Engenharia de Software

Orientação a Objetos

Conceitos básicos

Prof. MSc. Edilberto Silva

com adaptações **Clenio Emidio**

Tópicos

- Histórico
- Conceitos de OO
  - Abstração e Objetos
  - Encapsulamento
  - Herança
  - Polimorfismo
- Classes e objetos
  - Variáveis de instância
  - Variáveis de classe
  - Métodos e mensagens

Histórico

- Linguagem Simula (anos 60), derivada do Algol e desenvolvida no Centro Norueguês de Computação. Era utilizada para simulações e foi a pioneira na utilização de classes e subclasses, semelhantes às usadas atualmente em POO.
- *SmallTalk* (anos 70), foi a primeira linguagem autêntica orientada a objetos. Foi desenvolvida por cientistas do Xerox.
- C++, dos laboratórios da AT&T Bell, uma expansão da linguagem C, muito popular e conhecida. Suporta POO, mantendo as características do C tradicional. Fácil transição do C para o C++.
- A linguagem Java foi projetada e implementada por um pequeno grupo pessoas, coordenado por James Gosling, na Sun Microsystems em Mountain View, Califórnia, em 1991.

POO x Técnicas Convencionais

▪ Métodos	▪ Procedimentos e funções
▪ Variáveis de instância	▪ Dados
▪ Mensagens	▪ Chamadas de procedimento e funções
▪ Classes	▪ Tipos de Dados
▪ Herança	▪ Não existe
▪ Chamadas sob controle do sistema	▪ Chamada sob controle do programador

Conceitos

- O termo “**orientação a objetos**” significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados (atributos) e um conjunto de ações (métodos) que manipulam estes dados.
- Todas as linguagens orientadas a objetos possuem três características básicas:
  - **Encapsulamento**
  - **Herança**
  - **Polimorfismo**
- Para entender essas características é necessário antes compreender o conceito de **abstração e objetos**.

Abstração

- Focalizar o essencial, ignorando os pormenores.
- Deve ser executada com algum objetivo para saber o que é importante e o que não é.

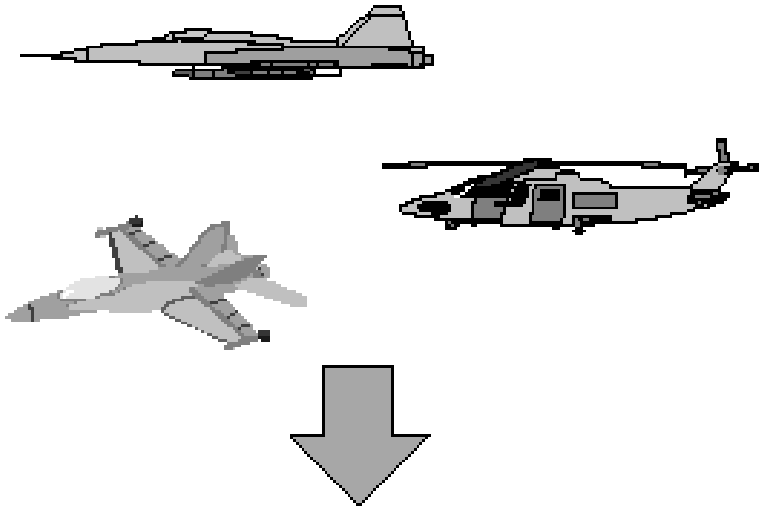
"A abstração é o processo de filtragem de detalhes sem importância do objeto, para que apenas as características apropriadas que o descrevem permanecam" (Peter Van)

"Extrair tudo o que for essencial e nada mais"

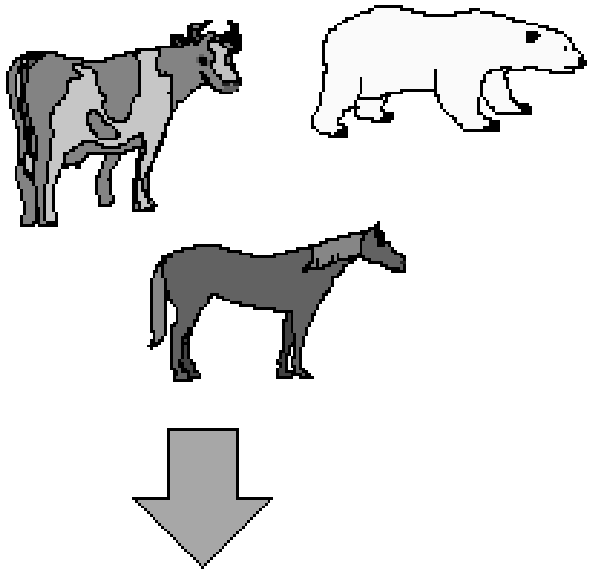
(Aaron Walsh)

Abstração

- **Exemplo**



Aeronave



Mamífero

Abstração

- Costumamos abstrair de objetos aquilo que nos interessa.

- placa
- cor
- númeroChassi

- aplicarMulta()

- cor
- cilindrada
- velocidadeMax

- acelerar()

Objetos

- Objetos são abstrações de entidades do mundo real (ou de algum sistema) que se auto-gerenciam.
- Objetos são independentes e encapsulam suas representações de estado e de informações
- A funcionalidade de um sistema é expressa em termos de serviços que os objetos prestam

Objetos

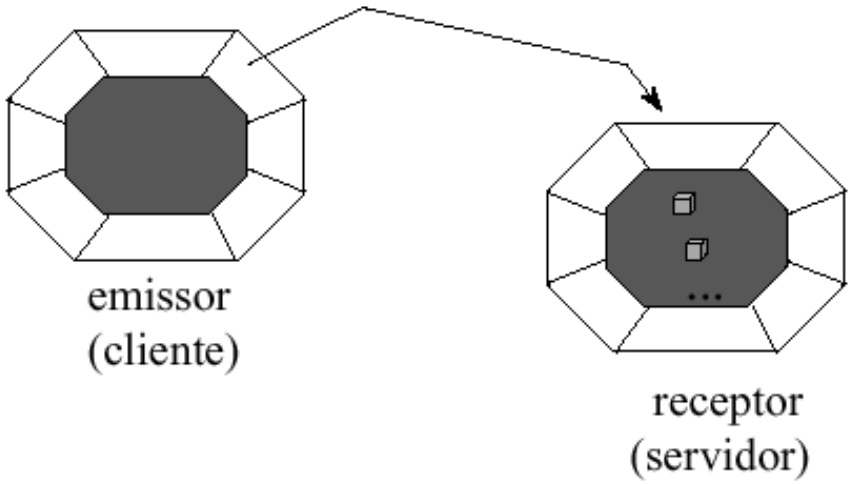
- É uma entidade lógica que contém atributos e métodos.
  - **Atributos**: são as características próprias dos objetos. Exemplo: nome da pessoa, cor do carro, valor da transação.
  - **Métodos**: são blocos de instruções que manipulam os atributos. Exemplo: falar (pessoa), acelerar (carro), debitar (transação).
- Exemplos de objetos
  - Coisas tangíveis: bicicleta, lápis, carro...
  - Incidente (evento/ocorrência): eleição, aniversário, missa...
  - Interação (transação/contrato): o débito de R$ 100,00 na conta “x” do dia 07/11/2002

Objetos

- **Classe**
  - É uma estrutura de dados, como as já conhecidas, para declarar variáveis. Essa estrutura serve para definir objetos.
  - Uma variável de uma classe é chamada de Objeto.
  - Definir uma classe não cria um objeto, assim como um tipo de variável NÃO é uma variável.
- **Pacote**
  - é um agrupador de classes de determinada área. É semelhante às pastas e subpastas de um sistema de arquivos.
- **Instância**
  - temos uma instância de uma classe quando declaramos um objeto a partir de uma classe. É semelhante à declaração de uma variável.
- **Variável de instância:**
  - São os atributos de um objeto em particular

Objetos

- **Mensagens**
  - Objetos interagem e comunicam-se através de mensagens.
  - As mensagens identificam os métodos a serem executados no objeto receptor.



Exemplo:
Imagine dois objetos: um Carro e um Motorista.
O objeto Motorista pode enviar a mensagem "ligar()" para o objeto Carro.
O objeto Carro recebe a mensagem e executa o método ligar(), que irá, por exemplo, acionar a partida e o motor.

Objetos

- **Mensagens**
  - Para invocar um método de um objeto, deve-se enviar uma mensagem para este objeto.
  - Para enviar uma mensagem deve-se:
    - Identificar o objeto que receberá a mensagem
    - Identificar o método que o objeto deve executar
    - Passar os argumentos requeridos pelo método
  - Um objeto possui:
    - Um estado (definido pelo conjunto de valores dos seus atributos em determinado instante)
    - Um comportamento (definido pelo conjunto de métodos definido na sua interface)
    - Uma identidade única

Objetos

- Métodos construtores e destrutores
  - **Construtores:** alocam dinamicamente na memória uma instância de uma classe.
  - **Destrutores:** retiram a instância da memória, liberando-a. (ausente no Java)
- A **alocação** das instâncias é **dinâmica**.
  - Outras instâncias são necessárias apenas por algum tempo.
  - Quando um objeto não é mais necessário, ele é destruído e seu espaço na memória liberado.
  - A recuperação do espaço é denominada Coleta de Lixo (Garbage Collection).

Encapsulamento

- Base da abordagem OO - Um dado está encapsulado quando envolvido por código de forma que, só é visível na rotina onde foi criado.
- Não interessa saber como é o funcionamento interno da classe e sim sua função
  - ex: a tecla de play do som, não interessa como funciona internamente mas sim que para qualquer marca sua função será a mesma.

```java
public class ContaBancaria {
    private String numero;
    private double saldo;

    public ContaBancaria(String numero, double saldo) {
        this.numero = numero;
        this.saldo = saldo;
    }

    public String getNumero() {
        return numero;
    }

    public double getSaldo() {
        return saldo;
    }

    public void depositar(double valor) {
        if (valor > 0) {
            saldo += valor;
        }
    }

    public void sacar(double valor) {
        if (valor > 0 && valor <= saldo) {
            saldo -= valor;
        } else {
            System.out.println("Saldo insuficiente");
        }
    }
}
```

- Imagine uma classe **ContaBancaria** com os atributos numero (número da conta) e saldo (saldo da conta) e os métodos **depositar()** e **sacar().**
- Em vez de permitir acesso direto aos atributos, o encapsulamento garante que o saldo só seja alterado através dos métodos depositar() e sacar(), que podem incluir validações, como verificar se há saldo suficiente para um saque antes de efetuar a operação.

Encapsulamento

- Todo o acesso aos dados do objeto é feito através da chamada a uma operação (**método**) da sua interface
- Mudanças na implementação de um objeto, que preservem a sua interface externa, não afetam o resto do sistema.
- A interface (**pública**) de um objeto declara todas as operações permitidas (**métodos**). Esses métodos públicos poderão manipular e controlar o acesso aos atributos privados.
- Não podemos acessar os **atributos** de um objeto diretamente. (Fenômeno da caixa preta). Para isso devemos fazer uso do **métodos**.

Encapsulamento

- Os tipos de visibilidade que podem ser determinados em uma classe para atributos e serviços são:
  - \+ **public:** os elementos são acessíveis por todas as classes;
  - # **protected:** os elementos são acessíveis por subclasses, ou pela própria classe;
  - – **private:** os elementos são acessíveis somente pela própria classe;

Herança

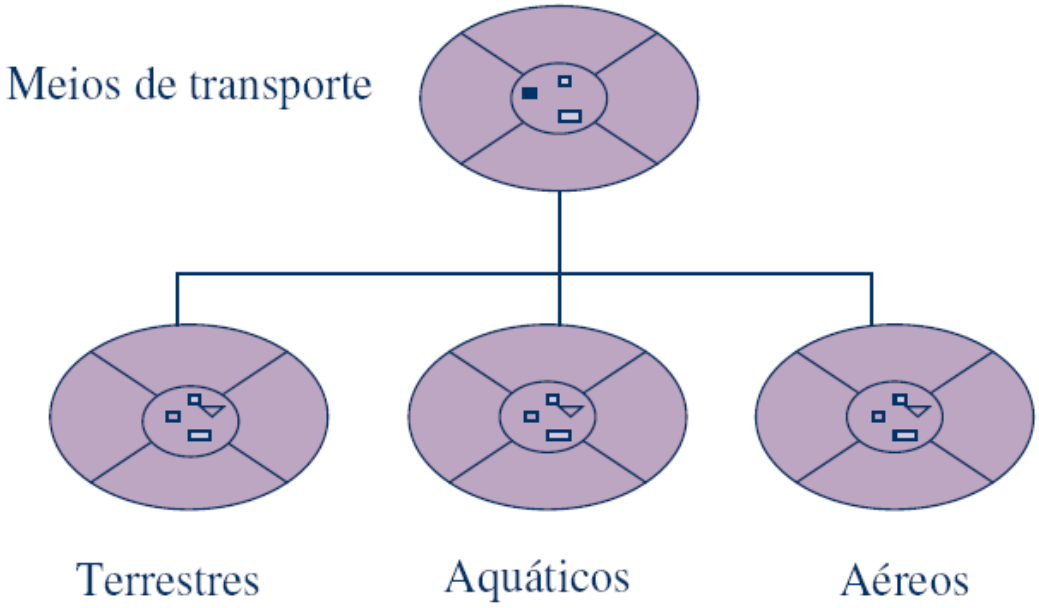
- **Reuso**
  - Principal motivação da herança.
  - Incialmente um um código em Java pode ter um tempo de desenvolvimento maior que em uma linguagem de programação procedural. Depois, com a propriedade de reutilização de código o tempo diminui bastante.
  - Evita retrabalho e facilita as manutenções corretivas e evolutivas.
  - O reuso hoje é possível de duas maneiras:
    - usando composição/agregação (“tem um”) e ;
    - através de herança (“é um”).

Herança

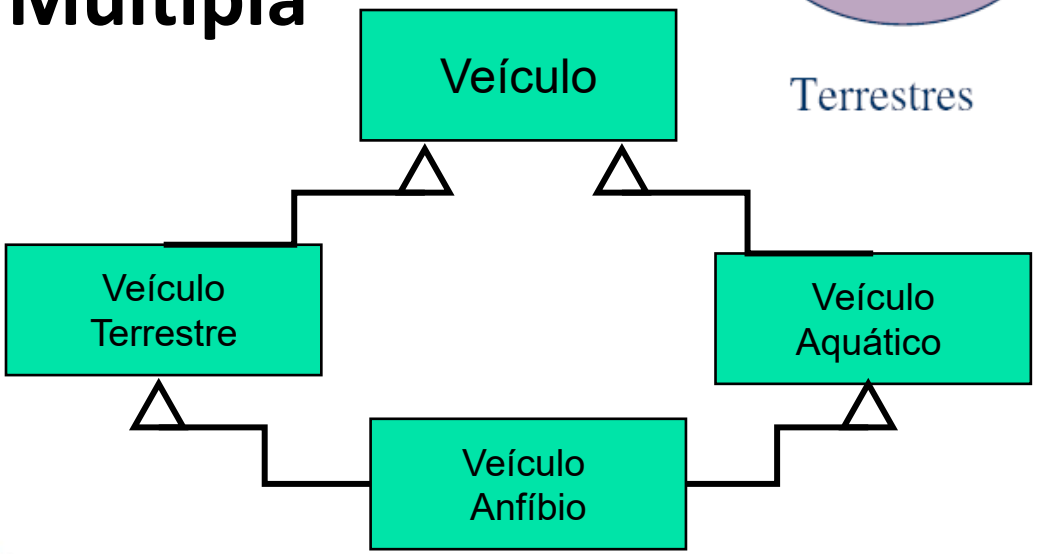
- Herança significa ser capaz incorporar os atributos e métodos de uma classe previamente definida.
- Através da herança pode-se reutilizar ou alterar os métodos de classes existentes, bem como adicionar novos atributos a fim de adaptá-los a novas situações. Isso se chama especialização das classes.
- Tipos
  - Herança **simples**: herda de um objeto
  - Herança **múltipla**: herda de vários objetos

Herança

- **Simples**

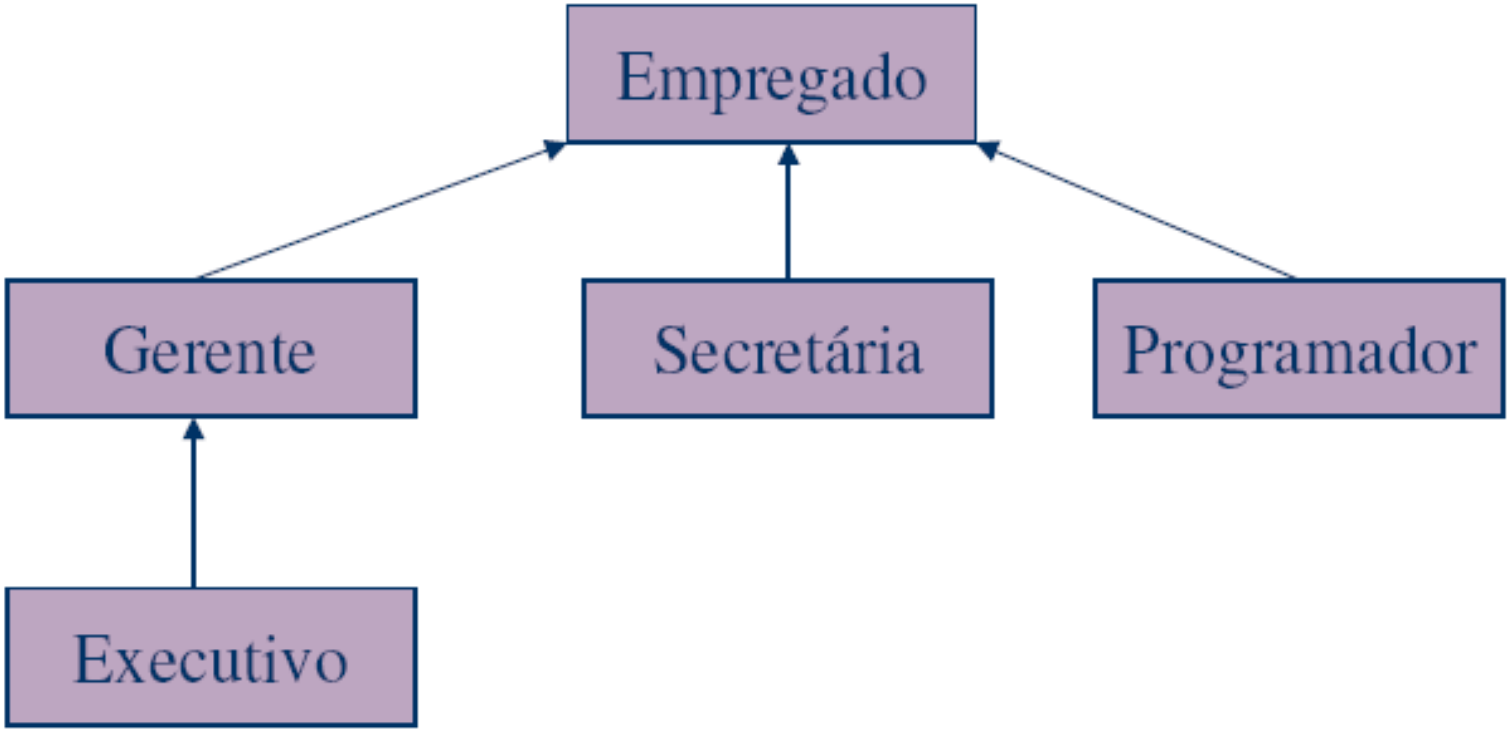


- **Múltipla**



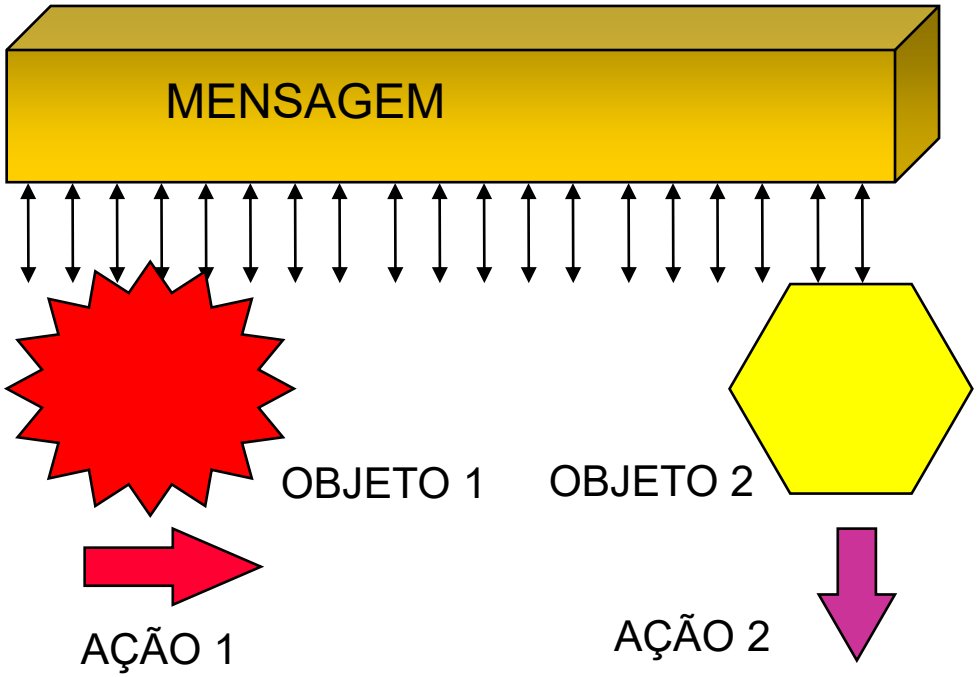
Herança

- **Exemplo 2**



Polimorfismo

- O Polimorfismo ocorre quando uma mesma mensagem chegando a objetos diferentes provoca respostas diferentes.



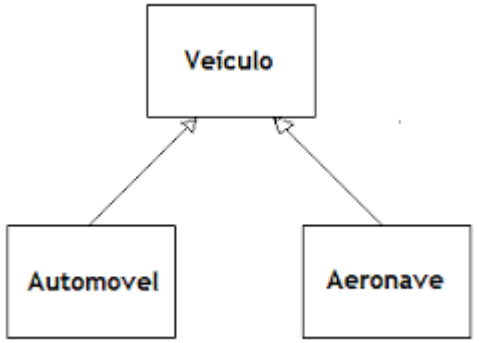
Em outras palavras, "um nome, várias formas" ou "muitas formas".

Exemplo:
Imagine um sistema que gerencia diferentes tipos de formas geométricas: círculos, quadrados e triângulos. Cada forma tem um método chamado **desenhar**(), mas a implementação desse método é diferente para cada forma.

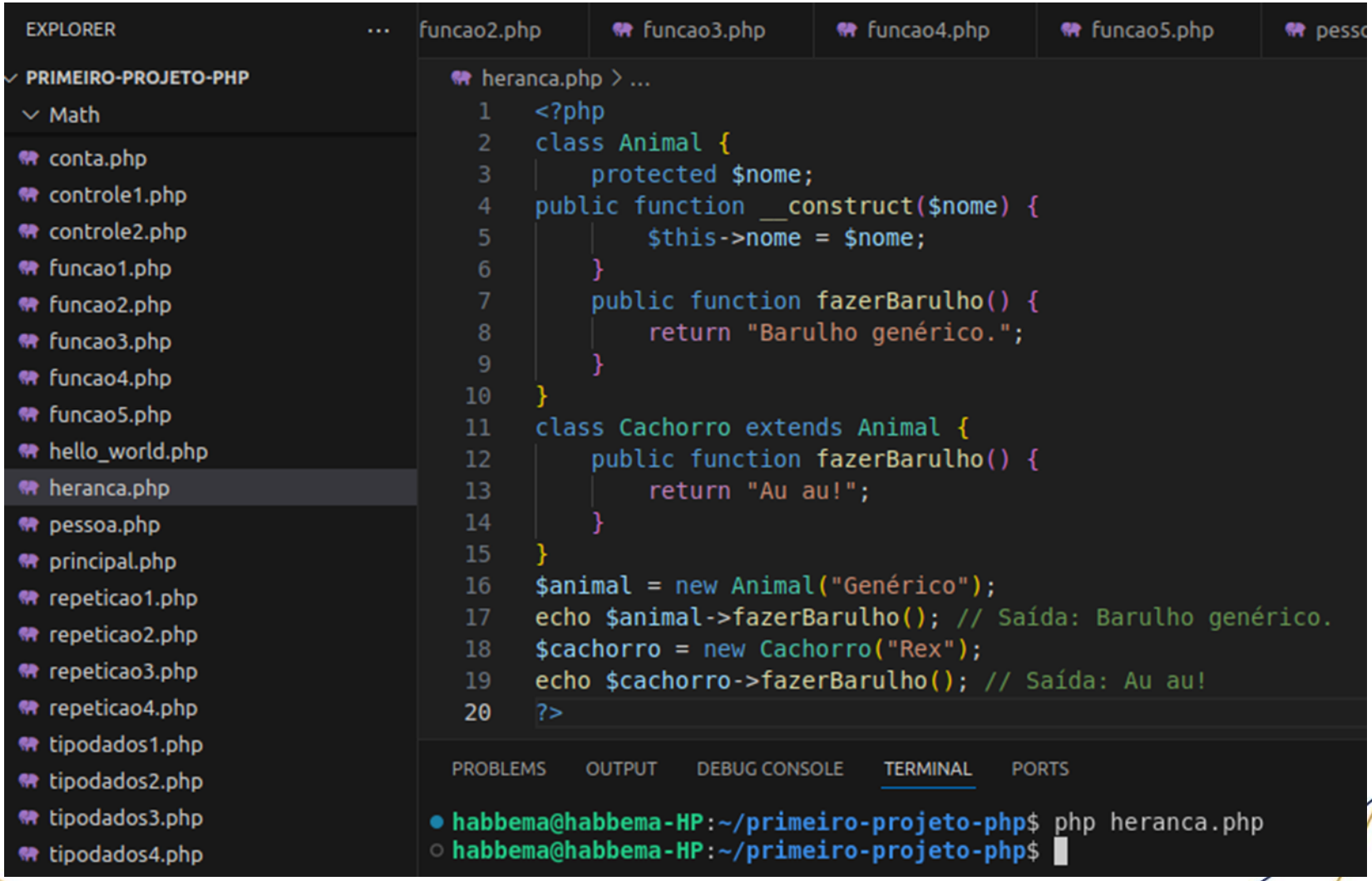
Polimorfismo

Python

```python
class Forma:
    def desenhar(self):
        pass  # Método abstrato

class Circulo(Forma):
    def desenhar(self):
        print("Desenhando um círculo...")

class Quadrado(Forma):
    def desenhar(self):
        print("Desenhando um quadrado...")

class Triangulo(Forma):
    def desenhar(self):
        print("Desenhando um triângulo...")

# Lista de formas
formas = [Circulo(), Quadrado(), Triangulo()]

# Polimorfismo em ação:
for forma in formas:
    forma.desenhar()
```

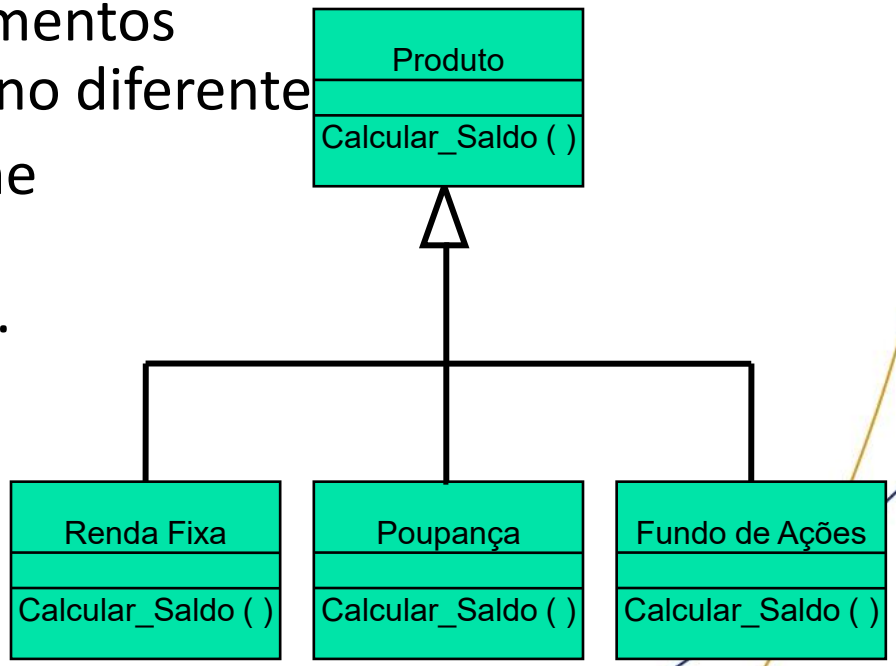


Polimorfismo



Polimorfismo

- A implementação consiste em utilizar o mesmo nome do método dentro de uma classe, ou em uma hierarquia de classes, com comportamentos diferentes.
- Há duas maneiras de implementação em Java:
  - **Sobrecarga**: reutilizar o mesmo nome para o método com argumentos diferentes e talvez tipo de retorno diferente
  - **Sobrescrita**: usar o mesmo nome para o método como o mesmo retorno e argumentos idênticos.



Videos para complementar

- https://www.youtube.com/watch?v=x9aLELLVzjQ
- https://www.youtube.com/watch?v=dG7LlYne2VA

Atividades

- Questão 01
- A Programação Orientada a Objetos (POO) é um paradigma que organiza o software em unidades chamadas “objetos”, instâncias de classes que combinam dados e comportamentos. A POO visa promover a modularidade, a reutilização de código e a facilidade de manutenção. Técnicas como sobrescrita (override) e sobrecarga (overload) são usadas para permitir que um método se comporte de maneira flexível, dependendo da classe ou dos parâmetros utilizados.
- Considerando essas técnicas, assinale a alternativa que apresenta o conceito de POO relacionado ao uso dessas práticas.
- A Abstração.
- B Associação.
- C Encapsulamento.
- D Herança.
- E Polimorfismo.

Questão 02

- A Orientação a Objetos (OO) é um paradigma de programação baseado na modelagem de sistemas por meio de objetos, que representam entidades do mundo real. Cada objeto possui atributos (dados) e métodos (comportamentos), encapsulando informações e funcionalidades. Um dos conceitos fundamentais de OO é o conceito de polimorfismo. Acerca do polimorfismo no contexto de OO, assinale a alternativa correta.
- A- Habilita métodos com o mesmo nome a terem comportamentos diferentes dependendo do contexto, facilitando a flexibilidade do código.
- B- Define apenas os detalhes essenciais de um objeto, ocultando sua implementação interna.
- C- Trata-se de um método especial de uma classe usado para inicializar objetos, atribuindo valores iniciais aos seus atributos no momento de sua criação.
- D-É um método usado para liberar recursos quando um objeto não é mais necessário, sendo destruído ou descartado.
- E-Restringe o acesso direto aos atributos de um objeto, permitindo a manipulação apenas por meio de métodos específicos.

Questão 03

- Na programação orientada a objetos, um dos conceitos mais importantes é o de encapsulamento, essencial para garantir a segurança e a integridade dos dados. Esse recurso provê
- Alternativas
- A
- restrição do acesso direto aos dados internos de um objeto, expondo apenas os métodos desejados para manipulá-los.
- B
- capacidade de uma classe herdar atributos e métodos de outra classe.
- C
- capacidade de um método se comportar de formas diferentes, de acordo com o objeto que o invoca.
- D
- processo de criação de novas classes com base em classes existentes, sem alterações.
- E
- técnica de dividir um programa em funções e procedimentos menores para facilitar o desenvolvimento.

Questão 04

- Um desenvolvedor está criando um sistema de gerenciamento de pedidos em PHP e precisa organizar as funcionalidades em módulos reutilizáveis. Para isso, ele decide estruturar seu código utilizando programação orientada a objetos, garantindo maior organização e reutilização dos componentes do sistema. Assinale a alternativa correspondente à sintaxe que o desenvolvedor deve utilizar para declarar corretamente uma classe em PHP.
- Alternativas
- A - new NomeClasse {}
- B- define NomeClasse {}
- C- class NomeClasse {}
- D- create NomeClasse {}
- E- object NomeClasse {}

Questão 05

- Utilizando a linguagem de programação Java, qual palavra-chave permite herdar uma classe?
- Alternativas
- A- implements
- B- inherits
- C- super
- D- extends
- E- override

Questão 06

- Assinale a alternativa que descreve o principal conceito da programação orientada a objetos.
- Alternativas
- A- Dividir o programa em funções independentes e reutilizáveis.
- B- Organizar o código em classes e objetos, permitindo maior modularidade, reutilização.
- C- Escrever o código diretamente em linguagem de máquina, facilitando a manutenção.
- D-Utilizar apenas variáveis globais para facilitar o acesso aos dados.
- E-Criar estruturas exclusivamente sequenciais para reduzir a complexidade do código.

Questão 07

- Na programação orientada a objetos, herança especifica que uma classe (subclasse) herda características (atributos e métodos) de outra classe (superclasse). Nesse contexto, as características da superclasse que são comuns às subclasses e as características específicas das subclasses, são conhecidas, respectivamente, por:
- Alternativas
- A- Abstração e encapsulamento.
- B- Encapsulamento e abstração.
- C- Especialização e generalização.
- D- Generalização e especialização.
- E- Sobrecarga e sobrescrita.

Questão 08

- No que se refere à herança na programação orientada a objetos, assinale a opção correta.
- Alternativas
- A-A herança promove acoplamento fraco entre as classes.
- B-A herança elimina completamente a necessidade de composição.
- C- A herança permite que uma classe derivada reutilize métodos e atributos da superclasse.
- D-A herança permite que uma classe filha acesse diretamente os atributos privados da classe pai.
- E-Na herança, uma classe pode herdar múltiplas superclasses diretamente em todas as linguagens.

Questão 09

- Acerca da programação orientada a objetos, assinale a opção correta.
- Alternativas
- A- A classe é a estrutura que define os atributos e comportamentos dos objetos.
- B- Toda linguagem de programação define automaticamente todas as classes necessárias em tempo de compilação.
- C- Um objeto não pode ser instanciado a partir de uma classe.
- D- A classe representa uma instância específica de um objeto.
- E- Atributos e métodos são definidos apenas no objeto, não na classe.

Questão 10

- Na programação orientada a objetos, princípios fundamentais desempenham um papel específico na construção de software modular, reutilizável e de fácil manutenção.
- Com base no exposto, o princípio que permite que objetos de diferentes classes sejam tratados de maneira uniforme através de uma interface comum é
- Alternativas
- A- encapsulamento.
- B- polimorfismo.
- C- composição.
- D- abstração.
- E- herança.

Gab

- 1-E
- 2-A
- 3-A
- 4-C
- 5-D
- 6-B
- 7-D
- 8-C
- 9-A
- 10-B

Obrigado!