UNIVERSIDADE FEDERAL DE MATO GROSSO
Faculdade de Engenharia (FAENG)

Programação Orientada a Objetos

classes · objetos · atributos · métodos

PROVA DIDÁTICA
Concurso Público — Edital nº 01/PROGEP/UFMT/2026
Área: Ciência da Computação · Campus de Várzea Grande

Candidato: **Dr. Willian Garcias de Assunção**

Cuiabá / Várzea Grande — MT / 2026

Plano de aula

IDENTIFICAÇÃO

Disciplina:

Introdução à Programação Orientada a Objetos

Curso:

Bacharelado em Ciência da Computação

Público-alvo:

Estudantes do 2° semestre

Duração:

45 minutos

Pré-requisitos:

Lógica de programação e estruturada em Java

OBJETIVO GERAL

- *Compreender os conceitos fundamentais da POO, classes, objetos, atributos e métodos e aplicá-los na modelagem e implementação de soluções em Java.*

OBJETIVOS ESPECÍFICOS

- *Distinguir classe e objeto;*
- *Identificar atributos e métodos em uma classe;*
- *Aplicar modificadores de acesso e encapsulamento;*
- *Implementar uma classe em Java.*

Roteiro da aula

- ***Quatro blocos conectados: fundamentos → classes → instâncias → aplicação prática.***

1. **Fundamentos da POO**
   Paradigma, pilares, abstração e modelagem do mundo real
2. **Classes**
   Definição, anatomia, sintaxe Java e modificadores de acesso
3. **Objetos, atributos e métodos**
   Instâncias, estado, comportamento, getters/setters e construtores
4. **Aplicação e síntese**
   Estudo de caso, questão ENADE e fechamento

Tempo total previsto: 45 minutos

Por que estudar Orientação a Objetos?

- ***POO é hoje o paradigma dominante no desenvolvimento profissional de software.***

Linguagens dominantes

Java, C#, Python, C++, Kotlin, Swift — todas adotam POO como paradigma central.

Frameworks e padrões

Spring, .NET, Android SDK, Django e padrões de projeto (GoF) são construídos sobre POO.

Sistemas complexos

Modulariza grandes sistemas em entidades coesas, facilitando manutenção e evolução.

Empregabilidade

Vagas para desenvolvedor exigem POO como requisito básico no mercado.

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed. Pearson, 2016, cap. 1.

O problema da programação estruturada

- ***Conforme o software cresce, dados e funções soltos viram um caos de manutenção.***

Programação estruturada (procedural)

- **Dados separados** das funções que os manipulam
- **Variáveis globais** acessadas de qualquer lugar
- **Repetição** de código entre módulos
- Dificuldade de **reuso** e **manutenção**
- Modelo distante das entidades do mundo real

Programação orientada a objetos

- Dados e comportamento unidos em objetos
- Acesso controlado via **encapsulamento**
- **Reuso por herança** e composição
- **Manutenção** localizada na classe responsável
- Modelagem próxima das entidades do mundo real

Mudança de paradigma: pensar em objetos

Na programação estruturada, perguntamos:

"Quais funções o programa precisa executar?"

Na orientada a objetos, a pergunta muda para:

"Quais entidades existem no domínio e como elas se relacionam?"

> "
>
> *Objetos do mundo real compartilham duas características: todos têm estado e comportamento. Pessoas têm estado (nome, idade) e comportamento (responder a perguntas). A POO modela essas duas características em uma única entidade: o objeto.*
>
> *— adaptado de Barnes & Kölling (2009)*

BARNES, D.; KÖLLING, M. Programação Orientada a Objetos com Java. 4ª ed., 2009, cap. 1.

Paradigma orientado a objetos

- ***Programar em POO é descrever entidades autônomas que colaboram para resolver um problema.***

DEFINIÇÃO

POO é o paradigma de programação em que um sistema é organizado como uma coleção de objetos que se comunicam por troca de mensagens. Cada objeto encapsula estado (atributos) e comportamento (métodos) e pertence a uma classe que define seu tipo.

Objeto

Entidade com estado e comportamento

Classe

Tipo / molde dos objetos

Mensagem

Chamada de método entre objetos

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 1.6.

Os quatro pilares da POO

- ***Toda linguagem orientada a objetos sustenta-se sobre esses quatro princípios.***

1 Abstração

Foco nos aspectos essenciais, ignorando detalhes irrelevantes ao problema.

2 Encapsulamento

Ocultação dos detalhes internos; acesso controlado por interface pública.

3 Herança

Mecanismo pelo qual uma classe pode reutilizar e estender outra.

4 Polimorfismo

Capacidade de um mesmo método assumir comportamentos distintos.

Destaque: abstração e encapsulamento são o foco desta aula.

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 1.6.

Abstração: o ponto de partida

- ***Abstrair é decidir o que importa no problema e ignorar o resto.***

EXEMPLO

Um sistema de RH e um sistema médico modelam a mesma pessoa de formas distintas:

- *O RH abstrai salário e cargo;*
- *O sistema médico abstrai histórico clínico e alergias.*

Pessoa

✔ `nome`
✔ `idade`
✔ `cpf`

✘ `cor dos olhos`
✘ `altura do calçado`
✘ `marca do celular`

(contexto: sistema de RH)

BARNES, D.; KÖLLING, M. Programação Orientada a Objetos com Java. 4ª ed., 2009, cap. 1.1.

Do mundo real para o software

- ***Cada entidade relevante do problema vira uma classe; cada instância concreta vira um objeto.***

A modelagem é a etapa criativa: decidir quais classes existem e o que cada uma contém é o que separa um bom design de um ruim.

BARNES, D.; KÖLLING, M. Programação Orientada a Objetos com Java. 4ª ed., 2009, cap. 1.2.

O que é uma classe?

DEFINIÇÃO FORMAL

Uma classe é uma unidade de software que descreve um conjunto de objetos compartilhando a mesma estrutura (atributos) e o mesmo comportamento (métodos). É a especificação a partir da qual objetos são criados.

Uma classe define:

NOME	*Identificador único da classe (PascalCase)*
ATRIBUTOS	*Variáveis que armazenam o estado de cada objeto*
MÉTODOS	*Operações que definem o comportamento dos objetos*

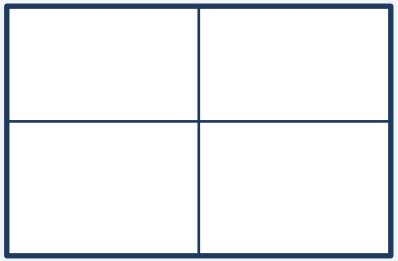
DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 3.2.

Analogia: a planta e as casas

- *A classe é como a planta arquitetônica; cada objeto, uma casa construída a partir dela.*

PLANTA (classe)

Casa



um único molde

instanciar

CASAS (objetos)

muitas instâncias possíveis

Anatomia de uma classe em Java

- ***Toda classe tem um cabeçalho e um corpo — o corpo contém atributos, construtores e métodos.***

```java
public class Pessoa {
    // 1. atributos (estado)
    private String nome;
    private int idade;

    // 2. construtor
    public Pessoa(String n, int i) {
        this.nome = n;
        this.idade = i;
    }

    // 3. métodos (comportamento)
    public String getNome() {
        return nome;
    }
}
```

1. **Atributos**
   Variáveis privadas que guardam o estado

2. **Construtor**
   Inicializa os atributos ao criar o objeto

3. **Métodos**
   Operações que o objeto pode executar

Sintaxe Java: declaração de classe

- ***A palavra-chave class introduz uma classe; PascalCase é a convenção para o nome.***

```
public class Carro {

}
```

`public` → *modificador de acesso (visível em qualquer pacote)*

`class` → *palavra-chave que declara uma classe*

`Carro` → *identificador (PascalCase: cada palavra inicia em maiúscula)*

`{ }` → *delimitam o corpo da classe (atributos, construtores, métodos)*

> ***Convenção:*** *o arquivo deve ter o mesmo nome da classe pública (Carro.java).*

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 3.2.

Modificadores de acesso

- *Os modificadores controlam quem pode ver e usar cada membro da classe.*

Modificador	Própria classe	Mesmo pacote	Subclasses	Qualquer classe
`private`	✓	—	—	—
`(default)`	✓	✓	—	—
`protected`	✓	✓	✓	—
`public`	✓	✓	✓	✓

Boa prática: atributos sempre private; acesso via métodos públicos (getters/setters).

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 3.5.

O que é um objeto?

- ***Objeto é uma instância concreta de uma classe — existe na memória durante a execução.***

DEFINIÇÃO

Um objeto é uma entidade individual, criada em tempo de execução a partir de uma classe, que possui identidade própria, mantém um estado em seus atributos e responde a mensagens por meio de seus métodos.

Identidade

Cada objeto é único, mesmo com atributos iguais

Estado

Valores atuais dos seus atributos

Comportamento

Métodos que ele sabe executar

BARNES, D.; KÖLLING, M. Programação Orientada a Objetos com Java. 4ª ed., 2009, cap. 1.3.

Classe × Objeto

- ***Existe uma classe, podem existir muitos objetos dela.***

CLASSE	OBJETO
Conceito abstrato (molde)	Entidade concreta (instância)
Existe em tempo de compilação	Existe em tempo de execução
Definida uma vez no código	Pode ser criada várias vezes
Não ocupa memória de instância	Ocupa memória no heap
`Carro`	`carro1, carro2, carro3...`

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 1.6.2.

Criação de objetos: o operador new

- *O operador new aloca memória, chama o construtor e devolve uma referência ao novo objeto.*

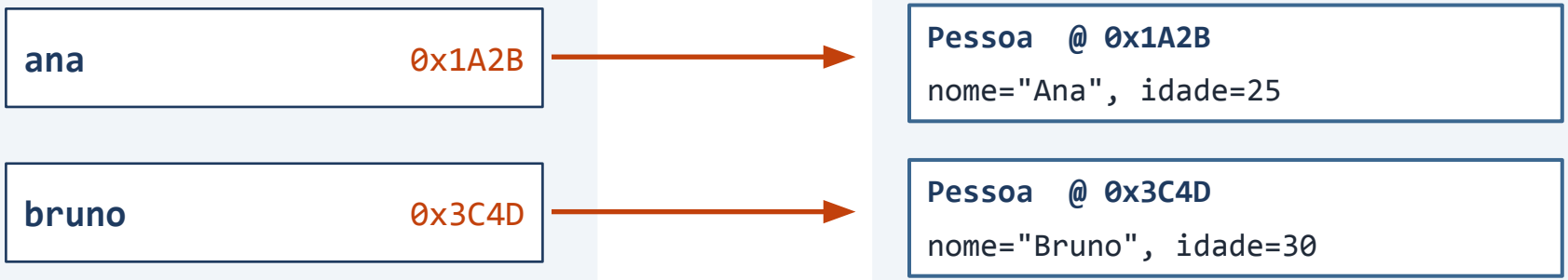
```
// Criando objetos a partir da classe Pessoa
Pessoa ana   = new Pessoa("Ana", 25);
Pessoa bruno = new Pessoa("Bruno", 30);
Pessoa carla = new Pessoa("Carla", 22);

// Cada chamada cria um objeto independente
```

1. **Aloca memória**
   Reserva espaço no heap para o novo objeto
2. **Chama o construtor**
   Inicializa os atributos com os valores informados
3. **Devolve referência**
   Armazena o endereço na variável (ana, bruno, carla)

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 3.3.

Referências e memória: stack × heap

- *Variáveis de objeto guardam referências (endereços), não o objeto em si.*



Quando duas variáveis recebem a mesma referência, ambas apontam para o mesmo objeto.

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 3.3.

Atributos: o estado do objeto

- ***Atributos são as variáveis declaradas dentro da classe; cada objeto tem sua própria cópia.***

```
public class Carro {
    private String modelo;
    private String cor;
    private int ano;
    private double velocidade;
}
```

Tipos de atributos

Primitivos — int, double, boolean, char...

Referência — objetos de outras classes (String, Data...)

Cada objeto Carro criado terá seu próprio valor para modelo, cor, ano e velocidade.

Dois objetos, dois estados independentes:

```
carro1.cor = "vermelho";
carro2.cor = "azul";
```

BARNES, D.; KÖLLING, M. Programação Orientada a Objetos com Java. 4ª ed., 2009, cap. 2.2.

Atributos de instância × de classe (static)

- ***Atributos de instância pertencem a cada objeto; atributos static pertencem à classe.***

Atributo de instância

```
private String nome;
```

- Cada objeto tem sua cópia
- Valor depende da instância
- Acessado por: `objeto.atributo`

Atributo de classe (static)

```
private static int total;
```

- Compartilhado por todos
- Único na memória
- Acessado por: `Classe.atributo`

Exemplo clássico: contador de instâncias criadas — usar static.

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 6.10.

Encapsulamento dos atributos

- ***Atributos privados + métodos públicos: o objeto controla seu próprio estado.***

__Encapsulamento__ é o princípio que esconde os detalhes internos de um objeto, permitindo que outros objetos interajam com ele apenas através de uma interface bem-definida __(métodos públicos)__.

Por que tornar atributos privados?

- ✓ **Proteção**: Impede que outros objetos modifiquem o estado de forma inconsistente.
- ✓ **Validação**: Garante que valores atribuídos passem por checagens (ex.: idade > 0).
- ✓ **Manutenibilidade**: Mudanças internas não afetam quem usa a classe.

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 3.5.

Métodos: o comportamento do objeto

- ***Métodos definem o que um objeto sabe fazer — são as operações executáveis sobre seu estado.***

Um método é um bloco de código nomeado, pertencente a uma classe, que pode receber parâmetros, executar instruções e opcionalmente retornar um valor. Métodos são acionados por mensagens enviadas a um objeto.

Exemplo:

```
public void acelerar(int incremento) {
    this.velocidade = this.velocidade + incremento;
}
```

BARNES, D.; KÖLLING, M. Programação Orientada a Objetos com Java. 4ª ed., 2009, cap. 2.3.

Anatomia de um método

- ***Assinatura = nome + lista de parâmetros. Define a identidade do método na classe.***

```
public double calcularJuros(double taxa, int meses) {
  ...
}
```

`public` → modificador de acesso — define quem pode chamar o método

`double` → tipo de retorno — tipo do valor devolvido (ou void se nada retorna)

`calcularJuros` → nome do método — verbo em camelCase, descreve a ação

`(double taxa, int meses)` → lista de parâmetros — entradas que o método recebe

`{ ... }` → corpo — instruções executadas quando o método é chamado

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 3.2.2.

Getters e setters

- *São os métodos públicos que dão acesso controlado aos atributos privados.*

```
private int idade;

// getter – devolve o valor do atributo
public int getIdade() {
    return this.idade;
}

// setter – atribui um novo valor, com validação
public void setIdade(int novaIdade) {
    if (novaIdade >= 0) {
        this.idade = novaIdade;
    }
}
```

Construtores

- ***Construtor é o método especial chamado automaticamente pelo new para inicializar o objeto.***

```
public class Pessoa {
    private String nome;
    private int idade;

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }
}
```

Regras do construtor:

- Tem o mesmo nome da classe;
- Não declara tipo de retorno (nem void);
- É invocado automaticamente pelo operador new.

Sobrecarga de métodos (overloading)

- *Uma classe pode ter vários métodos com o mesmo nome, desde que assinaturas sejam distintas.*

```
public class Calculadora {
    public int somar(int a, int b) {
        return a + b;
    }
    public double somar(double a, double b) {
        return a + b;
    }
    public int somar(int a, int b, int c) {
        return a + b + c;
    }
}
```

O compilador escolhe a versão correta pelos tipos e quantidade de argumentos passados.

DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, seção 6.12.

Estudo de caso: classe ContaBancaria

- ***Vamos integrar todos os conceitos modelando uma conta bancária simples.***

Requisitos do problema:

- Cada conta tem um número, titular e saldo.
- Operações: depositar, sacar (com validação) e consultar saldo.
- Saldo nunca pode ser modificado diretamente, apenas via operações.

Atributos (estado):

`numero` (String)
`titular` (String)
`saldo` (double)

Métodos (comportamento):

`depositar(valor)`
`sacar(valor)`
`getSaldo()`

Adaptado de DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, cap. 3.

Implementação da classe ContaBancaria

- ***Atributos privados, construtor parametrizado, métodos públicos com validação.***

```
public class ContaBancaria {

  private String numero;
  private String titular;
  private double saldo;

  public ContaBancaria(String numero, String titular) {
        this.numero  = numero;
        this.titular = titular;
        this.saldo   = 0.0;
  }

  public double getSaldo() {
       return this.saldo;
  }
```

Implementação da classe ContaBancaria

- ***Atributos privados, construtor parametrizado, métodos públicos com validação.***

```
    // método depositar
    public void depositar(double valor) {
        if (valor > 0) {
            this.saldo = this.saldo + valor;
        }
    }

    // método sacar (com validação)
    public boolean sacar(double valor) {
        if (valor > 0 && valor <= this.saldo) {
            this.saldo = this.saldo - valor;
            return true;
        }
        return false;
    }
}
```

Traço de execução: usando a classe

- ***Acompanhe o estado do objeto a cada operação.***

```
ContaBancaria c = new ContaBancaria("001", "Ana");

c.depositar(500.00);
c.depositar(200.00);
c.sacar(100.00);

System.out.println(c.getSaldo());
```

Estado do objeto c:

após new	R$ 0,00
depositar(500)	R$ 500,00
depositar(200)	R$ 700,00
sacar(100)	R$ 600,00

Saída no console: `600.0`

O encapsulamento garante que o saldo só seja modificado pelas regras dos métodos.

Adaptado de DEITEL, P.; DEITEL, H. Java: Como Programar. 10ª ed., 2016, cap. 3.

Verificação de aprendizagem

- *Questão ENADE 2017 — Ciência da Computação · Tema: encapsulamento e modificadores.*

O encapsulamento é um mecanismo da POO no qual os membros de uma classe (atributos e métodos) constituem uma caixa-preta. Sobre o comportamento gerado pelos modificadores de visibilidade, assinale a opção correta.

A Atributo privado pode ser acessado por métodos protegidos das subclasses.

B Atributo privado pode ser acessado por métodos públicos das subclasses.

C Membro público é visível na própria classe, mas não nas descendentes.

D Método protegido não pode acessar atributos privados da própria classe.

E Membro protegido é visível na própria classe e nas suas classes descendentes.

INEP. ENADE 2017 — Ciência da Computação. Questão 13 (adaptada).

Verificação de aprendizagem

- *Questão ENADE 2017 — Ciência da Computação · Tema: encapsulamento e modificadores.*

O encapsulamento é um mecanismo da POO no qual os membros de uma classe (atributos e métodos) constituem uma caixa-preta. Sobre o comportamento gerado pelos modificadores de visibilidade, assinale a opção correta.

A	**Atributo privado pode ser acessado por métodos protegidos das subclasses.**	
B	**Atributo privado pode ser acessado por métodos públicos das subclasses.**	
C	**Membro público é visível na própria classe, mas não nas descendentes.**	
D	**Método protegido não pode acessar atributos privados da própria classe.**	
E	**Membro protegido é visível na própria classe e nas suas classes descendentes.**	✓

INEP. ENADE 2017 — Ciência da Computação. Questão 13 (adaptada).

Síntese da aula

- ***Como os conceitos se conectam — mapa do que vimos hoje.***

ENCAPSULAMENTO (protege atributos via métodos)

Estes conceitos fundamentam todos os demais pilares: herança, polimorfismo e o design de sistemas reais.

Para casa

- ***Exercícios práticos e leitura prévia para a próxima aula.***

EXERCÍCIOS

1. Modelar e implementar em Java a classe **`Livro`** com atributos (título, autor, ano, disponível) e métodos para emprestar e devolver.
2. Criar a classe **`Aluno`** com matrícula, nome e notas. Implementar média e situação (aprovado/reprovado).
3. Justificar, em até 5 linhas, por que atributos devem ser privados em vez de públicos.

LEITURA PRÉVIA — próxima aula: herança e polimorfismo

- BARNES & KÖLLING — capítulo 8 (herança).
- DEITEL & DEITEL — capítulos 9 e 10.

Referências bibliográficas

- ***Bibliografia — base teórica da aula.***

BIBLIOGRAFIA BÁSICA

BARNES, David J.; KÖLLING, Michael. *Programação Orientada a Objetos com Java (BlueJ)*. 4ª ed. São Paulo: Pearson Prentice Hall, 2009.

DEITEL, Paul; DEITEL, Harvey. *Java: Como Programar*. 10ª ed. São Paulo: Pearson, 2016.

BIBLIOGRAFIA COMPLEMENTAR

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML: Guia do Usuário*. 2ª ed.

INEP. *ENADE 2017 — Prova de Ciência da Computação*. Brasília, 2017.

Material de apoio disponível no QR code do plano de aula.

UNIVERSIDADE FEDERAL DE MATO GROSSO

Faculdade de Engenharia (FAENG)

Obrigado!

PROVA DIDÁTICA

Concurso Público — Edital nº 01/PROGEP/UFMT/2026

Área: Ciência da Computação · Campus de Várzea Grande

Candidato: **Dr. Willian Garcias de Assunção**

Cuiabá / Várzea Grande — MT / 2026

UFMT

Aula em vídeo

Versão audiovisual completa