UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL

AULA 13

LINGUAGEM DE PROGRAMAÇÃO 2
JAVA

PROF. JANIHERYSON FELIPE

CONTEÚDO DESSA AULA

- CONHECER OS CONCEITOS DE ACOPLAMENTO E COESÃO DE CLASSES E MÉTODOS;
- CONHECER OS DESIGN DE CLASSES;
- CONHECER OS CONCEITOS DE PADRÕES DE PROJETO;
- CONHECER AS PRINCIPAIS BIBLIOTECAS GRÁFICA DO JAVA
- DISCUSSÕES E DÚVIDAS GERAIS.

PROJETO DE SOFTWARE

- Todo projeto de software bem sucedido deve seguir um conjunto de passos, de forma que os eventuais erros/prejuízos sejam evitados. A ideia central consiste em:
  - Entender as necessidades do público alvo do software;
  - Entender os requisitos funcionais do software;
  - Entender os custos envolvidos (financeiros e humanos) e as dependências que seu projeto terá .

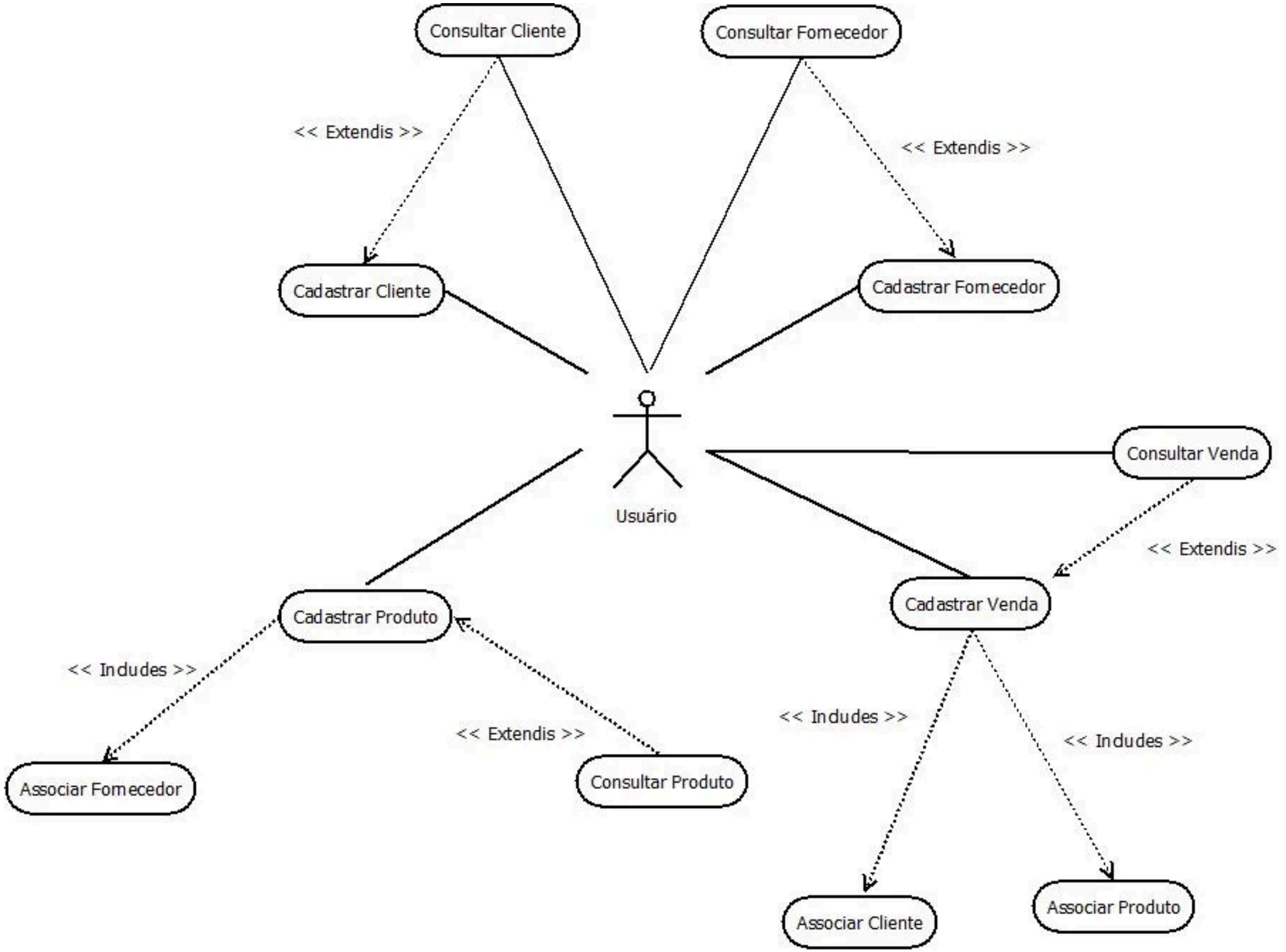
PROJETO DE SOFTWARE

- Quem vai usar o meu sistema e quais os papeis que esses agentes terão?
- Quais funcionalidades o meu sistema terá para atender a esses agentes?
- Quais classes o sistema deve ter para atender a todos essas demandas de funcionalidades?
- Quais métodos são necessários em cada classes?

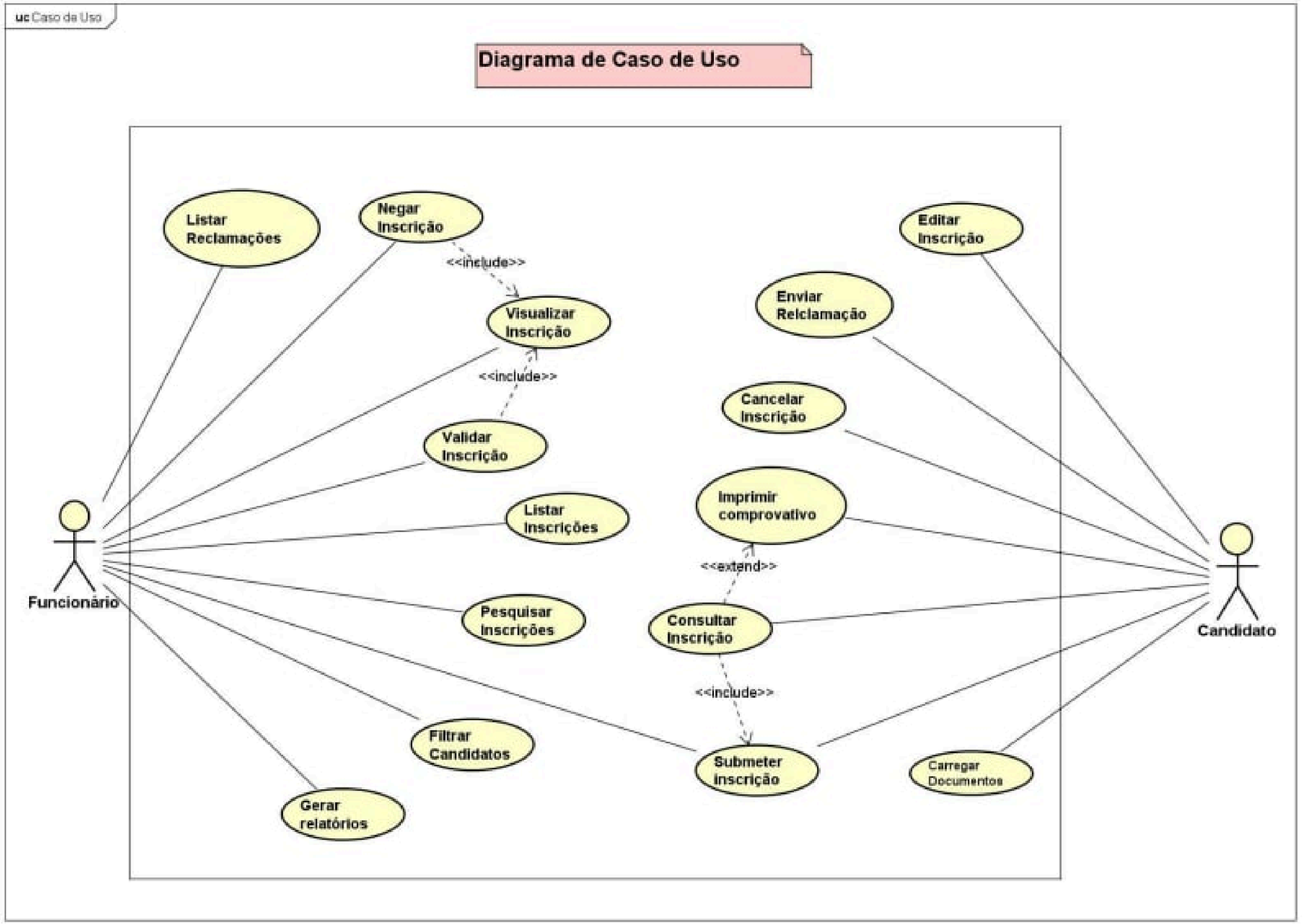
PROJETO DE SOFTWARE

- O software é claro e de facil utilização pelo usuario final (Usabilidade)?
- O software é bem estruturado e documentado para facilitar a manutenção e atualizações futuras?
- O software é compatível com diferentes plataformas e dispositivos, garantindo uma experiência consistente para todos os usuários?

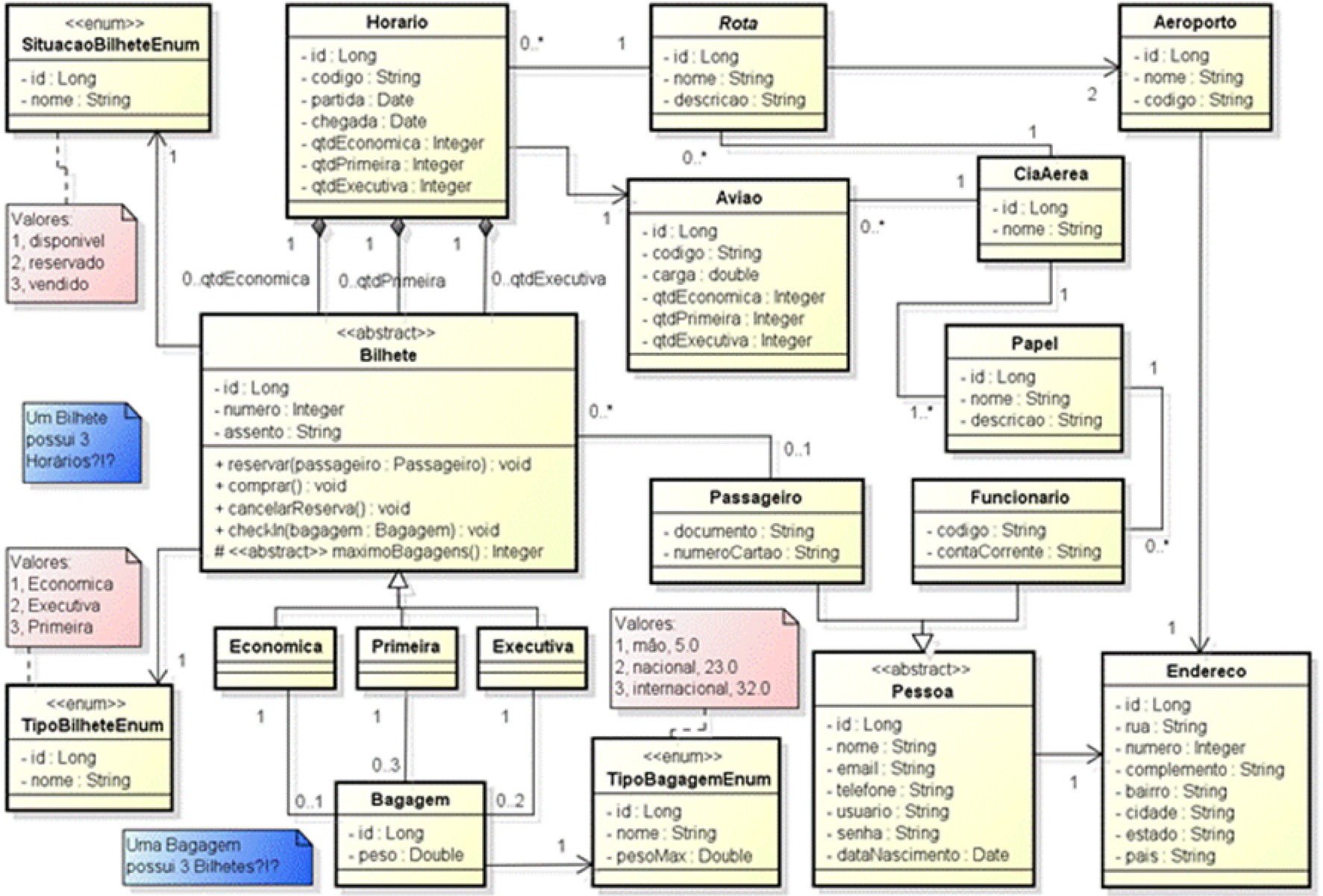
DIAGRAMAS UML



DIAGRAMAS UML

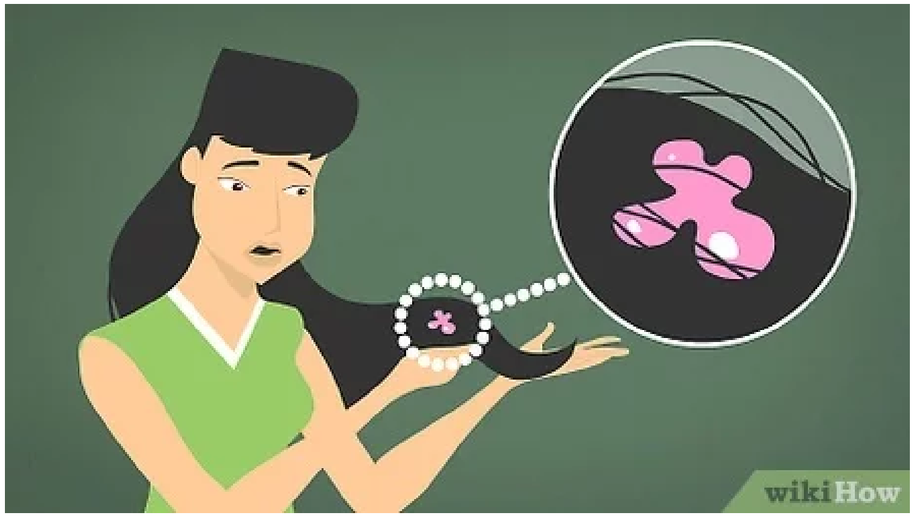


DIAGRAMAS UML



ACOPLAMENTO DE CLASSES

- O conceito de acoplamento forte e fraco de classes refere-se à interdependência entre as classes em um sistema de software. Podemos ter dois tipos de acoplamentos:
  - Acoplamento forte
  - Acoplamento fraco



ACOPLAMENTO DE CLASSES

- **O acoplamento forte:** ocorre quando duas classes estão fortemente interligadas, o que significa que uma classe depende muito da outra para funcionar corretamente.
- **O acoplamento fraco:** ocorre quando as classes estão menos interligadas, o que resulta em uma dependência mais flexível e fácil de modificar.

ACOPLAMENTO DE CLASSES

- Com forte acoplamento, temos os seguintes problemas:
  - Mudanças em uma classe acarretam mudanças em outras classes.
  - A classe é mais difícil de entender isoladamente.
  - A classe é mais difícil de ser reusada, já que depende da presença de outras classes.

ACOPLAMENTO DE CLASSES

```java
public class Produto {
    private String nome;
    private double preco;
    private BancoDeDados banco = new BancoDeDados();

    public void salvar() {
        banco.salvarProduto(this);
    }
}

public class Pedido {
    private List<Produto> produtos = new ArrayList<>();
    private BancoDeDados banco = new BancoDeDados();

    public void finalizar() {
        for (Produto produto : produtos) {
            produto.salvar();
        }
        banco.salvarPedido(this);
    }
}
```

- Dependência Direta: As classes Produto e Pedido possuem uma instância direta da classe BancoDeDados. Isso significa que qualquer alteração na classe BancoDeDados impactará diretamente as classes Produto e Pedido.

- Violação de Princípios: Esse exemplo viola o princípio de inversão de dependência (DIP), pois as classes de alto nível (Produto e Pedido) dependem de detalhes de baixo nível (BancoDeDados).

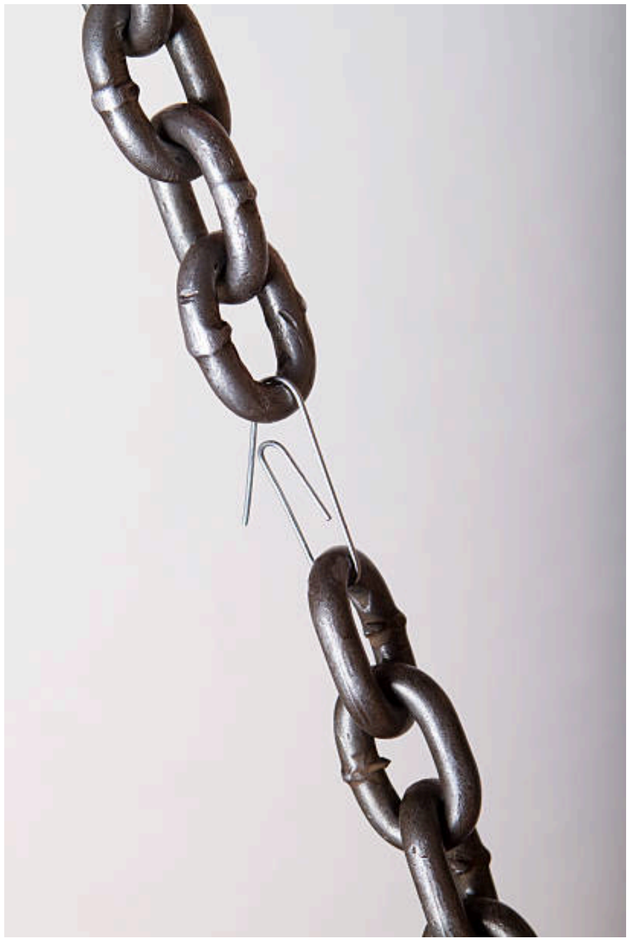
COESÃO DE CLASSES

- A coesão refere-se ao grau em que os elementos (como métodos e classes) de um módulo ou componente estão **relacionados** e trabalham juntos para alcançar um objetivo comum.

- A coesão em métodos refere-se à ideia de que um método deve realizar uma **única tarefa bem definida**. Métodos coesos são mais fáceis de entender, manter e reutilizar.

COESÃO DE CLASSES

- **Uma alta coesão** indica que os elementos estão fortemente relacionados e contribuem para uma única responsabilidade;
- **Uma baixa coesão** sugere que os elementos podem estar realizando múltiplas tarefas ou não estão diretamente relacionados.

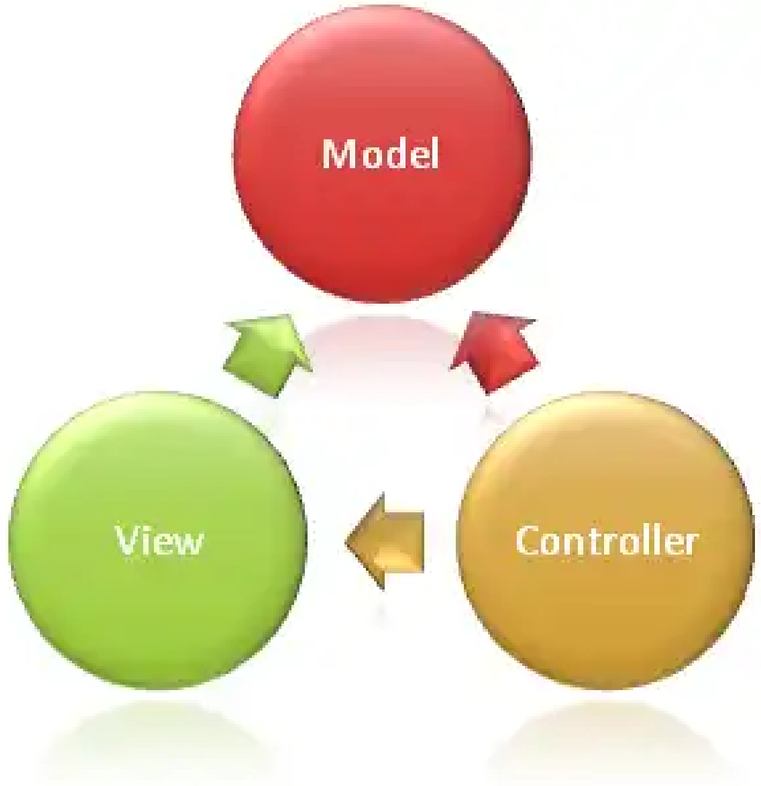


COESÃO DE CLASSES

- Baixa coesão
  - Difícil de entender;
  - Difícil de reusar;
  - Difícil de manter;
  - “Delicada": constantemente sendo afetada por outras mudanças
- Alta Coesão - possibilita:
  - Entender o que uma classe ou método faz.
  - Usar nomes descritivos.
  - Reusar classes ou métodos.

PADRAO DE PROJETO: MVC

- O padrão MVC (Model-View-Controller) é uma arquitetura de software comummente usada no desenvolvimento de aplicativos, especialmente em frameworks de desenvolvimento web. Ele separa os componentes de uma aplicação em três partes distintas:
  - Modelo (Model);
  - Visão (View);
  - Controlador (Controller).



PADRAO DE PROJETO: MVC

- **Modelo (Model):** O modelo representa os dados da aplicação e a lógica de negócios associada a esses dados. Ele é responsável por armazenar e gerenciar o estado da aplicação, bem como por realizar operações como validação, cálculos e interações com o banco de dados. O modelo não tem conhecimento da interface do usuário.

PADRAO DE PROJETO: MVC

- **Visão (View):** A visão é responsável por apresentar os dados ao usuário de forma visualmente atraente e compreensível. Ela exibe a interface do usuário e interage com o modelo para recuperar os dados a serem exibidos. A visão não contém lógica de negócios e geralmente é passiva, respondendo apenas a eventos gerados pelo usuário.

PADRAO DE PROJETO: MVC

- **Controlador (Controller):** O controlador atua como intermediário entre o modelo e a visão. Ele recebe entradas do usuário por meio da interface do usuário, processa essas entradas e atualiza o estado do modelo conforme necessário. Ele também atualiza a visão para refletir as mudanças no modelo. O controlador contém a lógica de aplicação e coordena as interações entre o modelo e a visão.

INTERFACES GRAFICAS NO JAVA

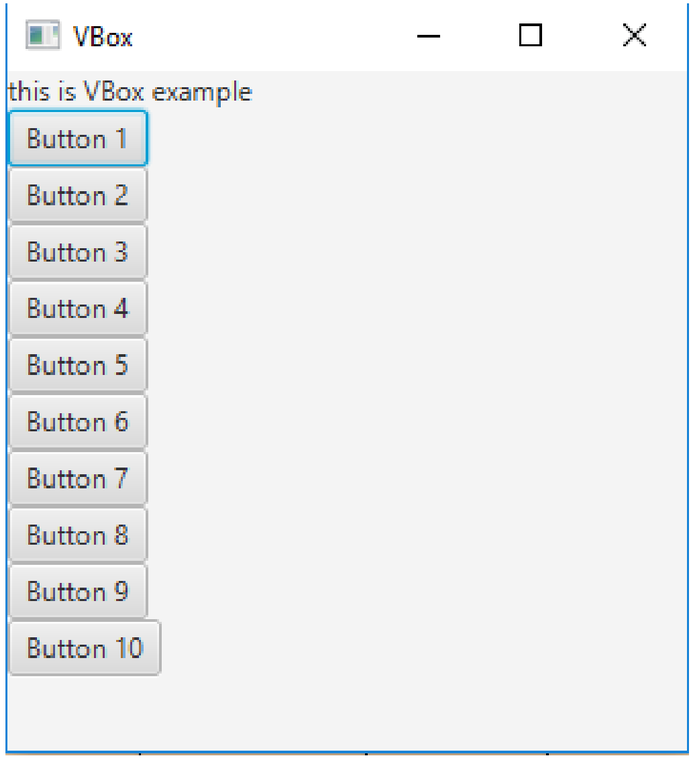
- A construção de interfaces graficas no java, por ser feito a partir de duas bibliotecas graficas:
  - **Swing**: é mais antiga e está presente desde as versões mais antigas do Java,
  - **JavaFX**: É uma biblioteca mais moderna, introduzida no Java 8.

INTERFACES GRAFICAS NO JAVA

- Ambas as bibliotecas oferecem uma variedade de componentes gráficos, como **botões**, **caixas de texto**, **listas**, **tabelas**, entre outros, que podem ser utilizados para construir interfaces de usuário - Componentes de Interface Gráfica (GUI components).

INTERFACES GRAFICAS NO JAVA

- As GUIs em Java normalmente usam gerenciadores de layout (layout managers) para organizar os componentes na tela de forma flexível e responsiva. Alguns exemplos de gerenciadores de layout incluem BorderLayout, FlowLayout, GridLayout e GroupLayout.



INTERFACES GRAFICAS NO JAVA

- Em Java, as interações do usuário com a interface gráfica são tratadas por meio de **eventos**. Você precisará entender como lidar com eventos de **mouse**, **teclado** e outros eventos de usuário para tornar sua interface interativa e responsiva.

JAVAFX

- Foi introduzido pela primeira vez pela Sun Microsystems (posteriormente adquirida pela Oracle) como uma alternativa moderna ao Swing. Ele foi projetado para oferecer uma experiência de desenvolvimento mais rica e produtiva para interfaces gráficas em comparação com o Swing.

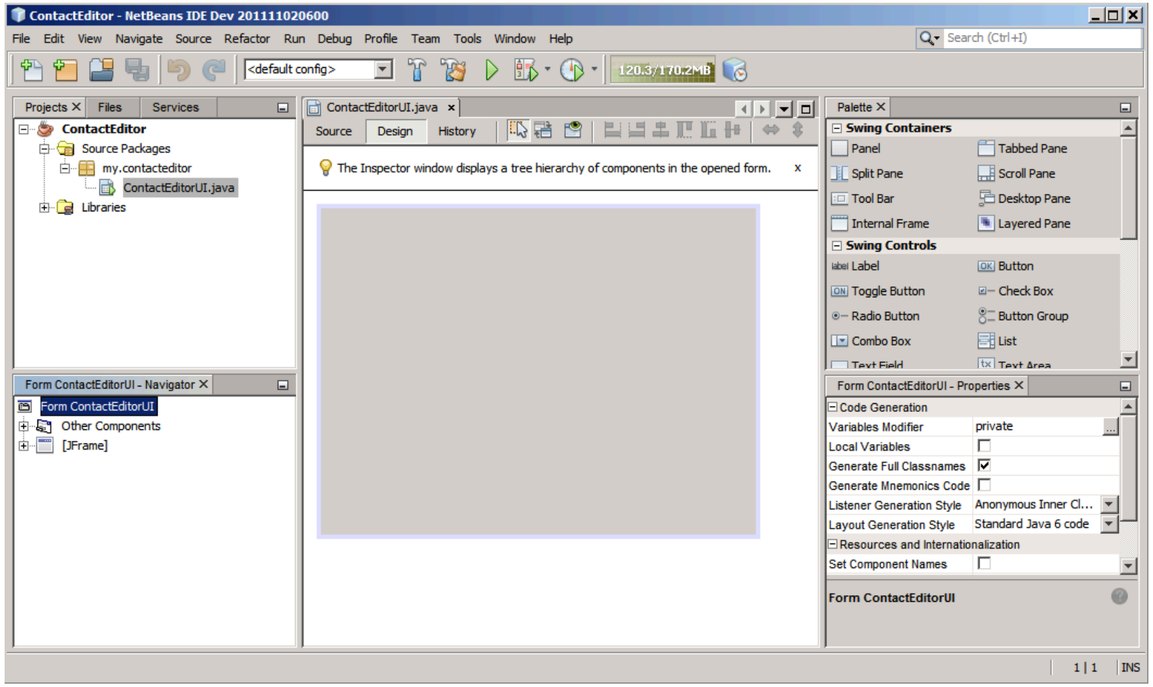
JAVAFX

- JavaFX introduziu o conceito de **FXML**, uma linguagem de marcação **XML** para definir interfaces de usuário de forma declarativa. O FXML permite separar a estrutura da interface do código Java, facilitando a manutenção e o design da interface. Além disso, o **Scene Builder** é uma ferramenta visual que permite projetar interfaces JavaFX de forma intuitiva, arrastando e soltando componentes.

INTERFACE GRAFICA EM JAVA

NetBeans GUI Builder (Matisse):

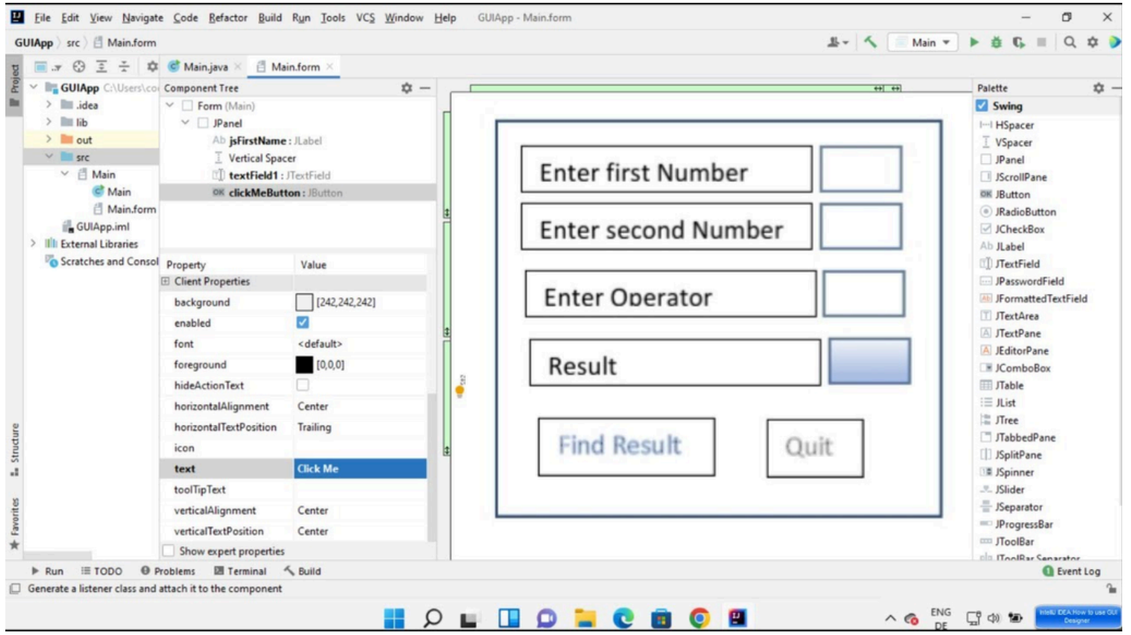
- NetBeans oferece uma ferramenta de construção de interfaces gráficas chamada Matisse, que permite o design visual de interfaces usando o Swing. Matisse tem uma interface de arrastar e soltar



INTERFACE GRAFICA EM JAVA

IntelliJ IDEA GUI Designer

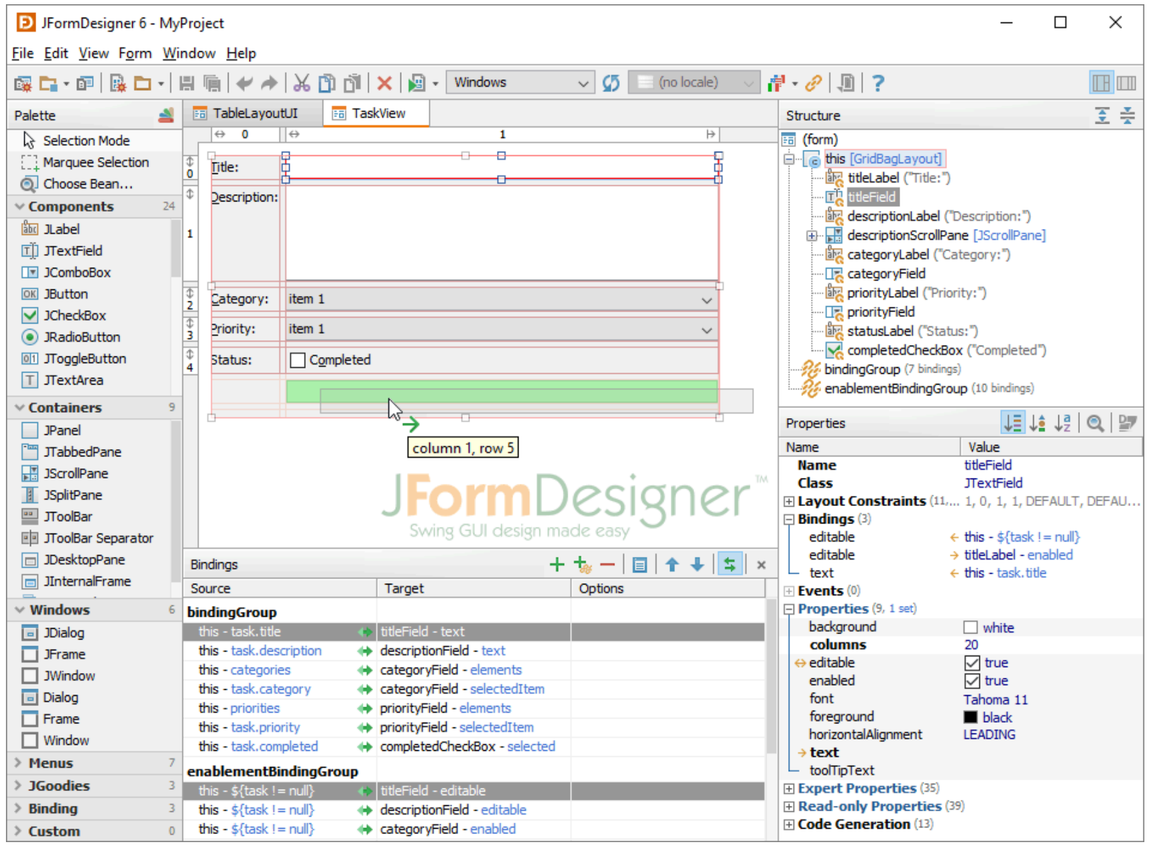
- IntelliJ IDEA possui um designer de GUI incorporado para o Swing. Ele permite a criação de interfaces gráficas visualmente, com ferramentas de arrastar e soltar, além de gerar o código necessário para o layout.



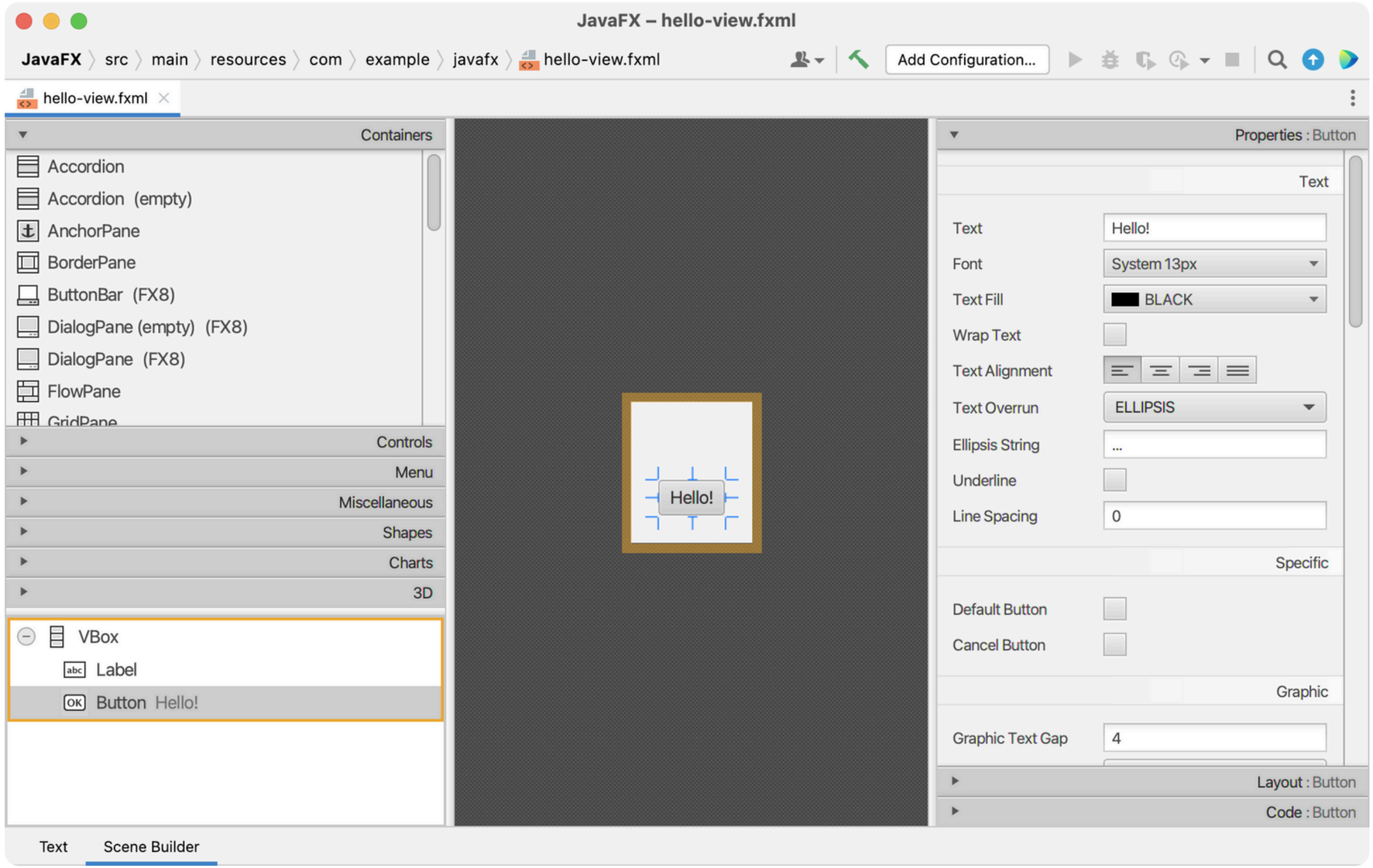
INTERFACE GRAFICA EM JAVA

JFormDesigner

- JFormDesigner é uma ferramenta popular para criar interfaces gráficas com Swing e JavaFX. Ela oferece uma interface visual para o design e é compatível com várias versões do Java.



SCENE BUILDER



INTEGRAÇÃO DO SCENE BUILDER NO VSCODE