

# **Programação Orientada a Objetos**

---

## **JavaFX – Controlos II**

Prof. Rui César das Neves, Prof. José Cordeiro

Departamento de Sistemas e Informática

Escola Superior de Tecnologia de Setúbal – Instituto Politécnico de Setúbal

2014/2015

# Sumário

- **ComboBox vs ListView**
- **Tabs**
- **Accordion & TitledPane**
- **Menus**
- **Radio Buttons**

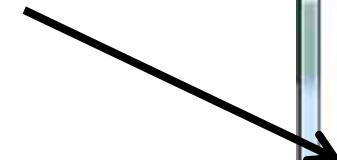
# JavaFX- Revendo o exemplo da ListView

## ☐ ListView –

Criação e  
Utilização

1. Criar um objeto  
do tipo  
**ListView**.

**ListView**



2. Criar uma  
**Observable  
List** de  
**Strings** e  
associá-la à  
**ListView**

```
Listview<String> listaNomes= new ListView<>();  
  
ObservableList<String> items=FXCollections.observableArrayList();  
listaNomes.setItems(items);
```

3. Adicionar linhas de texto à **ObservableList**  
para preencher a **ListView**

```
items.add(nome);
```

# JavaFX- Usando uma ComboBox

## □ ComboBox -

1. Criar um  
objeto do tipo  
**ComboBox**.

**ComboBox**

2. Criar uma  
**Observable**  
**List** de  
**Strings** e  
associá-la à  
**ComboBox**



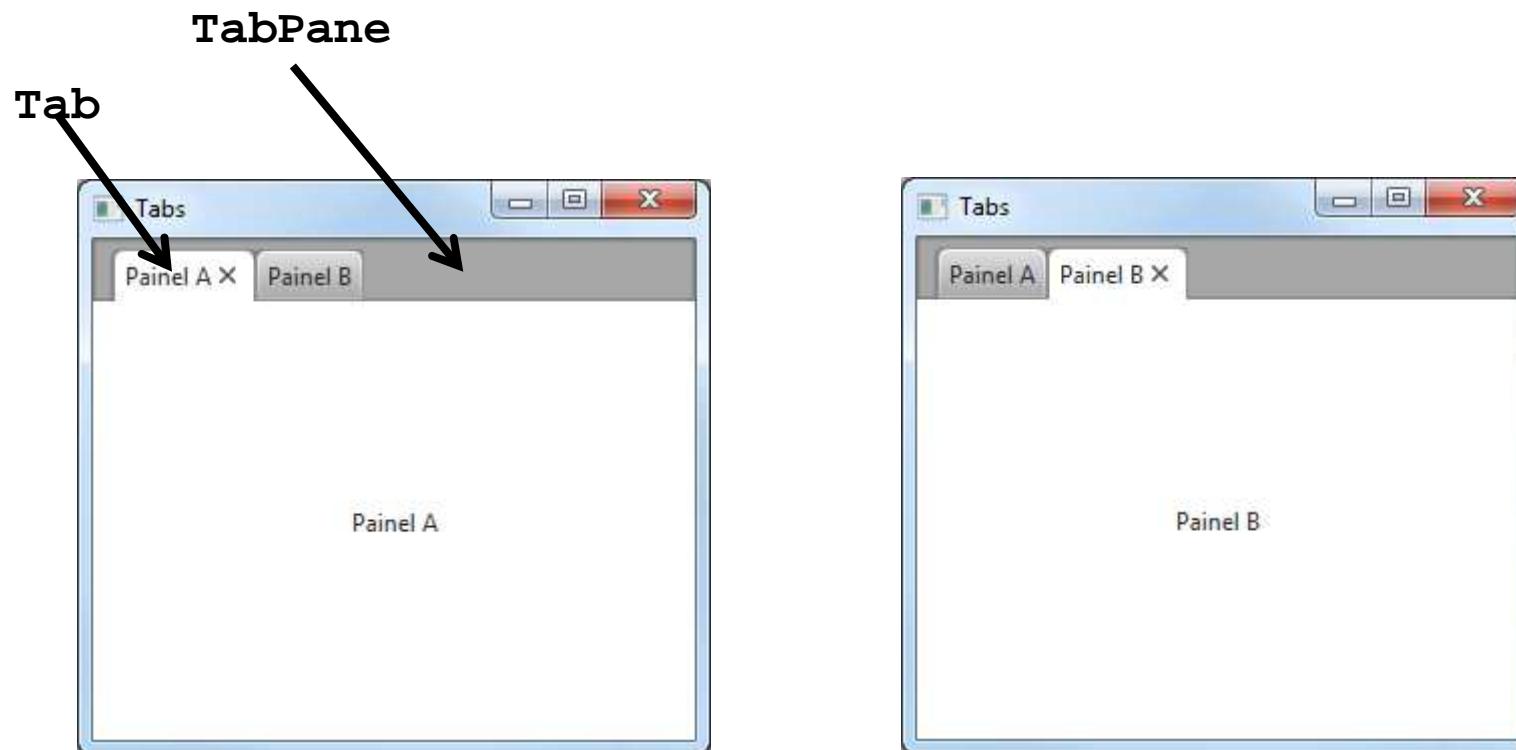
```
ComboBox<String> listaNomes= new ComboBox<>();  
  
ObservableList<String> items=FXCollections.observableArrayList();  
listaNomes.setItems(items);
```

3. Adicionar linhas de texto à  
**ObservableList** para preencher a  
**ComboBox**

```
items.add(nome);
```

# JavaFX- Exemplo - Tab

Uma forma simples de gerir a alternância entre vários painéis é através da utilização de um **TabPane** com vários **Tab** associados.



# JavaFX- Exemplo - Tab

Objetivo: Ter dois Painéis A e B, e alternar entre um e outro consoante o **Tab** selecionado.

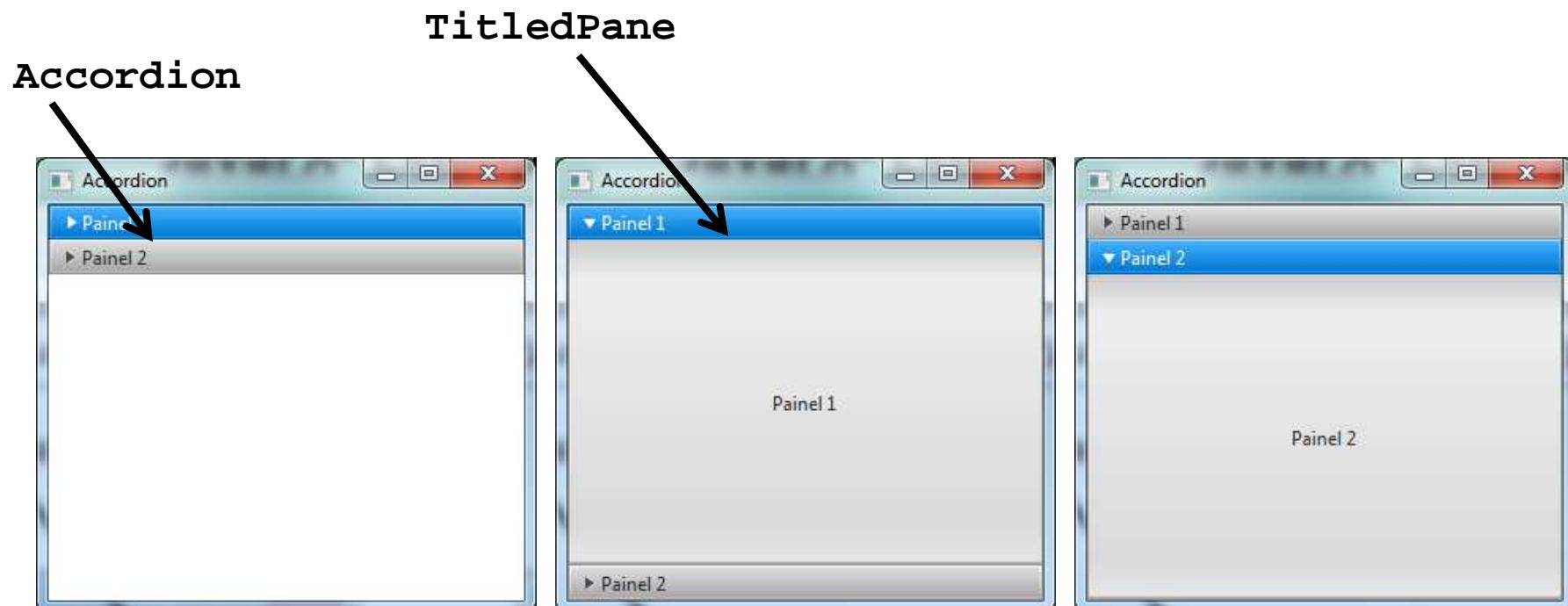
```
public class PainelComTab extends StackPane {  
  
    public PainelComTab() {  
        TabPane tabPane = new TabPane();  
        Tab tabA= new Tab("Painel A");  
        tabA.setContent(new PainelA());  
        Tab tabB= new Tab("Painel B");  
        tabB.setContent(new PainelB());  
        tabPane.getTabs().addAll(tabA,tabB);  
        this.getChildren().add(tabPane);  
    }  
  
    public class PainelA extends StackPane {  
  
        public PainelA() {  
            this.getChildren().add(new Label("Painel A"));  
        }  
    }  
}
```

1. Definir a **TabPane**
2. Definir objetos do tipo **Tab**.
3. Associar a cada **Tab**, o painel pretendido.  
Nota: **PainelA** e **PainelB** são classes derivadas de **Node**, normalmente painéis (**GridPane**, **HBox**, **VBox**, ...).
4. Associar cada **Tab** ao **TabPane**

Para indicar o tab selecionado:  
**tabPane.getSelectionModel().select(1);**

# JavaFX- Exemplo - Accordion

Outra forma simples de gerir a alternância entre vários painéis é através da utilização de um **Accordion** com vários **TitledPane** associados.



# JavaFX- Exemplo - Accordion

Objetivo: Ter dois Painéis 1 e 2, e alternar entre um e outro consoante a seleção.

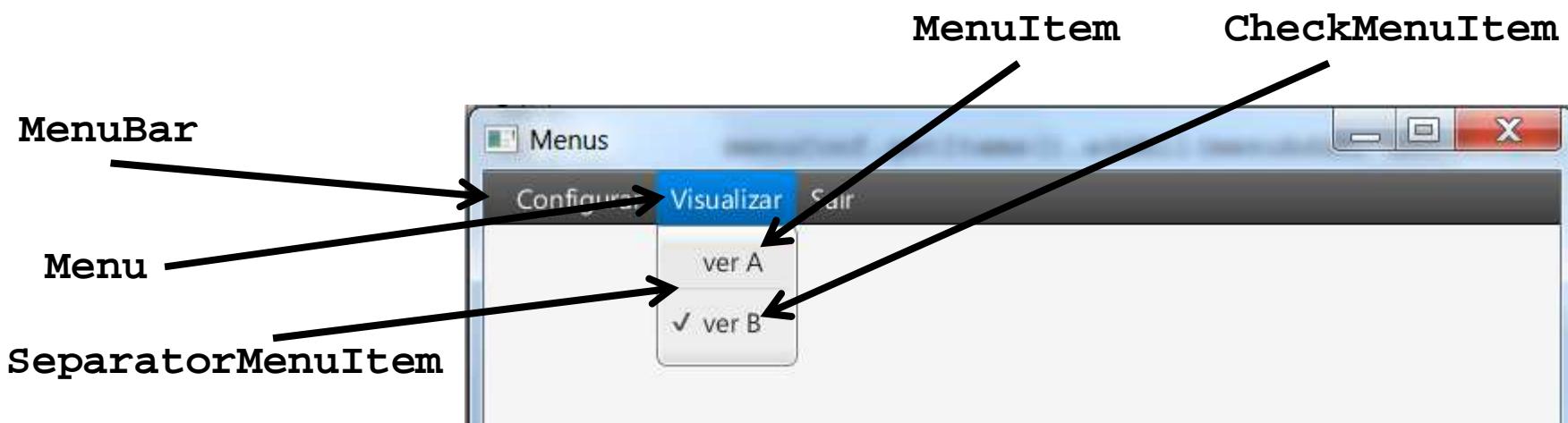
```
public class PainelComAccordion extends StackPane {  
  
    public PainelComAccordion() {  
        Accordion accordion = new Accordion();  
        TitledPane painel1 = new Painel1();  
        TitledPane painel2 = new Painel2();  
        accordion.getPanels().add(painel1);  
        accordion.getPanels().add(painel2);  
        this.getChildren().add(accordion);  
    }  
  
    public class Painel1 extends TitledPane {  
  
        public Painel1() {  
            this.setText("Painel 1");  
            this.setContent(new Label("Painel 1"));  
        }  
    }  
}
```

1. Definir o **Accordion**
2. Definir objetos do tipo **TitledPane**.  
Nota: É preciso definir o título (**setText**) e o conteúdo (**setContent**) dos **TitledPane**.
3. Associar a cada **TitledPane** ao **Accordion**.  
Para indicar o **TitledPane** selecionado:  
`accordion.setExpandedPane(painel2);`

# JavaFX- Exemplo - Menus

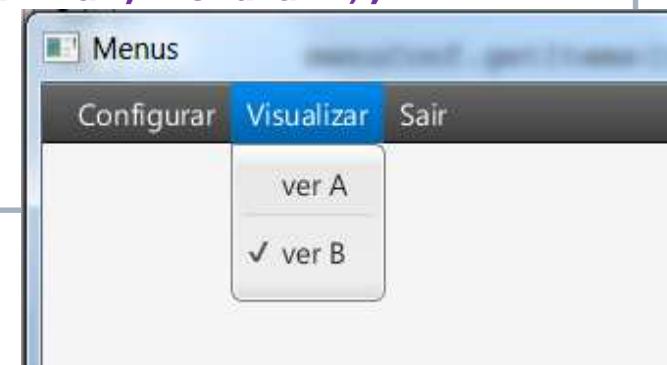
Em JavaFX podem usar-se as seguintes classes para definir menus:

- **MenuBar**
  - **Menu**
    - **MenuItem**
    - **CheckMenuItem**
    - **RadioMenuItem**
    - **SeparatorMenuItem**
  - **ContextMenu**



# JavaFX- Exemplo - Menus

```
public class PainelComMenu extends BorderPane {  
    public PainelComMenu() {  
        MenuBar menuBar = new MenuBar();  
  
        Menu menuConfigurar = new Menu("Configurar");  
        MenuItem menuConfigurarA = new MenuItem("Adicionar A");  
        MenuItem menuConfigurarB = new MenuItem("Adicionar B");  
        menuConfigurar.getItems().addAll(menuConfigurarA, menuConfigurarB);  
  
        Menu menuVisualizar = new Menu("Visualizar");  
        MenuItem menuVerA = new MenuItem("ver A");  
        CheckMenuItem menuVerB = new CheckMenuItem("ver B");  
        menuVisualizar.getItems().addAll(menuVerA,new SeparatorMenuItem(), menuVerB);  
  
        Menu menuSair = new Menu("Sair");  
        MenuItem menuFechar = new MenuItem("Fechar");  
        menuSair.getItems().add(menuFechar);  
  
        menuBar.getMenus().addAll(menuConfigurar, menuVisualizar, menuSair);  
        this.setTop(menuBar);  
    }  
}
```

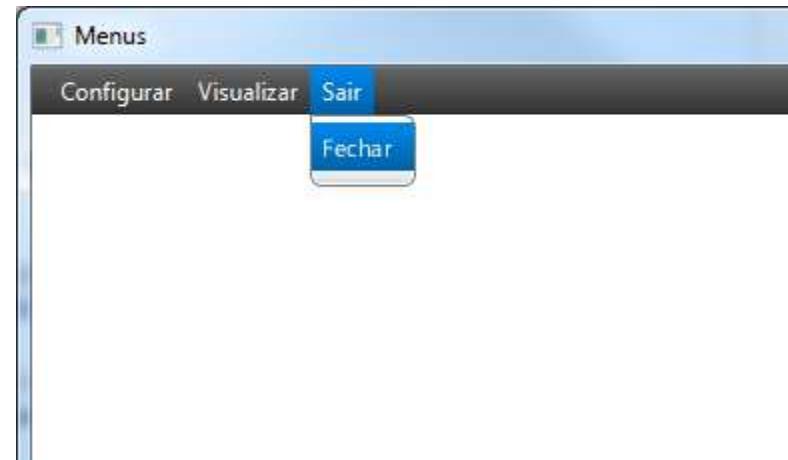


# JavaFX- Exemplo - Menus

Adicionar eventos ao **Menu**

Sair da aplicação através da opção de **Menu Sair**

```
menuFechar.setOnAction(e -> Platform.exit());
```



# JavaFX- Exemplo - Menus

**Objetivo:** Alternar entre painéis, consoante a opção do **Menu** selecionada.

**Solução:** Usar o layout **BorderPane** para a Janela. Definir dois painéis (**painelA** e **painelB**), consoante a opção do menu, colocar na zona central o **painelA** ou o **painelB**.

```
public class PainelComMenu extends BorderPane {  
  
    private final PanelA painelA;  
    private final PanelB painelB;  
  
    public PainelComMenu() {  
  
        MenuBar menuBar = new MenuBar();  
        ...  
        this.setTop(menuBar);  
  
        this.painelA = new PanelA();  
        this.painelB = new PanelB();  
    }  
}
```

1. Usar o layout **BorderPane** para a Janela.
2. Associar o menu ao topo
3. Definir dois painéis (**painelA** e **painelB**)

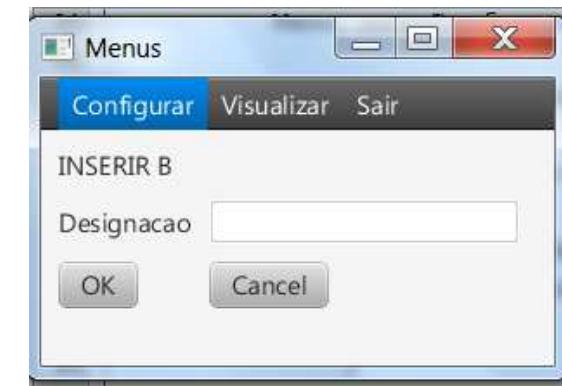
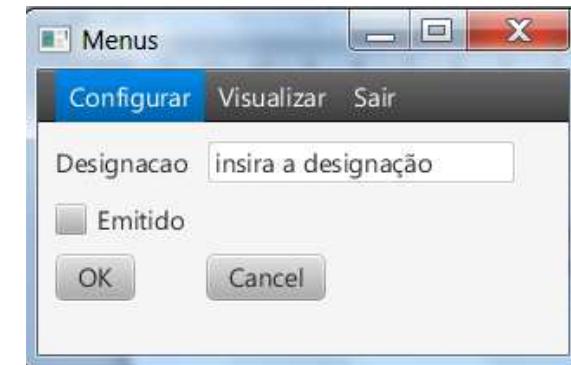


# JavaFX- Exemplo - Menus

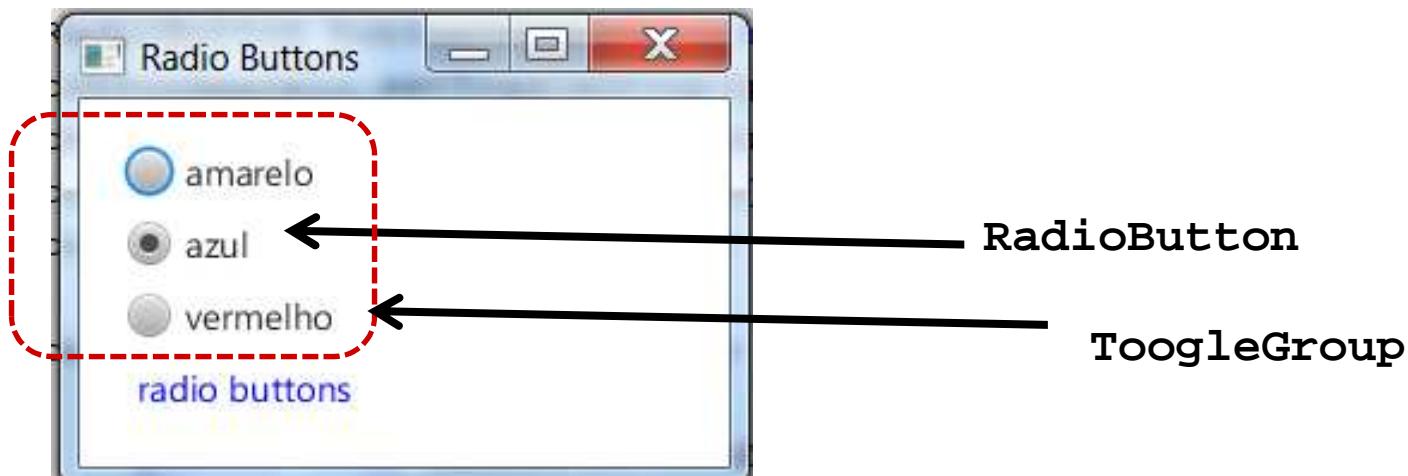
- Adicionar um evento ao **MenuItem menuConfigurarA**, que irá mostrar o **painelA**.
- Adicionar um evento ao **MenuItem menuConfigurarB**, que irá mostrar o **painelB**.

```
public PainelComMenu() {  
    ...  
    menuConfigurarA.setOnAction(e -> mostrarPainelA());  
    ...  
    menuConfigurarB.setOnAction(e -> mostrarPainelB());  
    ...  
}
```

```
public class PainelComMenu extends BorderPane {  
    ...  
    public final void mostrarPainelA() {  
        this.setCenter(painelA);  
    }  
  
    public final void mostrarPainelB() {  
        this.setCenter(painelB);  
    }  
}
```



## JavaFX- Exemplo – Radio Buttons



- Um **RadioButton** pode estar selecionado ou não selecionado.
- De forma a podermos ter um grupo de botões a trabalhar “sincronizados”: só um poderá estar selecionado de cada vez, (para implementarmos escolhas exclusivas) temos que definir um **ToggleGroup**.

# JavaFX- Exemplo – Radio Buttons



**Objetivo:** Ter uma aplicação que em função do **RadioButton** selecionado, muda a cor do texto apresentado.

Passo 1

1. Definir os **RadioButton**
2. Definir o **ToggleGroup**
3. Associar os **RadioButton** ao **ToggleGroup**
4. Criar o **Text**

```
RadioButton buttonAmarelo = new RadioButton("amarelo");
RadioButton buttonAzul = new RadioButton("azul");
RadioButton buttonVermelho = new RadioButton("vermelho");
final ToggleGroup group = new ToggleGroup();
buttonAmarelo.setToggleGroup(group);
buttonAzul.setToggleGroup(group);
buttonVermelho.setToggleGroup(group);

final Text texto = new Text("radio buttons");
texto.setFill(Color.BLUE);
buttonAzul.setSelected(true);
```

# JavaFX- Exemplo – Radio Buttons



## Passo 2

Definir as ações a realizar quando existe um evento nos botões. Uma ação para cada botão.

```
buttonAmarelo.setOnAction(e -> texto.setFill(Color.YELLOW));  
  
buttonAzul.setOnAction(e -> texto.setFill(Color.BLUE));  
  
buttonVermelho.setOnAction(e -> texto.setFill(Color.RED));
```

# Resumindo

- **ListView e ComboBox**
  - Permitem apresentar e manipular coleções de elementos.
  - A **ListView** permite seleção Múltipla ou Simples.
  - A **ComboBox** só permite a seleção de um elemento (Simples).
- **Tab**
  - Através de um controlo do tipo **TabPane**, podemos implementar facilmente a alternância entre vários painéis.
  - Associando um painel a um **Tab** e por sua vez os vários **Tabs** a um **TabPane**.
- **Accordion**
  - Através de um controlo do tipo **Accordion**, também é possível implementar a alternância entre vários painéis.
  - Criando os vários **TitledPane** e associando-os ao **Accordion**.
- **Menu**
  - É possível definir uma hierarquia de menus e submenus, usando as classes **MenuBar**, **Menu**, **MenuItem**, **CheckMenuItem**, **RadioMenuItem**, **SeparatorMenuItem**.
- **RadioButton**
  - Para implementar a escolha exclusiva através de **RadioButton**, temos que definir um **ToggleGroup** e associar cada **RadioButton** ao grupo criado.

# Leitura Complementar

Chapter 4 – Layouts and UI Controls Pgs 101 a 108

<http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/combo-box.htm#BABJCCIB>

[http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/menu\\_controls.htm#BABGHADI](http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/menu_controls.htm#BABGHADI)

<http://docs.oracle.com/javase/8/javafx/user-interface-tutorial/radio-button.htm#BABBJBDA>

