

# PROGRAMAÇÃO ORIENTADA A OBJETOS

---

Prof. Bernardo Copstein

Profa. Isabel H. Manssour

# INTRODUÇÃO AO JAVAFX

## PARTE I: LAYOUTS BÁSICOS

---

Leitura recomendada:

- <http://docs.oracle.com/javase/8/javase-clienttechnologies.htm>
- [http://docs.oracle.com/javafx/2/get\\_started/jfxpub-get\\_started.htm](http://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm)

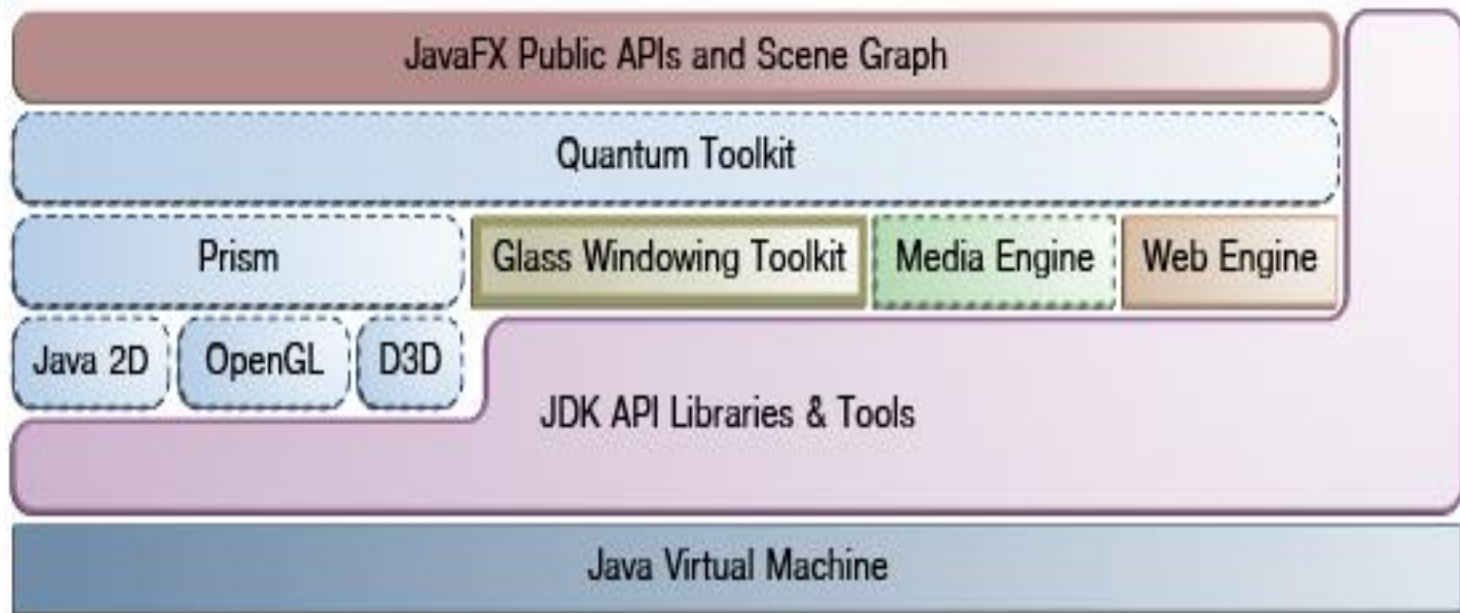
# O que é JavaFX

- JavaFX é um conjunto de pacotes de gráficos e mídia que permite desenvolver aplicações de cliente rico que executam de maneira consistente em diferentes plataformas.
- APIs para desenvolvimento de GUI em Java:
  - JDK (classes básicas)
  - SWING
  - JavaFX
    - Trabalha com componentes
    - Permite usar CSS
    - Permite usar FXML
    - Permite integrar componentes WEB

# Integração com IDEs

- JavaFX integra-se com diferentes IDEs e possui ferramenta de edição de *forms*
- Neste curso o objetivo é entender o funcionamento da API e os padrões de projeto utilizado e por esta razão este tipo de recurso não será utilizado

# Arquitetura do JavaFX



# Estrutura de uma aplicação

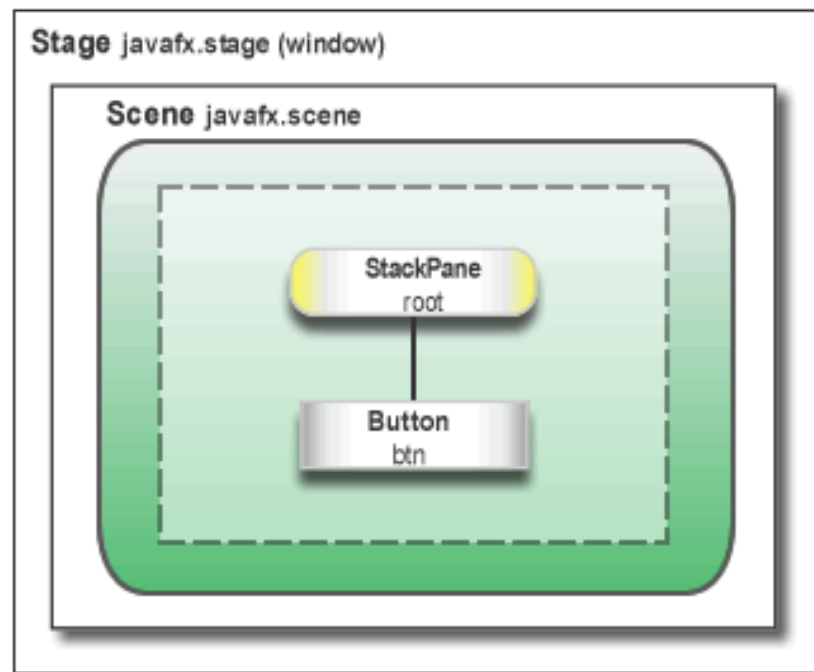
- O “**grafo de uma cena**” é o ponto de partida para construir uma aplicação JavaFX
- Consiste de uma árvore de nodos que representam todos os elementos visuais da interface com o usuário.
- Cada elemento pode tratar entrada com o usuário e ser “renderizado”.
- Todos os nodo tem um identificador.
- Com exceção do nodo raiz todos os demais podem ter zero ou mais filhos.

# *Containers* e Componentes

- Uma interface gráfica em Java é baseada em:
  - *Containers*
    - Servem para agrupar e exibir outros componentes
  - Componentes
    - São os botões, campos de textos, etc.

# Containers

- O “*stage*” (palco) é o container principal que corresponde a uma janela
- Uma “*scene*” (cena) é organizada na forma de uma árvore de componentes com relação de hierarquia entre si
- O componente “*root*” (raiz) normalmente é um “gerenciador de layout” (define a maneira pela qual os componentes subordinados a ele serão exibidos na cena).





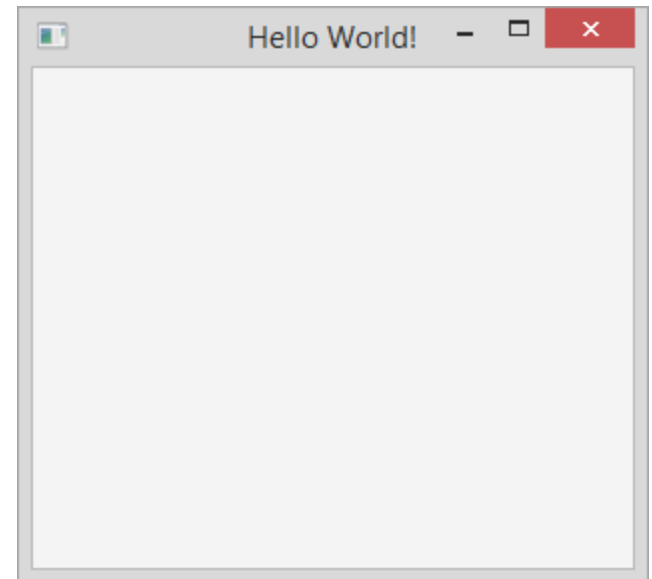
# Componentes

- Exemplos de componentes que podem ser organizados sobre uma cena



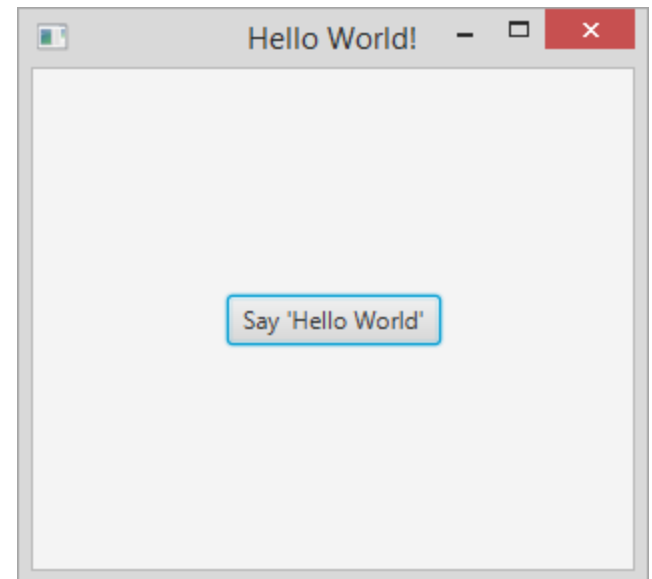
# Less than “Hello World”

```
public class HelloWorldJavaFX extends Application {  
    @Override  
    public void start(Stage primaryStage) {  
        StackPane root = new StackPane();  
        Scene scene = new Scene(root, 300, 250);  
        primaryStage.setTitle("Hello World!");  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```



# Hello World

```
public class HelloWorldJavaFX extends Application {  
    @Override  
    public void start(Stage primaryStage) {  
        Button btn = new Button();  
        btn.setText("Say 'Hello World'");  
        StackPane root = new StackPane();  
        root.getChildren().add(btn);  
        Scene scene = new Scene(root, 300, 250);  
        primaryStage.setTitle("Hello World!");  
        primaryStage.setScene(scene);  
        primaryStage.show();  
    }  
  
    public static void main(String[] args) {  
        launch(args);  
    }  
}
```



# Os *imports*

```
import javafx.application.Application;  
import javafx.event.ActionEvent;  
import javafx.event.EventHandler;  
import javafx.scene.Scene;  
import javafx.scene.control.Button;  
import javafx.scene.layout.StackPane;  
import javafx.stage.Stage;
```

# Criando um formulário simples



The image shows a JavaFX application window titled "JavaFX Welcome". Inside the window, there is a simple login form. The form consists of a "Welcome" label, followed by a "User Name:" label and a text input field, and then a "Password:" label and a text input field. The text input fields are empty and have a light blue border. The window has a standard macOS-style title bar with a red close button, a yellow maximize button, and a green minimize button.

JavaFX Welcome

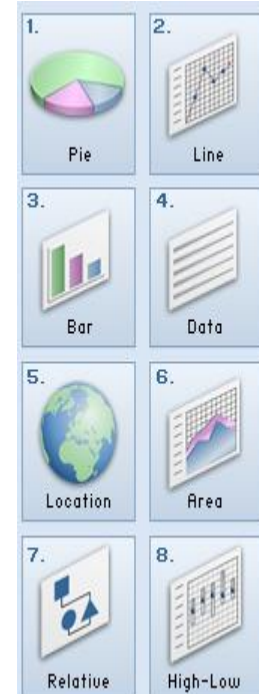
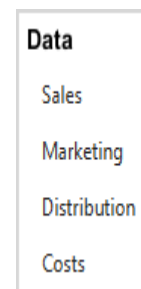
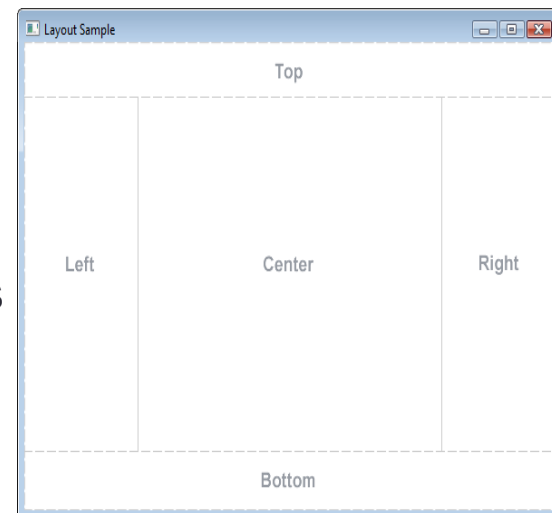
Welcome



User Name:

Password:

# Layouts

- Determinam a disposição dos componentes
- JavaFX oferece um conjunto de layouts
  - **BorderPane**: divide em 5 regiões para colocar os nodos
  - **HBox**: organiza um conjunto de nodos em uma linha
  - **VBox**: organiza um conjunto de nodos em uma coluna
  - **StackPane**: coloca todos os nodos em uma pilha
  - **GridPane**: cria uma grade flexível de linhas e colunas nas quais os nodos são colocados
  - **FlowPane**: os nodos são dispostos consecutivamente de acordo com o limite do painel e podem "fluir" vertical ou horizontalmente.
  - **TilePane**: posiciona todos os nodos em uma grade na qual todas as células possuem o mesmo tamanho
  - **AnchorPane**: permite "ancorar" os nodos na parte superior, inferior, lado esquerdo, lado direito ou centro do painel.



	Sales: Current Year		
	Goods and	Services	
			
Goods 80%		Services 20%	

# Definindo o layout

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

    primaryStage.setTitle("JavaFX - Welcome");

    GridPane grid = new GridPane();
    grid.setAlignment(Pos.CENTER);
    grid.setHgap(10);
    grid.setVgap(10);
    grid.setPadding(new Insets(25, 25, 25, 25));
    //grid.setGridLinesVisible(true);
    ...
}
```

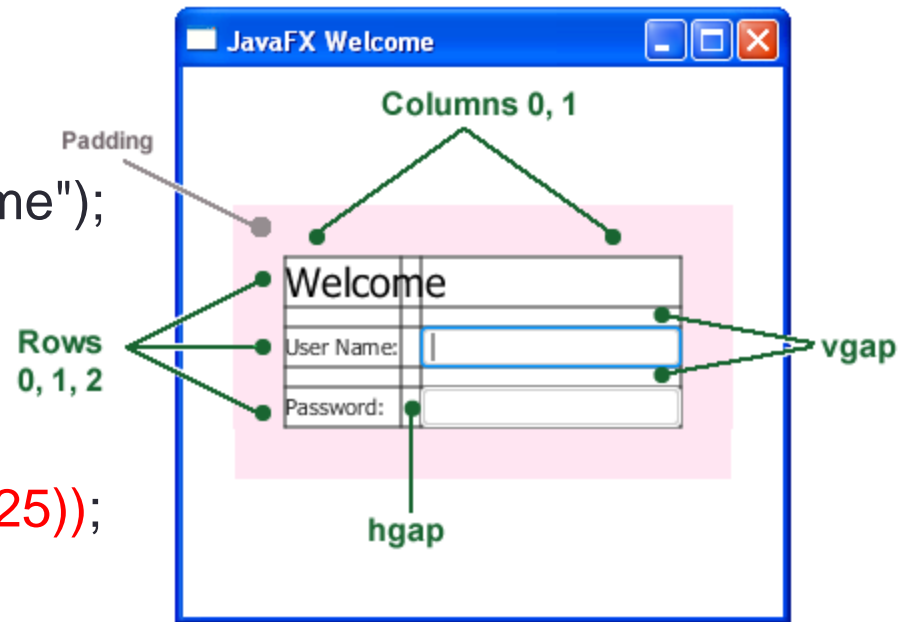
- Define um *GridPane* como *root*
- Um *GridPane* permite localizar os componentes em linhas e colunas
- É possível definir o tipo de **alinhamento** e o **espaçamento entre as células (*gap*)**
- Também se pode definir o ***padding*** (espaçamento ao redor do painel) usando *insets* (espaçamento que deve ser respeitado de cada uma das bordas do container)
- Para efeitos de depuração pode-se deixar as linhas do *GridPane* visíveis.

# Definindo o layout

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

    primaryStage.setTitle("JavaFX - Welcome");

    GridPane grid = new GridPane();
    grid.setAlignment(Pos.CENTER);
    grid.setHgap(10);
    grid.setVgap(10);
    grid.setPadding(new Insets(25, 25, 25, 25));
    //grid.setGridLinesVisible(true);
    ...
}
```





# Formatando um texto

```
Text scenetitle = new Text("Welcome");  
scenetitle.setId("welcome-text");  
scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));  
grid.add(scenetitle, 0, 0);
```



Linha e coluna do "grid"  
No caso: coluna 0, linha 0

# Acrescentando componentes

```
Label userName = new Label("User Name:");  
grid.add(userName, 0, 1);  
TextField userTextField = new TextField();  
grid.add(userTextField, 1, 1);
```

- Label + TextField

```
Label pw = new Label("Password:");  
grid.add(pw, 0, 2);  
PasswordField pwBox = new PasswordField();  
grid.add(pwBox, 1, 2);
```

- Label + PasswordField

```
Button btn = new Button("Sign in");  
HBox hbBtn = new HBox(10);  
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);  
hbBtn.getChildren().add(btn);  
grid.add(hbBtn, 1, 4);
```

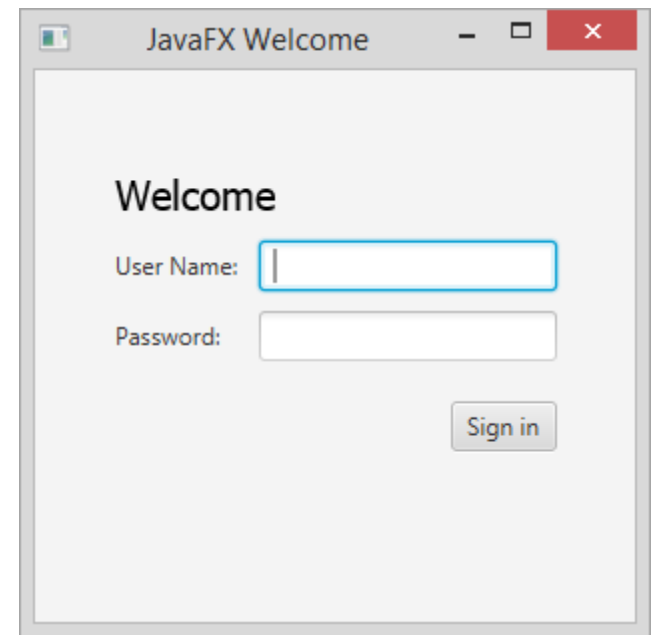
- Button (com alinhamento distinto ajustado dentro de um "HBox")

# Finalizando

...

```
Scene scene = new Scene(grid);  
    primaryStage.setScene(scene);  
    primaryStage.show();  
}
```

```
public static void main(String[] args) {  
    launch(args);  
}
```



# EXERCÍCIOS

---

# Exercícios

1. Crie uma aplicação usando JavaFX que exibe um formulário para colocar os dados relativos a um candidato ao vestibular, a saber:
  - Nome
  - Endereço (com rua, número, complemento e CEP separados)
  - Telefone (DDD e número separados)
  - Sexo (“radio buttons” em um “radio group” – pesquisar)
  - Curso (String)
  - Botões de “Enviar” e “Cancelar”

