

TI-220 Java Orientado a Objetos – Java FX

ANTONIO CARVALHO - TREINAMENTOS

A solid blue horizontal bar spanning the width of the slide, located at the bottom.

Criando a aplicação

Estrutura da janela JavaFX

No JavaFX o container superior, ou seja aquele que contém todos os elementos visuais é chamado de **Stage**

O Stage se torna uma janela no sistema operacional

Cada **Application** criado no JavaFX já possui um **Stage** próprio que é recebido como parâmetro no método **start**

Estrutura da janela JavaFX

No Java FX a hierarquia funciona como um Teatro, tendo o palco a cena, o painel de fundo e os elementos gráficos



Estrutura da janela JavaFX

A janela JavaFX consiste em uma hierarquia de elementos gráficos sendo o **Stage** o pai de todos eles.

Dentro do **Stage** fica definido um **Scene** que por sua vez pode conter um **Parent** normalmente um **Pane** ou **Group** e estes podem conter um ou mais elementos gráficos.

Stage

- Scene
 - Parent Node
 - Elemento Gráfico 1
 - Elemento Gráfico 2
 - Elemento Gráfico 3



Estrutura da janela JavaFX

A aplicação JFX é executada por meio da classe `Application`, considerada o ponto de entrada para as aplicações JavaFX.

Portanto para criar uma aplicação JFX é preciso criar uma classe que herde da classe **`javafx.application.Application`**

O método **`void start(Stage s)`** deve ser sobrescrito, para criar e colocar os elementos gráficos no **`Stage`**

A instância desta classe será criada automaticamente ao invocar o método **`Application.launch()`**

```
import javafx.application.Application;
import javafx.stage.Stage;
public class ExemploApp1 extends Application {
    @Override
    public void start(Stage stage)
        throws Exception {

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

Estrutura da janela JavaFX

No método `start()` deve ser definido o comportamento de exibição da janela. Ou seja toda a hierarquia dos componentes a serem visualizados.

Stage

- Scene
 - Parent (Region ou Group)
 - Elemento Gráfico

```
import javafx.application.Application;
import javafx.stage.Stage;
public class ExemploApp1 extends Application {
    @Override
    public void start(Stage stage)
        throws Exception {
        stage.setTitle("Titulo da Janela");
        stage.show();
    }
    public static void main(String[] args) {
        Application.launch(args);
    }
}
```

Estrutura da janela JavaFX

O elemento **Scene** representa todo o conteúdo visual da aplicação JavaFX. Os elementos visuais são guardados dentro do **Scene** através de uma estrutura do tipo árvore, onde o há um elemento raiz (**root**) que deve conter todos os demais elementos.

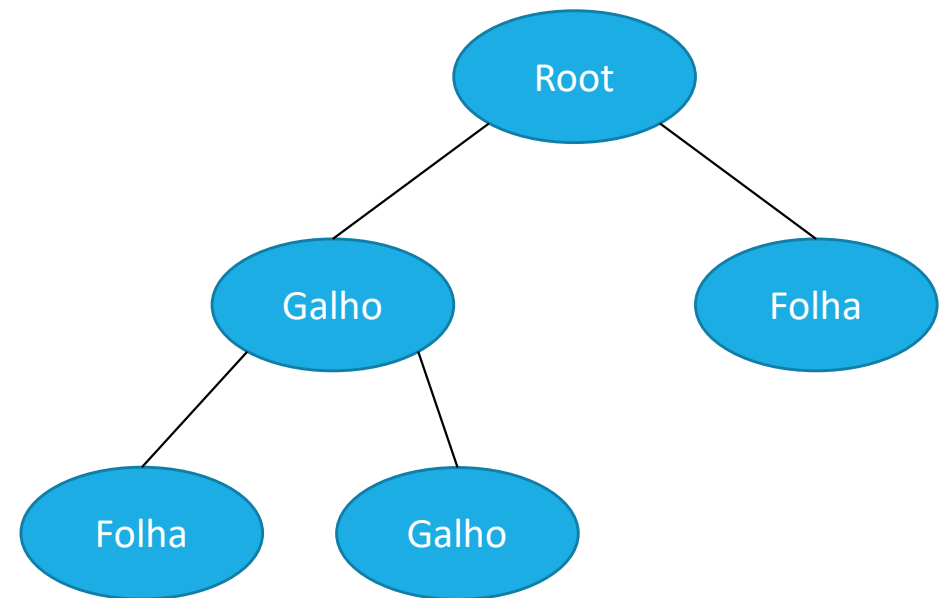
Ao criar um objeto do tipo **Scene** é preciso informar o elemento **root**, e é possível informar também o tamanho visual da cena em largura (**width**) e altura (**height**)

```
import javafx.application.Application;
import javafx.stage.Stage;
public class ExemploApp1 extends Application {
    @Override
    public void start(Stage stage)
        throws Exception {
        Pane panel = new Pane();
        Scene scn = new Scene(panel);
        stage.setScene(scn);
        stage.setTitle("Titulo da Janela");
        stage.show();
    }
    public static void main(String[] args) {
        Application.launch(args);
    }
}
```


Elemento Scene

No exemplo anterior o objeto **Scene** foi criado e seu **root** foi uma instância de **Parent** -> **Region** -> **Pane** porém existe outros tipos de nós que podem ser passados como **root**.

O elemento root é a ponta inicial de uma estrutura do tipo árvore, que pode conter folhas ou galhos.



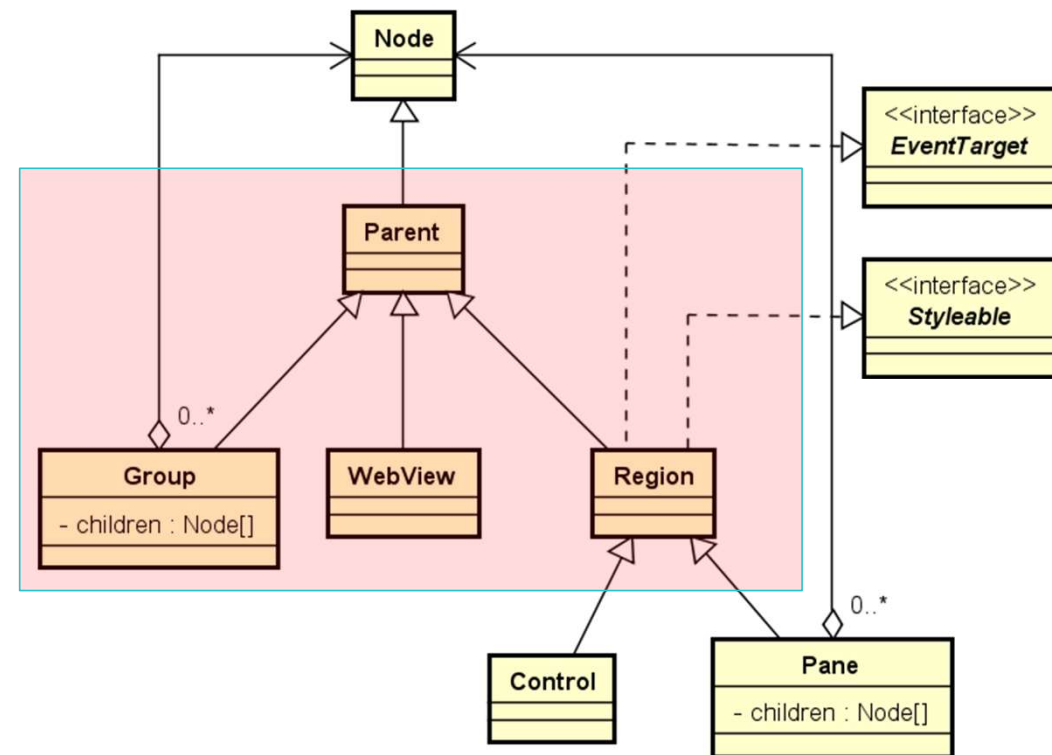
Elemento Scene

O elemento root pode ser um objeto do tipo **javafx.scene.Node** ou do tipo **javafx.scene.Parent** e esta última classe possui 3 filhos conhecidos.

Region

Group

WebView



Group

A classe **Group** consiste em uma coleção de **Node** que ao serem renderizados são desenhados na tela, porém ocupam o espaço definido pelo elemento **Group**. O objeto do tipo **Group** cria uma cerca envolvendo seus componentes, portanto seu tamanho varia conforme seus componentes internos, ou seja ele não muda o tamanho dos componentes internos.

Qualquer **transformação** aplicada ao **Group** será aplicado também aos objetos **filhos** deste **Group**

Estas características tornam o componente **Group** mais leve.

A sintaxe para criar um objeto do tipo **Group** é:

```
Group <nome do objeto> = new Group();
```

Exemplo:

```
Group grp = new Group();
```

Region

A classe **Region** é a base de todos os componentes **UI** (User Interface), renderiza conteúdos **CSS** próprio do JavaFX.

As Regions podem organizar seus componentes internos por meio de layouts, para isto é preciso usar uma classe específica de layout como **StackPane**, **HBox**, **VBox**, **TilePane**, **FlowPane**, **BorderPane**, **GridPane**, or **AnchorPane**.

A sintaxe para criar um objeto do tipo **Region** é:

```
Region <nome do objeto> = new Region();
```

Exemplo:

```
Region reg = new Region();
```

ou

```
Pane painel = new Pane();
```

UI Components

Os componentes do tipo UI no JavaFx possuem conteúdo visual e comportamentos padrões para lidar com as ações do usuários.

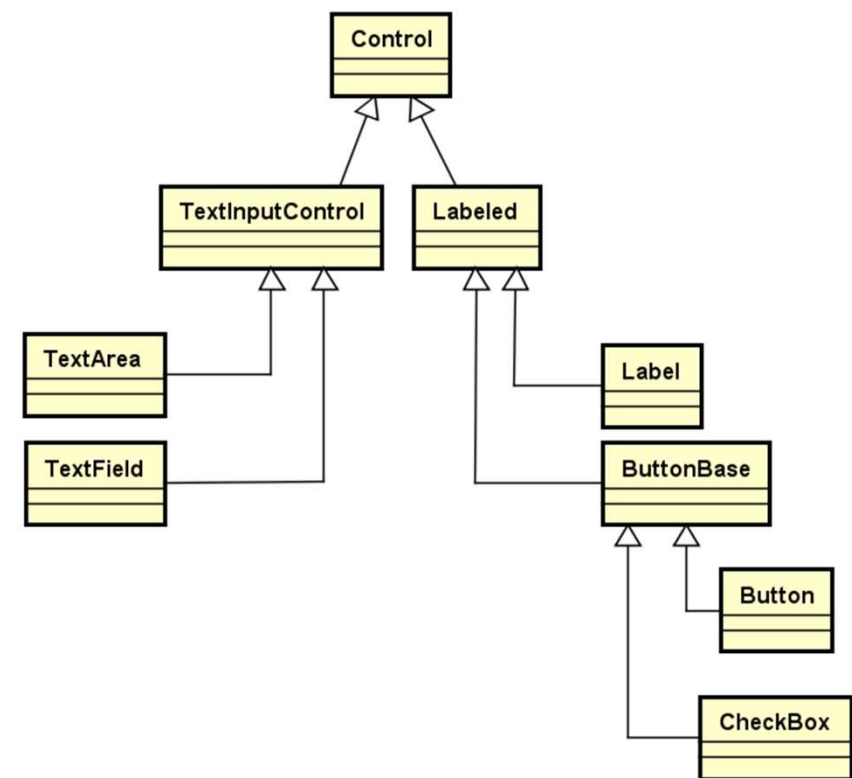
Existem diversos tipos de componentes do tipo UI para as mais variadas necessidades, porém nesta disciplina serão abordados apenas os tipos Label, TextField, TextArea, Button e CheckBox.

A sintaxe para criar um objeto do tipo Control é:

```
Label <nome do objeto> = new Label();
```

Exemplo:

```
Label hello = new Label("Hello World");
```



UI Components

Uma vez criado o objeto do tipo **Control**, basta inserí-lo no objeto do tipo **Pane** para que ele seja exibido na tela quando estiver dentro de cena.



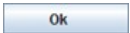

Para inserir, pega-se o objeto do tipo **Pane** e insere na **Collection** de **Nodes** por meio do método **getChildren()**

Exemplo:

```
<objPane>.getChildren()  
.add(<objeto control>);
```

```
public class ExemploApp1 extends Application {  
    @Override  
    public void start(Stage stage)  
        throws Exception {  
        Pane panel = new Pane();  
        Scene scn = new Scene(panel);  
        stage.setScene(scn);  
        Label hello = new Label("Hello");  
        panel.getChildren().add(hello);  
        stage.show();  
    }  
    public static void main(String[] args) {  
        Application.launch(args);  
    }  
}
```

UI Components

JavaFX UI Component	Descrição	Métodos Importantes
TextField 	<p>Cria uma caixa de texto que pode ser utilizada pelo usuário para digitar informações.</p> <p>Sintaxe : <code>TextField <objeto> = new TextField(<texto inicial>);</code></p> <p>Exemplo : <code>TextField nome = new TextField("Informe seu nome");</code></p>	<p>String getText() retorna um String com o texto digitado dentro da caixa.</p> <p>void setText(String t) define um texto dentro da caixa.</p> <p>boolean isEditable() retorna um boolean indicando se este objeto é editável</p> <p>void setEditable(boolean b) define se o objeto será editável ou não.</p>
Label 	<p>Cria um espaço onde pode ser exibido um texto curto e/ou uma imagem.</p> <p>Sintaxe : <code>Label <objeto> = new Label(<texto inicial>);</code></p> <p>Exemplo : <code>Label hello = new Label("hello world");</code></p>	<p>String getText() retorna um String com o conteúdo do rótulo.</p> <p>void setText(String t) coloca um novo texto no rótulo.</p>
Button 	<p>Implementa um botão o qual mostra um texto e/ou uma imagem. Sua utilização é para acionar uma determinada ação quando for pressionado.</p> <p>Sintaxe : <code>Button <objeto> = new Button(<texto inicial>);</code></p> <p>Exemplo : <code>Button name = new Button("Ok");</code></p>	<p>String getText() retorna um String com o conteúdo do botão.</p> <p>void setText(String t) define um novo texto para ser o conteúdo do botão.</p>
TextArea 	<p>Cria uma área de texto permitindo a visualização ou edição de um grande texto.</p> <p>Sintaxe : <code>TextArea <objeto> = new TextArea(<texto inicial>);</code></p> <p>Exemplo : <code>TextArea name = new TextArea("Texto de descricao");</code></p>	<p>String getText() retorna uma String com o texto digitado dentro da area.</p> <p>void setText(String t) define um texto dentro da area</p> <p>boolean isEditable() retorna um boolean indicando se este objeto é editável</p> <p>void setEditable(boolean b) define se o objeto será editável ou não.</p>