

PROGRAMAÇÃO ORIENTADA A OBJETOS

PROGRAMAÇÃO ORIENTADA A OBJETOS

Público-alvo: Estudantes do Centro Universitário FMU.

Objetivo: Esta apostila tem como objetivo conduzir os alunos recém-ingressos na graduação em Tech na transição do raciocínio lógico e estruturado para o Paradigma de Programação Orientada a Objetos (POO). O material foi desenhado de forma altamente didática, com linguagem clara, analogias do cotidiano e exemplos progressivos de código em Java.

Além de cobrir os pilares fundamentais da POO (abstração, encapsulamento, herança, polimorfismo e coleções), a apostila culmina na aplicação prática de persistência de dados utilizando um banco de dados moderno NoSQL (MongoDB), capacitando o estudante a desenvolver aplicações completas de nível básico a intermediário com operações de CRUD (Create, Read, Update, Delete) integradas. Toda a estrutura do material está alinhada com o cronograma de 66 horas semestrais e o plano de ensino oficial.

Lattes: <http://lattes.cnpq.br/8753112683856964>

Orcid: <https://orcid.org/0009-0002-7102-4993>

LLM utilizada: Gemini Flash – Agosto de 2026.

Unidade 2: Fundamentos do Java — Entrada, Processamento e Saída de Dados

Exercícios de Fixação

Questão 1

(A). A asserção I descreve exatamente o papel do compilador javac em produzir o Bytecode (.class). A asserção II justifica perfeitamente a I, explicando como a JVM atua em cada sistema operacional garantindo a portabilidade.

Questão 2

(B). Na expressão $22 / 3$, operando e divisor são do tipo int, portanto o Java trunca o resultado para o inteiro 7. Ao atribuir à variável double media, ele converte para 7.0. A utilização de 3.0 transforma a operação em uma divisão de ponto flutuante.

Questão 3

(C). O casting explícito (int) descarta a parte fracionária (.85) e converte o valor diretamente para o inteiro 1250, sem realizar arredondamento.

Questão 4

(C). A avaliação ocorre da esquerda para a direita. Primeiro é realizada a soma aritmética $a + b$ ($5 + 10 = 15$). Em seguida, o resultado 15 é concatenado com a String "result", gerando "15 result".

Questão 5

(B). O item I utiliza snake_case (convenção de outras linguagens). O item III é inválido na sintaxe do Java pois começa com um número (1aParcela). Os itens II e IV seguem rigorosamente a sintaxe e a convenção camelCase.

Exercícios Práticos de Programação

Questão 1

```
import java.util.Scanner;

public class ConversorTemperatura {
    public static void main(String[] args) {
        Scanner leitor = new Scanner(System.in);

        System.out.print("Digite a temperatura em Fahrenheit (°F): ");
        double fahrenheit = leitor.nextDouble();

        // Usamos 5.0 / 9.0 para garantir divisão com casas decimais
    }
}
```

```
double celsius = (5.0 / 9.0) * (fahrenheit - 32);

System.out.printf("Temperatura em Celsius: %.2f °C\n", celsius);

leitor.close();
}
}
```

Questão 2

```
import java.util.Scanner;

public class CalculadoraSalario {
    public static void main(String[] args) {
        Scanner leitor = new Scanner(System.in);

        System.out.print("Digite o valor ganho por hora (R$): ");
        double valorHora = leitor.nextDouble();

        System.out.print("Digite as horas trabalhadas no mês: ");
        double horasTrabalhadas = leitor.nextDouble();

        System.out.print("Digite o percentual de desconto do INSS (ex: 11): ");
        double percentualInss = leitor.nextDouble();

        // Processamento
        double salarioBruto = valorHora * horasTrabalhadas;
        double valorDesconto = salarioBruto * (percentualInss / 100);
        double salarioLiquido = salarioBruto - valorDesconto;

        // Saída de dados formatada
        System.out.println("\n=== DEMONSTRATIVO DE PAGAMENTO ===");
        System.out.printf("Salário Bruto: R$ %.2f\n", salarioBruto);
        System.out.printf("Desconto INSS: R$ %.2f\n", valorDesconto);
        System.out.printf("Salário Líquido: R$ %.2f\n", salarioLiquido);

        leitor.close();
    }
}
```

Questão 3

```
import java.util.Scanner;

public class DecomposicaoTempo {
    public static void main(String[] args) {
        Scanner leitor = new Scanner(System.in);

        System.out.print("Digite o tempo total em segundos: ");
        int totalSegundos = leitor.nextInt();

        // 1 hora possui 3600 segundos
        int horas = totalSegundos / 3600;

        // O resto da divisão por 3600 são os segundos que sobraram das horas
        int restoSegundos = totalSegundos % 3600;

        // 1 minuto possui 60 segundos
        int minutos = restoSegundos / 60;

        // O resto da divisão por 60 são os segundos finais
        int segundos = restoSegundos % 60;

        // Exibe o tempo no formato HH:MM:SS
        System.out.printf("Tempo formatado: %02d:%02d:%02d\n", horas, minutos, segundos);

        leitor.close();
    }
}
```

Unidade 3: Lógica de Controle — Estruturas Condicionais e Laços de Repetição

Exercícios de Fixação

Questão 1

(B). Vamos rastrear a execução passo a passo (teste de mesa):

Volta 1: $x = 10$. É par ($10 \% 2 == 0$). y recebe $0 + 10 = 10$. x vira $10 - 3 = 7$.

Volta 2: $x = 7$. É ímpar. y continua 10. x vira $7 - 3 = 4$.

Volta 3: $x = 4$. É par ($4 \% 2 == 0$). y recebe $10 + 4 = 14$. x vira $4 - 3 = 1$.

Volta 4: $x = 1$. É ímpar. y continua 14. x vira $1 - 3 = -2$.

Condição do while ($-2 > 0$) é falsa. O laço encerra e imprime $y = 14$.

Questão 2

(C). Como a primeira condição ($2 > 5$) é falsa, o operador `&&` já sabe que toda a expressão será falsa, sem precisar avaliar o segundo termo ($c++ > 5$). Por conta da avaliação em curto-circuito, o incremento `c++` não é executado, mantendo $c = 10$ e resultado = false.

Questão 3

(D). Como o programa entra no case 2:, resultado recebe $0 + 20 = 20$. Como o case 2 não possui a instrução `break`, a execução continua para o case 3: (fall-through), adicionando mais 30 ao resultado ($20 + 30 = 50$). No case 3 existe o `break`, encerrando o switch. O resultado impresso é 50.

Questão 4

(A). No for original, a variável começa em 0, a impressão ocorre antes do incremento, e o laço para quando i chega a 5 (imprimindo 01234). A alternativa (B) replica exatamente essa sequência.

Questão 5

(C). A condição de ter média suficiente (m) OU renda baixa (r) deve ser agrupada entre parênteses: $(m || r)$. A condição obrigatória final de NÃO ter reprovação por falta (!f) deve ser combinada com o operador E (&&), resultando em $(m || r) \&\& !f$.

Exercícios Práticos de Programação

Questão 1

```
import java.util.Scanner;

public class CalculadoraIMC {
    public static void main(String[] args) {
        Scanner leitor = new Scanner(System.in);

        System.out.print("Digite seu peso (kg): ");
        double peso = leitor.nextDouble();

        System.out.print("Digite sua altura em metros (ex: 1,75): ");
        double altura = leitor.nextDouble();

        double imc = peso / (altura * altura);

        System.out.printf("Seu IMC é: %.2f\nClassificação: ", imc);

        if (imc < 18.5) {
            System.out.println("Abaixo do peso");
        } else if (imc < 25.0) {
            System.out.println("Peso normal");
        } else if (imc < 30.0) {
            System.out.println("Sobrepeso");
        } else {
            System.out.println("Obesidade");
        }

        leitor.close();
    }
}
```

Questão 2

```
import java.util.Scanner;

public class AnaliseNumeros {
    public static void main(String[] args) {
        Scanner leitor = new Scanner(System.in);

        int soma = 0;
        int maior = Integer.MIN_VALUE; // Inicializado com o menor inteiro possível
        int menor = Integer.MAX_VALUE; // Inicializado com o maior inteiro possível

        System.out.println("=== LEITURA DE 5 NÚMEROS ===");

        for (int i = 1; i <= 5; i++) {
            System.out.print("Digite o " + i + "º número: ");
            int numero = leitor.nextInt();

            soma += numero;

            if (numero > maior) {
                maior = numero;
            }

            if (numero < menor) {
                menor = numero;
            }
        }

        double media = (double) soma / 5;
        System.out.println("\n=== RESULTADOS ===");
        System.out.println("Maior número digitado: " + maior);
        System.out.println("Menor número digitado: " + menor);
        System.out.printf("Média dos números: %.2f\n", media);

        leitor.close();
    }
}
```

Questão 3

```
import java.util.Scanner;

public class ValidacaoCaixaEletronico {
    public static void main(String[] args) {
        Scanner leitor = new Scanner(System.in);
        final int SENHA_CORRETA = 1234;
        int tentativas = 0;
        boolean acessoConcedido = false;

        while (tentativas < 3 && !acessoConcedido) {
            tentativas++;
            System.out.print("Digite sua senha de 4 dígitos (Tentativa " + tentativas + "/" + 3 + "): ");
            int senhaDigitada = leitor.nextInt();

            if (senhaDigitada == SENHA_CORRETA) {
                acessoConcedido = true;
                System.out.println("\n>>> Acesso Permitido! Bem-vindo ao Banco FMU. <<<");
            } else {
                System.out.println("Senha incorreta!");
            }
        }

        if (!acessoConcedido) {
            System.out.println("\n>>> Limite de tentativas excedido. Cartão Bloqueado! <<<");
        }

        leitor.close();
    }
}
```

Unidade 4: O Coração da POO — Classes, Objetos, Atributos, Métodos e Encapsulamento

Exercícios de Fixação

Questão 1

(A). O encapsulamento exige a restrição de acesso direto aos atributos (private) e a exposição de uma interface controlada via métodos getters e setters públicos para manter a integridade dos dados.

Questão 2

(B). Em Java, variáveis de objetos armazenam referências de memória e não cópias do objeto. Quando é feito `a2 = a1`, ambas as variáveis passam a apontar para o mesmo objeto na memória RAM. Assim, alterar o nome via `a2` reflete na leitura por `a1`, imprimindo "Roberto".

Questão 3

(C). As afirmações I e III estão corretas. A afirmação II está falsa porque atributos estáticos são compartilhados globalmente por todas as instâncias da classe, existindo em uma única cópia na memória.

Questão 4

(A). O Java provê automaticamente o construtor default `public NomeClasse() {}` caso nenhum outro seja explicitamente escrito pelo desenvolvedor. Se o desenvolvedor criar qualquer construtor personalizado, o padrão deixa de ser inserido automaticamente.

Questão 5

(B). Na teoria de POO, atributos representam o estado do objeto (suas características e informações armazenadas num determinado instante) e métodos representam seu comportamento (suas ações e operações).

Exercícios Práticos de Programação

Questão 1

```
public class Produto2 {
    private String nome;
    private double preco;
    private int quantidadeEstoque;

    public Produto2(String nome, double preco, int quantidadeEstoque) {
        this.nome = nome;
        setPreco(preco);
        setQuantidadeEstoque(quantidadeEstoque);
    }

    public void adicionarEstoque(int qtd) {
        if (qtd > 0) {
            this.quantidadeEstoque += qtd;
            System.out.println("Estoque atualizado: " + this.quantidadeEstoque + " unidades.");
        }
    }

    public void vender(int qtd) {
        if (qtd > 0 && qtd <= this.quantidadeEstoque) {
            this.quantidadeEstoque -= qtd;
            double totalVenda = qtd * this.preco;
            System.out.printf("Venda realizada! Total: R$ %.2f. Estoque restante: %d\n",
totalVenda, this.quantidadeEstoque);
        } else {
            System.out.println("Venda não realizada: Quantidade indisponível no estoque.");
        }
    }

    // Getters e Setters
    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }

    public double getPreco() { return preco; }
    public void setPreco(double preco) {
        if (preco >= 0) this.preco = preco;
    }

    public int getQuantidadeEstoque() { return quantidadeEstoque; }
    public void setQuantidadeEstoque(int quantidadeEstoque) {
        if (quantidadeEstoque >= 0) this.quantidadeEstoque = quantidadeEstoque;
    }
}
```

Questão 2

```
public class Retangulo {
    private double largura;
    private double altura;

    public Retangulo(double largura, double altura) {
        this.largura = (largura > 0) ? largura : 1.0;
        this.altura = (altura > 0) ? altura : 1.0;
    }

    public double calcularArea() {
        return this.largura * this.altura;
    }

    public double calcularPerimetro() {
        return 2 * (this.largura + this.altura);
    }

    public boolean eQuadrado() {
        return this.largura == this.altura;
    }
}
```

```
// Getters e Setters
public double getLargura() { return largura; }
public double getAltura() { return altura; }
}
```

Questão 3

```
public class TestaProduto {
    public static void main(String[] args) {
        // Criando dois objetos do tipo Produto
        Produto2 p1 = new Produto2("Notebook Gamer", 4500.00, 10);
        Produto2 p2 = new Produto2("Mouse Óptico", 80.00, 50);

        System.out.println("=== TESTANDO PRODUTO 1 ===");
        p1.vender(3); // Venda bem-sucedida
        p1.vender(15); // Venda bloqueada (estoque insuficiente)
        p1.adicionarEstoque(5); // Reabastecimento

        System.out.println("\n=== TESTANDO PRODUTO 2 ===");
        p2.vender(10);
        System.out.println("Estoque atual do " + p2.getNome() + ": " +
            p2.getQuantidadeEstoque());
    }
}
```

Questão 4

Cliente.java

```
public class Cliente {
    private String razaoSocial;
    private String cnpj;
    private String inscricaoEstadual;
    private String endereco;
    private String email;
    private String celular;

    // Construtor
    public Cliente(String razaoSocial, String cnpj, String inscricaoEstadual,
        String endereco, String email, String celular) {
        setRazaoSocial(razaoSocial);
        setCnpj(cnpj);
        this.inscricaoEstadual = inscricaoEstadual;
        this.endereco = endereco;
        this.email = email;
        this.celular = celular;
    }

    // Getters e Setters com Validação
    public String getRazaoSocial() { return razaoSocial; }
    public void setRazaoSocial(String razaoSocial) {
        if (razaoSocial != null && !razaoSocial.trim().isEmpty()) {
            this.razaoSocial = razaoSocial;
        } else {
            System.out.println("Erro: Razão Social não pode ser vazia.");
        }
    }

    public String getCnpj() { return cnpj; }
    public void setCnpj(String cnpj) {
        if (cnpj != null && !cnpj.trim().isEmpty()) {
            this.cnpj = cnpj;
        } else {
            System.out.println("Erro: CNPJ é obrigatório.");
        }
    }

    public String getInscricaoEstadual() { return inscricaoEstadual; }
}
```

```
    public void setInscricaoEstadual(String inscricaoEstadual) { this.inscricaoEstadual = inscricaoEstadual; }

    public String getEndereco() { return endereco; }
    public void setEndereco(String endereco) { this.endereco = endereco; }

    public String getEmail() { return email; }
    public void setEmail(String email) { this.email = email; }

    public String getCelular() { return celular; }
    public void setCelular(String celular) { this.celular = celular; }
}
```

Produto3.java

```
public class Produto3 {
    private int codigo;
    private String nome;
    private String ncm;
    private double valor;

    // Construtor
    public Produto3(int codigo, String nome, String ncm, double valor) {
        this.codigo = codigo;
        this.nome = nome;
        this.ncm = ncm;
        setValor(valor);
    }

    // Getters e Setters
    public int getCodigo() { return codigo; }
    public void setCodigo(int codigo) { this.codigo = codigo; }

    public String getNome() { return nome; }
    public void setNome(String nome) { this.nome = nome; }

    public String getNcm() { return ncm; }
    public void setNcm(String ncm) { this.ncm = ncm; }

    public double getValor() { return valor; }
    public void setValor(double valor) {
        if (valor > 0) {
            this.valor = valor;
        } else {
            System.out.println("Erro: O valor do produto deve ser maior que zero.");
        }
    }
}
```

Fatura.java

```
import java.util.ArrayList;
import java.util.List;

public class Fatura {
    private int codigo;
    private Cliente cliente;
    private String dataEmissao;
    private List<Produto3> produtos;
    private double valorTotal;

    // Construtor
    public Fatura(int codigo, Cliente cliente, String dataEmissao) {
        this.codigo = codigo;
        this.cliente = cliente;
        this.dataEmissao = dataEmissao;
        this.produtos = new ArrayList<>();
        this.valorTotal = 0.0;
    }
}
```

```
// Métodos de negócio
public void adicionarProduto(Produto3 p) {
    if (p != null) {
        this.produtos.add(p);
        this.valorTotal += p.getValor();
        System.out.println("Produto '" + p.getNome() + "' adicionado à fatura.");
    }
}

public void removerProduto(Produto3 p) {
    if (this.produtos.remove(p)) {
        this.valorTotal -= p.getValor();
        System.out.println("Produto '" + p.getNome() + "' removido da fatura.");
    } else {
        System.out.println("Produto não encontrado na fatura.");
    }
}

public void imprimirFatura() {
    System.out.println("\n=====");
    System.out.println("FATURA COMERCIAL Nº " + this.codigo);
    System.out.println("=====");
    System.out.println("Data de Emissão: " + this.dataEmissao);
    System.out.println("-----");
    System.out.println("DADOS DO CLIENTE:");
    System.out.println("Razão Social: " + this.cliente.getRazaoSocial());
    System.out.println("CNPJ: " + this.cliente.getCnpj() + " | IE: " +
this.cliente.getInscricaoEstadual());
    System.out.println("Endereço: " + this.cliente.getEndereco());
    System.out.println("Contato: " + this.cliente.getEmail() + " | Tel: " +
this.cliente.getCelular());
    System.out.println("-----");
    System.out.println("ITENS DA FATURA:");
    System.out.printf("%-6s | %-20s | %-10s | %-10s\n", "CÓD", "NOME", "NCM", "VALOR (R$)");
    System.out.println("-----");

    for (Produto3 p : this.produtos) {
        System.out.printf("%-6d | %-20s | %-10s | R$ %8.2f\n",
            p.getCodigo(), p.getNome(), p.getNcm(), p.getValor());
    }

    System.out.println("-----");
    System.out.printf("VALOR TOTAL DA FATURA: R$ %8.2f\n",
this.valorTotal);
    System.out.println("=====");
}

// Getters e Setters
public int getCodigo() { return codigo; }
public Cliente getCliente() { return cliente; }
public String getDataEmissao() { return dataEmissao; }
public List<Produto3> getProdutos() { return produtos; }
public double getValorTotal() { return valorTotal; }
}
```

TestarFatura.java

```
public class TestaFatura {
    public static void main(String[] args) {
        // 1. Instanciando o Cliente
        Cliente cliente = new Cliente(
            "Tech Solutions Brasil LTDA",
            "12.345.678/0001-90",
            "110.123.456.115",
            "Av. Paulista, 1000 - São Paulo/SP",
            "contato@techsolutions.com.br",
            "(11) 98765-4321"
        );

        // 2. Instanciando os Produtos
        Produto3 p1 = new Produto3(101, "Servidor Rack 2U", "8471.50.10", 12500.00);
        Produto3 p2 = new Produto3(102, "Switch Gerenciável 24P", "8517.62.54", 3200.50);
    }
}
```

```
Produto3 p3 = new Produto3(103, "Nobreak Senoidal 3kVA", "8504.40.10", 4150.00);

// 3. Criando a Fatura
Fatura fatura = new Fatura(5001, cliente, "11/09/2026");

// 4. Associando produtos à fatura
System.out.println("=== PROCESSANDO EMISSÃO DE FATURA ===");
fatura.adicionarProduto(p1);
fatura.adicionarProduto(p2);
fatura.adicionarProduto(p3);

// 5. Exibindo a Fatura Impressa
fatura.imprimirFatura();
}
```