Desenvolvimento Desktop

Rafael Vieira Coelho

Introdução JavaFX

`Chapter 1`, *Stages, Scenes, and Layout*, starts from the essential basis of the JavaFX application—Stage, Scene, JavaFX Event Thread, and the layout graph concept.

`Chapter 2`, *Building Blocks – Shapes, Text, and Controls*, fills the window we created in the previous chapter with various building blocks provided by the JavaFX API.

`Chapter 3`, *Connecting Pieces – Binding*, explains that binding is a new method to greatly simplify communication between components.

`Chapter 4`, *FXML*, introduces FXML is a powerful tool for building a complex JavaFX UI and separating business logic from UI design. We will design an FXML application and try SceneBuilder—an FXML-based UI designer tool.

`Chapter 5`, *Animation*, demonstrates how to use the JavaFX Animation API to create various types of dynamic content.

`Chapter 6`, *Styling Applications with CSS*, outlines that JavaFX supports **Cascading Style Sheets** (**CSS**) to style applications similar to web pages.

Introdução JavaFX

`Chapter 7`, *Building a Dynamic UI*, reviews several JavaFX practices to build a modern adaptive UI.

`Chapter 8`, *Effects*, explains that effects such as a reflection or shadow are an essential part of modern applications.

`Chapter 9`, *Media and WebView*, showcases how JavaFX provides special controls dedicated to video, audio, and web content.

`Chapter 10`, *Advanced Controls and Charts*, reviews complex controls such as tables and charts.

`Chapter 11`, *Packaging with Java9 Jigsaw*, goes through the next step after writing an application—deploying it to end users. We will review JavaFX app building, packaging, and Jigsaw modularization options.

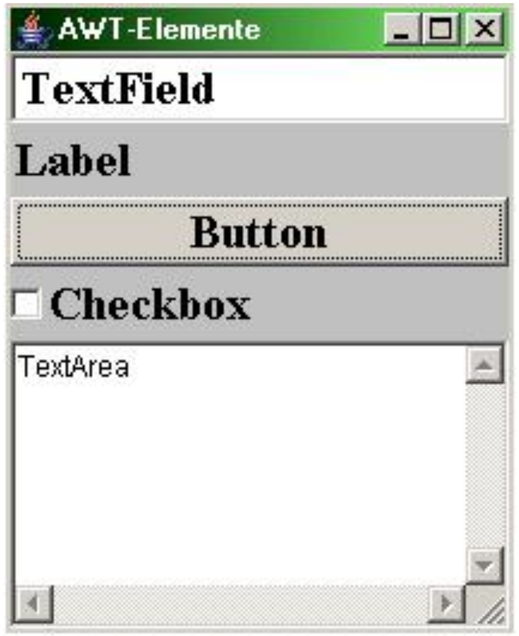
`Chapter 12`, *3D at a Glance*, teaches us to create several small 3D applications and study JavaFX's 3D capabilities.

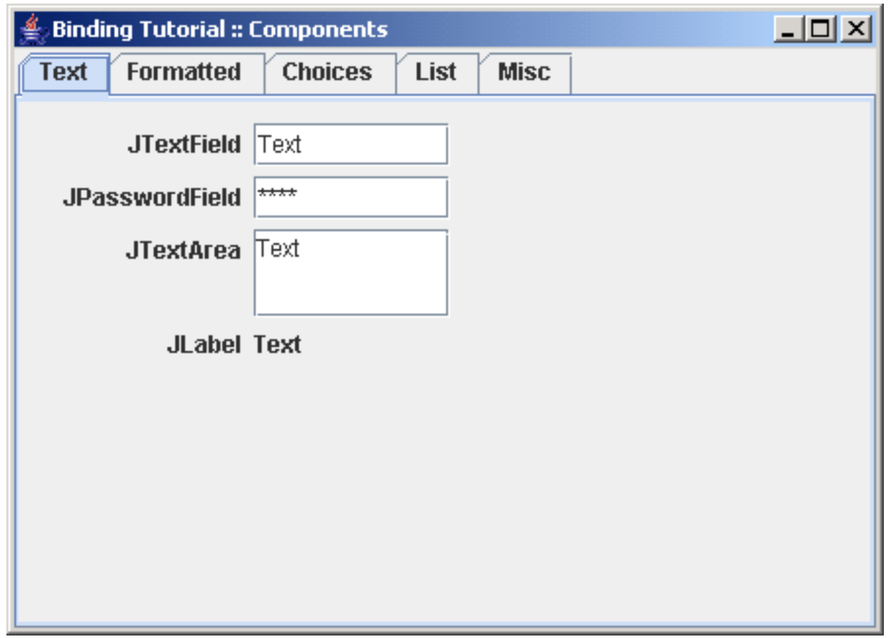
Roteiro

- **Conceitos de JavaFX**
- Gerenciadores de Layouts
- Exemplo de Relógio (Clock Demo)
- Criando um Projeto FX no IntelliJ

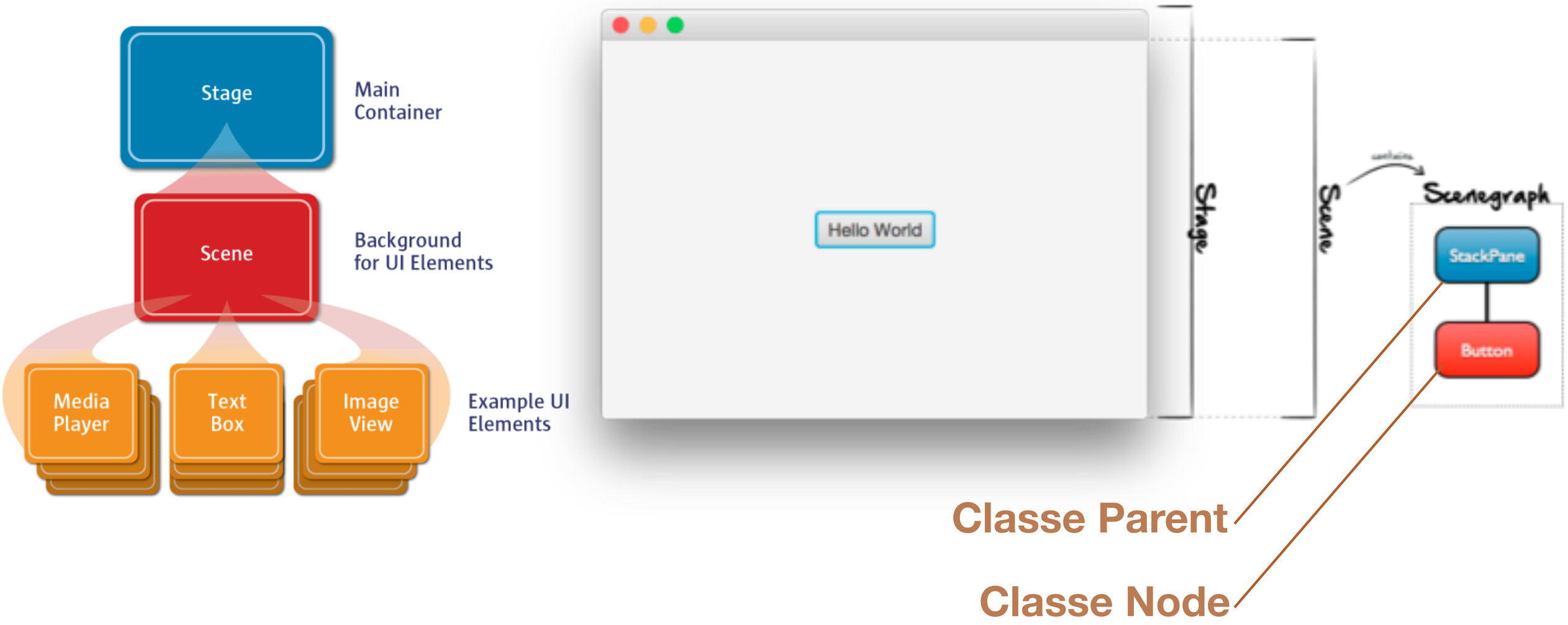
Aplicações Gráficas em Java: Histórico

1. Primeiramente tivemos o AWT (roda em cima do Java2D)
2. Depois o Swing (que ainda usava algumas coisas do AWT)
3. Agora o JavaFX (que não usa mais nada do Swing ou AWT e roda em cima do Prisma).





Hierarquia de Componentes no JavaFX



Classe Application

- Para ter uma aplicação em JavaFX, devemos criar uma classe filha de Application (javafx.application)

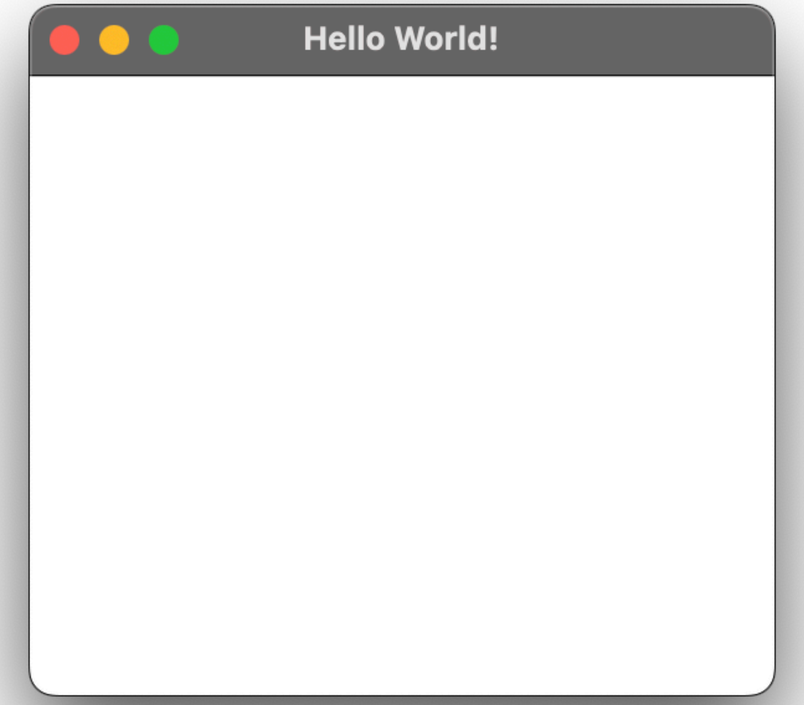
- Desta forma, podemos implementar 3 métodos da classe Application:

1) **public void init()**: tudo que deve ser feito antes de mostrar a janela para o usuário.

2) **public abstract void start(Stage)**: o que deve ser feito para criar a janela da aplicação.

3) **public void stop()**: após a finalização do programa. Tudo que deve ser feito depois de fechar totalmente a aplicação.

```java
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class FXApplicationDemo extends Application {

    @Override
    public void init() { System.out.println("Before"); }

    @Override
    public void start(Stage stage) {
        Scene scene = new Scene(new Group(), 300, 250);
        stage.setTitle("Hello World!");
        stage.setScene(scene);
        stage.show();
    }

    public void stop() { System.out.println("After"); }

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



Como Maximizar a Janela

```
25  public class StageFullScreen extends Application {
26
27      @Override
28      public void start(Stage primaryStage) {
29          Button btn = new Button();
30          btn.setText("Say 'Hello World'");
31          btn.setOnAction((ActionEvent event) -> {
32              System.out.println("Hello World!");
33          });
34
35          StackPane root = new StackPane();
36          root.getChildren().add(btn);
37
38          Scene scene = new Scene(root, v: 300, v1: 250);
39
40          primaryStage.setTitle("Hello World!");
41          primaryStage.setScene(scene);
42          primaryStage.setFullScreenExitHint("Exit code is Ctrl+B");
43          primaryStage.setFullScreenExitKeyCombination(KeyCombination.valueOf("Ctrl+B"));
44          primaryStage.setFullScreen(true);
45          primaryStage.show();
```

Como Fechar a Aplicação

- Para que uma aplicação em JavaFX seja fechada, todos os stages criados devem ser fechados.

- Uma forma de fazer isto explicitamente é chamando o método estático exit() da classe Platform.

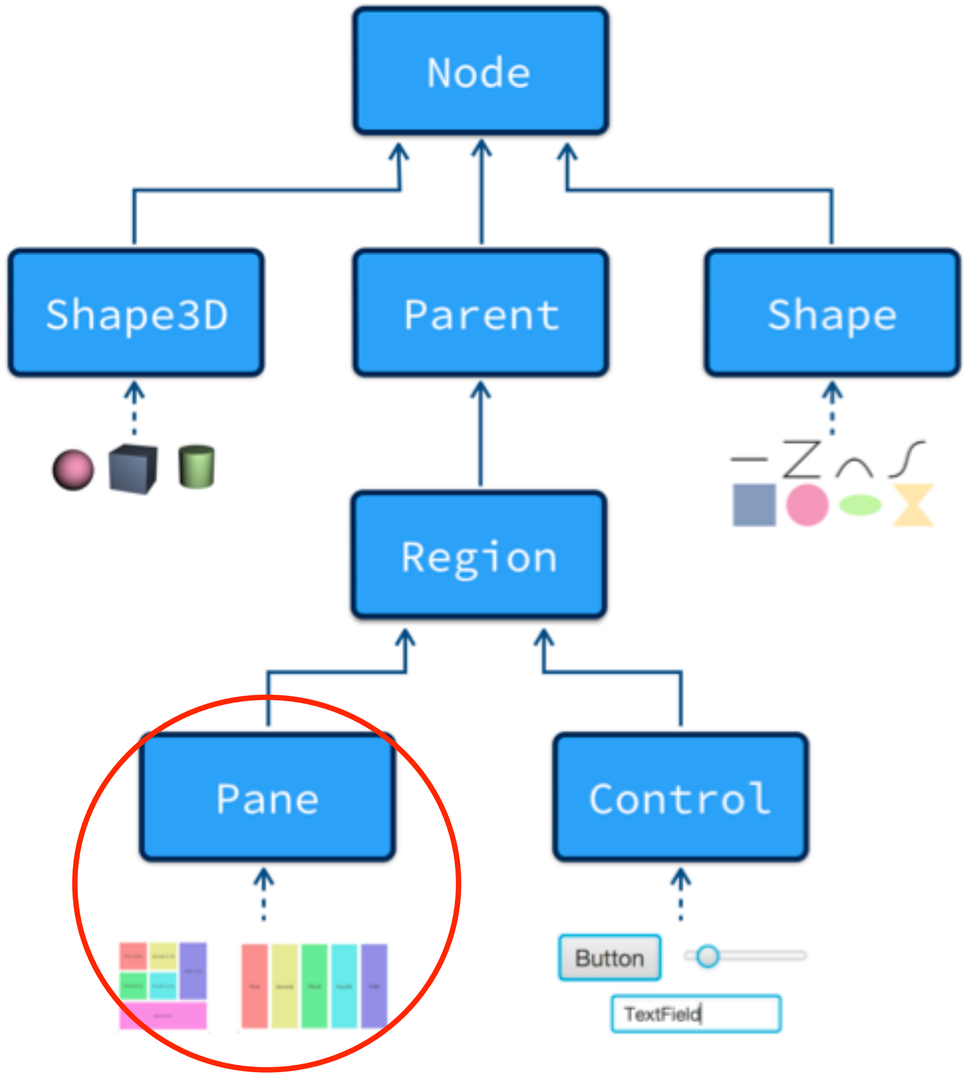
```
javafx.application.Platform.exit()
```

Roteiro

- Conceitos de JavaFX
- **Gerenciadores de Layouts**
- Exemplo de Relógio (Clock Demo)
- Criando um Projeto FX no IntelliJ

Tipos de Componentes no JavaFX

- ✓ Todos componentes do JavaFX são filhos da classe Node.
- ✓ Existem diferentes tipos de nodos.
- ✓ Todas formas geométricas e texto são filhas da classe Shape ou Shape3D.
- ✓ Painéis (Pane) e nodos de controle (Control) são filhas de Region.
- ✓ Todos nodos suportam tratamento de eventos e formatação CSS.



Gerenciadores de Layout

Estáticos:

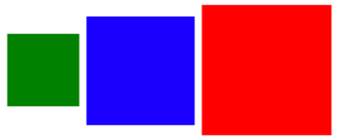
- Free Layout
- Group
- Region
- AnchorPane
- StackPane
- VBOX
- HBOX

Posicionais:

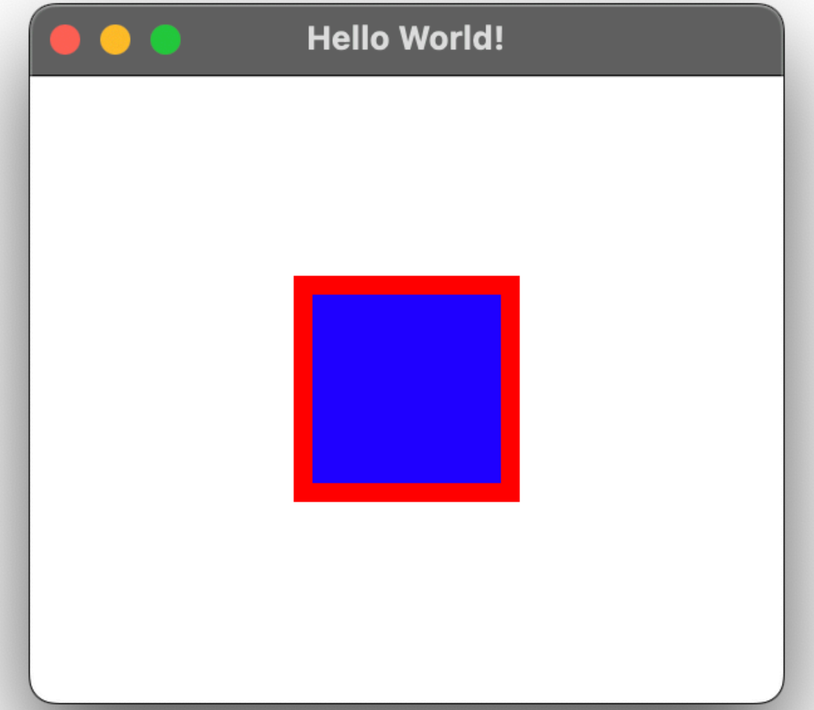
- FlowPane
- BorderPane
- TilePane
- GridPane

Exemplo de StackPane

Empilha os componentes na ordem que são adicionados, um em cima do outro.



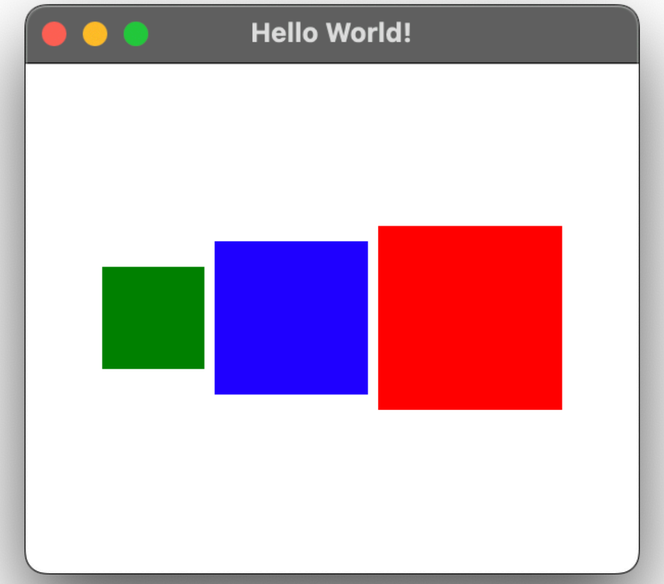
```java
public class StackPaneDemo extends Application {

    @Override
    public void start(Stage primaryStage) {
        Pane root = new StackPane();
        Rectangle red;
        root.getChildren().addAll(
                new Rectangle( v: 50,  v1: 50, Color.GREEN), // is behind blue and invisible
                new Rectangle( v: 75,  v1: 75, Color.BLUE),
                red = new Rectangle( v: 90,  v1: 90, Color.RED));
        red.toBack();

        Scene scene = new Scene(root,  v: 300,  v1: 250);

        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```



Exemplo de HBox

- Organiza horizontalmente os componentes na sequência que foram adicionados.

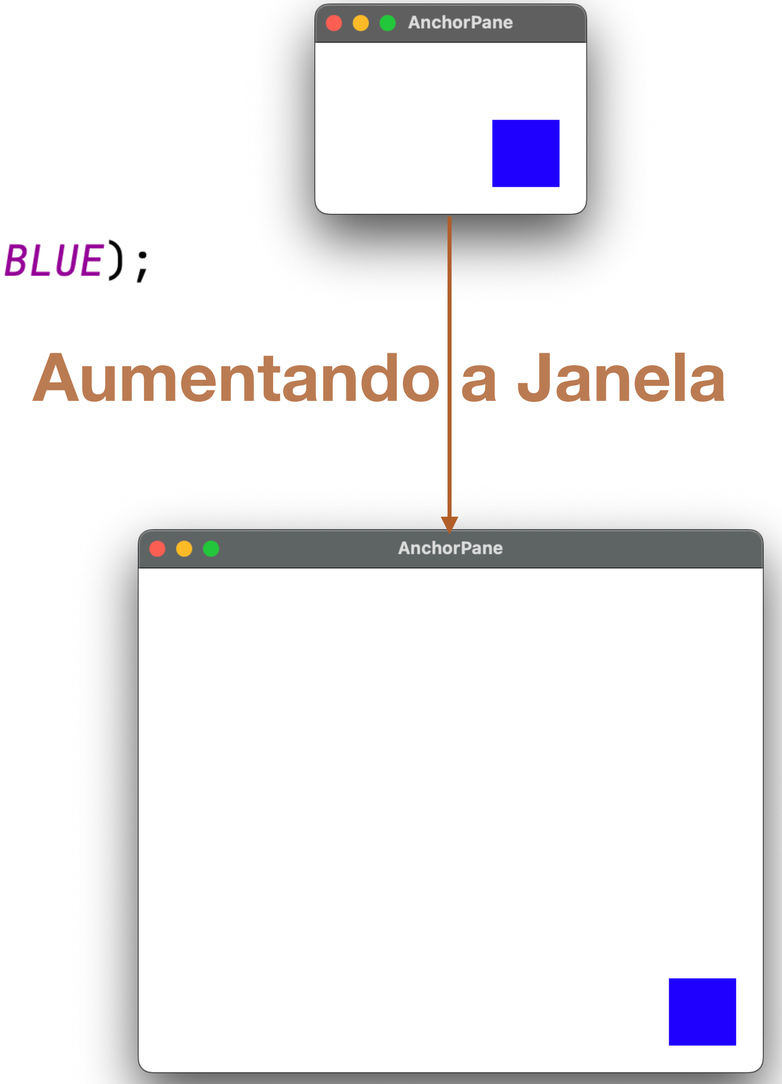
```java
public class HBoxDemo extends Application {

    @Override
    public void start(Stage primaryStage) {
        HBox hbox = new HBox(5);
        hbox.setAlignment(Pos.CENTER);
        hbox.getChildren().addAll(
                new Rectangle(50, 50, Color.GREEN),
                new Rectangle(75, 75, Color.BLUE),
                new Rectangle(90, 90, Color.RED));
        Scene scene = new Scene(hbox, 300, 250);

        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```



Exemplo de AnchorPane

- Ancora os componentes em posições específicas (cantos/lados).

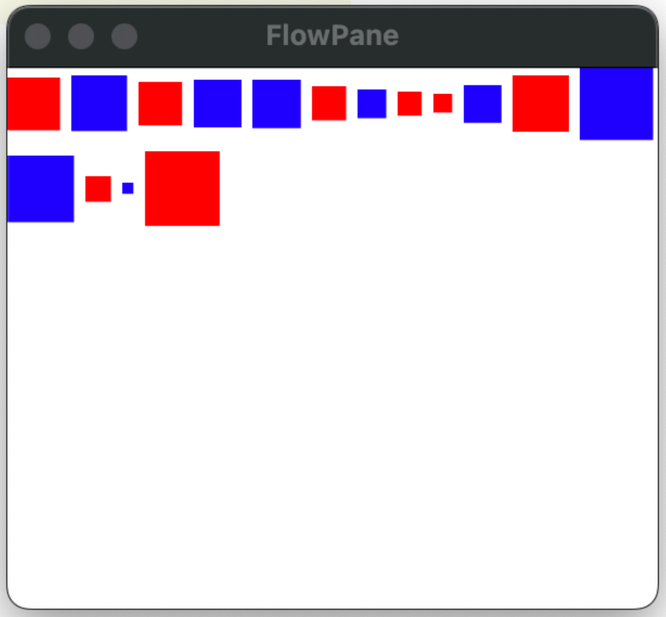
```java
23  public class AnchorPaneDemo extends Application {
24
25      @Override
26      public void start(Stage primaryStage) {
27
28          Rectangle rect = new Rectangle( v: 50,  v1: 50, Color.BLUE);
29
30          Pane root = new AnchorPane(rect);
31          AnchorPane.setRightAnchor(rect,  aDouble: 20.);
32          AnchorPane.setBottomAnchor(rect,  aDouble: 20.);
33
34          Scene scene = new Scene(root,  v: 150,  v1: 100);
35
36          primaryStage.setTitle("AnchorPane");
37          primaryStage.setScene(scene);
38          primaryStage.show();
39      }
```



Exemplo de FlowPane

- Adiciona lateralmente em um fluxo horizontal até o limite da janela. E segue abaixo.

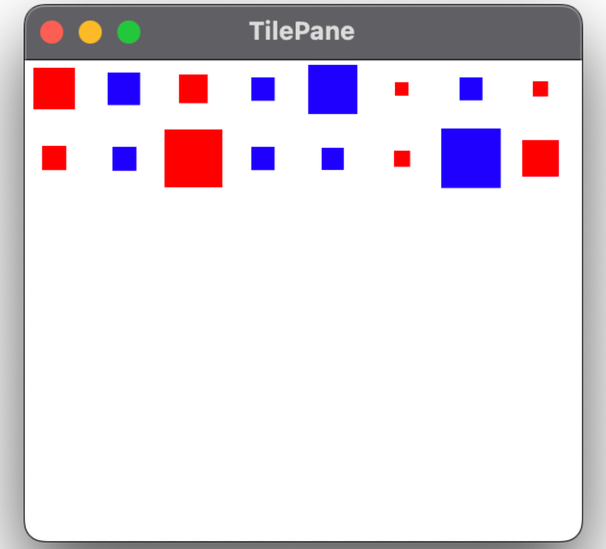
```java
public class FlowPaneDemo extends Application {

    @Override
    public void start(Stage primaryStage) {
        FlowPane root = new FlowPane(5, 5);

        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                double size = 5 + 30 * Math.random();
                Rectangle rect = new Rectangle(size, size, (i + j) % 2 == 0 ?
                        Color.RED : Color.BLUE);
                root.getChildren().add(rect);
            }

        }

        Scene scene = new Scene(root, 300, 250);

        primaryStage.setTitle(root.getClass().getSimpleName());
        primaryStage.setScene(scene);
        primaryStage.show();
```



Exemplo de TilePane

- Dispõe similarmente ao FlowPane, mas cada espaço para componente (tile) tem o mesmo tamanho.

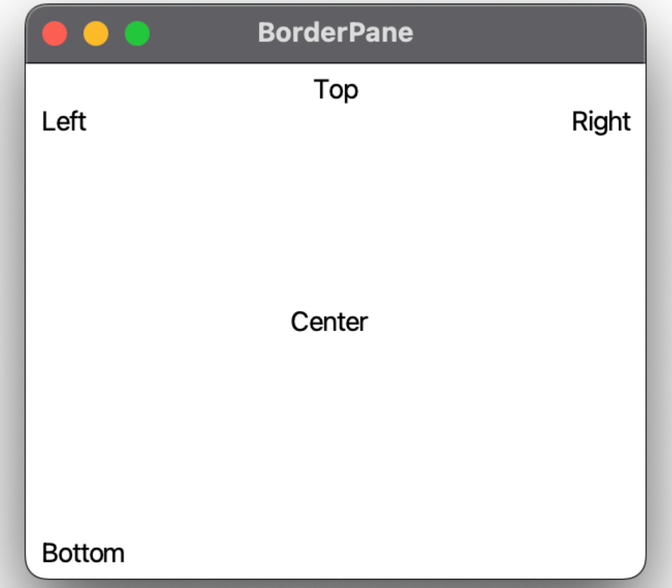
```java
public class TilePaneDemo extends Application {

    @Override
    public void start(Stage primaryStage) {
        TilePane root = new TilePane(5, 5);
        root.setPrefColumns(4);
        root.setPrefRows(4);

        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                double size = 5 + 30 * Math.random();
                Rectangle rect = new Rectangle(size, size, (i + j) % 2 == 0 ?
                        Color.RED : Color.BLUE);
                root.getChildren().add(rect);
            }
        }
        Scene scene = new Scene(root, 300, 250);

        primaryStage.setTitle(root.getClass().getSimpleName());
        primaryStage.setScene(scene);
        primaryStage.show();
```



Exemplo de BorderPane

```java
22  public class BorderPaneDemo extends Application {
23
24      @Override
25      public void start(Stage primaryStage) {
26          BorderPane root = new BorderPane();
27          root.setRight(new Text( s: "Right "));
28          root.setCenter(new Text( s: "Center"));
29          root.setBottom(new Text( s: " Bottom"));
30          root.setLeft(new Text( s: " Left"));
31
32          Text top = new Text( s: "Top");
33          root.setTop(top);
34          BorderPane.setAlignment(top, Pos.CENTER);
35          root.setStyle("-fx-padding: 5px");
36
37          Scene scene = new Scene(root, v: 300, v1: 250);
38          primaryStage.setTitle("BorderPane");
39          primaryStage.setScene(scene);
40          primaryStage.show();
41      }
```

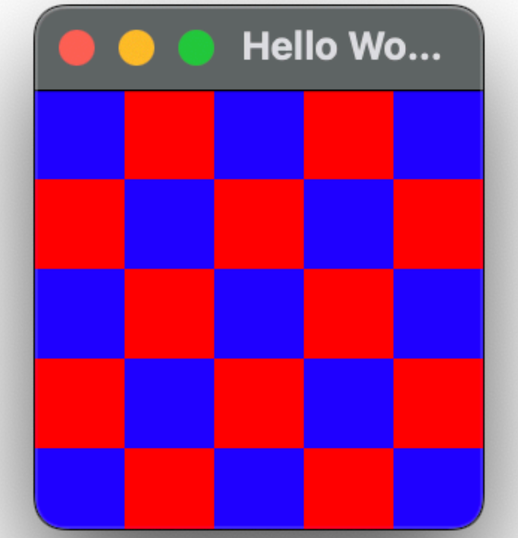
- Separa a janela em 5 partes (top, bottom, left, right, center) e permite a adição de componentes em posições específicas da tela.



Exemplo de GridPane

- Separa a janela em uma matriz (grid) no qual podemos adicionar elementos em uma determinada linha e coluna

```java
24  public class GridPaneDemo extends Application {
25      @Override
26      public void start(Stage primaryStage) {
27          GridPane root = new GridPane();
28
29          for (int i = 0; i < 5; i++) {
30              root.getColumnConstraints().add(new ColumnConstraints( v: 30));
31              root.getRowConstraints().add(new RowConstraints( v: 30));
32          }
33          for (int i = 0; i < 5; i++) {
34              for (int j = 0; j < 5; j++) {
35                  if ((i + j) % 2 == 0) {
36                      root.add(new Rectangle( v: 30,  v1: 30, Color.BLUE), i, j);
37                  } else {
38                      root.add(new Rectangle( v: 30,  v1: 30, Color.RED), i, j);
39                  }
40              }
41          }
42          Scene scene = new Scene(root,  v: 250,  v1: 250);
43          primaryStage.setTitle("Hello World!");
44          primaryStage.setScene(scene);
45          primaryStage.show();
```



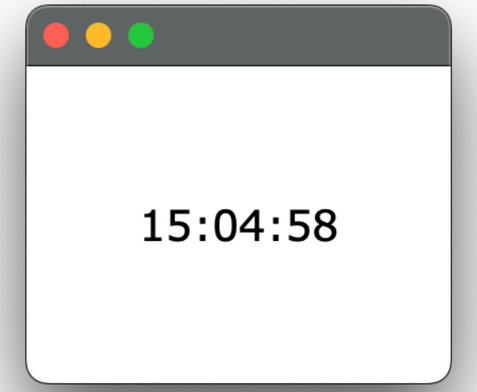
Roteiro

- Conceitos de JavaFX
- Gerenciadores de Layouts
- **Exemplo de Relógio (Clock Demo)**
- Criando um Projeto FX no IntelliJ

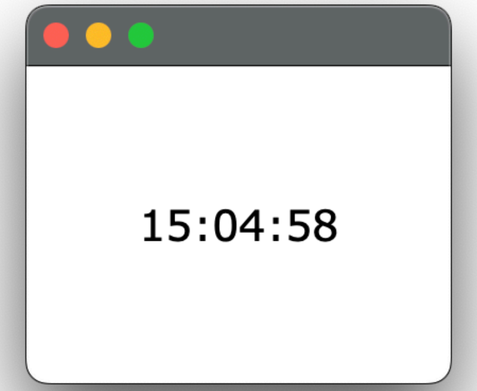
Executando Tarefas Simultâneas (threads)

- Para aplicações simples, não é necessário realizar tarefas assíncronas.
- Ou seja, o programa só faz uma tarefa por vez.
- Para fazer várias ações ao mesmo tempo, precisamos instanciar um novo objeto da classe **Thread** e invocar o método **start**.

```java
26  public class ClockOne extends Application {
27
28      private final Text txtTime = new Text();
29
30      private volatile boolean enough = false;
31
32      private Thread timer = new Thread(() -> {
33          while (!enough) {
34              final String time = LocalDateTime.now().format(
35                      DateTimeFormatter.ofPattern("HH:MM:ss"));
36              Platform.runLater(() -> {
37                  txtTime.setText(time);
38              });
39              try {
40                  Thread.sleep( millis: 1000);
41              } catch (InterruptedException ex) {
42              }
43          }
44      });
```



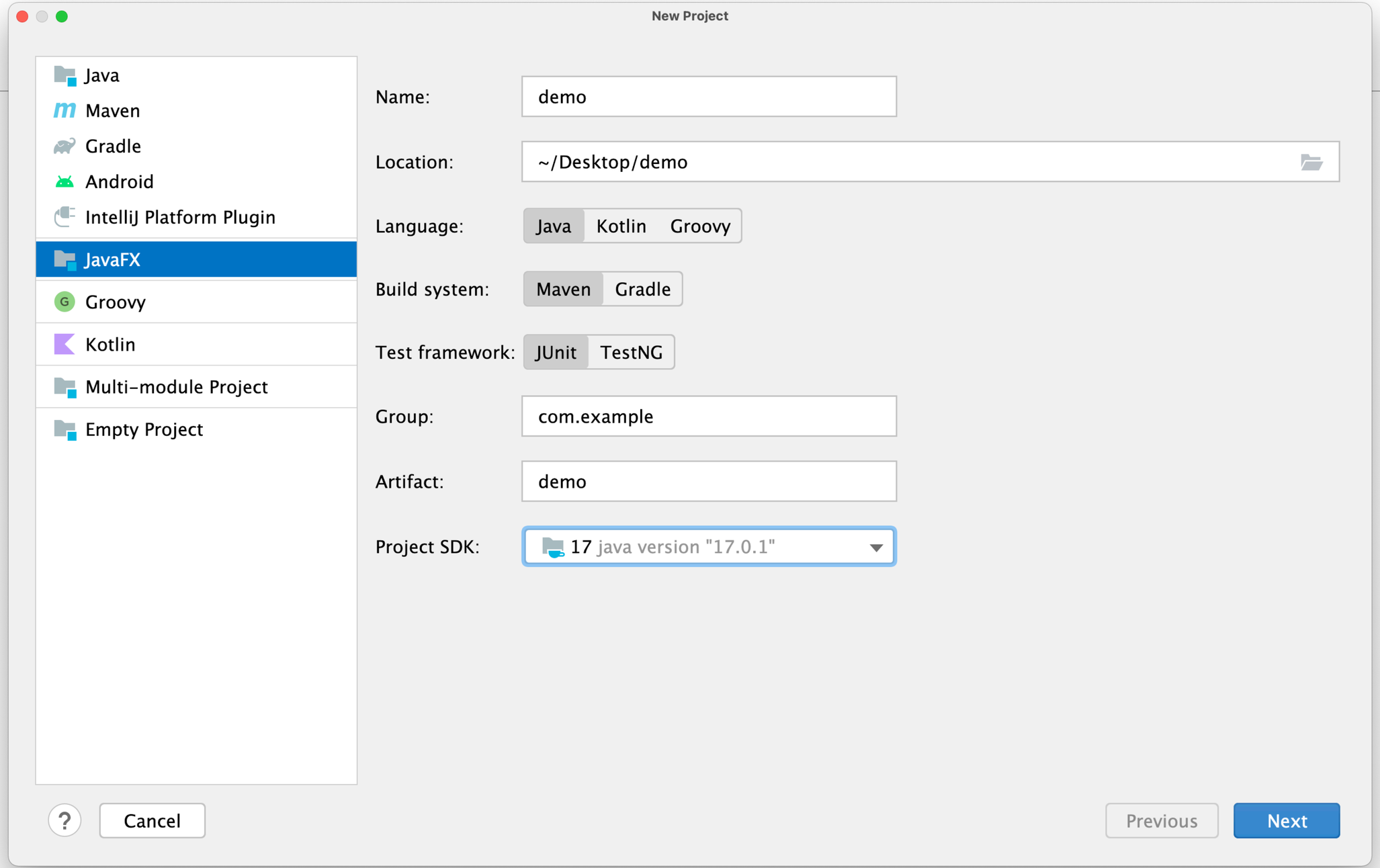
```java
46          @Override
47          public void start(Stage stage) {
48              BorderPane root = new BorderPane();
49              root.setCenter(txtTime);
50              txtTime.setFont(Font.font( s: "Verdana",  v: 20));
51              Scene scene = new Scene(root,  v: 200,  v1: 150);
52              stage.setScene(scene);
53              timer.start();
54              stage.show();
55          }
56
57          @Override
58          public void stop() {
59              enough = true;
60          }
```



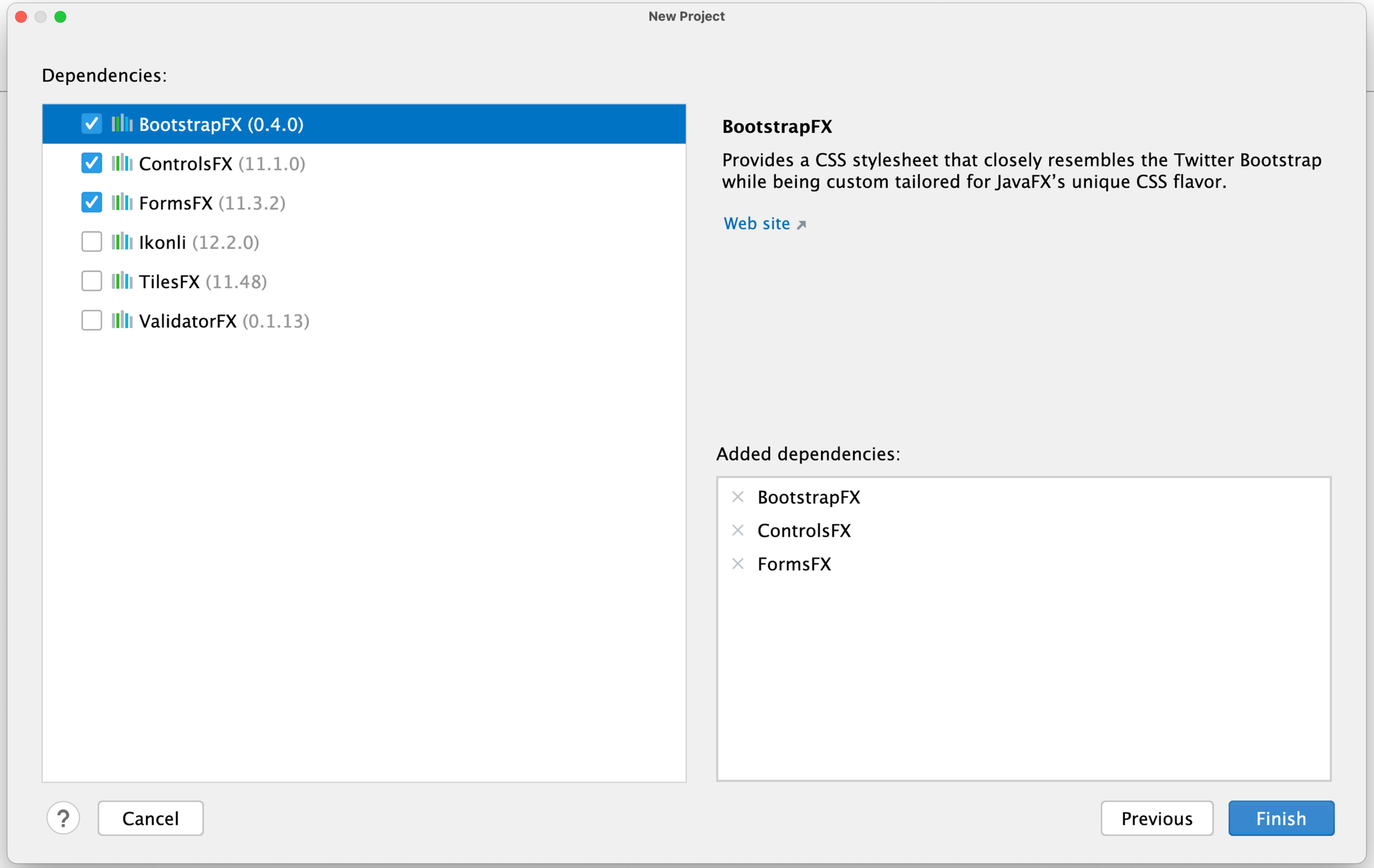
Roteiro

- Conceitos de JavaFX
- Gerenciadores de Layouts
- Exemplo de Relógio (Clock Demo)
- **Criando um Projeto FX no IntelliJ**

Criando um Projeto FX no IntelliJ



Criando um Projeto FX no IntelliJ



Criando um Projeto FX no IntelliJ

demo – HelloApplication.java

demo

Add Configuration...

Project

demo ~/Desktop/demo
- .idea
- src
  - main
    - java
      - com.example.demo
        - HelloApplication
        - HelloController
      - module-info.java
    - resources
      - com.example.demo
        - hello-view.fxml
  - demo.iml
  - pom.xml
- External Libraries
- Scratches and Consoles

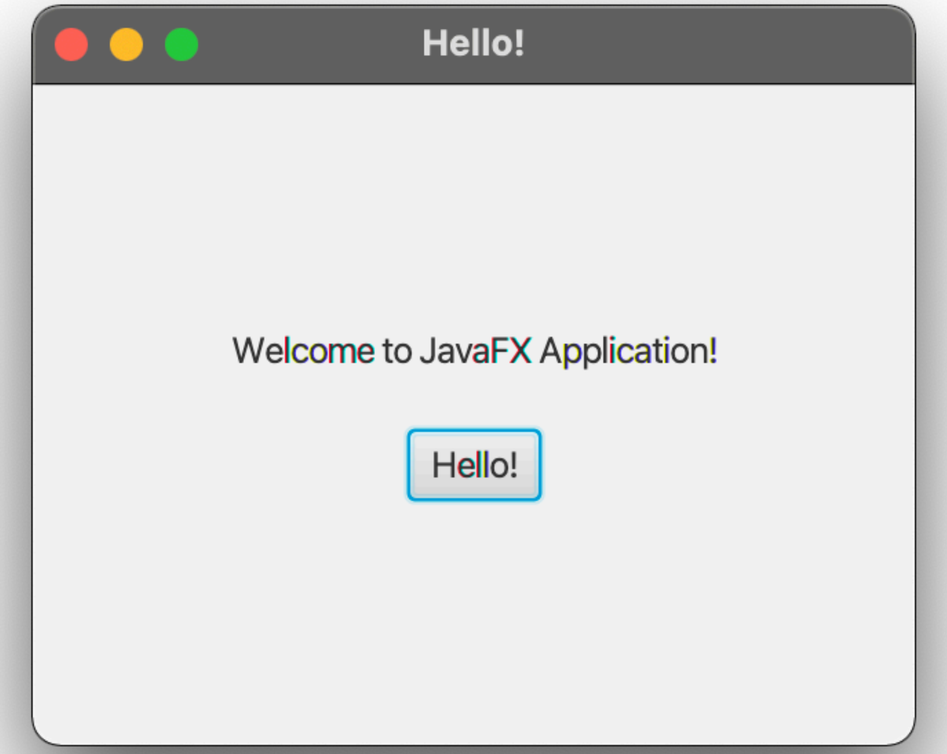
pom.xml (demo) | hello-view.fxml | HelloController.java | HelloApplication.java

```
1   package com.example.demo;
2
3   import ...
9
10  public class HelloApplication extends Application {
11      @Override
12      public void start(Stage stage) throws IOException {
13          FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class
14          Scene scene = new Scene(fxmlLoader.load(), v: 320, v1: 240);
15          stage.setTitle("Hello!");
16          stage.setScene(scene);
17          stage.show();
18      }
19
20      public static void main(String[] args) { launch(); }
23  }
```

Problems | Version Control | Terminal | TODO | Build | Dependencies | Event Log

1:1 LF UTF-8 4 spaces

Criando um Projeto FX no IntelliJ



Temos 3 Arquivos Importantes…

demo – HelloApplication.java

demo

Add Configuration…

Project

pom.xml (demo) | hello-view.fxml | HelloController.java | HelloApplication.java

- demo ~/Desktop/demo
  - .idea
  - src
    - main
      - java
        - com.example.demo
          - HelloApplication
          - HelloController
          - module-info.java
      - resources
        - com.example.demo
          - hello-view.fxml
  - demo.iml
  - pom.xml
- External Libraries
- Scratches and Consoles

```java
package com.example.demo;

import ...

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class
        Scene scene = new Scene(fxmlLoader.load(), v: 320, v1: 240);
        stage.setTitle("Hello!");
        stage.setScene(scene);
        stage.show();
    }

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

Problems | Version Control | Terminal | TODO | Build | Dependencies | Event Log

1:1 LF UTF-8 4 spaces

Links

https://www.oracle.com/java/technologies/downloads/#jdk17-windows