...";
    }
}
```

```java
public class TestaPersonalidade {
    public static void main(String[] args) {
        ObjetoPersonalidade personalidade = new ObjetoPersonalidade
        ();
        ObjetoPessimista pessimista = new ObjetoPessimista();
        ObjetoOtimista otimista = new ObjetoOtimista();
        ObjetoIntrovertido introvertido = new ObjetoIntrovertido();
        ObjetoExtrovertido extrovertido = new ObjetoExtrovertido();
        //capacidade de substituicao permite o seguinte:
        ObjetoPersonalidade[] personalidades = new
        ObjetoPersonalidade[5];
        personalidades[0] = personalidade;
        personalidades[1] = pessimista;
        personalidades[2] = otimista;
        personalidades[3] = introvertido;
        personalidades[4] = extrovertido;
        //o polimorfismo faz com que a Persona
        muitos comportamentos diferentes
        for (int i = 0; i < 5; i++) {
            System.out.println("\n Personalidade[" + (i + 1) + "]
            diz: " + personalidades[i].falar());
        }
    }
}
```

```
Personalidade[1] diz: Eu sou um Objeto

Personalidade[2] diz: O copo esta meio vazio

Personalidade[3] diz: O copo esta meio cheio

Personalidade[4] diz: Oi...

Personalidade[5] diz: E ai, bla, bla, bla, voce sabia...
```

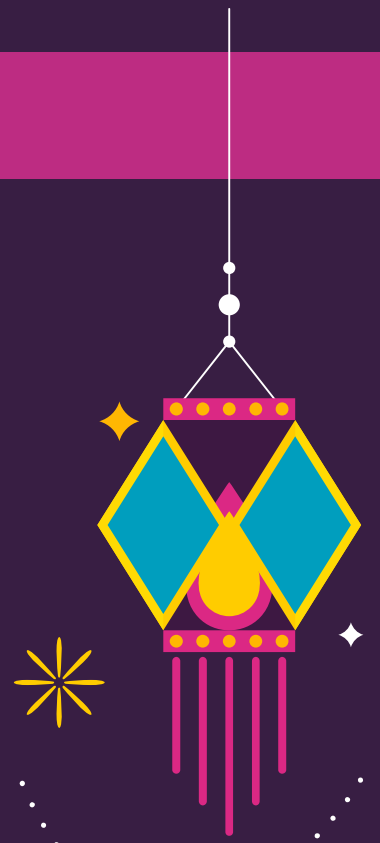
```java
class ObjetoPersonalidade {
    public String falar() {
        return "Eu sou um Objeto";
    }

    public void mandeFalar(ObjetoPersonalidade objeto) {
        System.out.println(objeto.falar());
    }
}
```

```java
for (int i = 0; i < 5; i++) {
    System.out.println("\n Personalidade[" + (i + 1) + "] diz: ");
    personalidade.mandeFalar(personalidades[i]);
}
```

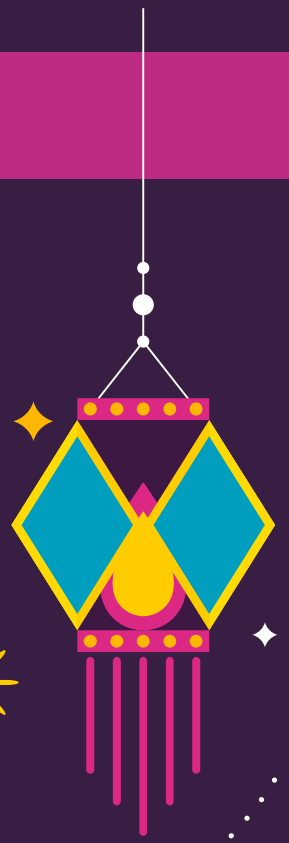
O Polimorfismo correto é Polimórfico

- Adicionar novas funcionalidades a qualquer momento
- Adicionar novas classes que tenham funcionalidades jamais imaginadas
- Sem mudar o código
- Software à Prova do Futuro

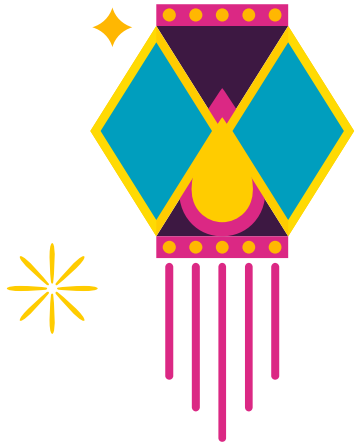
```java
public class Pessoa {
    private String nome;
    private int cpf;

    Pessoa(String nome, int cpf) {
        this.nome = nome;
        this.cpf = cpf;
    }
    public String getNome() {
        return nome;
    }
    public int getCpf() {
        return cpf;
    }
    public String toString() {
        return "\nNome: " + nome + " CPF: " + cpf;
    }
}
```



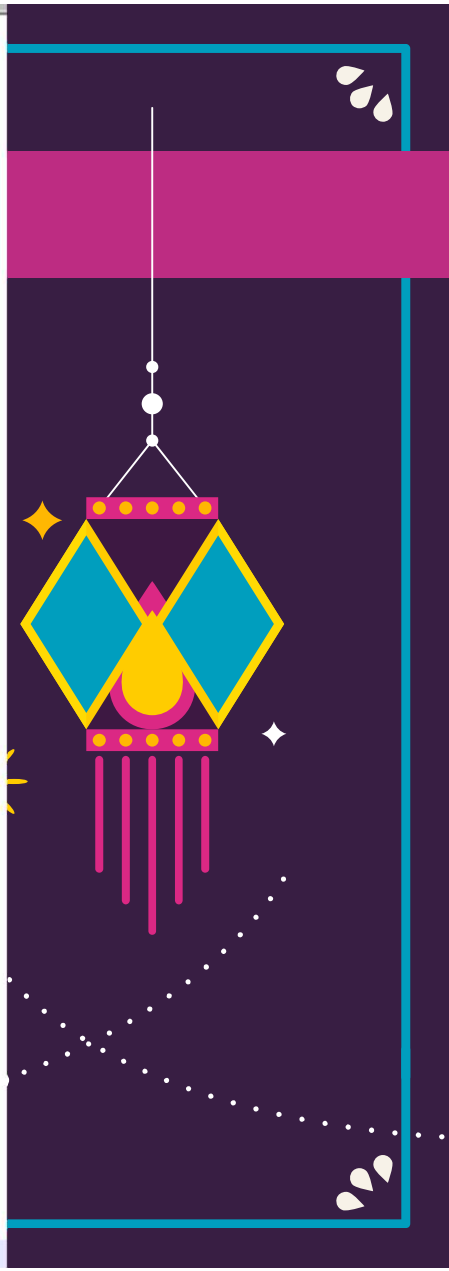
```java
public class Empregado extends Pessoa {
    private float salario;

    public Empregado(String nome, int cpf, float salario) {
        super(nome, cpf);
        this.salario = salario;
    }

    public float getSalario() {
        return salario;
    }

    public String toString() {
        String resultado;
        resultado = super.toString() + "\n";
        resultado += "Salario: R$ " + salario + " reais\n\n";
        return resultado;
    }
}
```



```java
public class ChefeDepartamento extends Empregado {
    private String departamento;
    public ChefeDepartamento(String nome, int cpf, float salario, String departamento) {
        super(nome, cpf, salario);
        this.departamento = departamento;
    }

    public String getDepartamento() {
        return departamento;
    }

    public String toString() {
        String resultado;
        resultado = super.toString() + "\n";
        resultado += "Departamento: " + departamento + "\n";
        return resultado;
    }
}
```



```java
class Emprestimo {
    public static void main(String[] args) {
        Pessoa pessoa = new Pessoa("lucilia", 123);
        Empregado func1 = new Empregado("bruna", 456, 10000f);
        Empregado func2 = new Empregado("andre", 789, 15000f);
        Empregado func3 = new Empregado("fulano", 777, 1251.3f);
        ChefeDepartamento chefe = new ChefeDepartamento("boss", 654, 25.25f, "CPD");
        System.out.print(pessoa);
        System.out.println("\nValor do emprestimo: R$ " + calculaEmprestimo(pessoa));
        System.out.print(func1);
        System.out.println("\nValor do emprestimo: R$ " + calculaEmprestimo(func1));
        System.out.print(func2);
        System.out.println("\nValor do emprestimo: R$ " + calculaEmprestimo(func2));
        System.out.print(func3);
        System.out.println("\nValor do emprestimo: R$ " + calculaEmprestimo(func3));
        System.out.print(chefe);
        System.out.println("\nValor do emprestimo: R$ " + calculaEmprestimo(chefe));
    }

    public static float calculaEmprestimo(Pessoa pessoa) {
        return 1000f;
    }

    public static float calculaEmprestimo(Empregado funcionario) {
        float emprestimo = 0f;
        if (funcionario instanceof ChefeDepartamento) {
            emprestimo = 4.0f * funcionario.getSalario();
        } else {
            if (funcionario instanceof Empregado) {
                emprestimo = 2.0f * funcionario.getSalario();
            }
        }
        return emprestimo;
    }
}
```



```java
class Concessionaria {
    public static void main(String[] args) {
        Automovel carro1 = new Automovel("Fusca", "azul", Automovel.GASOLINA);
        AutomovelBasico carro2 = new AutomovelBasico("Corsa", "prata", Automovel.GASOLINA);
        AutomovelBasico carro3 = new AutomovelBasico("Gol", "branco", Automovel.ALCOOL, true, true, false);
        AutomovelDeLuxo carro4 = new AutomovelDeLuxo("Ibiza", "vermelho", Automovel.DIESEL);
        AutomovelDeLuxo carro5 = new AutomovelDeLuxo("Honda Fit", "branco", Automovel.ALCOOL, true, true, false, true, false, true);
        imprime(carro1);
        imprime(carro2);

    }

    public static void imprime(Automovel carro) {
        System.out.println("Dados do automovel escolhido:");
        System.out.print(carro); //chamada implicita a toString
        System.out.println("Valor: " + carro.quantoCusta());
        System.out.println(carro.quantasPrestacoes() + " prestacoes de " + (carro.quantoCusta() / carro.quantasPrestacoes()));
    }
}
```

Obrigada!

Perguntas?

professora@lucilia.com.br