

Fundamentos de Desenvolvimento com Java

Professor: Armenio cardoso



Kaike Torres da silva

16.03.2025

Análise e Desenvolvimento de Sistemas (ADS)

INTRODUÇÃO

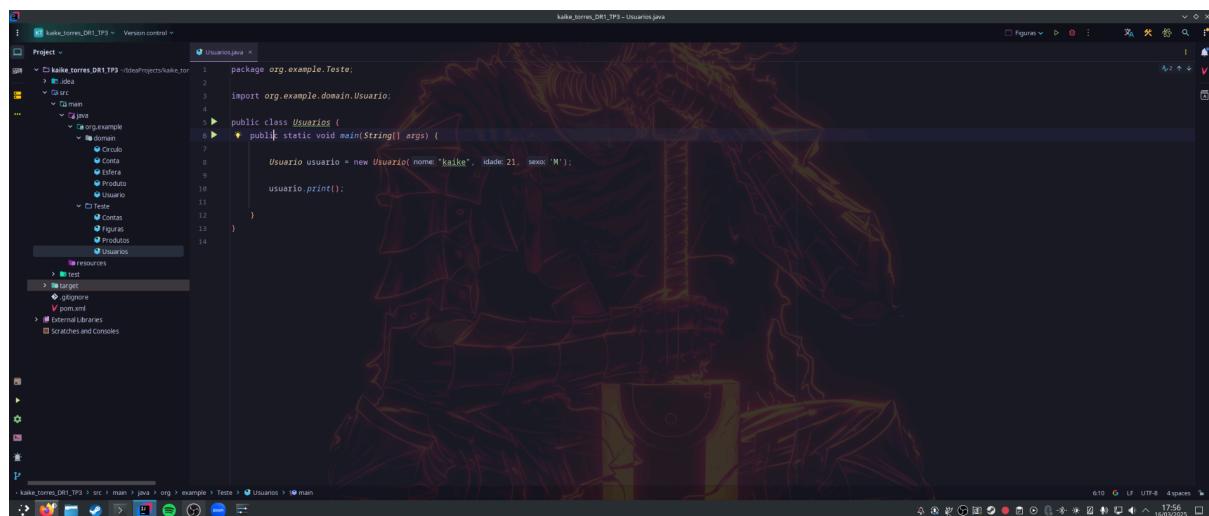
Neste conjunto de exercícios, explorei conceitos fundamentais de Programação Orientada a Objetos (POO) em Java, como classes, objetos, atributos e métodos. A partir de um contexto de gerenciamento de produtos e contas bancárias, criei classes e métodos para manipular atributos como preço, quantidade e saldo, aplicando o conhecimento de forma prática. Além disso, implementei métodos como getters, setters e construtores para organizar melhor o código e facilitar a criação e manipulação dos objetos.

Exercício 1 - Conceitos de Classe, Objeto, Campos e Métodos

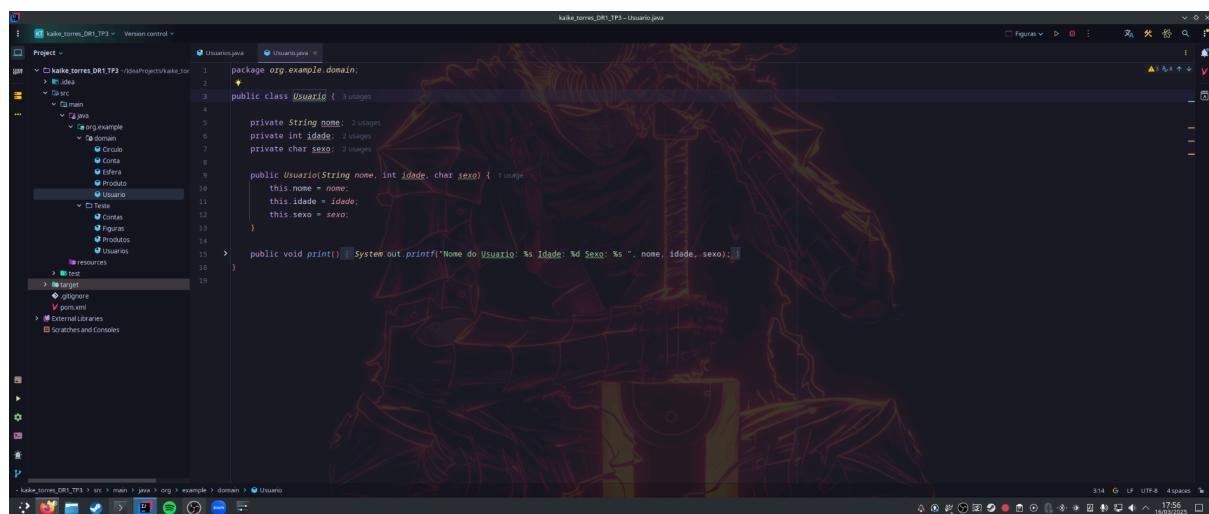
Neste código, foram criadas duas classes em pacotes distintos, com a principal funcionalidade de exibir as informações de um usuário.

Pacote: org.example.Teste

A classe `Usuarios` é a principal do programa e contém o método `main`. Dentro deste método, é criado um objeto da classe `Usuario` com os parâmetros "kaike", 21 e 'M' para nome, idade e sexo, respectivamente. O método `print()` da classe `Usuario` é então chamado para exibir as informações desse usuário.



```
package org.example.Teste;
import org.example.domain.Usuario;
public class Usuarios {
    public static void main(String[] args) {
        Usuario usuario = new Usuario("kaike", 21, 'M');
        usuario.print();
    }
}
```



```
package org.example.domain;
public class Usuario {
    private String nome;
    private int idade;
    private char sexo;
    public Usuario(String nome, int idade, char sexo) {
        this.nome = nome;
        this.idade = idade;
        this.sexo = sexo;
    }
    public void print() {
        System.out.printf("Nome do Usuario: %s Idade: %d Sexo: %s", nome, idade, sexo);
    }
}
```

```

package org.example.Teste;
import org.example.domain.Usuario;
public class Usuarios {
    public static void main(String[] args) {
        Usuario usuario = new Usuario("kaike", 21, 'M');
        usuario.print();
    }
}

```

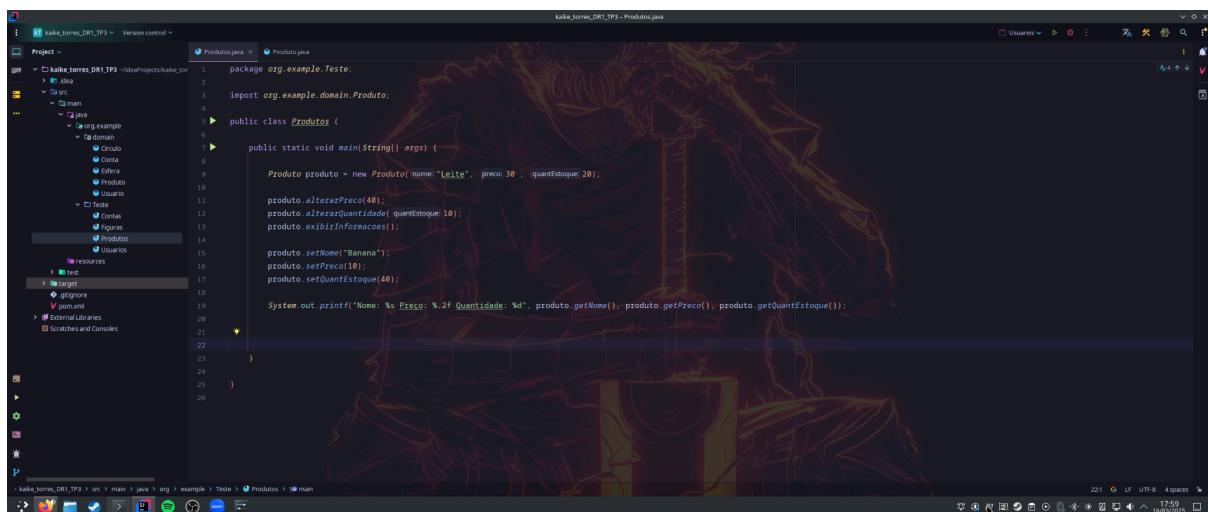
Exercício 2 - Criando a Classe “Produto” (Com Contexto de Usuário)

Este código envolve a criação de duas classes em pacotes diferentes, sendo uma destinada ao gerenciamento de produtos e a outra à exibição e manipulação das informações desses produtos.

Pacote: org.example.Teste

A classe `Produtos` contém o método `main`, onde a interação com a classe `Produto` ocorre. Dentro deste método, um objeto `produto` é criado, e diversas operações são realizadas sobre ele:

- O objeto `produto` é instanciado com os valores `nome = "Leite"`, `preco = 30` e `quantEstoque = 20`.
- O método `alterarPreco()` altera o preço do produto para `40`.
- O método `alterarQuantidade()` altera a quantidade em estoque para `10`.
- O método `exibirInformacoes()` é chamado para mostrar as informações do produto com os valores alterados.
- O nome, preço e quantidade do produto são atualizados diretamente utilizando os métodos `setNome()`, `setPreco()` e `setQuantEstoque()`.
- Por fim, as informações do produto atualizado são exibidas com `System.out.printf()`.



```
package org.example.Teste;

import org.example.domain.Produto;

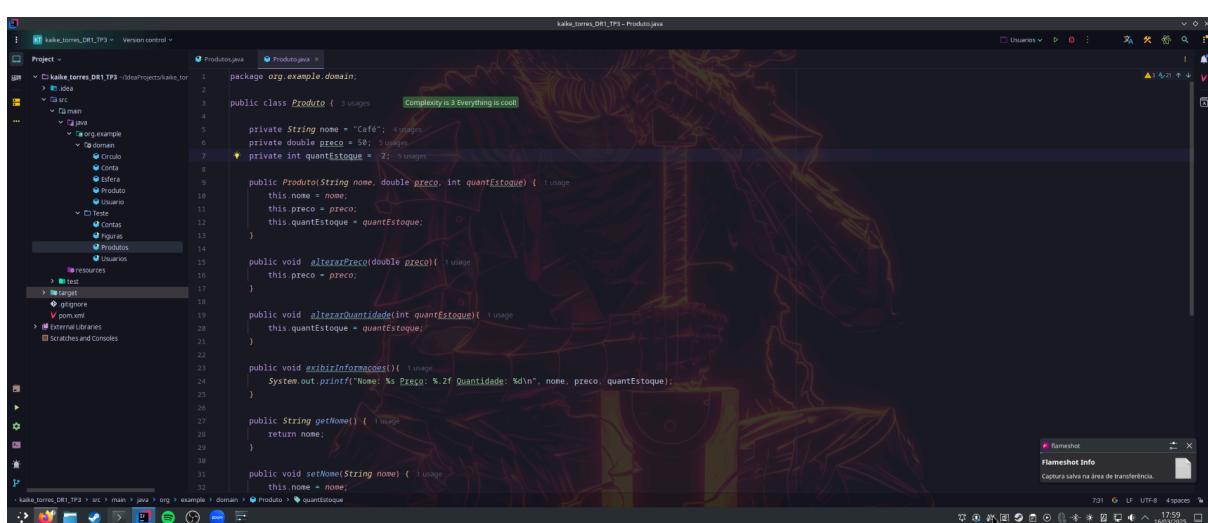
public class Produtos {

    public static void main(String[] args) {
        Produto produto = new Produto(nome: "Leite", preco: 10.00, quantEstoque: 10);

        produto.alterarPreco(40);
        produto.alterarQuantidade(quantEstoque: 10);
        produto.exibirInformacoes();

        produto.setNome("Banana");
        produto.setPreco(10);
        produto.setQuantEstoque(40);

        System.out.printf("Nome: %s Preco: %.2f Quantidade: %d", produto.getNome(), produto.getPreco(), produto.getQuantEstoque());
    }
}
```



```
package org.example.domain;

public class Produto {
    private String nome;
    private double preco;
    private int quantEstoque;

    public Produto(String nome, double preco, int quantEstoque) {
        this.nome = nome;
        this.preco = preco;
        this.quantEstoque = quantEstoque;
    }

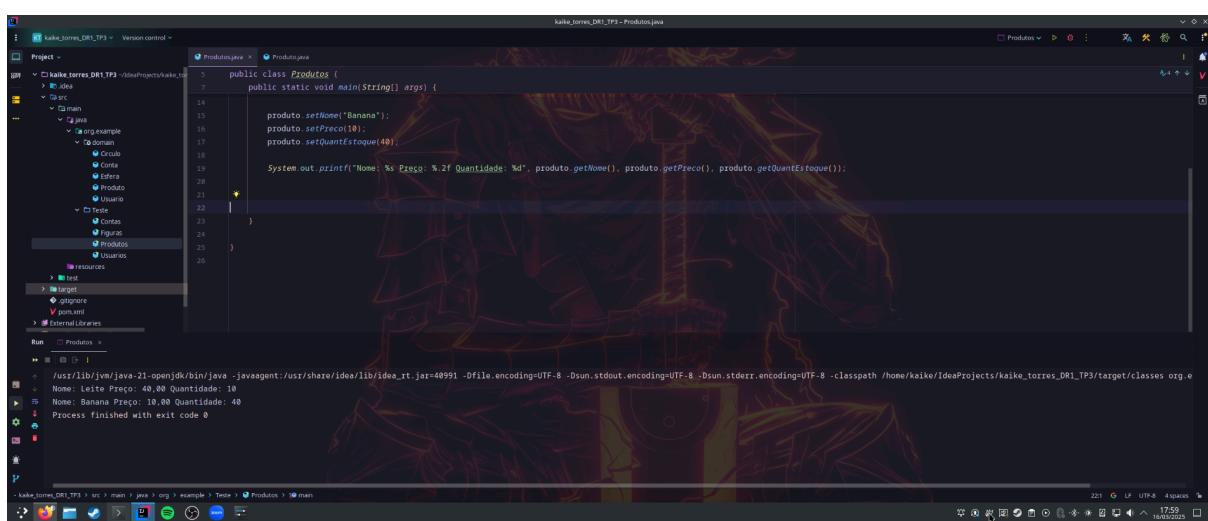
    public void alterarPreco(double preco) {
        this.preco = preco;
    }

    public void alterarQuantidade(int quantEstoque) {
        this.quantEstoque = quantEstoque;
    }

    public void exibirInformacoes() {
        System.out.printf("Nome: %s Preco: %.2f Quantidade: %d\n", nome, preco, quantEstoque);
    }

    public String getNome() {
        return nome;
    }

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



```
public class Produtos {
    public static void main(String[] args) {
        Produto.setNome("Banana");
        Produto.setPreco(10);
        Produto.setQuantEstoque(40);

        System.out.printf("Nome: %s Preco: %.2f Quantidade: %d", Produto.getNome(), Produto.getPreco(), Produto.getQuantEstoque());
    }
}
```

Output of the run command:

```
> /usr/lib/jvm/java-21-openjdk/bin/java -javaagent:/usr/share/idea/lib/idea_rt.jar=40991:file.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /home/kaike_torres/IdeaProjects/kaike_torres_DR1_TP3/target/classes org.example.Teste
Name: Leite Preco: 10.00 Quantidade: 10
Name: Banana Preco: 10.00 Quantidade: 40
Process finished with exit code 0
```

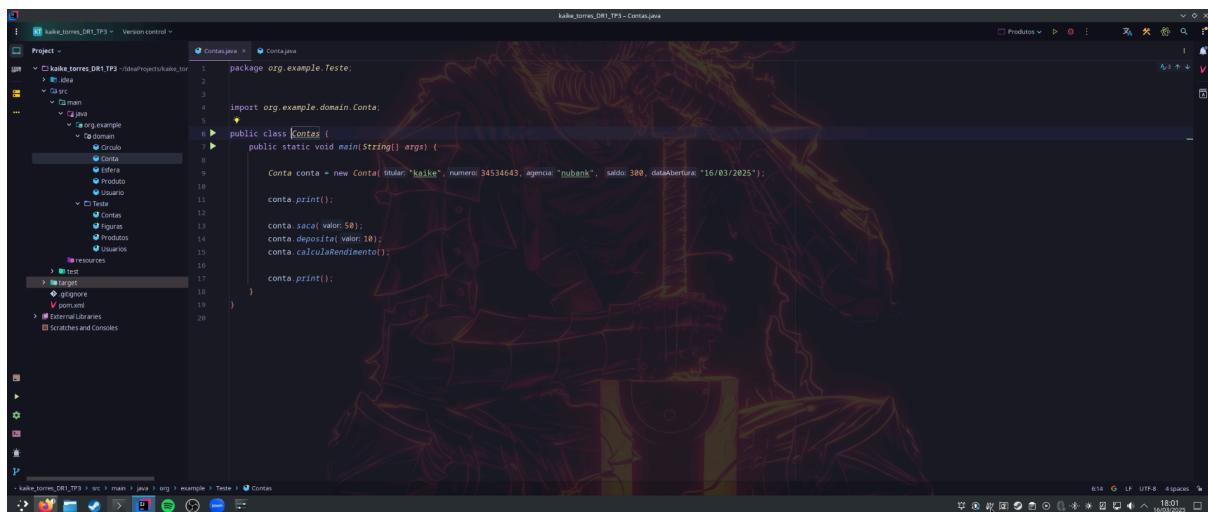
Exercício 7 - Modelando uma Conta Bancária

Este código define um sistema simples para simular operações bancárias, como saque, depósito e cálculo de rendimento. Ele usa duas classes, uma para definir a conta bancária e outra para manipular e exibir as informações dessa conta.

Pacote: org.example.Teste

A classe `Contas` contém o método `main`, onde o objeto `conta` é instanciado e várias operações bancárias são realizadas:

1. Um objeto `conta` é criado com o titular "kaike", o número 34534643, a agência "nubank", saldo inicial de 300 e data de abertura "16/03/2025".
2. O método `print()` é chamado para exibir as informações iniciais da conta, incluindo o saldo e o rendimento calculado.
3. O método `saca(50)` realiza um saque de 50 da conta.
4. O método `deposita(10)` realiza um depósito de 10 na conta.
5. O método `calculaRendimento()` é chamado para calcular o rendimento da conta.
6. O método `print()` é chamado novamente para exibir as informações da conta após as operações de saque e depósito.



```
package org.example.Teste;

import org.example.domain.Conta;

public class Contas {
    public static void main(String[] args) {
        Conta conta = new Conta(titular:"kaike", numero:34534643, agencia:"nubank", saldo:300, dataAbertura:"16/03/2025");

        conta.print();

        conta.saca(valor:50);
        conta.deposita(valor:10);
        conta.calculaRendimento();

        conta.print();
    }
}
```

```

package org.example.domain;

public class Conta {
    private String titular;
    private int numero;
    private String agencia;
    private double saldo;
    private String dataAbertura;

    public Conta(String titular, int numero, String agencia, double saldo, String dataAbertura) {
        this.titular = titular;
        this.numero = numero;
        this.agencia = agencia;
        this.saldo = saldo;
        this.dataAbertura = dataAbertura;
    }

    public void print() {
        System.out.printf("Titular: %s\nNúmero: %d\nAgência: %s\nSaldo: %.2f\nData de abertura: %s\nRendimento: %.2f\n", titular, numero, agencia, saldo, dataAbertura, calculaRendimento());
        System.out.println("-----");
    }

    public void saca(double valor) {
        saldo -= valor;
    }

    public void deposita(double valor) {
        saldo += valor;
    }
}

```

```

package org.example.Teste;

import org.example.domain.Conta;

public class Contas {
    public static void main(String[] args) {
        Conta conta = new Conta(titular: "kaike", numero: 34534643, agencia: "nubank", saldo: 300, dataAbertura: "16/03/2025");
        conta.print();
    }
}

```

Run Output:

```

Titular: kaike
Número: 34534643
Agência: nubank
Saldo: 300.00000
Data de abertura: 16/03/2025
Rendimento: 30.00

Titular: kaike
Número: 34534643
Agência: nubank
Saldo: 260.000000
Data de abertura: 16/03/2025
Rendimento: 26.00

```

Exercício 10 - Definindo Classes para Formas Geométricas

```
package org.example.domain;

public class Circulo { Complexity is 4 everything is cool

    private double rai0; 2 usages

    public Circulo(double rai0) { 1 usage
        this.raio = rai0;
    }

    public double calcularArea(){ Complexity is 3 everything is cool! 1 usage
        return Math.PI * Math.pow(raio, 2);
    }
}
```

```
package org.example.domain;

public class Esfera { Complexity is 4 Everything is cool!

    private double rai0; 2 usages

    public Esfera(double rai0) { 1 usage
        this.raio = rai0;
    }

    public double calcularVolume(){ Complexity is 3 Everything is cool!
        return (4.0 / 3.0) * Math.PI * Math.pow(rai0, 3);
    }
}
```

```
package org.example.Figuras;

import org.example.domain.Circulo;
import org.example.domain.Esfera;

public class Figuras {
    public static void main(String[] args) {
        Circulo circulo = new Circulo(rai0: 3.0);
        Esfera esfera = new Esfera(rai0: 5.0);

        System.out.printf("Área Circulo: %.2f\nVolume Esfera: %.2f", circulo.calcularArea(), esfera.calcularVolume());
    }
}
```

Conclusão

Esses exercícios permitiram consolidar o entendimento sobre os conceitos de POO e a prática de manipulação de dados dentro de classes em Java. A implementação de métodos de alteração de valores, exibição de informações e cálculos demonstrou a aplicação direta desses conceitos, tornando o código mais organizado e fácil de entender. A utilização de getters, setters e construtores também mostrou como simplificar a interação com os atributos de objetos, tornando o gerenciamento de dados mais eficiente.