ânima EDUCAÇÃO

UNISUL
UNIVERSIDADE DO SUL DE SANTA CATARINA

PROGRAMAÇÃO DE SOLUÇÕES COMPUTACIONAIS

Prof. Ricardo Ribeiro Assink

PROGRAMAÇÃO ORIENTADA A OBJETOS

O QUE É ORIENTAÇÃO A OBJETOS?

- Uma forma de Construir Sistemas Computacionais.
- Uma técnica que facilita a reutilização de código entre sistemas.
- Uma técnica que facilita a modularização de um sistema.
- Permite processo de abstração.

PROGRAMAÇÃO ORIENTADA A OBJETOS

O QUE É ORIENTAÇÃO A OBJETOS?

O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados.

PROGRAMAÇÃO ORIENTADA A OBJETOS

PORQUE USAR ORIENTAÇÃO A OBJETOS?

01 **Simplicidade:**
Os objetos escondem a complexidade do código.

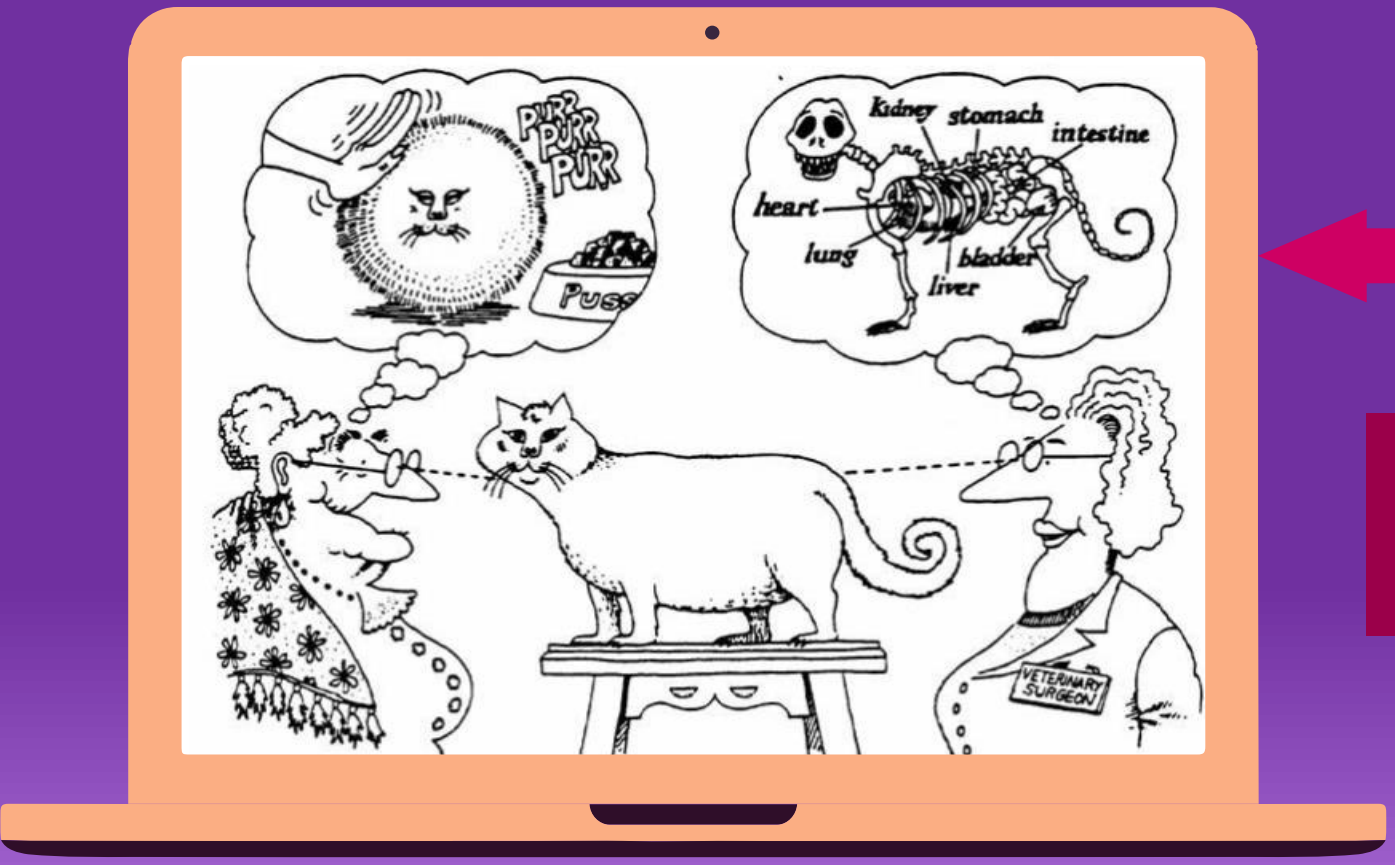
02 **Reutilização de código:**
Objetos podem ser reutilizados por outras aplicações, ter suas funções estendidas e também usados como blocos fundamentais em sistemas mais complexos.

03 **Inclusão Dinâmica:**
Objetos podem ser incluídos dinamicamente no programa, durante a execução. Permite que vários programas compartilhem os mesmos objetos e classes, reduzindo o seu tamanho final.

PROGRAMAÇÃO ORIENTADA A OBJETOS

O QUE É ABSTRAÇÃO?

- Não é um conceito implementável, depende diretamente do contexto e da análise do programador.
- O objetivo é utilizar a abstração para determinar os principais atributos e métodos ao definir uma classe.
- Permite MODELAR características do mundo real que estão diretamente relacionadas ao contexto analisado, agrupando elementos por relevância.



Como funciona a ABSTRAÇÃO.

Abstração foca nas características essenciais de um objeto, relativo à visão do programador.

Descrição de imagem:

Mostra dois pontos de vista sobre o mesmo objeto (Gato), um expectador vê o gato e pensa em fofura, o outro expectador vê o gato e pensa na sua composição biológica como coração, ossos, etc.

CLASSES E OBJETOS

Formalmente uma CLASSE é definida como uma descrição que abstrai um conjunto de OBJETOS com características similares.

É como se fosse uma forma de biscoito.

A forma é a CLASSE e suas "cópias", os biscoitos, são os OBJETOS.

A CLASSE define quais dados e quais comportamentos as suas “cópias” (OBJETOS) devem assumir.

Descrição de imagem:

Mostra uma forma de biscoito representando a classe e vários biscoitos representando os objetos.

COMPONENTES DE UMA CLASSE

ATRIBUTOS

O conjunto de propriedades de classe. Para cada propriedade, especifica-se:

NOME:
Um identificador para o atributo

TIPO:
O tipo de atributo (Inteiro, real, texto, etc.)

VALOR_DEFAULT:
Opcionalmente, pode-se especificar um valor inicial para o atributo.

VISIBILIDADE:
Opcionalmente, pode-se especificar o quão acessível é um atributo de um objeto a partir de outros objetos.

VALORES POSSÍVEIS SÃO:

- **(privativo)**, nenhuma visibilidade externa.
+ **(público)**, visibilidade externa total.
 # (protegido), visibilidade externa limitada.

COMPONENTES DE UMA CLASSE

MÉTODOS

O conjunto de funcionalidades da classe, ou seja, seu comportamento.
Cada método é composto de:

NOME:
Um identificador para o método

TIPO:
Quando o método tem um valor de retorno, o tipo desse valor

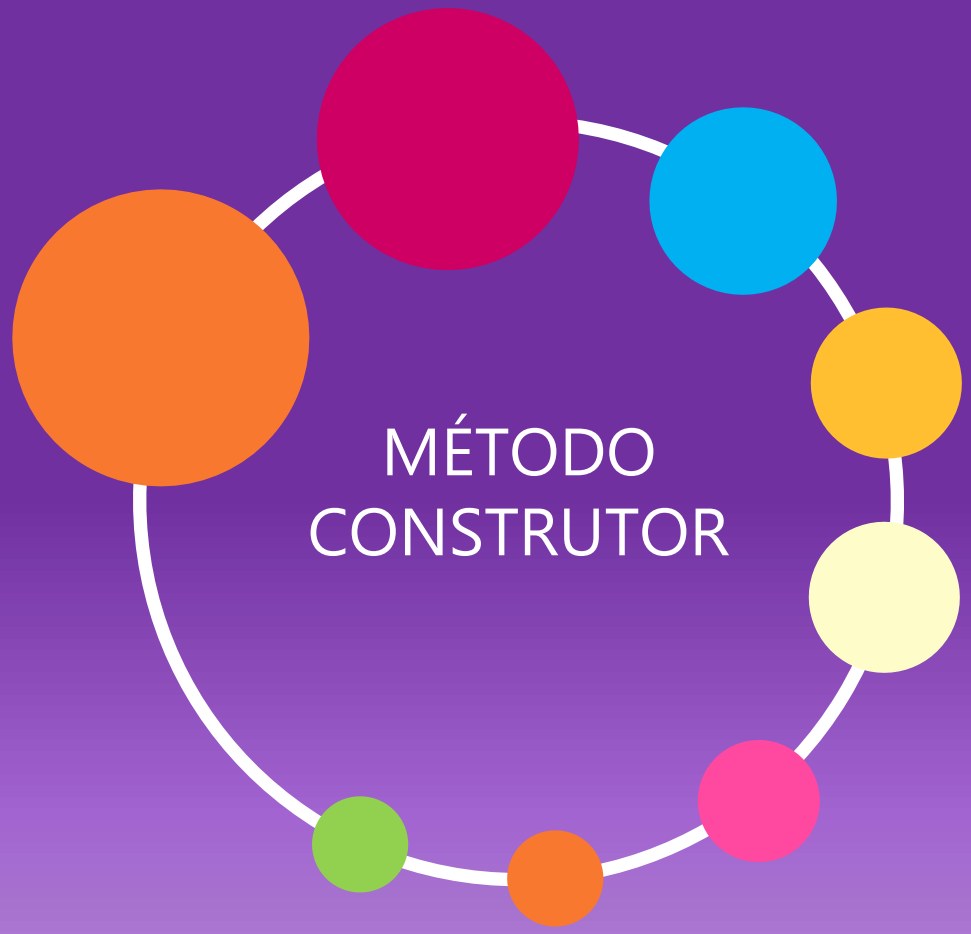
LISTA DE ARGUMENTOS:
Quando o método recebe parâmetros para sua execução, o tipo e um identificador para cada parâmetro

VISIBILIDADE:
Como para atributos, define o quão visível é um método a partir de objetos de outras classes

VALORES POSSÍVEIS SÃO:

- **(privativo)**, nenhuma visibilidade externa.
+ **(público)**, visibilidade externa total.
 # (protegido), visibilidade externa limitada.

COMPONENTES DE UMA CLASSE



Em Java, o construtor é definido como um método cujo nome deve ser o mesmo nome da classe e sem indicação do tipo de retorno (nem mesmo void).

O construtor é unicamente invocado no momento da **criação do objeto** através do operador new.

O retorno do operador new é uma referência para o **OBJETO** recém-criado.

Com os métodos construtores é possível criar um objeto vazio ou criar um objeto já informando os valores.

CÓDIGO COMENTADO – CLASSE PRODUTO

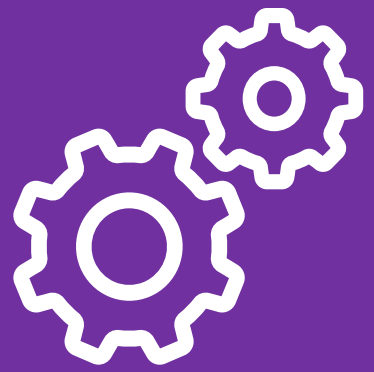
Acompanhe o professor para explicações mais detalhadas. Observe o funcionamento e uso dos itens estudados anteriormente.

```java
public class Produto {

    // Atributos
    private int id;
    private String nome;
    private String descricao;
    private double preco;
    private int estoque;

    // Método Construtor de Objeto Vazio
    public Produto() {
    }

    // Método Construtor de Objeto, inserindo dados
    public Produto(int id, String nome, String descricao, double preco, int estoque) {
        this.id        = id;
        this.nome      = nome;
        this.descricao = descricao;
        this.preco     = preco;
        this.estoque   = estoque;
    }
```

```java
    // Métodos GET e SET
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNome() {
        return nome;
    }

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

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public double getPreco() {
        return preco;
    }

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

    public int getEstoque() {
        return estoque;
    }

    public void setEstoque(int estoque) {
        this.estoque = estoque;
    }

}
```

Descrição de imagem:
Mostra o conteúdo da classe Produto, consulte ANEXO 1 no final deste documento, para acesso total ao código fonte em JAVA.

USO DE OBJETOS



OK!!!!!!
Mas agora que já construímos uma classe básica chamada Produto, como eu crio e utilizo OBJETOS dessa CLASSE?

Para isso vamos criar uma classe principal de exemplo.

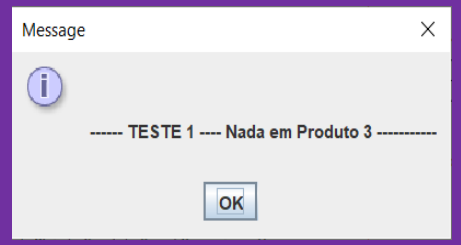
Lá, vamos INSTANCIAR (criar) objetos da classe Produto.
Vamos executar uma rotina que demonstra a manipulação dos objetos da classe Produto.

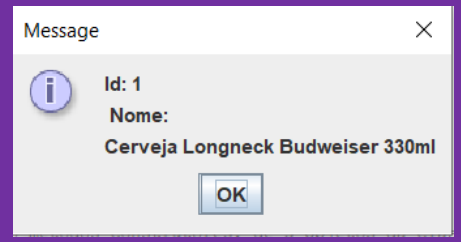
Sacou agora o que significa Orientação à Objetos ? :P

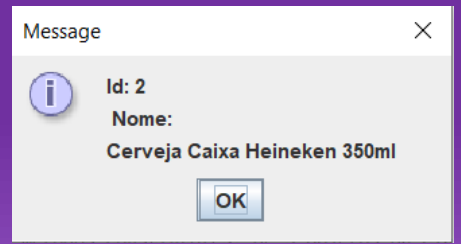
CÓDIGO COMENTADO – CLASSE PRINCIPAL

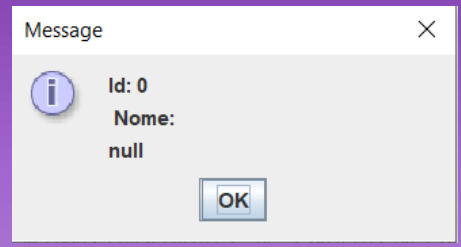
Acompanhe o professor para explicações mais detalhadas sobre a utilização prática de objetos.

```java
// Inclui as definições de outras classes
import Model.Produto;
import javax.swing.JOptionPane;

// Define nome da Classe
public class Principal {

    // Método Principal, este determina a ORDEM das ações.
    public static void main(String[] args) {

        // Instanciação, usando métodos construtores, de 3 Objetos da Classe Produto do Pacote Model
        Produto ObjetoProduto1 = new Produto(1,"Cerveja Longneck Budweiser 330ml","Fardo com 6 unidades",25.0,60);
        Produto ObjetoProduto2 = new Produto(2,"Cerveja Caixa Heineken 350ml","Caixa com 12 unidades",30.0,40);
        // Instanciação de produto vazio
        Produto ObjetoProduto3 = new Produto();

        // Uso dos métodos GET
        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 1 ---- Nada em Produto 3 ----------\n\n");
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto1.getId()+ "\n Nome: \n" + ObjetoProduto1.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto2.getId()+ "\n Nome: \n" + ObjetoProduto2.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto3.getId()+ "\n Nome: \n" + ObjetoProduto3.getNome());

        // Uso dos métodos SET
        ObjetoProduto3.setId(3);
        ObjetoProduto3.setNome("Refrigerante Pureza 2 litros");
        ObjetoProduto3.setDescricao("1 unidade");
        ObjetoProduto3.setPreco(4.0);
        ObjetoProduto3.setEstoque(100);
```







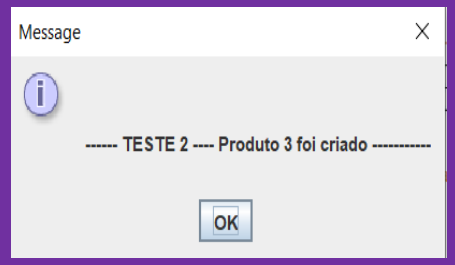


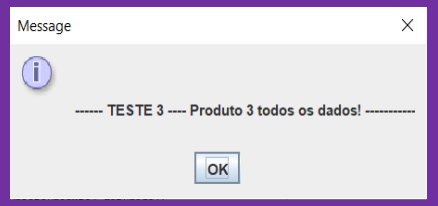
Descrição de imagem:
Mostra o conteúdo da classe Principal, consulte ANEXO 2 no final deste documento, para acesso total ao código fonte em JAVA.

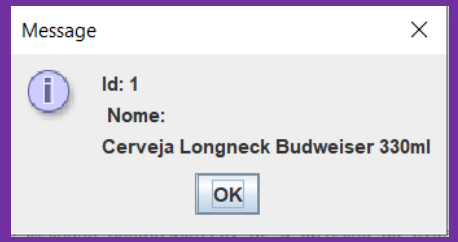
CÓDIGO COMENTADO – CLASSE PRINCIPAL CONTINUAÇÃO

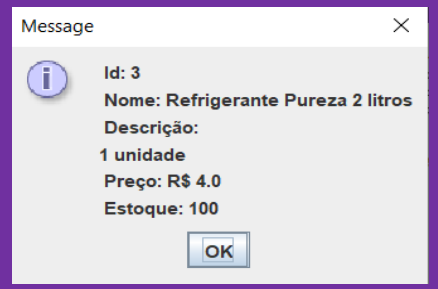
Acompanhe o professor para explicações mais detalhadas sobre a utilização prática de objetos.

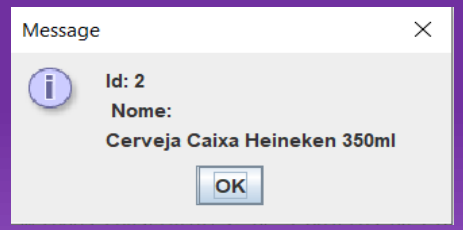
```
        // Teste dos métodos SET e GET
        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 2 ---- Produto 3 foi criado ----------\n\n");
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto1.getId()+ "\n Nome: \n" + ObjetoProduto1.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto2.getId()+ "\n Nome: \n" + ObjetoProduto2.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto3.getId()+ "\n Nome: \n" + ObjetoProduto3.getNome());

        // Exemplo todos os dados
        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 3 ---- Produto 3 todos os dados! ----------\n\n");
        JOptionPane.showMessageDialog(null,
                  " Id: "+ ObjetoProduto3.getId()
                + "\n Nome: " + ObjetoProduto3.getNome()
                + "\n Descrição: \n" + ObjetoProduto3.getDescricao()
                + "\n Preço: R$ " + ObjetoProduto3.getPreco()
                + "\n Estoque: " + ObjetoProduto3.getEstoque()
                );

        // modificando o preço de Produto 3 de R$ 4 reais para R$ 5 reais
        ObjetoProduto3.setPreco(5.0);

        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 4 ---- Produto 3 alteração de preço! ----------\n\n");
                JOptionPane.showMessageDialog(null,
                  " Id: "+ ObjetoProduto3.getId()
                + "\n Nome: " + ObjetoProduto3.getNome()
                + "\n Descrição: \n" + ObjetoProduto3.getDescricao()
                + "\n Preço: R$ " + ObjetoProduto3.getPreco()
                + "\n Estoque: " + ObjetoProduto3.getEstoque()
                );

    }

}
```

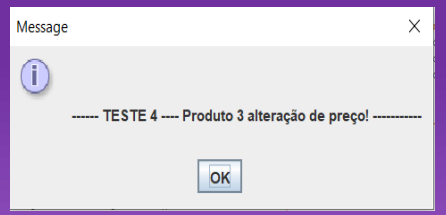


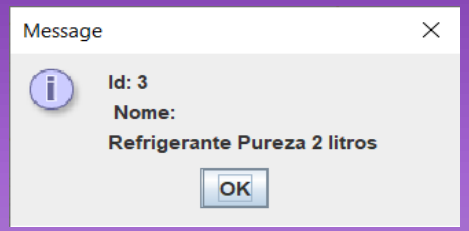


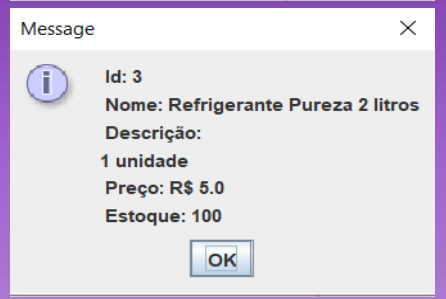












Descrição de imagem:

Mostra o conteúdo da classe Principal, consulte ANEXO 2 no final deste documento, para acesso total ao código fonte em JAVA.

CAMADAS E O PADRÃO MVC

MODEL, VIEW, CONTROLLER

01 Uma aplicação que usa POO normalmente gera várias classes e outros arquivos.

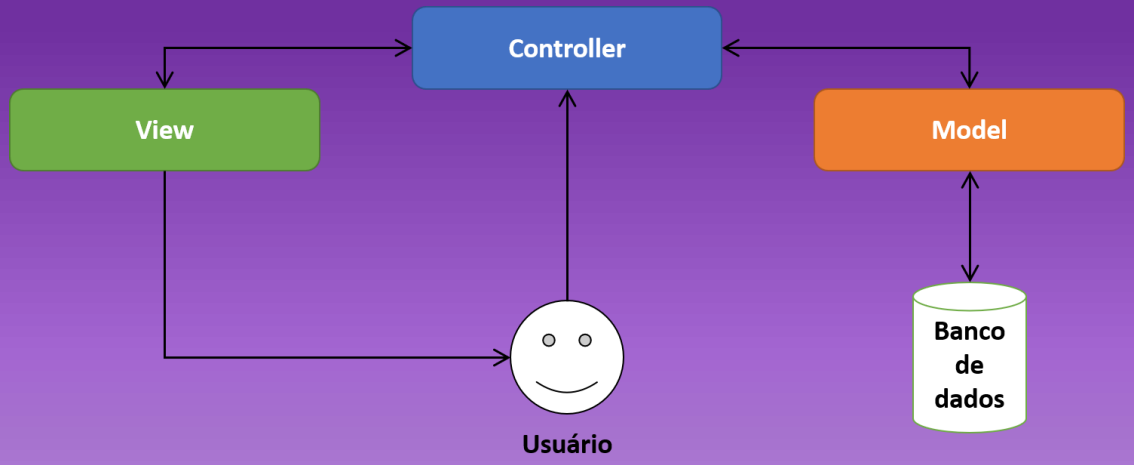
02 O ideal é utilizar algum padrão para organizar tudo.

03 Dividir o software em CAMADAS é uma solução.

CAMADAS E O PADRÃO MVC

MODEL, VIEW, CONTROLLER

MVC é o acrônimo de Model-View-Controller (em português: Arquitetura Modelo-Visão-Controle - MVC) é um padrão de projeto de software focado no reuso de código e a separação de conceitos em três camadas interconectadas, onde a apresentação dos dados e interação dos usuários (front-end) são separados dos métodos que interagem com o banco de dados (back-end).



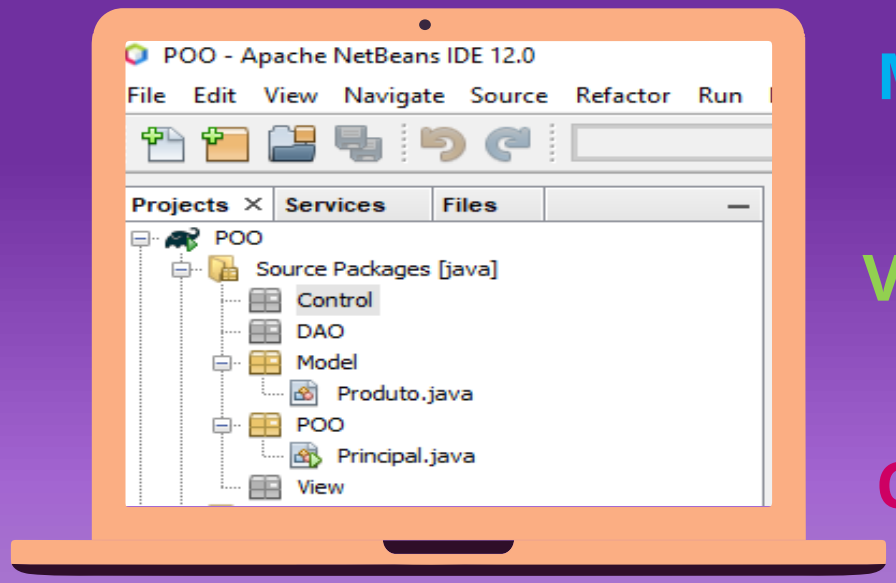
Descrição de imagem:
Apenas ilustra a ligação das camadas View, Controller, Model, Banco de Dados e usuário.

CAMADAS E O PADRÃO MVC

MODEL, VIEW, CONTROLLER

Então vamos agora colocar os arquivos no seu devido lugar ?

Vamos criar pacotes que representam as camadas e colocar os arquivos no local adequado.



MODEL As classes relacionadas aos modelos de dados e regras de negócio. Comunica-se com o DAO (Data access object) que veremos mais à frente com o uso dos bancos de dados.

VIEW As classes relacionadas as interfaces com o usuário final, tudo que o usuário vê e interage.

CONTROLLER As classes relacionadas ao PROCESSAMENTO do software e transporte de dados entre os objetos.

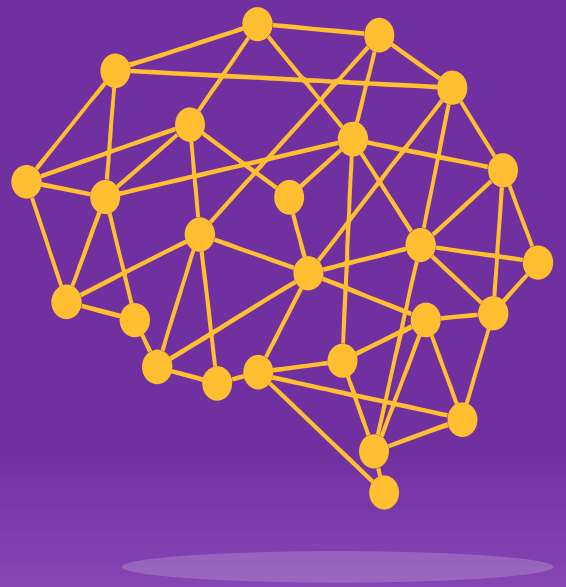
Descrição de imagem:
Mostra a árvore de diretórios que organiza os arquivos de acordo com sua função.

Busca Ativa!

1. Procure e assista vídeos na internet que falem de Orientação à Objetos e MVC.

2. Implemente e execute TODOS os exemplos da aula de hoje.

3. Adicione um atributo a classe Produto com seus métodos GET e SET e na classe Principal inclua linhas manipulando este novo dado.



FIM

ANEXO 1 – CLASSE PRODUTO

```
public class Produto {

    // Atributos
    private int id;
    private String nome;
    private String descricao;
    private double preco;
    private int estoque;

    // Método Construtor de Objeto Vazio
    public Produto() {
    }

    // Método Construtor de Objeto, inserindo dados
    public Produto(int id, String nome, String descricao, double preco, int estoque) {
        this.id        = id;
        this.nome      = nome;
        this.descricao = descricao;
        this.preco     = preco;
        this.estoque   = estoque;
    }

// continua na coluna ao lado.
```

```
// Métodos GET e SET
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getDescricao() {
        return descricao;
    }
    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
    public double getPreco() {
        return preco;
    }
    public void setPreco(double preco) {
        this.preco = preco;
    }
    public int getEstoque() {
        return estoque;
    }
    public void setEstoque(int estoque) {
        this.estoque = estoque;
    }

}
```

ANEXO 2 – CLASSE PRINCIPAL

```
// Inclui as definições de outras classes
import Model.Produto;
import javax.swing.JOptionPane;

// Define nome da Classe
public class Principal {

    // Método Principal, este determina a ORDEM das ações.
    public static void main(String[] args) {

        // Instanciação, usando métodos construtores, de 3 Objetos da Classe Produto do Pacote Model
        Produto ObjetoProduto1 = new Produto(1,"Cerveja Longneck Budweiser 330ml","Fardo com 6 unidades",25.0,60);
        Produto ObjetoProduto2 = new Produto(2,"Cerveja Caixa Heineken 350ml","Caixa com 12 unidades",30.0,40);
        // Instanciação de produto vazio
        Produto ObjetoProduto3 = new Produto();

        // Uso dos métodos GET
        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 1 ---- Nada em Produto 3 ----------\n\n");
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto1.getId()+ "\n Nome: \n" + ObjetoProduto1.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto2.getId()+ "\n Nome: \n" + ObjetoProduto2.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto3.getId()+ "\n Nome: \n" + ObjetoProduto3.getNome());

        // Uso dos métodos SET
        ObjetoProduto3.setId(3);
        ObjetoProduto3.setNome("Refrigerante Pureza 2 litros");
        ObjetoProduto3.setDescricao("1 unidade");
        ObjetoProduto3.setPreco(4.0);
        ObjetoProduto3.setEstoque(100);
```

ANEXO 2 – CLASSE PRINCIPAL
CONTINUAÇÃO

```
// Teste dos métodos SET e GET
        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 2 ---- Produto 3 foi criado ----------\n\n");
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto1.getId()+ "\n Nome: \n" + ObjetoProduto1.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto2.getId()+ "\n Nome: \n" + ObjetoProduto2.getNome());
        JOptionPane.showMessageDialog(null, "Id: "+ ObjetoProduto3.getId()+ "\n Nome: \n" + ObjetoProduto3.getNome());

        // Exemplo todos os dados
        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 3 ---- Produto 3 todos os dados! ----------\n\n");
        JOptionPane.showMessageDialog(null,
                  " Id: "+ ObjetoProduto3.getId()
                + "\n Nome: " + ObjetoProduto3.getNome()
                + "\n Descrição: \n" + ObjetoProduto3.getDescricao()
                + "\n Preço: R$ " + ObjetoProduto3.getPreco()
                + "\n Estoque: " + ObjetoProduto3.getEstoque()
                );

        // modificando o preço de Produto 3 de R$ 4 reais para R$ 5 reais
        ObjetoProduto3.setPreco(5.0);

        JOptionPane.showMessageDialog(null, "\n\n------ TESTE 4 ---- Produto 3 alteração de preço! ----------\n\n");
                JOptionPane.showMessageDialog(null,
                  " Id: "+ ObjetoProduto3.getId()
                + "\n Nome: " + ObjetoProduto3.getNome()
                + "\n Descrição: \n" + ObjetoProduto3.getDescricao()
                + "\n Preço: R$ " + ObjetoProduto3.getPreco()
                + "\n Estoque: " + ObjetoProduto3.getEstoque()
                );

    }

}
```