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.

Roteiro

- **Formas**
- Textos
- Controle Básico

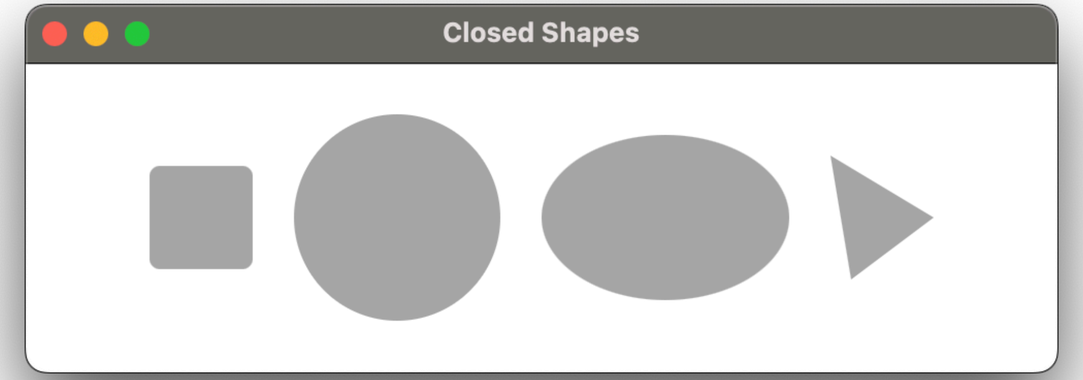
Shapes (Formas)

- São os componentes mais simples de se trabalhar em FX.
- Eles representam formas geométricas e podem ser usados para desenhar na janela.
- Pertencem ao pacote javafx.scene.shape
- Podemos subdividí-los em dois grupos: **linhas** e **formas fechadas** (Rectangle, Circle, Ellipse e Polygon)

Formas Fechadas

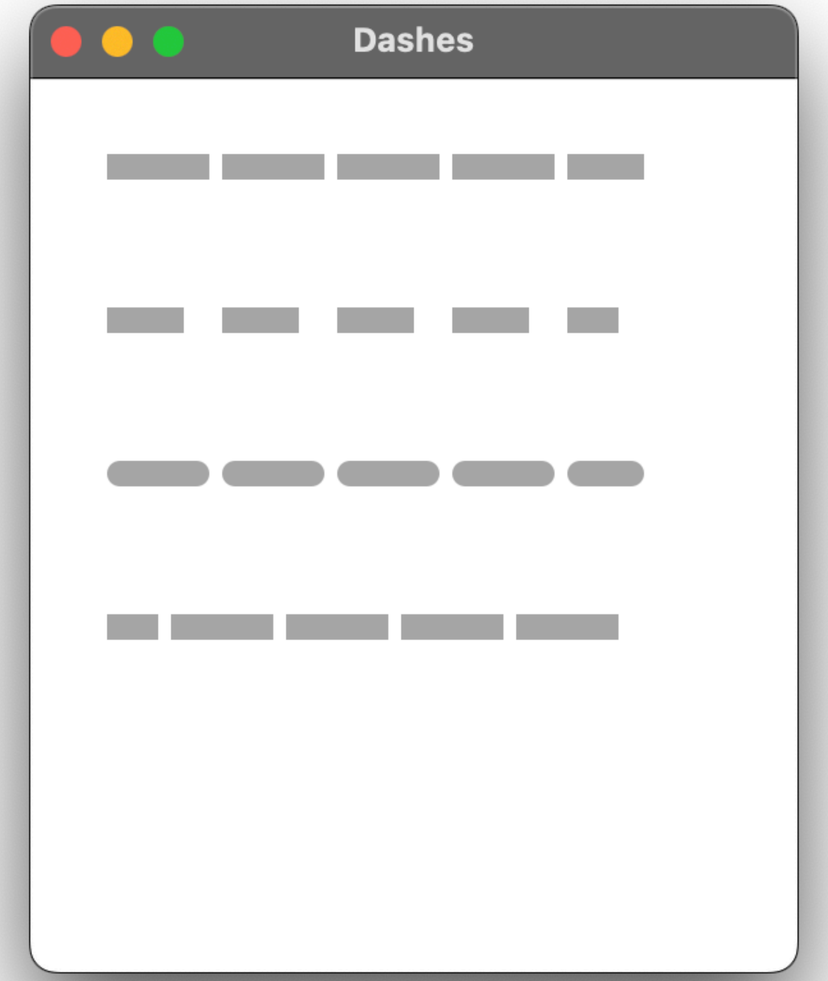
```java
public class ClosedShapes extends Application {

    @Override
    public void start(Stage primaryStage) {
        Rectangle rect = new Rectangle( v: 50,  v1: 50);
        rect.setArcHeight(10);
        rect.setArcWidth(10);
        rect.setFill(Color.DARKGREY);

        Circle circle = new Circle( v: 50);
        circle.setFill(Color.DARKGREY);

        Ellipse ellipse = new Ellipse();
        ellipse.setRadiusX(60);
        ellipse.setRadiusY(40);
        ellipse.setFill(Color.DARKGREY);

        Polygon polygon = new Polygon();
        polygon.setFill(Color.DARKGREY);
        polygon.getPoints().addAll(
                ...es: 0.0, 0.0,
                50.0, 30.0,
```

```java
        HBox hbox = new HBox( v: 20);
        hbox.setPadding(new Insets( v: 20));
        hbox.setAlignment(Pos.CENTER);
        hbox.getChildren().addAll(rect, circle, ellipse, polygon);

        primaryStage.setTitle("Closed Shapes");
        primaryStage.setScene(new Scene(hbox,  v: 500,  v1: 150));
        primaryStage.show();
    }

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

}
```



Linhas

```java
26          private VBox root = new VBox( v: 50);
27
28  @       private Line addLine() {
29              Line line = new Line( v: 50,  v1: 0,  v2: 250,  v3: 0);
30              line.setStrokeWidth(10);
31              line.setStroke(Color.DARKGRAY);
32              line.getStrokeDashArray().addAll( ...es: 30.0, 15.0);
33              root.getChildren().add(line);
34              return line;
35          }
36
37          @Override
38  @       public void start(Stage primaryStage) {
39              root.setPadding(new Insets( v: 30));
40
41              addLine().setStrokeLineCap(StrokeLineCap.SQUARE);
42              addLine().setStrokeLineCap(StrokeLineCap.BUTT);
43              addLine().setStrokeLineCap(StrokeLineCap.ROUND);
44
45              Line line = addLine();
46              line.setStrokeLineCap(StrokeLineCap.SQUARE);
47              line.setStrokeDashOffset(20);
```



Roteiro

- Formas
- **Textos**
- Controle Básico

Texto

```java
public class TextDemo extends Application {

    @Override
    public void start(Stage primaryStage) {
        Text txt = new Text( s: "Hello\nJavaFX!");
        txt.setFont(Font.font( s: "Courier New", FontWeight.BOLD, FontPosture.ITALIC, v: 30)
        Text txt1 = new Text( s: "Text1");
        txt1.setFont(Font.font( s: "Courier New", v: 15));
        Text txt2 = new Text( s: "Text2");
        txt2.setFont(Font.font( s: "Times New Roman", v: 20));
        Text txt3 = new Text( s: "Text3");
        txt3.setFont(Font.font( s: "Arial", v: 30));
        TextFlow textFlow = new TextFlow(txt1, txt2, txt3);

        VBox root = new VBox( v: 30, txt, textFlow);
        root.setPadding(new Insets( v: 20));
        Scene scene = new Scene(root, v: 300, v1: 250);
        primaryStage.setTitle("Hello World!");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```

Propriedades do Texto

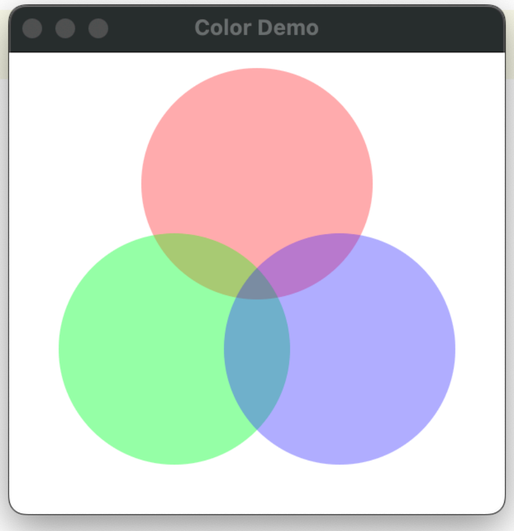
- Podemos chamar o método **setFill**() para definir como será preenchida a forma do texto.

- E podemos chamar o método **setStroke**() para definir como será feita a borda da forma contendo o texto.

- A classe **Color** provê acesso às cores básicas (Color.BLACK, etc.), métodos para definir nova combinação de cor (rgb(), hsb() e web()) e métodos de ajuste como: saturate(), darker() e deriveColor().

Cores

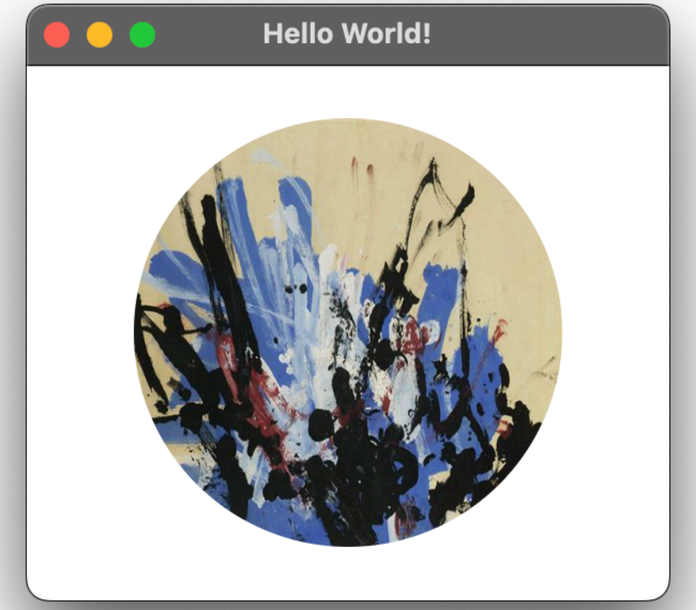
```java
lic class ColorsDemo extends Application {

    @Override
    public void start(Stage primaryStage) {

        Pane root = new Pane();
        root.getChildren().addAll(
                // RED, opacity 0.3
                new Circle( v: 150,  v1: 80,  v2: 70, Color.rgb( i: 255,  i1: 0,  i2: 0,  v: 0.3)),
                // GREEN, opacity 0.3
                new Circle( v: 100,  v1: 180,  v2: 70, Color.hsb( v: 120,  v1: 1.0,  v2: 1.0,  v3: 0.3))
                // BLUE, opacity 0.3
                new Circle( v: 200,  v1: 180,  v2: 70, Color.web( s: "0x0000FF",  v: 0.3))
        );

        Scene scene = new Scene(root,  v: 300,  v1: 280);

        primaryStage.setTitle("Color Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
```



Imagens

- Podemos adicionar imagens em formas através da classe **ImagePattern**

```
StackPane root = new StackPane();
root.getChildren().add(
    new Circle( v: 100,
        new ImagePattern(
            new Image( s: "https://upload.wikimedia.org/wikipedia/commons/3/3f/Chimpanzee_congo_painting.jpg"))));

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



Roteiro

- Formas
- Textos
- **Controle Básico**

Nodos de Controle

- Este tipo de nodo trata os eventos provenientes da interação do usuário com a nossa aplicação.

- Botões e evento de ação

```
Button btn = new Button();
btn.setText("Say 'Hello World'");
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.println("Hello World!");
    }
});
```

Comportamentos Padrões

- ENTER -> setDefaultButton com true
- ESC -> setCancelButton com true

```
btn.setDefaultButton(true);
//or
btn.setCancelButton(true);
```

Podemos Adicionar Nodos aos Botões

```
Button btn = new Button();
btn.setText("Say 'Hello World'");
btn.setGraphic(new Circle(10));
```

Outras Classes Derivadas de Botão

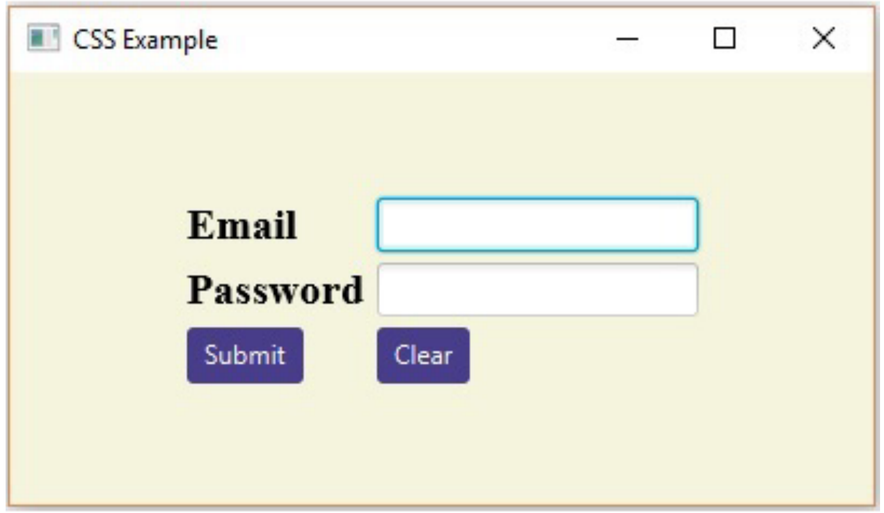
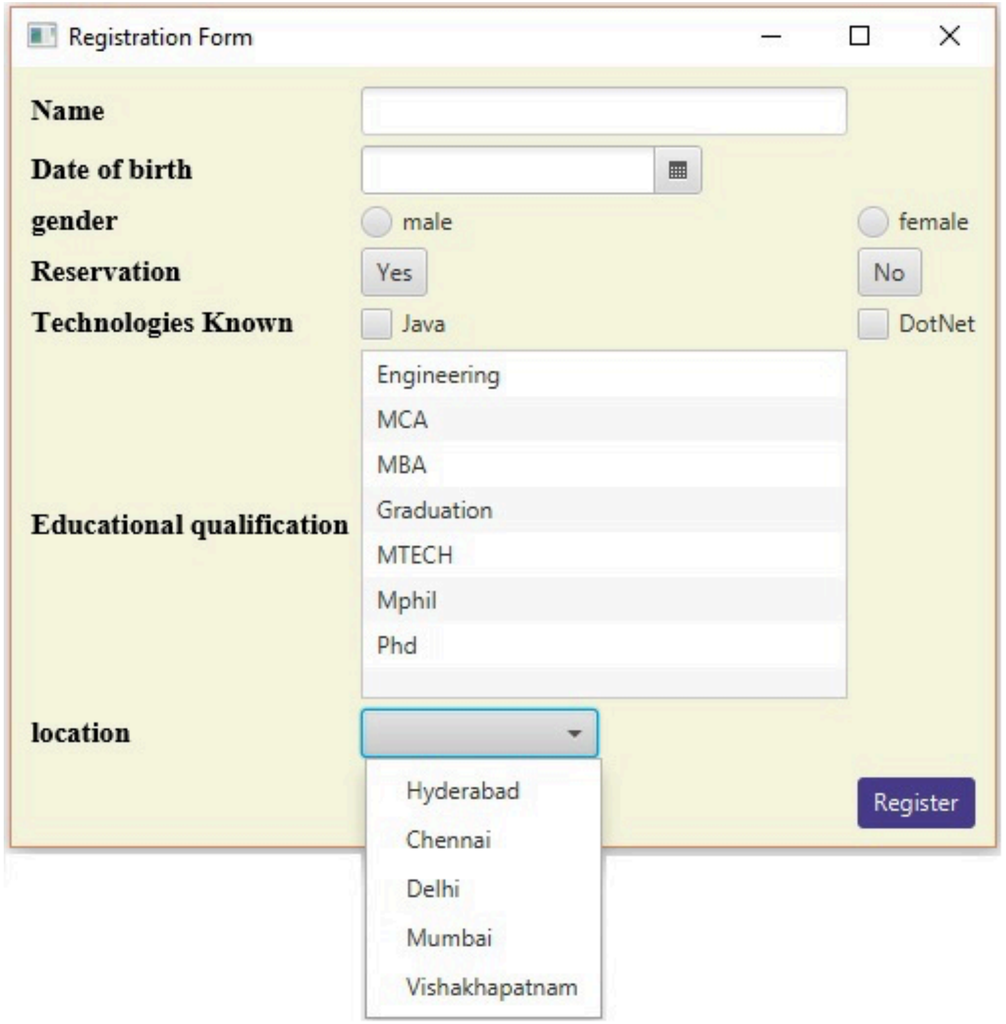
As quatro classes abaixo derivam da mesma classe mãe ButtonBase:

- **ToggleButton** deixa marcada se foi clicado ou não. Podemos verificar o status através do método isSelected().

- **CheckBox** tem o comportamento similar ao ToggleButton, mas é diferente visualmente. Devemos chamar o método getState().

- **Hyperlink** é um botão que fica marcado caso seja clicado.

- **MenuButton** faz parte da API Menu.

Exercícios

Crie um projeto e defina as duas janelas abaixo, mas sem usar arquivos FXML. No primeiro, pode usar o método setStyle() para definir como o nodo irá aplicar o CSS.