Herança POO

Profª Lucília Ribeiro

Herança é o alicerce da orientação a objetos, permitindo que classes descendam do passado para moldar o futuro do software

— chatGPT

Pilares da POO

abstração

encapsulamento

herança

```java
public class Carro {
    private String marca;
    private String modelo;
    private int ano;

    // Construtor
    public Carro(String marca, String modelo, int ano) {
        this.marca = marca;
        this.modelo = modelo;
        this.ano = ano;
    }

    public void ligar() {
        System.out.println("O carro está ligado.");
    }

    public void desligar() {
        System.out.println("O carro está desligado.");
    }

    public void acelerar() {
        System.out.println("Acelerando o carro.");
    }

    // Getters para acessar os atributos privados
    public String getMarca() {
        return marca;
    }

    public String getModelo() {
        return modelo;
    }

    public int getAno() {
        return ano;
    }
}
```

```java
import java.util.Scanner;
public class Ex1 {
    public static void main(String arg[ ]) {
        Scanner dado = new Scanner(System.in);
        // Criao de objetos
        Carro meuCarro = new Carro("Volks", "Fusca", 1969);
        Carro seuCarro = new Carro("Honda", "Civic", 2023);

        // Acesso aos atributos
        System.out.println("Meu Carro: " + meuCarro.getMarca() + " " + meuCarro.getModelo() +
         " " + meuCarro.getAno());
        System.out.println("Seu Carro: " + seuCarro.getMarca() + " " + seuCarro.getModelo() +
         " " + seuCarro.getAno());

        // Chamar mtodos
        meuCarro.ligar();
        meuCarro.acelerar();
        meuCarro.desligar();
    }
}
```

```
Meu Carro: Volks Fusca 1969
Seu Carro: Honda Civic 2023
O carro está ligado.
Acelerando o carro.
O carro está desligado.
```

Herança

O que é Herança

- Um dos pilares da POO
- Permite que uma classe (subclasse ou classe derivada) herde atributos e métodos de outra classe (superclasse ou classe base)
- Promove a reutilização de código e a hierarquia de classes
- Ajuda a modelar a relação “é um” entre objetos

```java
class Animal {
    String nome;
    int idade;

    Animal(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    void emitirSom() {
        System.out.println("O animal faz um som.");
    }
}

class Cachorro extends Animal {
    Cachorro(String nome, int idade) {
        super(nome, idade);
    }

    void latir() {
        System.out.println("AU AU AU.");
    }
}

public class Ex2 {
    public static void main(String[] args) {
        Cachorro meuCachorro = new Cachorro("Bidu", 3);

        System.out.println("Nome do cachorro: " + meuCachorro.nome);
        System.out.println("Idade do cachorro: " + meuCachorro.idade + " anos");

        meuCachorro.emitirSom();
        meuCachorro.latir();
    }
}
```

```
Nome do cachorro: Bidu
Idade do cachorro: 3 anos
O animal faz um som.
AU AU AU.
```

Importância da Herança

- Fundamental para criar abstrações e modelos mais complexos
- Permite criar novas classes com base em classes existentes, economizando tempo e esforço

```java
class Veiculo {
    private String marca;
    private String modelo;
    private int ano;

    Veiculo(String marca, String modelo, int ano) {
        this.marca = marca;
        this.modelo = modelo;
        this.ano = ano;
    }

    public void ligar() {
        System.out.println("Veículo está ligado.");
    }

    public void desligar() {
        System.out.println("Veículo está desligado.");
    }

    public void acelerar() {
        System.out.println("Vrum Vrum.");
    }

    // Getters para acessar os atributos privados
    public void mostrarInformacoes() {
        System.out.println("Marca: " + marca + " Modelo: " + modelo + " Ano: " + ano);
    }
}
```

```java
class Carro extends Veiculo {
    private int numeroPortas;

    Carro(String marca, String modelo, int ano, int numeroPortas) {
        super(marca, modelo, ano);
        this.numeroPortas = numeroPortas;
    }

    void abrirPortas() {
        System.out.println("Abrindo as " + numeroPortas + " portas do carro.");
    }
}

class Moto extends Veiculo {
    private boolean temPartidaEletrica;

    Moto(String marca, String modelo, int ano, boolean temPartidaEletrica) {
        super(marca, modelo, ano);
        this.temPartidaEletrica = temPartidaEletrica;
    }

    void ligarPartidaEletrica() {
        if (temPartidaEletrica) {
            System.out.println("Ligando a partida elétrica da moto.");
        }
        else {
            System.out.println("A moto não tem partida elétrica.");
        }
    }
}
```

```java
public class Ex3 {
    public static void main(String[] args) {
        Carro meuCarro = new Carro("Volks", "Fusca", 1969, 4);
        Moto minhaMoto = new Moto("Honda", "CB500", 2023, true);
        meuCarro.mostrarInformacoes();
        meuCarro.acelerar();
        meuCarro.abrirPortas();
        meuCarro.desligar();
        System.out.println();
        minhaMoto.mostrarInformacoes();
        minhaMoto.acelerar();
        minhaMoto.ligarPartidaEletrica();
        minhaMoto.desligar();
    }
}
```

```
Marca: Volks Modelo: Fusca Ano: 1969
Vrum Vrum.
Abrindo as 4 portas do carro.
Veículo está desligado.

Marca: Honda Modelo: CB500 Ano: 2023
Vrum Vrum.
Ligando a partida elétrica da moto.
Veículo está desligado.
```

Uso e importância

- Desenvolvimento de Jogos
- Frameworks Web
- Modelagem de Dados
- Desenvolvimento de Aplicativos para Mobiles
- Simulação e Modelagem de Sistemas

Desenvolvimento de Jogos

```java
class Personagem {
    private String nome;
    private int vida;

    Personagem(String nome, int vida) {
        this.nome = nome;
        this.vida = vida;
    }

    void atacar() {
        System.out.println(nome + " ataca!");
    }

    void receberDano(int dano) {
        vida -= dano;
        System.out.println(nome + " recebe " + dano + " de dano. Vida restante: " + vida);
    }
}
```

```java
class Guerreiro extends Personagem {
    Guerreiro(String nome) {
        super(nome, 100);
    }

    void usarEspada() {
        System.out.println("O guerreiro ataca com a espada!");
    }
}

class Mago extends Personagem {
    Mago(String nome) {
        super(nome, 80);
    }

    void lançarFeitiço() {
        System.out.println("O mago lançaa uma magia!");
    }
}

public class DesJogos {
    public static void main(String[] args) {
        Guerreiro guerreiro = new Guerreiro("Conan");
        Mago mago = new Mago("Merlin");

        guerreiro.atacar();
        guerreiro.usarEspada();
        guerreiro.receberDano(20);

        mago.atacar();
        mago.lançarFeitiço();
        mago.receberDano(30);
    }
}
```

```
Conan ataca!
O guerreiro ataca com a espada!
Conan recebe 20 de dano. Vida restante: 80
Merlin ataca!
O mago lançaa uma magia!
Merlin recebe 30 de dano. Vida restante: 50
```

Framework de Desenvolvimento Web

```java
class Autenticacao {
    void autenticar(String usuario, String senha) {
        System.out.println("Autenticando usuário: " + usuario);
        // Lógica de autenticação
    }
}

class UsuarioComum extends Autenticacao {
    void acoesUsuarioComum() {
        System.out.println("Realizando ações de usuário comum.");
    }
}

class Administrador extends Autenticacao {
    void acoesAdministrador() {
        System.out.println("Realizando ações de administrador.");
    }
}

public class DesWeb {
    public static void main(String[] args) {
        UsuarioComum usuario = new UsuarioComum();
        Administrador admin = new Administrador();

        usuario.autenticar("aluno", "aluno");
        usuario.acoesUsuarioComum();

        admin.autenticar("admin", "adminpass");
        admin.acoesAdministrador();
    }
}
```

```
Autenticando usuário: aluno
Realizando ações de usuário comum.
Autenticando usuário: admin
Realizando ações de administrador.
```

Modelagem de Dados

```java
class ItemBiblioteca {
    private String titulo;
    private boolean disponivel;

    ItemBiblioteca(String titulo) {
        this.titulo = titulo;
        disponivel = true;
    }

    void emprestar() {
        disponivel = false;
        System.out.println("Item emprestado.");
    }

    void devolver() {
        disponivel = true;
        System.out.println("Item devolvido.");
    }

    String getTitulo() {
        return titulo;
    }

    boolean getDisponivel() {
        return disponivel;
    }
}
```

```java
class Livro extends ItemBiblioteca {
    private String autor;

    Livro(String titulo, String autor) {
        super(titulo);
        this.autor = autor;
    }

    String getAutor() {
        return autor;
    }
}

class DVD extends ItemBiblioteca {
    private String diretor;

    DVD(String titulo, String diretor) {
        super(titulo);
        this.diretor = diretor;
    }

    String getDiretor() {
        return diretor;
    }
}
```

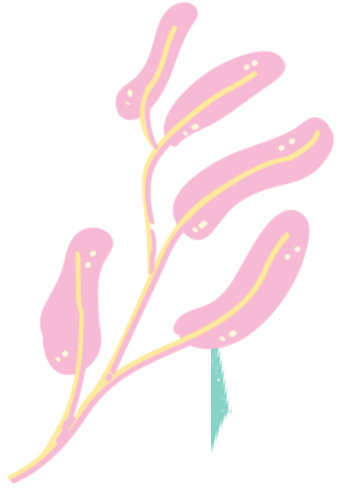
Modelagem de Dados

```java
public class DesMod {
    public static void main(String[] args) {
        Livro livro = new Livro("Dom Casmurro", "Machado de Assis");
        DVD dvd = new DVD("Pulp Fiction", "Quentin Tarantino");

        livro.emprestar();
        dvd.emprestar();
        livro.devolver();

        System.out.println("Livro: " + livro.getTitulo() + ", Autor: " + livro.getAutor());
        System.out.println("DVD: " + dvd.getTitulo() + ", Diretor: " + dvd.getDiretor());
        System.out.println("Status do Livro: " + (livro.getDisponivel() ? "Disponivel" : "Indisponível"));
        System.out.println("Status do DVD: " + (dvd.getDisponivel() ? "Disponivel" : "Indisponível"));
    }
}
```

```
Item emprestado.
Item emprestado.
Item devolvido.
Livro: Dom Casmurro, Autor: Machado de Assis
DVD: Pulp Fiction, Diretor: Quentin Tarantino
Status do Livro: Disponivel
Status do DVD: Indisponível
```

Aplicativos para Dispositivos Móveis

```
1  class ElementoUI {
2      protected int x;
3      protected int y;
4      protected int largura;
5      protected int altura;
6
7      ElementoUI(int x, int y, int largura, int altura) {
8          this.x = x;
9          this.y = y;
10         this.largura = largura;
11         this.altura = altura;
12     }
13
14     void exibir() {
15         System.out.println("ElementoUI - Posição: (" + x + ", " + y + "),
           Tamanho: " + largura + "x" + altura);
16     }
17 }
```



Aplicativos para Dispositivos Móveis

```
19  class Botao extends ElementoUI {
20      private String texto;
21
22      Botao(int x, int y, int largura, int altura, String texto) {
23          super(x, y, largura, altura);
24          this.texto = texto;
25      }
26
27      void clicar() {
28          System.out.println("Botão clicado: " + texto);
29      }
30  }
31
32  class CaixaDeTexto extends ElementoUI {
33      CaixaDeTexto(int x, int y, int largura, int altura) {
34          super(x, y, largura, altura);
35      }
36
37      void digitarTexto(String texto) {
38          System.out.println("Texto digitado na caixa de texto: " + texto);
39      }
40  }
```

Aplicativos para Dispositivos Móveis

```
42  public class DesMob {
43      public static void main(String[] args) {
44          Botao botao = new Botao(10, 20, 100, 40, "Clique-me");
45          CaixaDeTexto caixa = new CaixaDeTexto(30, 60, 150, 30);
46
47          botao.exibir();
48          botao.clicar();
49
50          caixa.exibir();
51          caixa.digitarTexto("Let´s Code!");
52      }
53  }
```

```
ElementoUI - Posição: (10, 20), Tamanho: 100x40
Botão clicado: Clique-me
ElementoUI - Posição: (30, 60), Tamanho: 150x30
Texto digitado na caixa de texto: Let´s Code!
```

EXERCÍCIOS

Exercício 1

Crie uma hierarquia de classes para representar figuras geométricas. A classe base deve ser "Figura" e ter um método para calcular a área. Crie classes derivadas, como "Retângulo“, "Círculo“ e mais uma de sua preferência, que herdam de "Figura" e implementam seus próprios métodos para calcular a área. Crie objetos dessas classes e calcule as áreas correspondentes.

Exercício 2

Desenvolva um sistema de modelagem de veículos.
Crie uma classe base "Veiculo" com atributos comuns, como marca e modelo, e métodos para acelerar e desacelerar.
Em seguida, crie classes derivadas, como "Carro“, "Moto“, “Caminhão”, que herdam de "Veiculo" e adicionam seus próprios métodos específicos, como "abrirCapota" para um carro conversível ou "ligarFarol" para uma moto.
Crie objetos dessas classes e demonstre seus métodos.

Exercício 3

1. Crie uma classe base chamada "Personagem" que contenha os seguintes atributos:
   - nome (string)
   - nivel (int)
   - pontosDeVida (int)
   - pontosDeAtaque (int)
2. Implemente métodos na classe base "Personagem":
   - atacar(): Simule um ataque do personagem imprimindo uma mensagem.
   - receberDano(int dano): Reduza os pontos de vida do personagem com base no dano recebido.

Exercício 3 (cont)

3.Crie duas classes derivadas: "Guerreiro" e "Mago". Cada uma dessas classes deve herdar de "Personagem" e adicionar atributos e métodos específicos:

- Guerreiro:
  - Atributo adicional: forca (int)
  - Método adicional: usarEspada(), que imprime uma mensagem de ataque com espada.
- Mago:
  - Atributo adicional: poderMagico (int)
  - Método adicional: lancarFeitico(), que imprime uma mensagem sobre o lançamento de um feitiço.

Exercício 3 (cont)

4. Crie personagens de diferentes classes (Guerreiro e Mago), defina seus atributos (nome, nível, pontos de vida, etc.) e demonstre as ações dos personagens. Eles devem realizar ataques, usar habilidades específicas da classe e simular a redução de pontos de vida quando recebem dano.
5. Crie uma pequena narrativa ou cena de batalha em que os personagens interajam e utilizem suas habilidades únicas.

Obrigada!

professora@lucilia.com.br