

*Desenvolvimento Rápido de Aplicações*

# *JavaFX MVC*

Profa. Joyce Miranda

Material de referência:

<https://www.youtube.com/watch?v=OPNiAZ3PjpM>

# JavaFX MVC

---

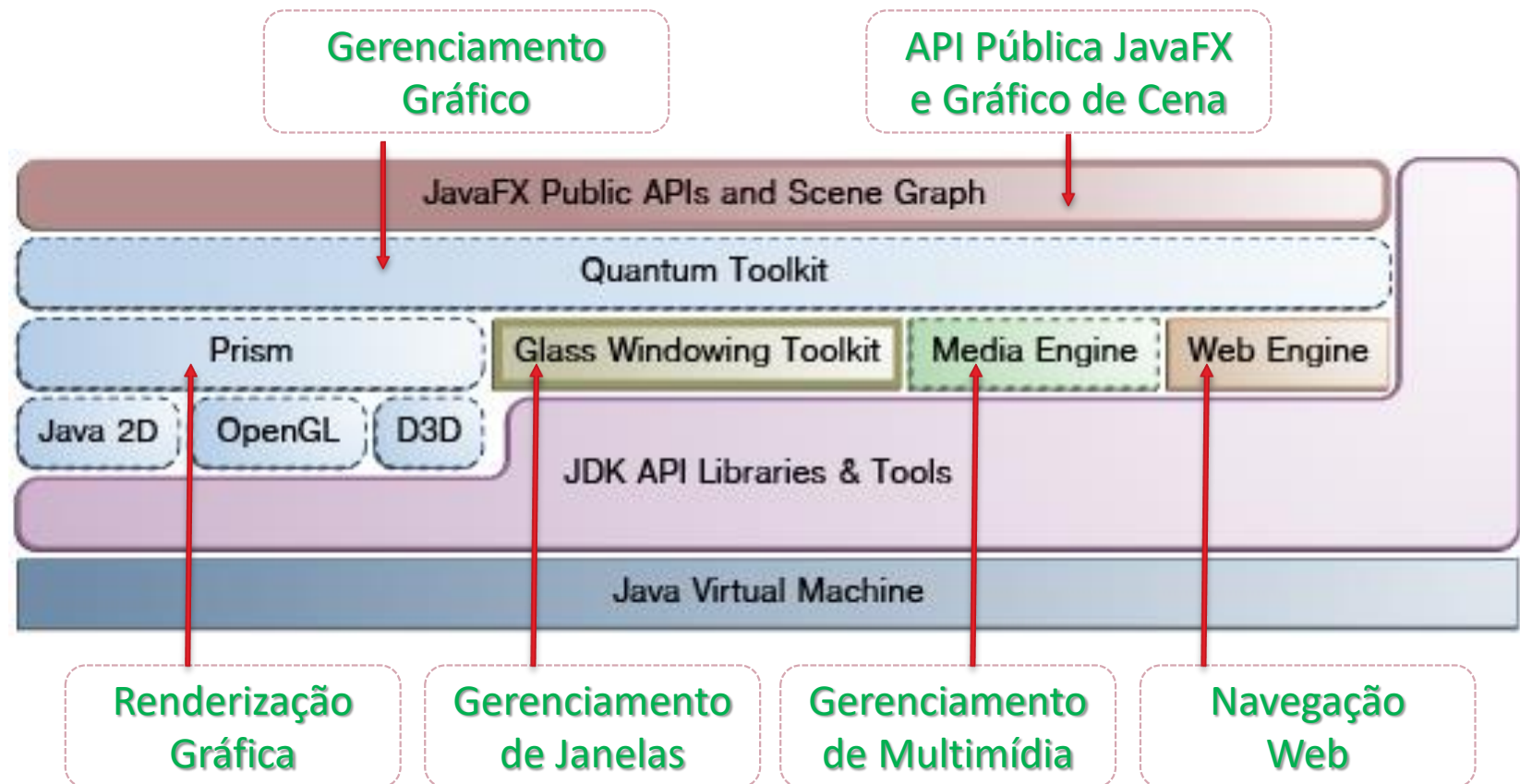
## ► Overview

- API gráfica do JAVA para criação de Aplicações Ricas
  - Aplicações com características e facilidades Desktop que podem ser executadas em diferentes plataformas.
- Características
  - FXML: linguagem declarativa usada para criação da interface.
  - Motor multimídia para renderizar modernas interfaces gráficas;
  - Webkit : componente capaz de embutir páginas HTML, interagir com códigos CSS e JavaScript;
  - Diversidade de componentes gráficos.



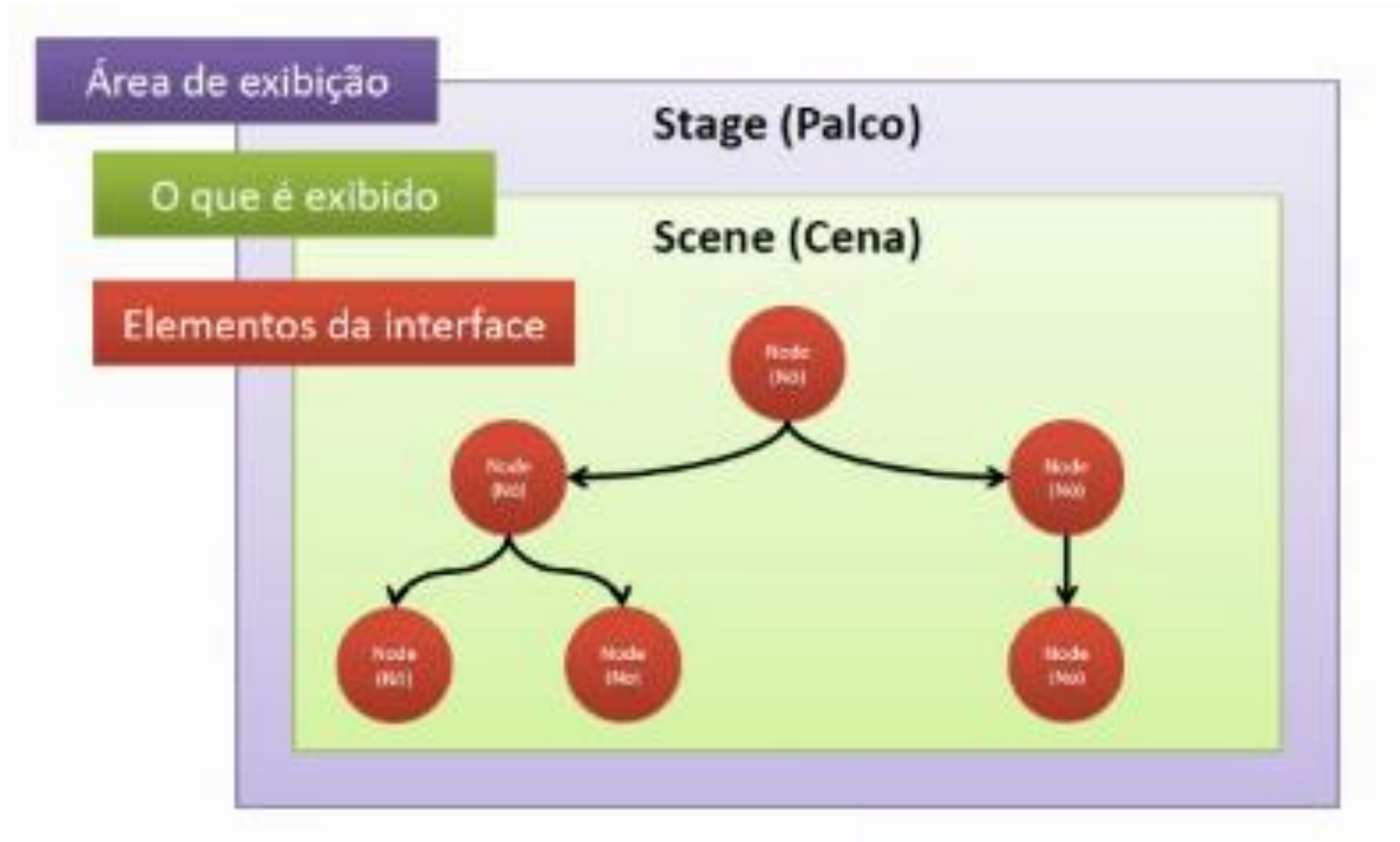
## JavaFX MVC

### ► Arquitetura



## JavaFX MVC

### ► Estrutura de cena



## JavaFX MVC

---

### ▶ Classe JavaFX

- ▶ Toda Classe JavaFX deve herdar da classe Application
- ▶ Deve-se implementar seus métodos abstratos
  - ▶ *start()*

```
public class MyHelloWorldJavaFX extends Application {  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
  
    }  
  
}
```

## JavaFX MVC

---

### ► Classe JavaFX

- Implementar método principal – *main()*
  - Iniciar o JavaFX – *launch()*

```
public class MyHelloWorldJavaFX extends Application {  
  
    @Override  
    public void start(Stage primaryStage) throws Exception {  
  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
  
}
```

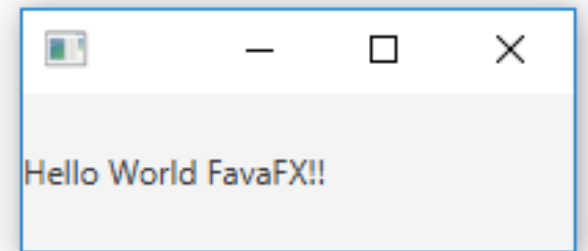
## JavaFX MVC

---

### ► Classe JavaFX

#### ► Definindo comportamento do método *start()*

```
@Override
public void start(Stage primaryStage) throws Exception {
    //Definindo nó raiz
    Label label = new Label("Hello World FavaFX!!");
    //Adicionando nó à cena
    Scene cena = new Scene(label, 200, 200);
    //Adicionando cena ao palco
    primaryStage.setScene(cena);
    //Tornando palco visível
    primaryStage.show();
}
```

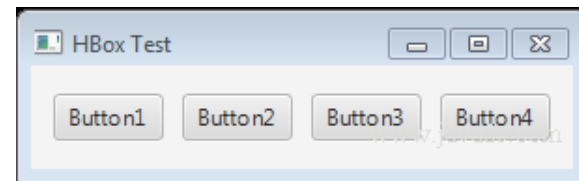


## JavaFX MVC

---

### ► Layout Panes

- Melhor forma de organizar os nós na interface gráfica
- Alguns exemplos
  - VBox
    - ❑ Organiza os nós verticalmente
  - Hbox
    - ❑ Organiza os nós horizontalmente





## JavaFX MVC

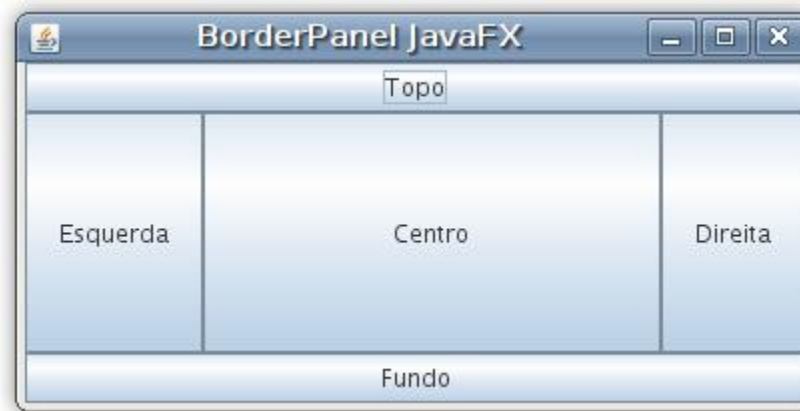
---

### ► Layout Panes

#### ► Alguns exemplos

##### ► BorderPane

- Organiza os nós em 5 regiões: *top*, *bottom*, *left*, *right* e *center*.



## JavaFX MVC

---

### ► Layout Panes

#### ► Alguns exemplos

##### ► GridPane

- Organiza os nós em linhas e colunas

```
+-----+  
| [label] [ field ] |  
| [label] [ field ] |  
|                [ button ] |  
+-----+
```

### Welcome

User Name:

Password:

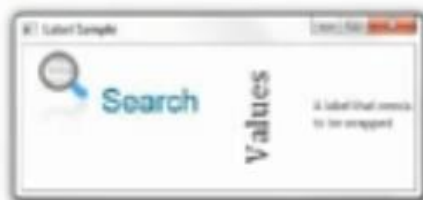
Sign in

# JavaFX MVC

## ▶ Controles

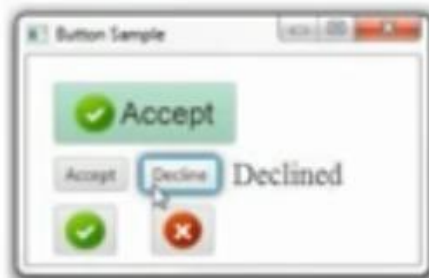


### Label



Fonte: oracle.com

### Button



Fonte: oracle.com

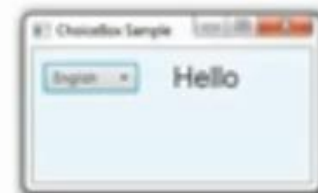
### ToggleButton



Selected toggle button

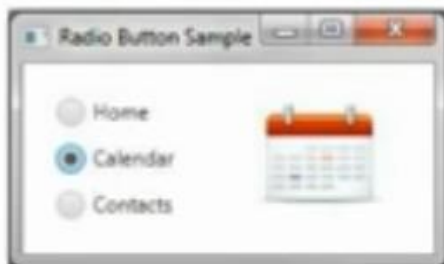
Fonte: oracle.com

### ChoiceBox



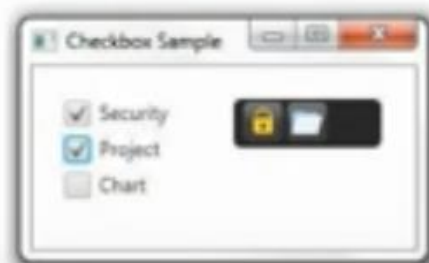
Fonte: oracle.com

### RadioButton

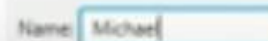


Fonte: oracle.com

### Checkbox



### TextField



# JavaFX MVC

## ▶ Controles

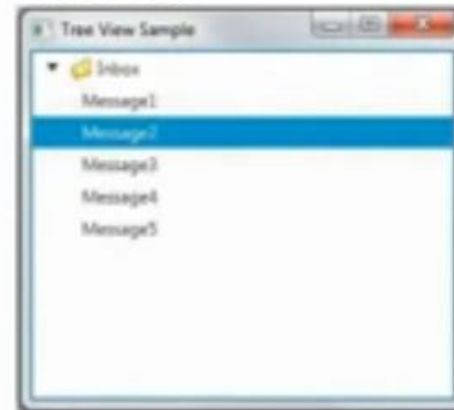


### TableView



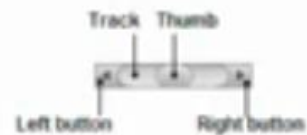
Fonte: oracle.com

### TreeView



Fonte: oracle.com

### ScrollBar



Fonte: oracle.com

### ListView



Fonte: oracle.com

## JavaFX MVC

---

### ► Formas de criar interfaces com JavaFX

#### Programação Direta

- Java puro
- Desenho da interface fica misturado com código JAVA

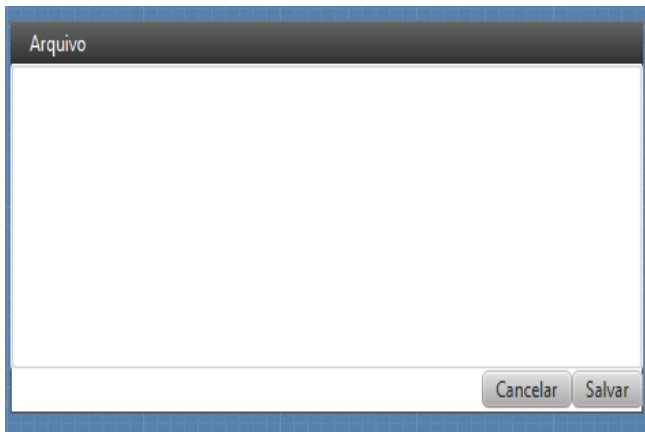
#### FXML

- Linguagem para representação de estrutura de cena
- Interface é definida em XML, separada do código JAVA
- Facilita a manutenção
- Não precisa recompilar o código

## JavaFX MVC

### ► Formas de criar interfaces com JavaFX

#### ► Via Programação Direta



```
BorderPane root = new BorderPane();

MenuBar menuBar = new MenuBar();
menuBar.getMenus().add(new Menu("Arquivo"));
root.setTop(menuBar);

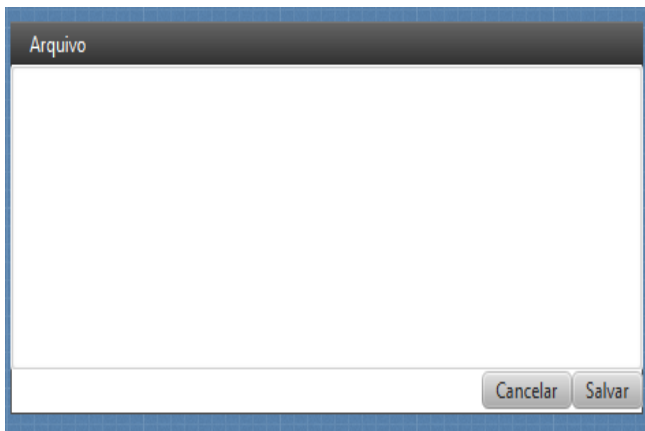
TextArea textArea = new TextArea();
root.setCenter(textArea);

Button btnCancel = new Button("Cancelar");
Button btnSalvar = new Button("Salvar");
HBox buttonBox = new HBox();
buttonBox.getChildren().add(btnCancel);
buttonBox.getChildren().add(btnSalvar);
root.setBottom(buttonBox);
```

## JavaFX MVC

### ► Formas de criar interfaces com JavaFX

#### ► Via FXML



```
<BorderPane>
  <top>
    <MenuBar>
      <menus>
        <Menu text="Arquivo">
          <items>
            <MenuItem text="Close" />
          </items>
        </Menu>
      </menus>
    </MenuBar>
  </top>
  <bottom>
    <HBox >
      <children>
        <Button text="Cancelar" />
        <Button text="Salvar" />
      </children>
    </HBox>
  </bottom>
  <center>
    <TextArea />
  </center>
</BorderPane>
```

## *JavaFX MVC*

---

### ▶ Formas de criar arquivos FXML

#### ▶ Manualmente

- ▶ Editores de texto (Estilo Matrix)

#### ▶ Ferramentas específicas para construção visual de interface

- ▶ SceneBuilder (Estilo arrastar e soltar)


□ <http://www.oracle.com/technetwork/java/javase/archive-139210.html>



## JavaFX MVC

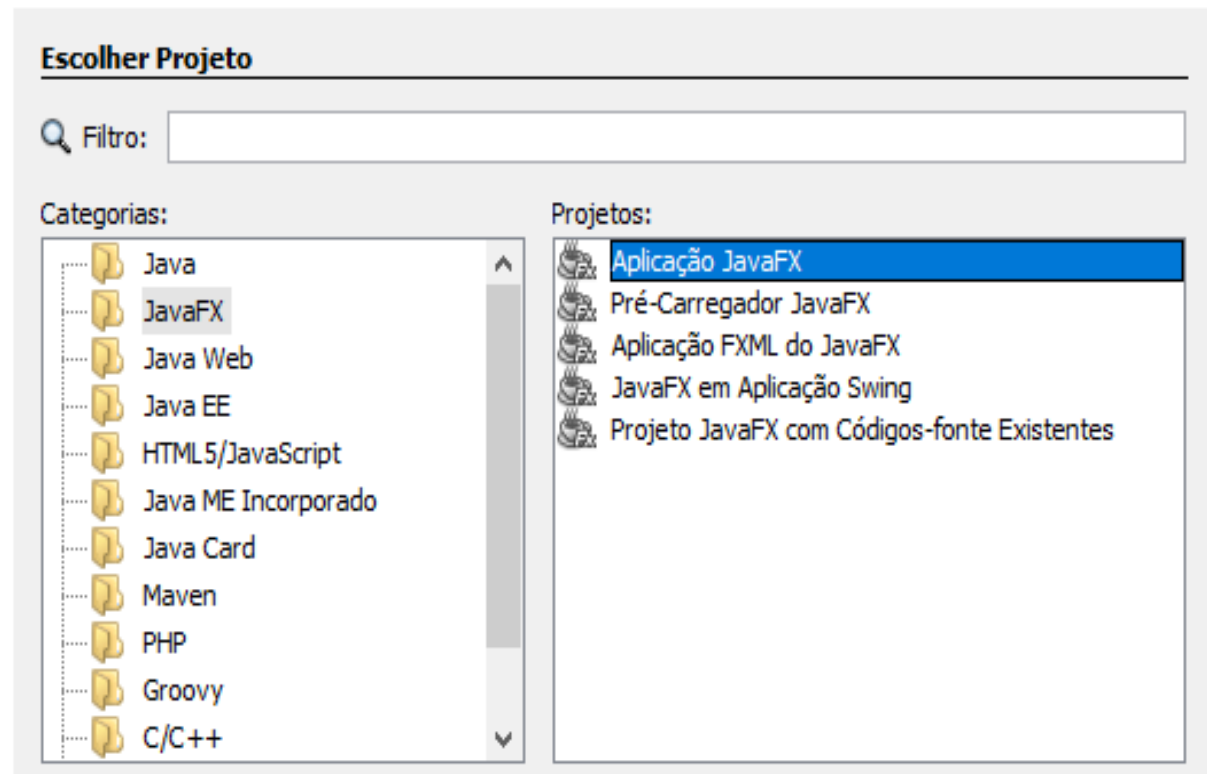
### ► Hello World JavaFx!

#### ► No Netbeans...

 Novo Projeto

#### Etapas

1. Escolher Projeto
2. ...



## JavaFX MVC

### ► Hello World JavaFx!

#### ► No Netbeans...

Novo Aplicação JavaFX

**Etapas**

1. Escolher Projeto
2. **Nome e Localização**

**Nome e Localização**

Nome do Projeto: HelloWorldJavaFX

Localização do Projeto: C:\Users\Jhoyce\Google Drive\NetBeansProjects Procurar...

Pasta do Projeto: C:\Users\Jhoyce\Google Drive\NetBeansProjects\HelloWorldJavaFX

☐ Criar Pré-Carregador Personalizado

Nome do Projeto: HelloWorldJavaFX-Preloader

☐ Usar Pasta Dedicada para Armazenar Bibliotecas

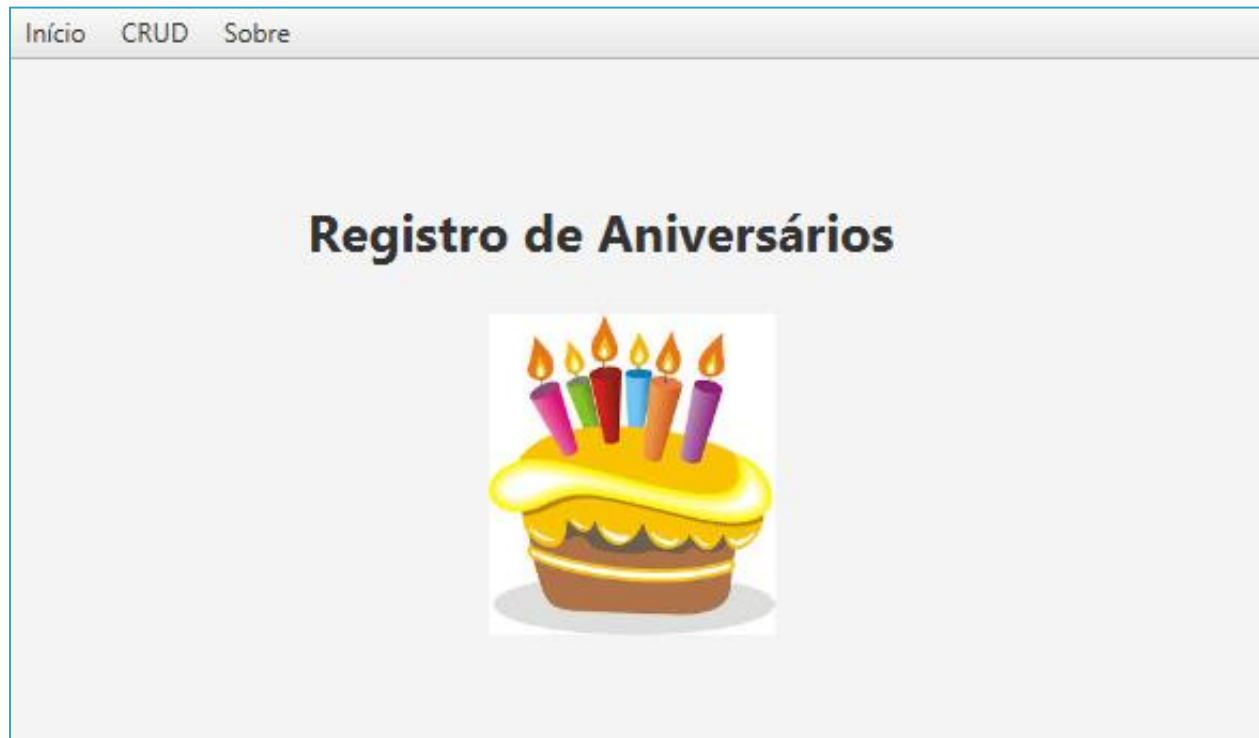
Pasta Bibliotecas: Procurar...

Usuários e projetos diferentes podem compartilhar as mesmas bibliotecas de compilação (consulte a Ajuda para obter detalhes).

## JavaFX MVC

---

- ▶ Carregando FXML para dentro da aplicação
  - ▶ Criando arquivo FXML
    - ▶ Novo FXML Vazio (**Layout.fxml**) – Abrir com SceneBuilder – Montar Layout



## JavaFX MVC

---

- ▶ Carregando FXML para dentro da aplicação
  - ▶ Linkando FXML

```
@Override
public void start(Stage primaryStage) throws Exception {

    Parent root = FXMLLoader.load(getClass().getResource("Layout.fxml"));

    Scene scene = new Scene(root, 600, 400);

    primaryStage.setScene(scene);
    primaryStage.show();

}
```

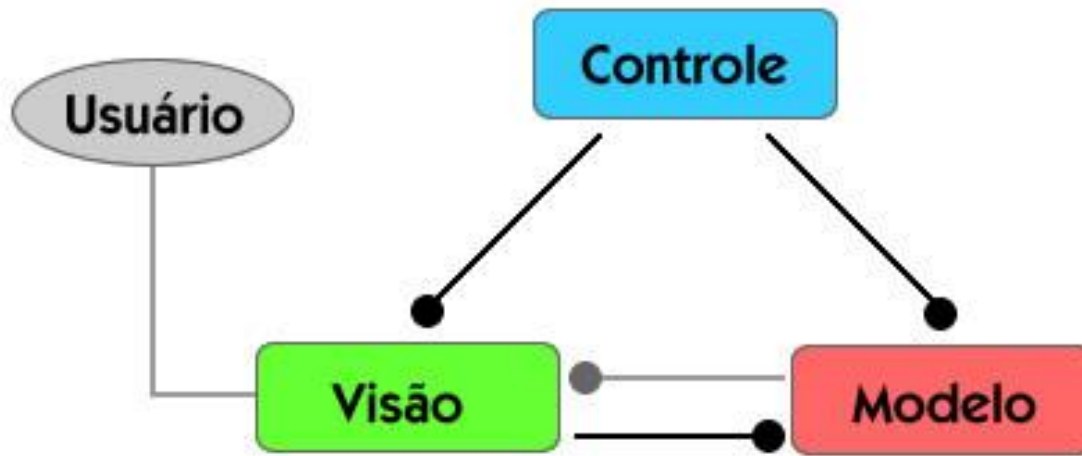
- ▶ Problema de Referência: Limpar e Construir Projeto

## JavaFX MVC

---

### ► Padrão MVC

- Propõe a separação lógica do código em camadas de acordo com sua funcionalidade: Model – View – Controller
- JavaFx foi projetado para se adequar a esse padrão
- Adota o sistema 1:1
  - Para cada arquivo de View deve existir um arquivo de Controller



## JavaFX MVC

---

### ► Padrão MVC

- 1º : Gerar uma classe de Controle

```
public class LayoutCRUDController {  
    public void initialize() {  
        System.out.println("Controller iniciado!");  
    }  
}
```

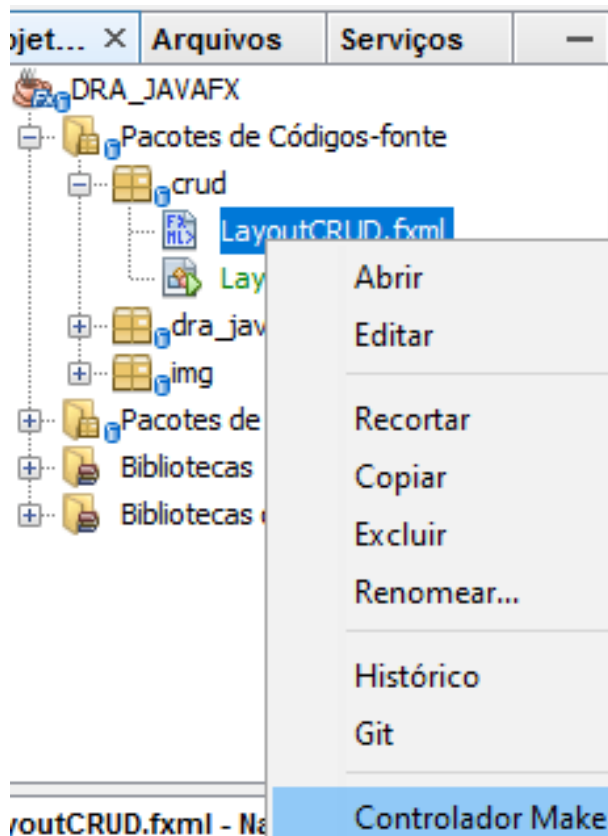
- 2º: Linkar a classe de Controle com a View
  - Editar a tag raiz do seu arquivo fxml.

```
<BorderPane fx:controller="crud.LayoutCRUDController"
```

## JavaFX MVC

### ► Padrão MVC

#### ► Alternativa: Gerando a classe de Controle automaticamente



```
public class LayoutCRUDController implements Initializable {  
    public void initialize(URL url, ResourceBundle rb) {  
        System.out.println("Controller iniciado");  
    }  
}
```

## JavaFX MVC

---

### ► Padrão MVC

- Adicionando ação ao botão – Fechar Janela





## JavaFX MVC

---

### ► Padrão MVC

#### ► Adicionando ação ao botão – Fechar Janela

```
<MenuItem mnemonicParsing="false" text="Sair" onAction="#closeApp"/>
```

```
public class LayoutCRUDController implements Initializable {  
    @Override  
    public void initialize(URL url, ResourceBundle rb) {  
        System.out.println("Controller iniciado");  
    }  
    public void closeApp(){  
        Platform.exit();  
    }  
}
```

## JavaFX MVC

---

### ► Padrão MVC

- Adicionando ação ao botão – Abrir outra Janela



## JavaFX MVC

### ► Implementando um CRUD

- Vamos criar esse layout com o SceneBuilder: LayoutCRUD

Pesquisa

Id	Nome	Aniversário
86	fulano	2017-06-26
73	beltrano	2017-06-26
60	ciclano	2017-06-26

Id

Nome

Aniversário

Cancelar Salvar

Novo Excluir

## JavaFX MVC

---

### ► Padrão MVC

- Adicionando ação ao botão – Abrir outra Janela

#### Layout.fxml

```
<MenuItem mnemonicParsing="false" onAction="#openLayoutCRUD" text="Aniversário" />
```

#### LayoutController.java

```
public void openLayoutCRUD() throws IOException{  
    Stage stage = new Stage();  
    Parent root = FXMLLoader.load(getClass().getResource("/crud/LayoutCRUD.fxml"));  
    Scene scene = new Scene(root, 600, 400);  
    stage.setScene(scene);  
    stage.initModality(Modality.APPLICATION_MODAL);  
    stage.show();  
}
```

## JavaFX MVC

### ► Padrão MVC

#### ► Linkando elementos da View no Controller

Pesquisa

Id	Nome	Aniversário
86	fulano	2017-06-26
73	beltrano	2017-06-26
60	ciclano	2017-06-26

Id

Nome

Aniversário

Cancelar Salvar

Novo Excluir

Carregando Valores

## JavaFX MVC

---

### ► Padrão MVC

- Linkando elementos da View no Controller
  - Defina um id para o componente da interface

```
<TableView fx:id="tabelaAniversarios"
```

- Crie no Controller um atributo com o mesmo nome

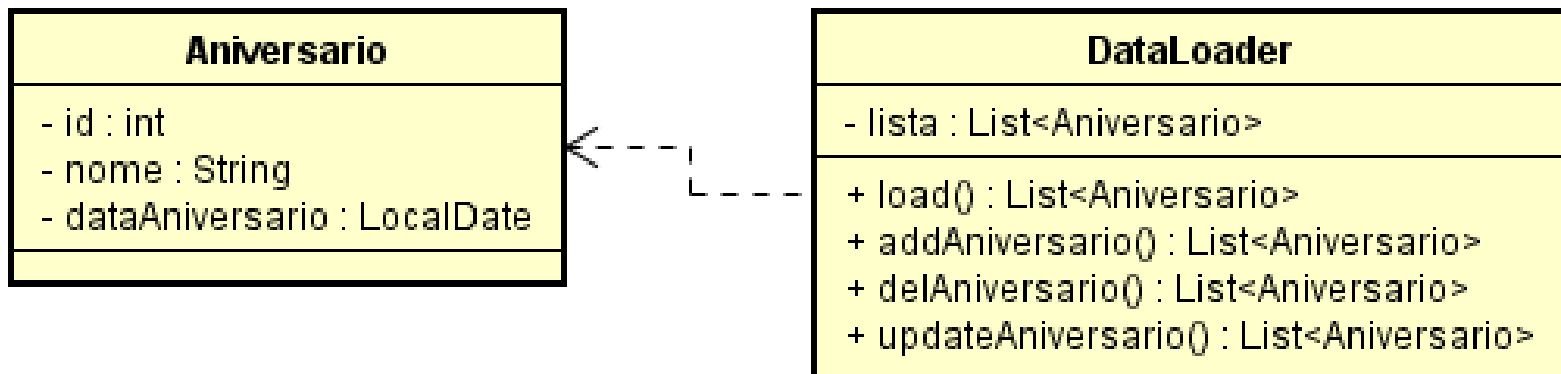
```
public class LayoutCRUDController implements Initializable {  
  
    @FXML  
    private TableView tabelaAniversarios;  
}
```

## JavaFX MVC

---

### ► Padrão MVC

- Linkando elementos da View no Controller
  - Utilizando Camada Model



## JavaFX MVC

---

### ► Padrão MVC

- Linkando elementos da View no Controller
  - Implementação da Classe DataLoader{}

```
public class DataLoader {  
  
    static List<Aniversario> lista ;  
  
    public static List<Aniversario> load(){  
        lista = new ArrayList<>();  
        lista.add(new Aniversario(gerarId(), "fulano" , LocalDate.now()));  
        lista.add(new Aniversario(gerarId(), "beltrano", LocalDate.now()));  
        lista.add(new Aniversario(gerarId(), "ciclano" , LocalDate.now()));  
        return lista;  
    }  
}
```



## JavaFX MVC

---

### ► Padrão MVC

- Linkando elementos da View no Controller
  - Carregando itens na tabela

```
public void loadTabela() {  
    //preenchendo tabela  
    listaAniversarios = DataLoader.load();  
    observableList = FXCollections.observableArrayList(listaAniversarios);  
    tabelaAniversarios.setItems(observableList);  
}
```

## JavaFX MVC

---

### ► Padrão MVC

- Linkando elementos da View no Controller
  - Carregando itens na tabela

```
public void loadTabela() {  
    //preenchendo tabela  
    listaAniversarios = DataLoader.load();  
    observableList = FXCollections.observableArrayList(listaAniversarios);  
    tabelaAniversarios.setItems(observableList);  
}
```

**ObservableList:** é  
possível definir ações  
(listeners) para quando  
a lista for alterada

## JavaFX MVC

---

### ► Padrão MVC

- Linkando elementos da View no Controller
  - ObservableList: adicionando Listener

```
//add listener da observableList  
observableList.addListener(new ObservableListAlterada());
```

```
class ObservableListAlterada implements ListChangeListener{  
    @Override  
    public void onChanged(Change c) {  
        //toda vez que a observablelist for alterada  
        //a tabela é atualizada  
        tabelaAniversarios.setItems(c.getList());  
    }  
}
```

## ▶ Linkando elementos do Controller na View

## Como saber qual valor vai aparecer em cada coluna?

Por meio das Properties

## JavaFX MVC

---

### ► Padrão MVC

#### ► Linkando elementos do Controller na View

##### ► Definindo valores que serão apresentados em cada coluna

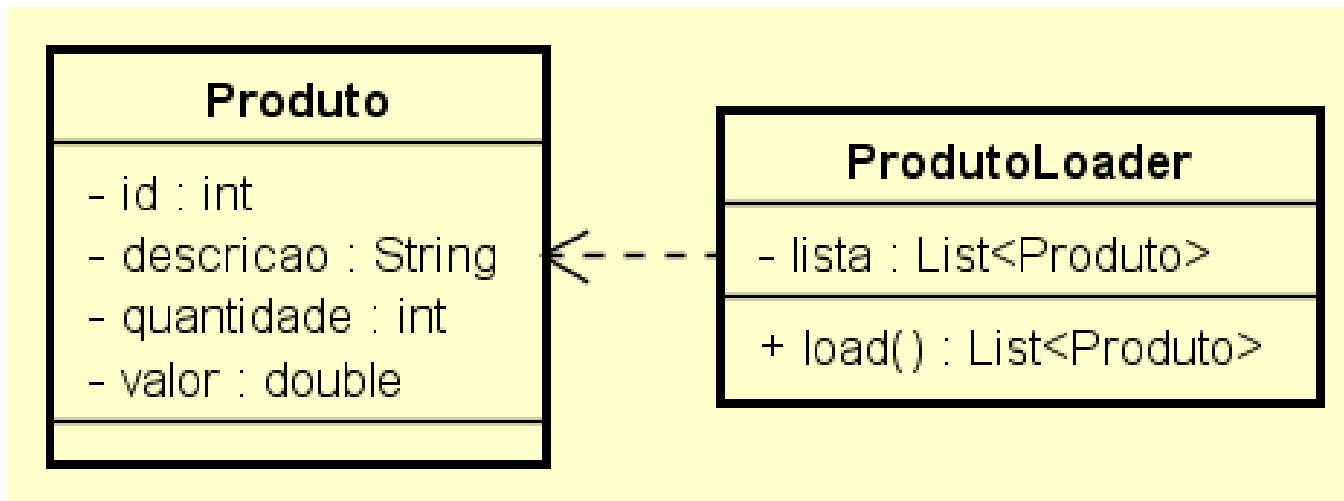
- Fazendo Binding de Properties - Chama os gets da classe Modelo

```
<TableColumn prefWidth="75.0" text="Id">
    <cellValueFactory>
        <PropertyValueFactory property="id" />
    </cellValueFactory>
</TableColumn>
<TableColumn prefWidth="75.0" text="Nome">
    <cellValueFactory>
        <PropertyValueFactory property="nome" />
    </cellValueFactory>
</TableColumn>
<TableColumn prefWidth="75.0" text="Aniversário">
    <cellValueFactory>
        <PropertyValueFactory property="dataAniversario" />
    </cellValueFactory>
</TableColumn>
```

## JavaFX MVC

---

### ► Exercício de Fixação



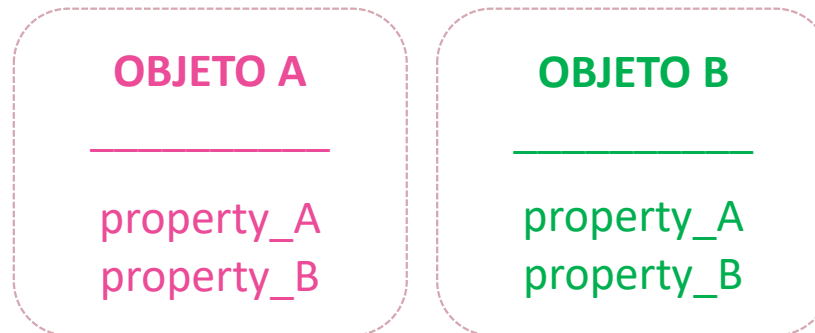
## JavaFX MVC

### ► Padrão MVC

#### ► *Properties*

- Característica de um objeto
- Atributo mais inteligente
  - Permite ligar *properties* com o objetivo de sincronização (*Binding*)

*a.property\_A.bind(b.property\_B)*



*Quando b.property\_B for alterado,  
a.property\_A também será.*

## JavaFX MVC

---

### ► Padrão MVC

#### ► *Properties*

- Para trabalhar com Properties será necessário utilizar o modelo **JavaBean**.
- **JavaBean** é uma classe que segue as seguintes convenções:
  - ❑ Construtor padrão sem parâmetros
  - ❑ Métodos getters e setters
- Essa classe ainda não está preparada para o uso de Properties

```
public class PessoaJavaBean {  
    private String nome;  
  
    public PessoaJavaBean() {  
    }  
  
    public String getNome() {  
        return nome;  
    }  
  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
}
```



# JavaFX MVC

---

## ► Padrão MVC

### ► *Properties*

#### ► Lembre-se:

- ❑ As properties podem ser vistas como atributos, mas precisam ser declaradas com tipos específicos.

#### ► Tipos

- ❑ *StringProperty*
- ❑ *IntegerProperty*
- ❑ *DoubleProperty*
- ❑ *BooleanProperty*
- ❑ *ObjectProperty<Object>*
  - ❑ Ex: *ObjectProperty<LocalDate>*

```
public class PessoaJavaBeanProperties {  
  
    private StringProperty nome = new SimpleStringProperty();  
  
}
```

## JavaFX MVC

### ► Padrão MVC

#### ► *Properties*

- Preparando o JavaBean para trabalhar com properties

```
public class PessoaJavaBeanProperties {  
    private StringProperty nome = new SimpleStringProperty();  
  
    public PessoaJavaBeanProperties(String nome) {  
        this.nome.set(nome);  
    }  
  
    public String getNome() {  
        return nome.get();  
    }  
  
    public void setNome(String value) {  
        nome.set(value);  
    }  
  
    public StringProperty nomeProperty() {  
        return nome;  
    }  
}
```

Importante para  
fazer o *Binding*

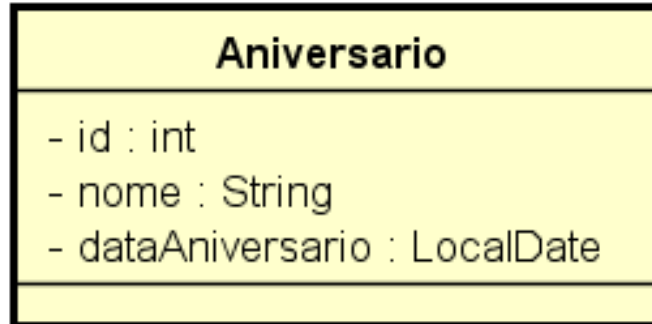
## JavaFX MVC

---

### ► Exercício de Fixação

#### ► *Properties*

- Vamos gerar o JavaBean referente ao modelo abaixo utilizando conceito de Property do JavaFX.



## JavaFX MVC

### ► Padrão MVC

#### ► Aplicando o *Binding*

- Vamos ligar os campos da tabela com os campos do formulário.
  - Quando estivermos editando alguma informação, isso vai ser refletido de forma síncrona na tabela.

The screenshot displays a JavaFX application interface. On the left, there is a table with three columns: 'Id', 'Nome', and 'Aniversário'. The table contains three rows of data: (16, fulano, 2017-06-26), (13, beltrano, 2017-06-26), and (8, ciclano, 2017-06-26). The row with Id 13 is highlighted in blue. To the right of the table is a search bar labeled 'Pesquisa'. To the right of the search bar is a form with three fields: 'Id' (containing 13), 'Nome' (containing beltrano), and 'Aniversário' (containing 26/06/2017). Below the form are two buttons: 'Cancelar' and 'Salvar'. At the bottom of the table are two buttons: 'Novo' and 'Excluir'.

Id	Nome	Aniversário
16	fulano	2017-06-26
13	beltrano	2017-06-26
8	ciclano	2017-06-26

## JavaFX MVC

---

### ► Padrão MVC

#### ► Aplicando o *Binding*

- Na classe Controller, no evento Item Selecionado da Tabela...

```
//add listener de selecao da tabela  
tabelaAniversarios.getSelectionModel().selectedItemProperty().addListener(  
    new ServicoSelecionadoListener());
```

```
class ServicoSelecionadoListener implements ChangeListener<Aniversario>{
    @Override
    public void changed(ObservableValue<? extends Aniversario> observable,
        Aniversario oldValue, Aniversario newValue) {
        //definindo binding
        if (oldValue !=null) {
            txtId.textProperty().unbindBidirectional(
                oldValue.idProperty());
            txtNome.textProperty().unbindBidirectional(
                oldValue.nomeProperty());
            txtDataAniversario.valueProperty().unbindBidirectional(
                oldValue.dataAniversarioProperty());
        }
        if (newValue !=null){
            txtId.textProperty().bindBidirectional(
                newValue.idProperty(), NumberFormat.getNumberInstance());
            txtNome.textProperty().bindBidirectional(
                newValue.nomeProperty());
            txtDataAniversario.valueProperty().bindBidirectional(
                newValue.dataAniversarioProperty());
        }
    }
}
```

Definindo  
Binding  
Bidirecional

## JavaFX MVC

---

### ► Padrão MVC

#### ► Aplicando o *Binding*

- Habilitar botões somente se algum registro da tabela for selecionado

Id	Nome	Aniversário	
16	fulano	2017-06-26	
13	beltrano	2017-06-26	
8	ciclano	2017-06-26	

Id	Nome	Aniversário	
16	fulano	2017-06-26	
13	beltrano	2017-06-26	
8	ciclano	2017-06-26	

## JavaFX MVC

---

### ► Padrão MVC

#### ► Aplicando o *Binding*

- Habilitar botões somente se algum registro da tabela for selecionado

Id	Nome	Aniversário	
16	fulano	2017-06-26	
13	beltrano	2017-06-26	
8	ciclano	2017-06-26	

```
//binding para habilitar/desabilitar botoes  
btnExcluir.disableProperty().bind(  
    tabelaServicos.getSelectionModel().selectedItemProperty().isNull());
```



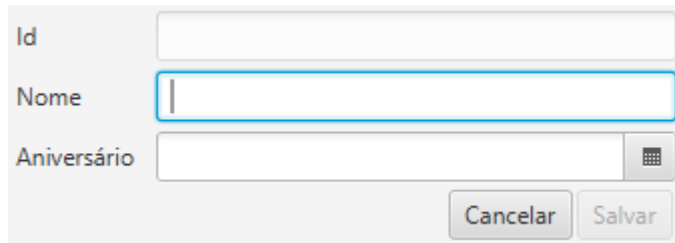
## JavaFX MVC

---

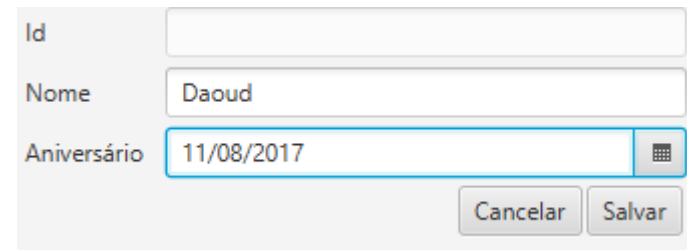
### ► Padrão MVC

#### ► Aplicando o *Binding*

- Habilitar botões se todos os campos estiverem preenchidos



A screenshot of a JavaFX form with three input fields: 'Id', 'Nome', and 'Aniversário'. The 'Nome' field is currently selected with a blue border. The 'Id' and 'Aniversário' fields are empty. Below the fields are two buttons: 'Cancelar' and 'Salvar'. Both buttons are disabled, indicated by a light gray background and no focus border.



A screenshot of the same JavaFX form, but now the 'Nome' field contains the text 'Daoud' and the 'Aniversário' field contains the date '11/08/2017'. The 'Cancelar' and 'Salvar' buttons are now enabled, shown with a darker gray background and a focus border.

```
btnSalvar.disableProperty().bind(txtLogin.textProperty().isEmpty() .  
    or(txtLogin.textProperty().isEmpty()) .  
    or(txtSenha.textProperty().isEmpty()));
```

## JavaFX MVC

### ► Padrão MVC

#### ► Adicionando Serviço

DataLoader
- lista : List<Aniversario>
+ load() : List<Aniversario> + addAniversario() : List<Aniversario> + delAniversario() : List<Aniversario> + updateAniversario() : List<Aniversario>

```
<Button fx:id="btnSalvar" mnemonicParsing="false" onAction="#addAniversario" text="Salvar" />
```

```
public void addAniversario() {  
    if(tabelaAniversarios.getSelectionModel().getSelectedItem() == null){  
        Aniversario ss = new Aniversario(  
            DataLoader.gerarId(),  
            txtNome.getText(),  
            txtDataAniversario.valueProperty().getValue() );  
  
        listaAniversarios = DataLoader.addAniversario(ss);  
        observableList.setAll(  
            FXCollections.observableArrayList(listaAniversarios));  
    }  
    limparCampos();  
    JOptionPane.showMessageDialog(null, "Salvo!");  
}
```

## JavaFX MVC

### ► Padrão MVC

#### ► Adicionando Filtro

Pesquisa

Id	Nome	Aniversário
13	beltrano	2017-06-26
58	beltrano 2	2017-06-27

```
//add listener de campo de busca  
txtAniversarioPesquisado.textProperty().addListener(  
    new ServicoPesquisadoListener());
```

## JavaFX MVC

### ► Padrão MVC

#### ► Adicionando Filtro

```
class ServicoPesquisadoListener implements ChangeListener<String>{
    @Override
    public void changed(ObservableValue<? extends String> observable,
                       String oldValue, String newValue) {

        //criando filtro
        FilteredList<Aniversario> filteredData = new FilteredList(observableList);

        //definindo criterio do filtro
        filteredData.setPredicate(aniversario ->
                                aniversario.getNome().contains(newValue));

        //aplicando filtro ao SortedList
        SortedList<Aniversario> sortedData = new SortedList<>(filteredData);

        //adicionando dados filtrados e ordenados à tabela
        tabelaAniversarios.setItems(sortedData);
    }
}
```

## JavaFX MVC

### ► Padrão MVC

#### ► Adicionando Filtro

```
class ServicoPesquisadoListener implements ChangeListener<String>{
    @Override
    public void changed(ObservableValue<? extends String> observable,
                        String oldValue, String newValue) {

        //criando filtro
        FilteredList<Aniversario> filteredData = new FilteredList(observableList);

        //definindo criterio do filtro
        filteredData.setPredicate(aniversario ->
                                aniversario.getNome().contains(newValue));

        //aplicando filtro ao SortedList
        SortedList<Aniversario> sortedData = new SortedList<>(filteredData);

        //adicionando dados filtrados e ordenados à tabela
        tabelaAniversarios.setItems(sortedData);
    }
}
```

## JavaFX MVC

### ► Padrão MVC

#### ► Adicionando Filtro

```
class ServicoPesquisadoListener implements ChangeListener<String>{
    @Override
    public void changed(ObservableValue<? extends String> observable,
                        String oldValue, String newValue) {

        //criando filtro
        FilteredList<Aniversario> filteredData = new FilteredList(observableList);

        //definindo criterio do filtro
        filteredData.setPredicate(aniversario ->
                                aniversario.getNome().contains(newValue));

        //aplicando filtro ao SortedList
        SortedList<Aniversario> sortedData = new SortedList<>(filteredData);

        //adicionando dados filtrados e ordenados à tabela
        tabelaAniversarios.setItems(sortedData);
    }
}
```

## JavaFX MVC

### ► Padrão MVC

#### ► Adicionando Filtro

```
class ServicoPesquisadoListener implements ChangeListener<String>{
    @Override
    public void changed(ObservableValue<? extends String> observable,
                        String oldValue, String newValue) {

        //criando filtro
        FilteredList<Aniversario> filteredData = new FilteredList(observableList);

        //definindo criterio do filtro
        filteredData.setPredicate(aniversario ->
                                aniversario.getNome().contains(newValue));

        //aplicando filtro ao SortedList
        SortedList<Aniversario> sortedData = new SortedList<>(filteredData);

        //adicionando dados filtrados e ordenados à tabela
        tabelaAniversarios.setItems(sortedData);
    }
}
```

### ► Tarefa de Implementação

- Crie a interface gráfica e implemente as funcionalidades referentes ao modelo apresentado abaixo.
- Utilize JavaFX e persista os dados em um banco de dados MySQL.

