JavaFX UI Controls and Multimedia

CSE 219, Computer Science III

Stony Brook University

http://www.cs.stonybrook.edu/~cse219

Labeled class

- A label is a display area for a short text, a node, or both.
  - It is often used to label other controls (usually text fields).
  - Labels and buttons share many common properties: these common properties are defined in the Labeled class.

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.Labeled

-alignment: ObjectProperty<Pos>	Specifies the alignment of the text and node in the labeled.
-contentDisplay: ObjectProperty<ContentDisplay>	Specifies the position of the node relative to the text using the constants TOP, BOTTOM, LEFT, and RIGHT defined in ContentDisplay.
-graphic: ObjectProperty<Node>	A graphic for the labeled.
-graphicTextGap: DoubleProperty	The gap between the graphic and the text.
-textFill: ObjectProperty<Paint>	The paint used to fill the text.
-text: StringProperty	A text for the labeled.
-underline: BooleanProperty	Whether text should be underlined.
-wrapText: BooleanProperty	Whether text should be wrapped if the text exceeds the width.
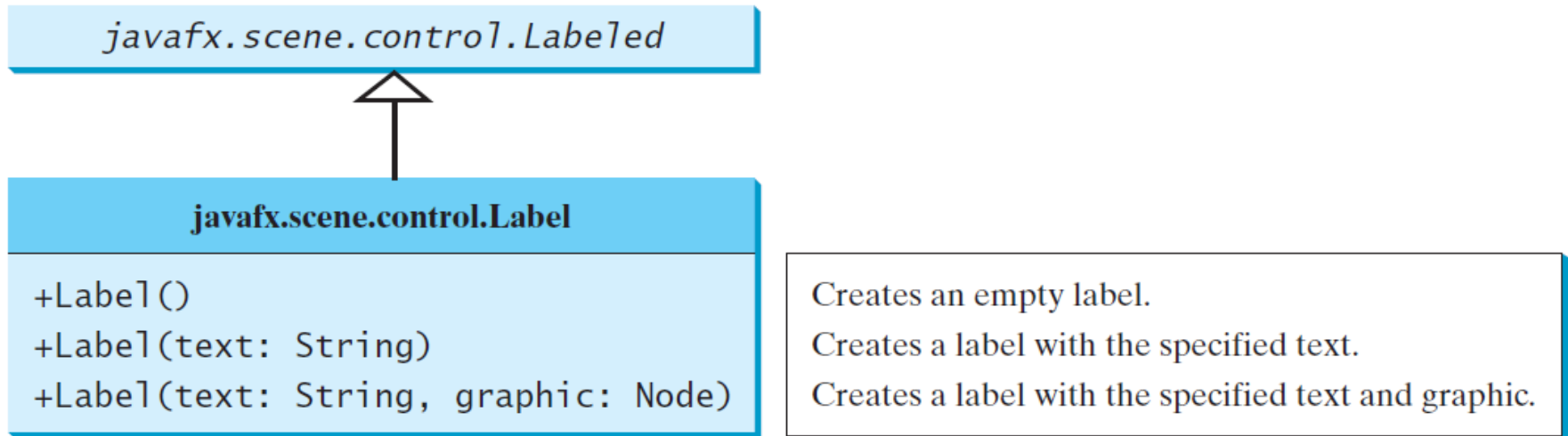
2

Label class

```
javafx.scene.control.Labeled
```

```
javafx.scene.control.Label

+Label()
+Label(text: String)
+Label(text: String, graphic: Node)
```

Creates an empty label.

Creates a label with the specified text.

Creates a label with the specified text and graphic.

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.ContentDisplay;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.Ellipse;

public class LabelWithGraphic extends Application {
    @Override
    public void start(Stage primaryStage) {
        ImageView us = new ImageView(new Image("us.jpg"));
        Label lb1 = new Label("US\n50 States", us);
        lb1.setStyle("-fx-border-color: green; -fx-border-width: 2");
        lb1.setContentDisplay(ContentDisplay.BOTTOM);
        lb1.setTextFill(Color.RED);
        Label lb2 = new Label("Circle", new Circle(50, 50, 25));
        lb2.setContentDisplay(ContentDisplay.TOP);
        lb2.setTextFill(Color.ORANGE);
        Label lb3 = new Label("Retangle", new Rectangle(10, 10, 50, 25));
        lb3.setContentDisplay(ContentDisplay.RIGHT);
        Label lb4 = new Label("Ellipse", new Ellipse(50, 50, 50, 25));
        lb4.setContentDisplay(ContentDisplay.LEFT);
```

(c) Paul Fodor and Pearson Inc.

```java
        Ellipse ellipse = new Ellipse(50, 50, 50, 25);
        ellipse.setStroke(Color.GREEN);
        ellipse.setFill(Color.WHITE);
        StackPane stackPane = new StackPane();
        stackPane.getChildren().addAll(ellipse, new Label("JavaFX"));
        Label lb5 = new Label("A pane inside a label", stackPane);
        lb5.setContentDisplay(ContentDisplay.BOTTOM);

        HBox pane = new HBox(20);
        pane.getChildren().addAll(lb1, lb2, lb3, lb4, lb5);

        Scene scene = new Scene(pane, 700, 150);
        primaryStage.setTitle("LabelWithGraphic");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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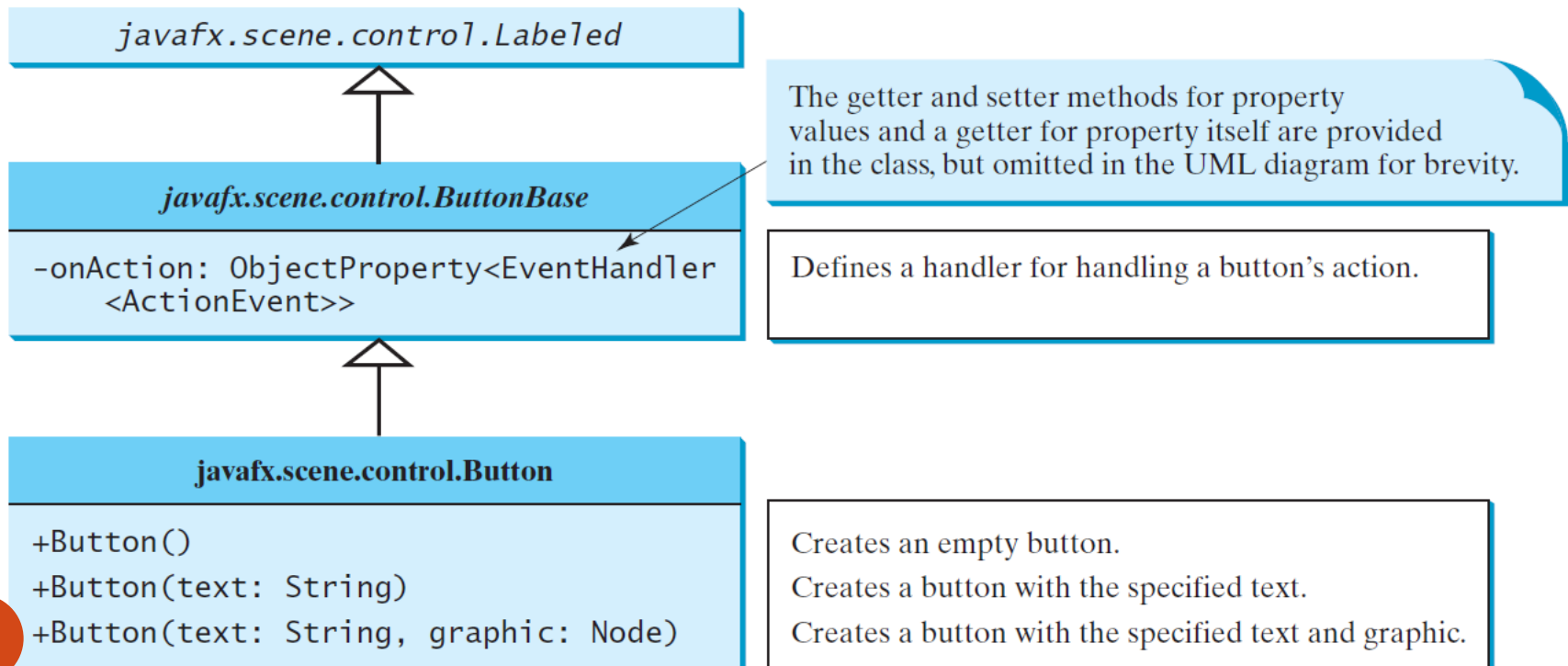
ButtonBase and Button

- A button is a control that triggers an action event when clicked.
- JavaFX provides regular buttons, toggle buttons, check box buttons, and radio buttons.
- The common features of these buttons are defined in ButtonBase and Labeled classes.

```
javafx.scene.control.Labeled
```

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

```
javafx.scene.control.ButtonBase

-onAction: ObjectProperty<EventHandler
    <ActionEvent>>
```

Defines a handler for handling a button's action.

```
javafx.scene.control.Button

+Button()
+Button(text: String)
+Button(text: String, graphic: Node)
```

Creates an empty button.
Creates a button with the specified text.
Creates a button with the specified text and graphic.

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.scene.control.Button;
import javafx.scene.image.ImageView;
import javafx.geometry.Pos;

public class ButtonDemo extends Application {
    @Override
    public void start(Stage primaryStage) {
        Scene scene = new Scene(getPane(), 450, 200);
        primaryStage.setTitle("ButtonDemo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }
    protected Text text = new Text(50, 50, "JavaFX Programming");
    protected BorderPane getPane() {
        HBox paneForButtons = new HBox(20);
        Button btLeft = new Button("Left", new ImageView("image/left.gif"));
        Button btRight = new Button("Right", new ImageView("image/right.gif"));
        paneForButtons.getChildren().addAll(btLeft, btRight);
        paneForButtons.setAlignment(Pos.CENTER);
        paneForButtons.setStyle("-fx-border-color: green");
        BorderPane pane = new BorderPane();
        pane.setBottom(paneForButtons);
```

7
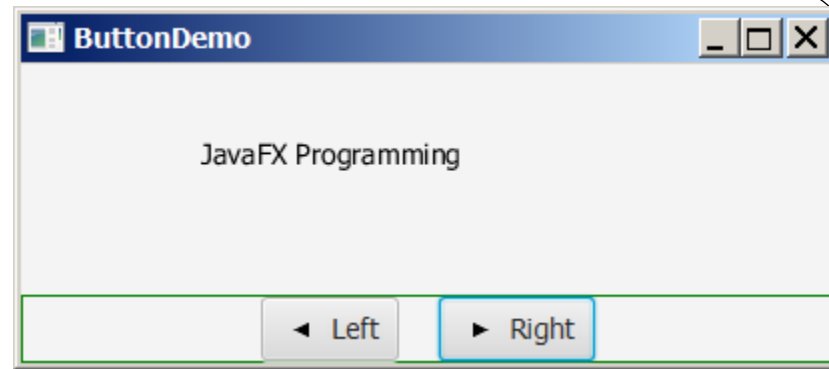
```
        Pane paneForText = new Pane();
        paneForText.getChildren().add(text);
        pane.setCenter(paneForText);

        btLeft.setOnAction(e -> text.setX(text.getX() - 10));
        btRight.setOnAction(e -> text.setX(text.getX() + 10));

        return pane;
    }

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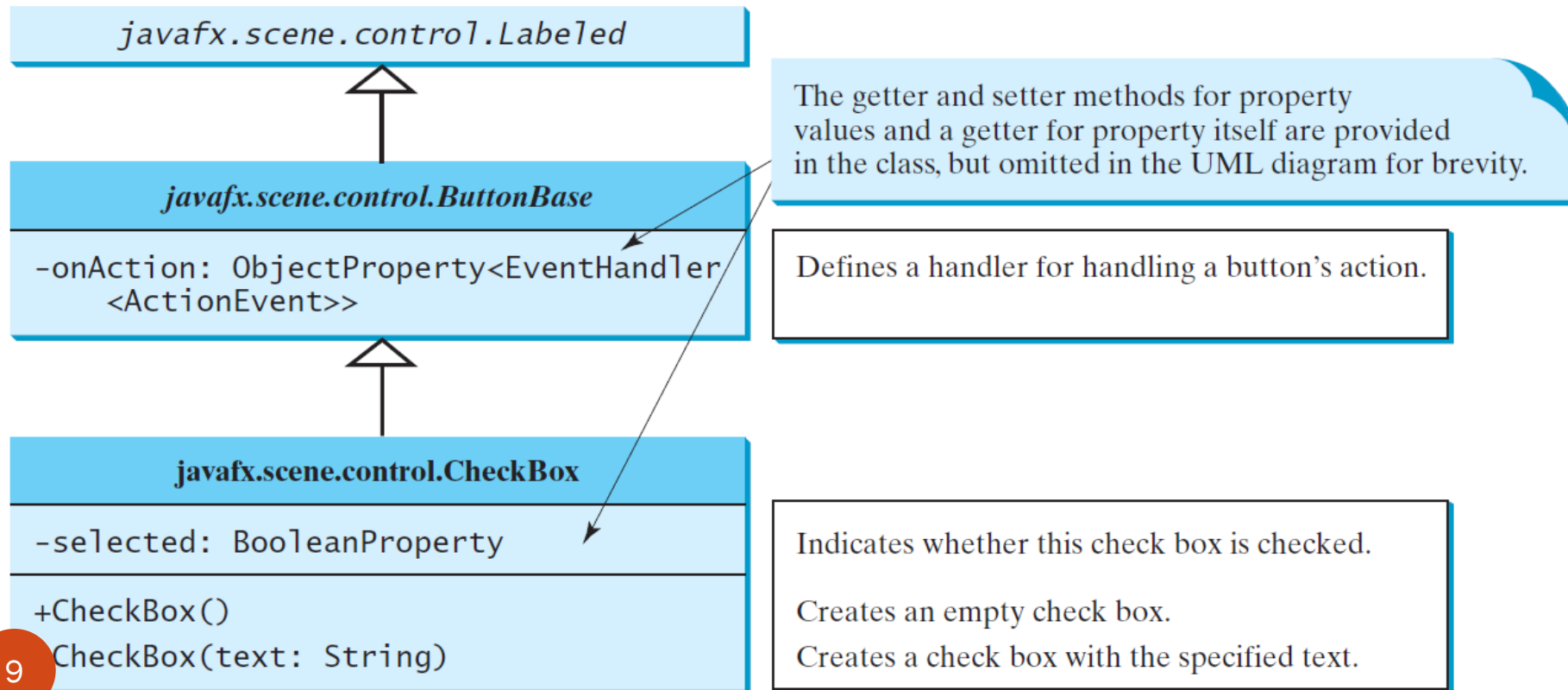
CheckBox

- A CheckBox is used for the user to make a selection (square box).
- CheckBox inherits all the properties from ButtonBase and Labeled: onAction, text, graphic, alignment, graphicTextGap, textFill, contentDisplay.

```
javafx.scene.control.Labeled
```

```
javafx.scene.control.ButtonBase
```
```
-onAction: ObjectProperty<EventHandler
    <ActionEvent>>
```

```
javafx.scene.control.CheckBox
```
```
-selected: BooleanProperty
```
```
+CheckBox()
CheckBox(text: String)
```

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

Defines a handler for handling a button's action.

Indicates whether this check box is checked.

Creates an empty check box.
Creates a check box with the specified text.

```java
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.control.CheckBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;

public class CheckBoxDemo extends ButtonDemo {
    @Override // Override the getPane() method in the super class
    protected BorderPane getPane() {
        BorderPane pane = super.getPane();

        Font fontBoldItalic = Font.font("Times New Roman",
                FontWeight.BOLD, FontPosture.ITALIC, 20);
        Font fontBold = Font.font("Times New Roman",
                FontWeight.BOLD, FontPosture.REGULAR, 20);
        Font fontItalic = Font.font("Times New Roman",
                FontWeight.NORMAL, FontPosture.ITALIC, 20);
        Font fontNormal = Font.font("Times New Roman",
                FontWeight.NORMAL, FontPosture.REGULAR, 20);

        text.setFont(fontNormal);

        VBox paneForCheckBoxes = new VBox(20);
        paneForCheckBoxes.setPadding(new Insets(5, 5, 5, 5));
        paneForCheckBoxes.setStyle("-fx-border-color: green");
```
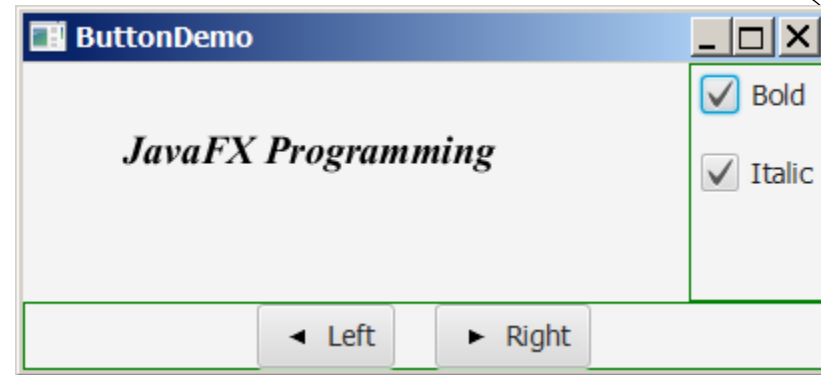
(c) Paul Fodor and Pearson Inc.

```java
        CheckBox chkBold = new CheckBox("Bold");
        CheckBox chkItalic = new CheckBox("Italic");
        paneForCheckBoxes.getChildren().addAll(chkBold, chkItalic);
        pane.setRight(paneForCheckBoxes);

        EventHandler<ActionEvent> handler = e -> {
            if (chkBold.isSelected() && chkItalic.isSelected()) {
                text.setFont(fontBoldItalic); // Both check boxes checked
            } else if (chkBold.isSelected()) {
                text.setFont(fontBold); // The Bold check box checked
            } else if (chkItalic.isSelected()) {
                text.setFont(fontItalic); // The Italic check box checked
            } else {
                text.setFont(fontNormal); // Both check boxes unchecked
            }
        };

        chkBold.setOnAction(handler);
        chkItalic.setOnAction(handler);

        return pane; // Return a new pane
    }

    // the start method is inherited from the superclass ButtonDemo

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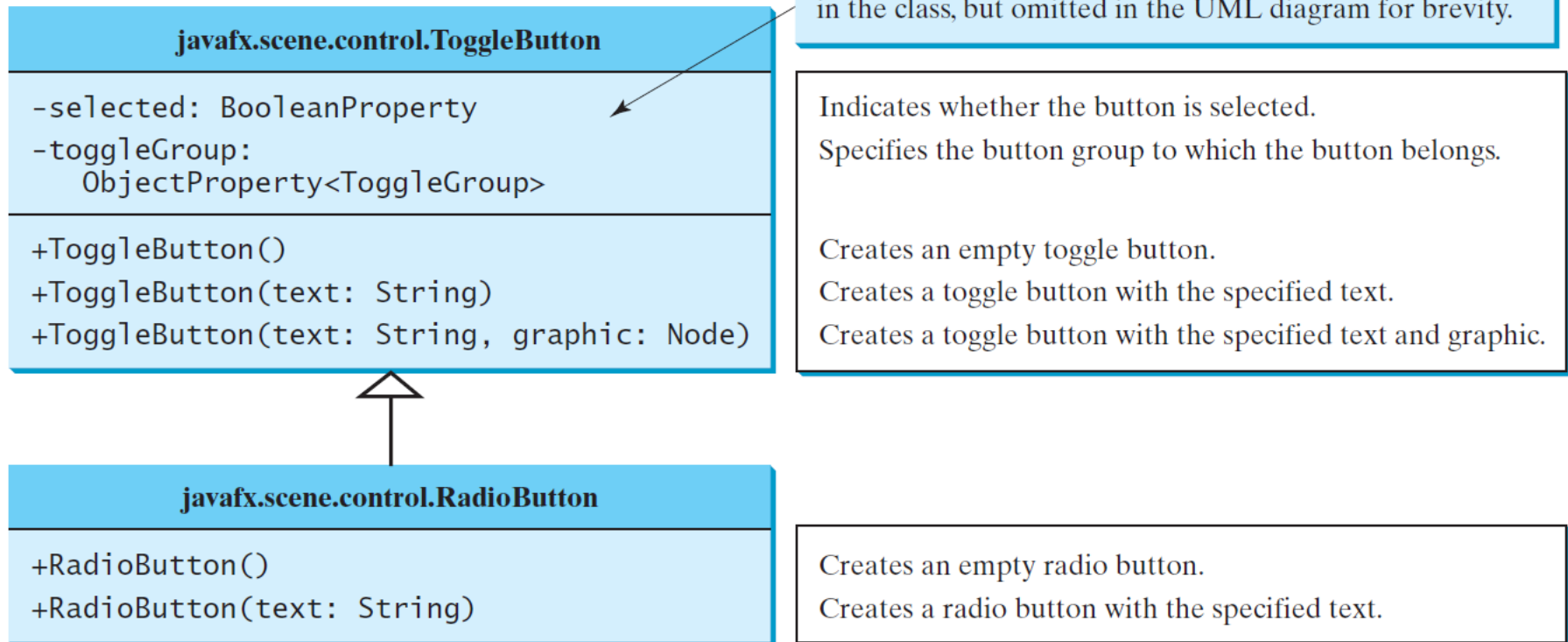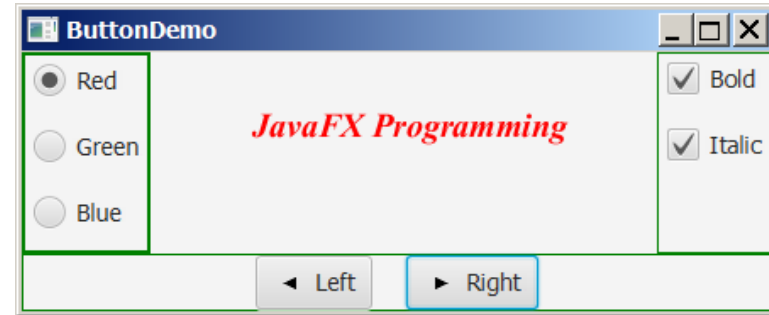
RadioButton

- Radio buttons allow to choose a **single** item from a group of choices.
  - Radio buttons display a circle that is either filled (if selected) or blank (if not selected).

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.ToggleButton
-selected: BooleanProperty
-toggleGroup: ObjectProperty<ToggleGroup>
+ToggleButton() +ToggleButton(text: String) +ToggleButton(text: String, graphic: Node)

Indicates whether the button is selected.
Specifies the button group to which the button belongs.

Creates an empty toggle button.
Creates a toggle button with the specified text.
Creates a toggle button with the specified text and graphic.

javafx.scene.control.RadioButton
+RadioButton() +RadioButton(text: String)

Creates an empty radio button.
Creates a radio button with the specified text.

12

```java
import static javafx.application.Application.launch;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.geometry.Insets;
import javafx.scene.paint.Color;

public class RadioButtonDemo extends CheckBoxDemo {
    @Override // Override the getPane() method in the super class
    protected BorderPane getPane() {
        BorderPane pane = super.getPane();
        VBox paneForRadioButtons = new VBox(20);
        paneForRadioButtons.setPadding(new Insets(5, 5, 5, 5));
        paneForRadioButtons.setStyle("-fx-border-color: green");
        RadioButton rbRed = new RadioButton("Red");
        RadioButton rbGreen = new RadioButton("Green");
        RadioButton rbBlue = new RadioButton("Blue");
        paneForRadioButtons.getChildren().addAll(rbRed, rbGreen, rbBlue);
        pane.setLeft(paneForRadioButtons);
        ToggleGroup group = new ToggleGroup();
        rbRed.setToggleGroup(group);
        rbGreen.setToggleGroup(group);
        rbBlue.setToggleGroup(group);

        rbRed.setOnAction(e -> {
            if (rbRed.isSelected()) {
                text.setFill(Color.RED);
            }
        });
```


13

```java
        rbGreen.setOnAction(e -> {
            if (rbGreen.isSelected()) {
                text.setFill(Color.GREEN);
            }
        });

        rbBlue.setOnAction(e -> {
            if (rbBlue.isSelected()) {
                text.setFill(Color.BLUE);
            }
        });

        return pane;
    }

    // the start method is inherited from the superclass ButtonDemo

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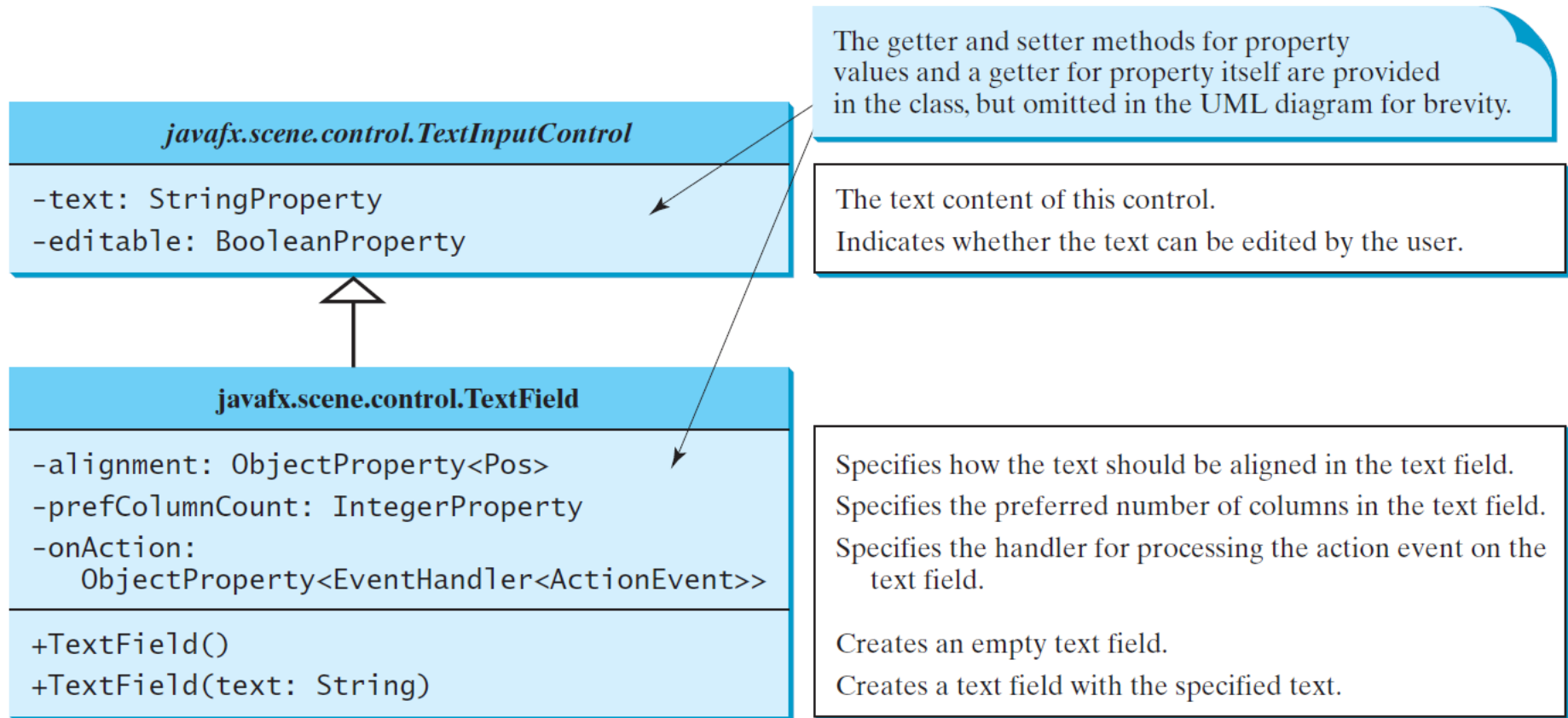
TextField

- A text field can be used to enter or display a string. TextField is a subclass of TextInputControl.

---

javafx.scene.control.TextInputControl

```
-text: StringProperty
-editable: BooleanProperty
```

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

The text content of this control.

Indicates whether the text can be edited by the user.

---

javafx.scene.control.TextField

```
-alignment: ObjectProperty<Pos>
-prefColumnCount: IntegerProperty
-onAction:
   ObjectProperty<EventHandler<ActionEvent>>

+TextField()
+TextField(text: String)
```

Specifies how the text should be aligned in the text field.
Specifies the preferred number of columns in the text field.
Specifies the handler for processing the action event on the text field.

Creates an empty text field.
Creates a text field with the specified text.

15

```java
import static javafx.application.Application.launch;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;
public class TextFieldDemo extends RadioButtonDemo{
    @Override
    protected BorderPane getPane() {
        BorderPane pane = super.getPane();

        BorderPane paneForTextField = new BorderPane();
        paneForTextField.setPadding(new Insets(5, 5, 5, 5));
        paneForTextField.setStyle("-fx-border-color: green");
        paneForTextField.setLeft(new Label("Enter a new message: "));

        TextField tf = new TextField();
        tf.setAlignment(Pos.BOTTOM_RIGHT);
        paneForTextField.setCenter(tf);
        pane.setTop(paneForTextField);

        tf.setOnAction(e -> text.setText(tf.getText()));

        return pane;
    }

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

16
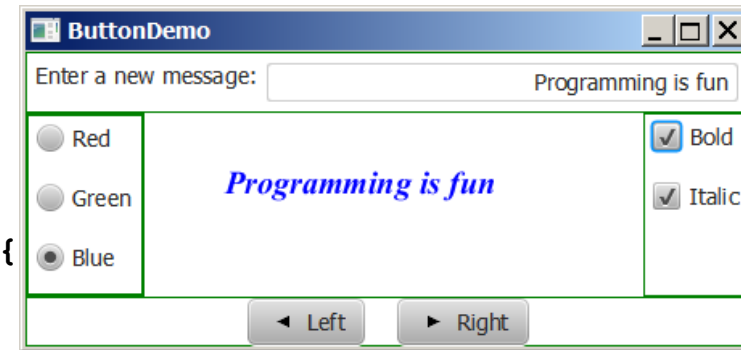
TextArea

- A TextArea enables the user to enter multiple lines of text.

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.TextInputControl

```
-text: StringProperty
-editable: BooleanProperty
```

The text content of this control.
Indicates whether the text can be edited by the user.

javafx.scene.control.TextArea

```
-prefColumnCount: IntegerProperty
-prefRowCount: IntegerProperty
-wrapText: BooleanProperty

+TextArea()
+TextArea(text: String)
```

Specifies the preferred number of text columns.
Specifies the preferred number of text rows.
Specifies whether the text is wrapped to the next line.

Creates an empty text area.
Creates a text area with the specified text.

ComboBox

Select a country: [Canada ▾]

- A combo box, also known as a choice list or drop-down list, contains a list of items from which the user can choose.

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

---

javafx.scene.control.ComboBoxBase<T>

```
-value: ObjectProperty<T>
-editable: BooleanProperty
-onAction:
    ObjectProperty<EventHandler<ActionEvent>>
```

The value selected in the combo box.

Specifies whether the combo box allows user input.

Specifies the handler for processing the action event.

---

javafx.scene.control.ComboBox<T>

```
-items: ObjectProperty<ObservableList<T>>
-visibleRowCount: IntegerProperty

+ComboBox()
+ComboBox(items: ObservableList<T>)
```

The items in the combo box popup.

The maximum number of visible rows of the items in the combo box popup.

Creates an empty combo box.

Creates a combo box with the specified items.

ListView

- A list view is a component that performs basically the same function as a combo box, but it enables the user to choose a single value or multiple values.

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.ListView\<T\>	
-items: ObjectProperty\<ObservableList\<T\>\>	The items in the list view.
-orientation: BooleanProperty	Indicates whether the items are displayed horizontally or vertically in the list view.
-selectionModel: ObjectProperty\<MultipleSelectionModel\<T\>\>	Specifies how items are selected. The SelectionModel is also used to obtain the selected items.
+ListView()	Creates an empty list view.
+ListView(items: ObservableList\<T\>)	Creates a list view with the specified items.

(c) Paul Fodor and Pearson Inc.

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.collections.FXCollections;
import javafx.scene.Scene;
import javafx.scene.control.ListView;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.SelectionMode;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;

public class ListViewDemo extends Application {
  // Declare an array of Strings for flag titles
  private String[] flagTitles = {"United States of America", "Canada", "China",
    "Denmark", "France", "Germany", "India"};
  // Declare an ImageView array for the national flags
  private ImageView[] ImageViews = {
    new ImageView("image/us.gif"),
    new ImageView("image/ca.gif"),
    new ImageView("image/china.gif"),
    new ImageView("image/denmark.gif"),
    new ImageView("image/fr.gif"),
    new ImageView("image/germany.gif"),
    new ImageView("image/india.gif")
  };
  @Override
  public void start(Stage primaryStage) {
    ListView<String> lv = new ListView<>(FXCollections
                                    .observableArrayList(flagTitles));
    lv.setPrefSize(400, 400);
```

(c) Paul Fodor and Pearson Inc.

```java
      lv.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);

      // Create a pane to hold image views
      FlowPane imagePane = new FlowPane(10, 10);
      BorderPane pane = new BorderPane();
      pane.setLeft(new ScrollPane(lv));
      pane.setCenter(imagePane);

      lv.getSelectionModel().selectedItemProperty().addListener(
        ov -> {
          imagePane.getChildren().clear();
          for (Integer i: lv.getSelectionModel().getSelectedIndices()) {
            imagePane.getChildren().add(ImageViews[i]);
          }
      });

      Scene scene = new Scene(pane, 450, 170);
      primaryStage.setTitle("ListViewDemo");
      primaryStage.setScene(scene);
      primaryStage.show();
  }


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

ScrollBar

- A scroll bar is a control that enables the user to select from a range of values. The scrollbar appears in two styles: horizontal and vertical.

Minimal value / Track / Maximal value / Left button / Thumb / Right button

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.
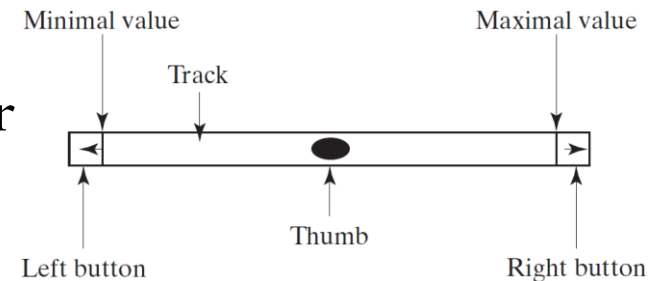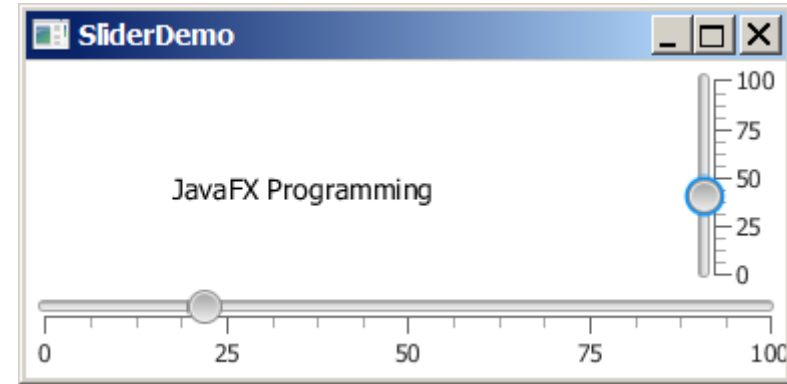
javafx.scene.control.ScrollBar

```
-blockIncrement: DoubleProperty
-max: DoubleProperty
-min: DoubleProperty
-unitIncrement: DoubleProperty

-value: DoubleProperty
-visibleAmount: DoubleProperty
-orientation: ObjectProperty<Orientation>

+ScrollBar()
+increment()
+decrement()
```

The amount to adjust the scroll bar if the track of the bar is clicked (default: 10).

The maximum value represented by this scroll bar (default: 100).

The minimum value represented by this scroll bar (default: 0).

The amount to adjust the scroll bar when the increment() and decrement() methods are called (default: 1).

Current value of the scroll bar (default: 0).

The width of the scroll bar (default: 15).

Specifies the orientation of the scroll bar (default: HORIZONTAL).

Creates a default horizontal scroll bar.

Increments the value of the scroll bar by unitIncrement.

Decrements the value of the scroll bar by unitIncrement.

(c) Paul Fodor and Pearson Inc.

Slider

- Slider is similar to ScrollBar, but Slider has more properties and can appear in many forms.

SliderDemo

JavaFX Programming

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.Slider

```
-blockIncrement: DoubleProperty
-max: DoubleProperty
-min: DoubleProperty
-value: DoubleProperty
-orientation: ObjectProperty<Orientation>
-majorTickUnit: DoubleProperty
-minorTickCount: IntegerProperty
-showTickLabels: BooleanProperty
-showTickMarks: BooleanProperty

+Slider()
+Slider(min: double, max: double,
    value: double)
```

The amount to adjust the slider if the track of the bar is clicked (default: 10).
The maximum value represented by this slider (default: 100).
The minimum value represented by this slider (default: 0).
Current value of the slider (default: 0).
Specifies the orientation of the slider (default: HORIZONTAL).
The unit distance between major tick marks.
The number of minor ticks to place between two major ticks.
Specifies whether the labels for tick marks are shown.
Specifies whether the tick marks are shown.

Creates a default horizontal slider.
Creates a slider with the specified min, max, and value.

23

Media

- The Media class is used to obtain the source of a media type.
- The MediaPlayer class is used to play and control the media.
- The MediaView class is used to display video.

The getter and setter methods for property
values and a getter for property itself are provided
in the class, but omitted in the UML diagram for brevity.

javafx.scene.media.Media

```
-duration: ReadOnlyObjectProperty
    <Duration>
-width: ReadOnlyIntegerProperty
-height: ReadOnlyIntegerProperty

+Media(source: String)
```

The durations in seconds of the source media.

The width in pixels of the source video.
The height in pixels of the source video.

Creates a Media from a URL source.

24

MediaPlayer

- The MediaPlayer class plays and controls media with properties: autoPlay, currentCount, cycleCount, mute, volume, and totalDuration.

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.media.MediaPlayer

```
-autoPlay: BooleanProperty
-currentCount: ReadOnlyIntegerProperty
-cycleCount: IntegerProperty
-mute: BooleanProperty
-volume: DoubleProperty
-totalDuration:
    ReadOnlyObjectProperty<Duration>

+MediaPlayer(media: Media)
+play(): void
+pause(): void
+seek(): void
```

Specifies whether the playing should start automatically.
The number of completed playback cycles.
Specifies the number of time the media will be played.
Specifies whether the audio is muted.
The volume for the audio.
The amount of time to play the media from start to finish.

Creates a player for a specified media.
Plays the media.
Pauses the media.
Seeks the player to a new playback time.

25

MediaView

- The MediaView class is a subclass of Node that provides a view of the Media being played by a MediaPlayer.
  - The MediaView class provides the properties for viewing the media.

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.media.MediaView

```
-x: DoubleProperty
-y: DoubleProperty
-mediaPlayer:
    ObjectProperty<MediaPlayer>
-fitWidth: DoubleProperty
-fitHeight: DoubleProperty

+MediaView()
+MediaView(mediaPlayer: MediaPlayer)
```

Specifies the current x-coordinate of the media view.
Specifies the current y-coordinate of the media view.
Specifies a media player for the media view.

Specifies the width of the view for the media to fit.
Specifies the height of the view for the media to fit.

Creates an empty media view.
Creates a media view with the specified media player.

Example: Using Media

- This example displays a video in a view: use the play/pause button to play or pause the video and use the rewind button to restart the video, and use the slider to control the volume of the audio.


```
media: Media ◇── mediaPlayer: MediaPlayer ◇── mediaView: MediaView
```

(c) Paul Fodor and Pearson Inc.

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Region;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.util.Duration;
public class MediaDemo extends Application {
    private static final String MEDIA_URL = "sample.mp4";
    @Override
    public void start(Stage primaryStage) {
        Media media = new Media(MEDIA_URL);
        MediaPlayer mediaPlayer = new MediaPlayer(media);
        MediaView mediaView = new MediaView(mediaPlayer);
        Button playButton = new Button(">");
        playButton.setOnAction(e -> {
            if (playButton.getText().equals(">")) {
                mediaPlayer.play();
                playButton.setText("||");
            } else {
                mediaPlayer.pause();
                playButton.setText(">");
            }
        });
```

(c) Paul Fodor and Pearson Inc.

```java
        Button rewindButton = new Button("<<");
        rewindButton.setOnAction(e -> mediaPlayer.seek(Duration.ZERO));
        Slider slVolume = new Slider();
        slVolume.setPrefWidth(150);
        slVolume.setMaxWidth(Region.USE_PREF_SIZE);
        slVolume.setMinWidth(30);
        slVolume.setValue(50);
        mediaPlayer.volumeProperty().bind(slVolume.valueProperty().divide(100));
        HBox hBox = new HBox(10);
        hBox.setAlignment(Pos.CENTER);
        hBox.getChildren().addAll(playButton, rewindButton,
                new Label("Volume"), slVolume);
        BorderPane pane = new BorderPane();
        pane.setCenter(mediaView);
        pane.setBottom(hBox);

        Scene scene = new Scene(pane, 650, 500);
        primaryStage.setTitle("MediaDemo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

CSS

- Cascading Style Sheets (CSS) is a language used for describing the look and formatting of a document.
  - CSS is designed primarily to enable the separation of document content from document presentation (layout, colors, and fonts).
    - It is used to style web pages and user interfaces written in HTML, XHTML, and any kind of XML document.
  - The CSS language specifications are Web standards maintained by the World Wide Web Consortium (W3C).
  - CSS rule set example:

Selector   Declaration   Declaration

h1 { color:blue; font-size:12px;}

Property   Value   Property   Value

(c) Paul Fodor and Pearson Inc.
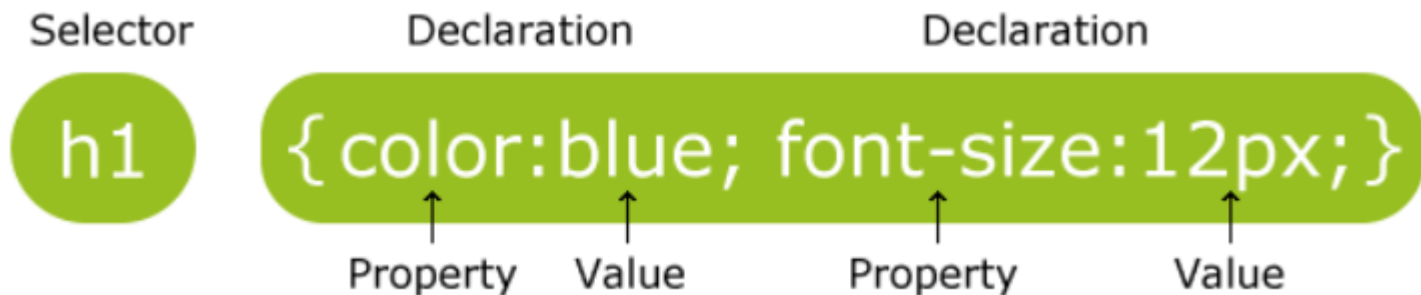
JavaFX CSS

- JavaFX Cascading Style Sheets (CSS) is based on the W3C CSS and allows to customize and develop themes for JavaFX controls and scene graph objects
  - http://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html
  - JavaFX uses the prefix "-fx-" to define its vendor CSS properties (separate from W3C CSS).
- A style sheet uses the style class or style id to define styles.
  - Mutiple style classes can be applied to a single node and a style id to a unique node.
  - The syntax .styleclass defines a style class.
  - The syntax #styleid defines a style id.

Style Class and Style ID

- **mystyle.css:**

```css
.plaincircle {
  -fx-fill: white;
  -fx-stroke: black;
}
.circleborder {
  -fx-stroke-width: 5;
  -fx-stroke-dash-array: 12 2 4 2;
}
.border {
  -fx-border-color: black;
  -fx-border-width: 5;
}
#redcircle {
  -fx-fill: red;
  -fx-stroke: red;
}
#greencircle {
  -fx-fill: green;
  -fx-stroke: green;
}
```

(c) Paul Fodor and Pearson Inc.

```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;
public class StyleSheetDemo extends Application {
  @Override
  public void start(Stage primaryStage) {
    HBox hBox = new HBox(5);
    Scene scene = new Scene(hBox, 300, 250);
    // Load the stylesheet
    scene.getStylesheets().add("mystyle.css");
    Pane pane1 = new Pane();
    Circle circle1 = new Circle(50, 50, 30);
    Circle circle2 = new Circle(150, 50, 30);
    Circle circle3 = new Circle(100, 100, 30);
    pane1.getChildren().addAll(circle1, circle2, circle3);
    pane1.getStyleClass().add("border");
    circle1.getStyleClass().add("plaincircle"); // Add a style class
    circle2.getStyleClass().add("plaincircle"); // Add a style class
    circle3.setId("redcircle"); // Add a style id
    Pane pane2 = new Pane();
    Circle circle4 = new Circle(100, 100, 30);
```

```java
        circle4.getStyleClass().addAll("circleborder", "plainCircle");
        circle4.setId("greencircle"); // Add a style class
        pane2.getChildren().add(circle4);
        pane2.getStyleClass().add("border");

        hBox.getChildren().addAll(pane1, pane2);

        primaryStage.setTitle("StyleSheetDemo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    // Lauch the program from command-line
    public static void main(String[] args) {
        launch(args);
    }
}
```
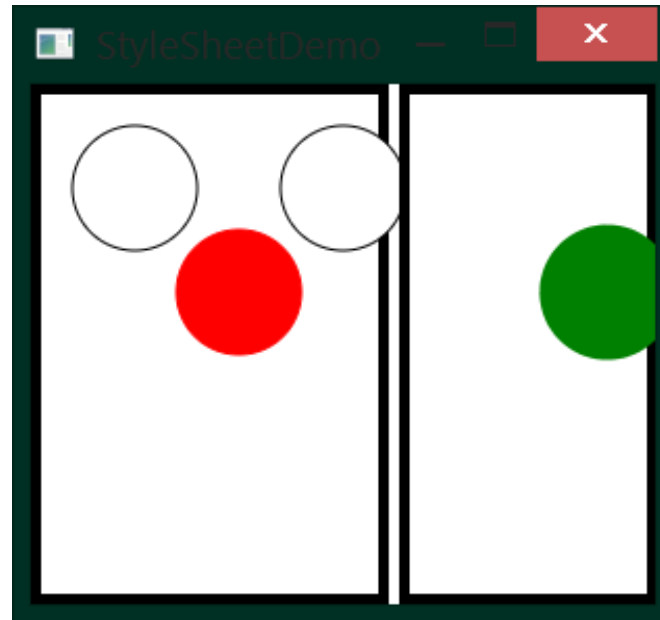
QuadCurve

- A quadratic curve is mathematically defined as a quadratic polynomial.

```
QuadCurve(double startX, double startY,
double controlX, double controlY, double
endX, double endY)
```

(controlX, controlY)

(startX, startY)

(endX, endY)
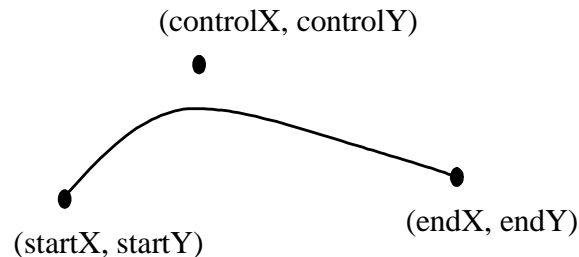
QuadCurve

javafx.scene.shape.QuadCurve
-startX: DoubleProperty
-startY: DoubleProperty
-endX: DoubleProperty
-endY: DoubleProperty
-controlX: DoubleProperty
-controlY: DoubleProperty
+QuadCurve()
+QuadCurve(startX: double, startY: double, controlX: double, controlY: double, endX: double, endY: double)

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

The x-coordinate of the start point (default 0).

The y-coordinate of the start point (default 0)..

The x-coordinate of the end point (default 0)..

The y-coordinate of the end point (default 0)..

The x-coordinate of the control point (default 0)..

The y-coordinate of the control point (default 0)..

Creates an empty quad curve.

Creates a quad curve with the specified arguments.

(c) Paul Fodor and Pearson Inc.

Menus

- Menus make selection easier and are widely used in window applications.
  - JavaFX provides five classes that implement menus: MenuBar, Menu, MenuItem, CheckMenuItem, and RadioButtonMenuItem.
- MenuBar is a top-level menu component used to hold the menus.
  - A menu consists of menu items that the user can select (or toggle on or off).
  - A menu item can be an instance of MenuItem, CheckMenuItem, or RadioButtonMenuItem.
  - Menu items can be associated with nodes and keyboard accelerators.
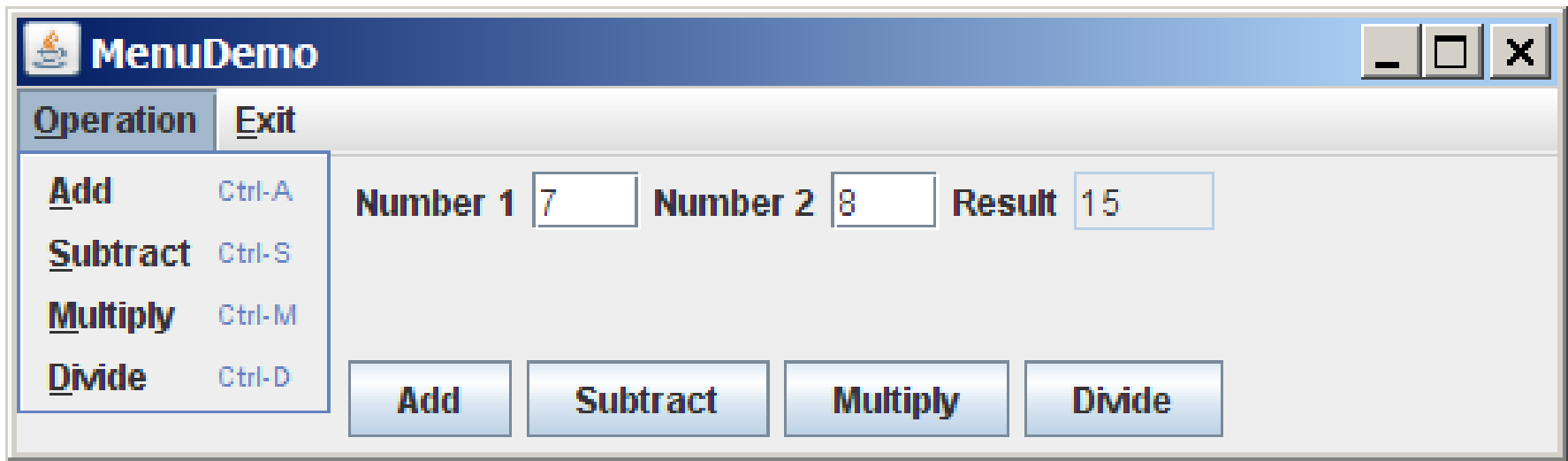
```java
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.MenuItem;
import javafx.scene.control.TextField;
import javafx.scene.input.KeyCombination;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class MenuDemo extends Application {
    private TextField tfNumber1 = new TextField();
    private TextField tfNumber2 = new TextField();
    private TextField tfResult = new TextField();
```

(c) Paul Fodor and Pearson Inc.

```java
@Override
 public void start(Stage primaryStage) {
    MenuBar menuBar = new MenuBar();

    Menu menuOperation = new Menu("Operation");
    Menu menuExit = new Menu("Exit");
    menuBar.getMenus().addAll(menuOperation, menuExit);

    MenuItem menuItemAdd = new MenuItem("Add");
    MenuItem menuItemSubtract = new MenuItem("Subtract");
    MenuItem menuItemMultiply = new MenuItem("Multiply");
    MenuItem menuItemDivide = new MenuItem("Divide");
    menuOperation.getItems().addAll(menuItemAdd, menuItemSubtract,
       menuItemMultiply, menuItemDivide);

    MenuItem menuItemClose = new MenuItem("Close");
    menuExit.getItems().add(menuItemClose);

    menuItemAdd.setAccelerator(
       KeyCombination.keyCombination("Ctrl+A"));
    menuItemSubtract.setAccelerator(
       KeyCombination.keyCombination("Ctrl+S"));
    menuItemMultiply.setAccelerator(
       KeyCombination.keyCombination("Ctrl+M"));
    menuItemDivide.setAccelerator(
       KeyCombination.keyCombination("Ctrl+D"));
    HBox hBox1 = new HBox(5);
    tfNumber1.setPrefColumnCount(2);
    tfNumber2.setPrefColumnCount(2);
    tfResult.setPrefColumnCount(2);
```

(c) Paul Fodor and Pearson Inc.

```java
    hBox1.getChildren().addAll(new Label("Number 1:"), tfNumber1,
      new Label("Number 2:"), tfNumber2, new Label("Result:"),
      tfResult);
    hBox1.setAlignment(Pos.CENTER);

    HBox hBox2 = new HBox(5);
    Button btAdd = new Button("Add");
    Button btSubtract = new Button("Subtract");
    Button btMultiply = new Button("Multiply");
    Button btDivide = new Button("Divide");
    hBox2.getChildren().addAll(btAdd, btSubtract, btMultiply, btDivide);
    hBox2.setAlignment(Pos.CENTER);

    VBox vBox = new VBox(10);
    vBox.getChildren().addAll(menuBar, hBox1, hBox2);
    Scene scene = new Scene(vBox, 300, 250);
    primaryStage.setTitle("MenuDemo"); // Set the window title
    primaryStage.setScene(scene); // Place the scene in the window
    primaryStage.show(); // Display the window
    // Handle menu actions
    menuItemAdd.setOnAction(e -> perform('+'));
    menuItemSubtract.setOnAction(e -> perform('-'));
    menuItemMultiply.setOnAction(e -> perform('*'));
    menuItemDivide.setOnAction(e -> perform('/'));
    menuItemClose.setOnAction(e -> System.exit(0));
    // Handle button actions
    btAdd.setOnAction(e -> perform('+'));
    btSubtract.setOnAction(e -> perform('-'));
    btMultiply.setOnAction(e -> perform('*'));
    btDivide.setOnAction(e -> perform('/'));
  }
```

```java
    private void perform(char operator) {
      double number1 = Double.parseDouble(tfNumber1.getText());
      double number2 = Double.parseDouble(tfNumber2.getText());

      double result = 0;
      switch (operator) {
        case '+': result = number1 + number2; break;
        case '-': result = number1 - number2; break;
        case '*': result = number1 * number2; break;
        case '/': result = number1 / number2; break;
      }

      tfResult.setText(result + "");
    };

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

(c) Paul Fodor and Pearson Inc.

Context Menu

- A context menu (also known as a popup menu) is like a regular menu, but does not have a menu bar and can float anywhere on the screen.
  - Creating a context menu is similar to creating a regular menu.
    - First, create an instance of ContextMenu, then add MenuItem, CheckMenuItem, and RadioMenuItem to the context menu.

```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.ContextMenu;
import javafx.scene.control.MenuItem;
import javafx.scene.image.ImageView;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
public class ContextMenuDemo extends Application {
  @Override
  public void start(Stage primaryStage) {
    ContextMenu contextMenu = new ContextMenu();
    MenuItem menuItemNew = new MenuItem("New",
      new ImageView("image/new.gif"));
    MenuItem menuItemOpen = new MenuItem("Open",
      new ImageView("image/open.gif"));
    MenuItem menuItemPrint = new MenuItem("Print",
      new ImageView("image/print.gif"));
    MenuItem menuItemExit = new MenuItem("Exit");
    contextMenu.getItems().addAll(menuItemNew, menuItemOpen,
      menuItemPrint, menuItemExit);
    Pane pane = new Pane();
    Scene scene = new Scene(pane, 300, 250);
    primaryStage.setTitle("ContextMenuDemo");
    primaryStage.setScene(scene);
    primaryStage.show();
```
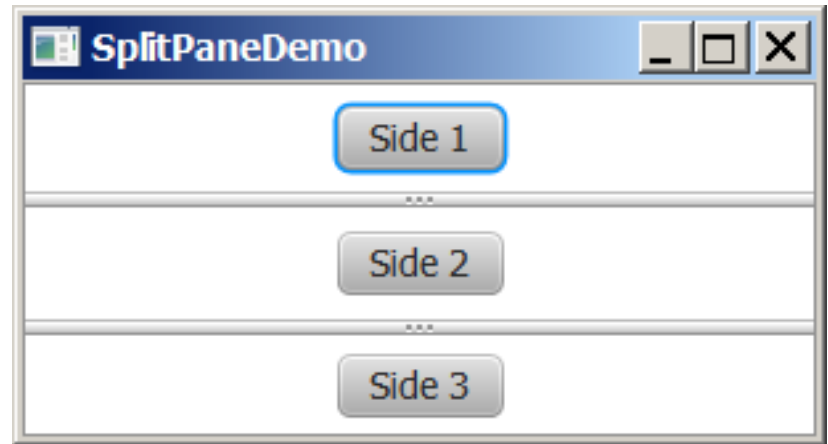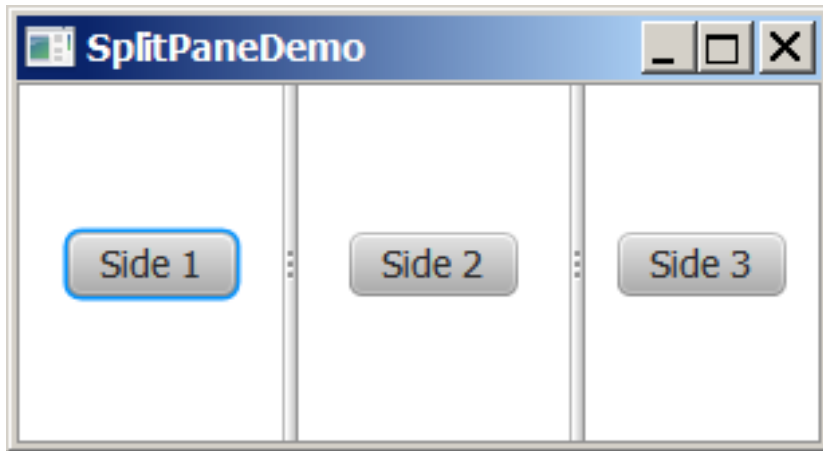

43
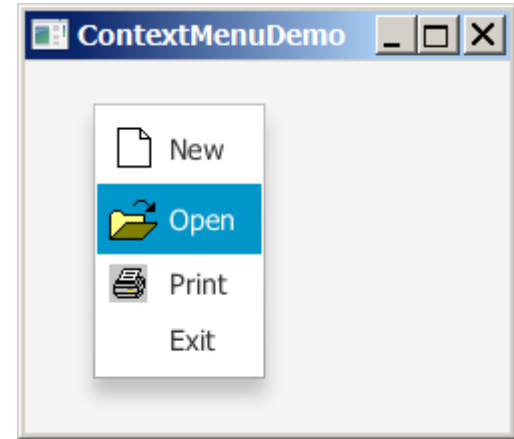
```java
    pane.setOnMousePressed(
      e -> contextMenu.show(pane, e.getScreenX(), e.getScreenY()));

    menuItemNew.setOnAction(e -> System.out.println("New"));
    menuItemOpen.setOnAction(e -> System.out.println("Open"));
    menuItemPrint.setOnAction(e -> System.out.println("Print"));
    menuItemExit.setOnAction(e -> System.exit(0));
  }

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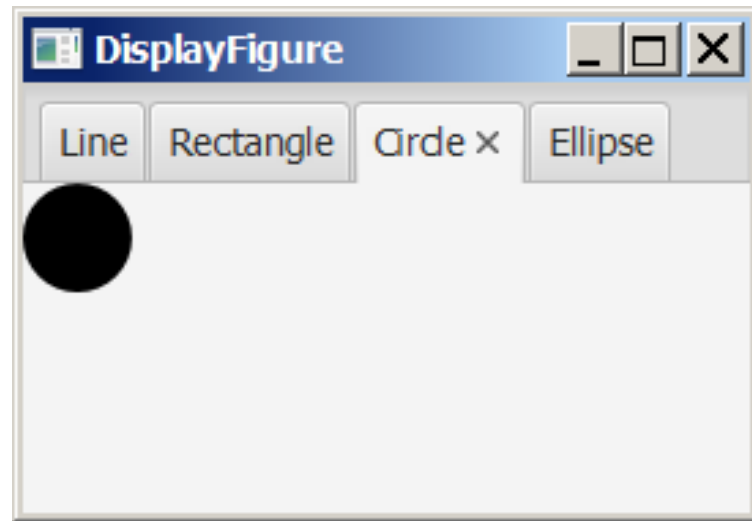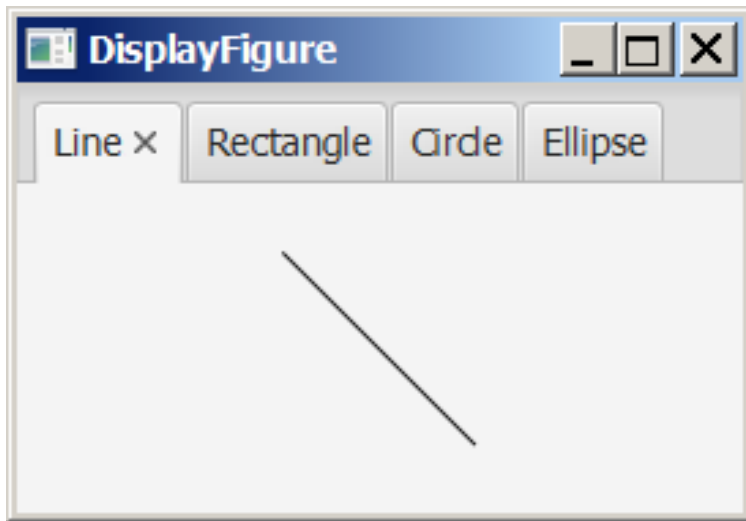
SplitPane

- The SplitPane class can be used to display multiple panes and allow the user to adjust the size of the panes.
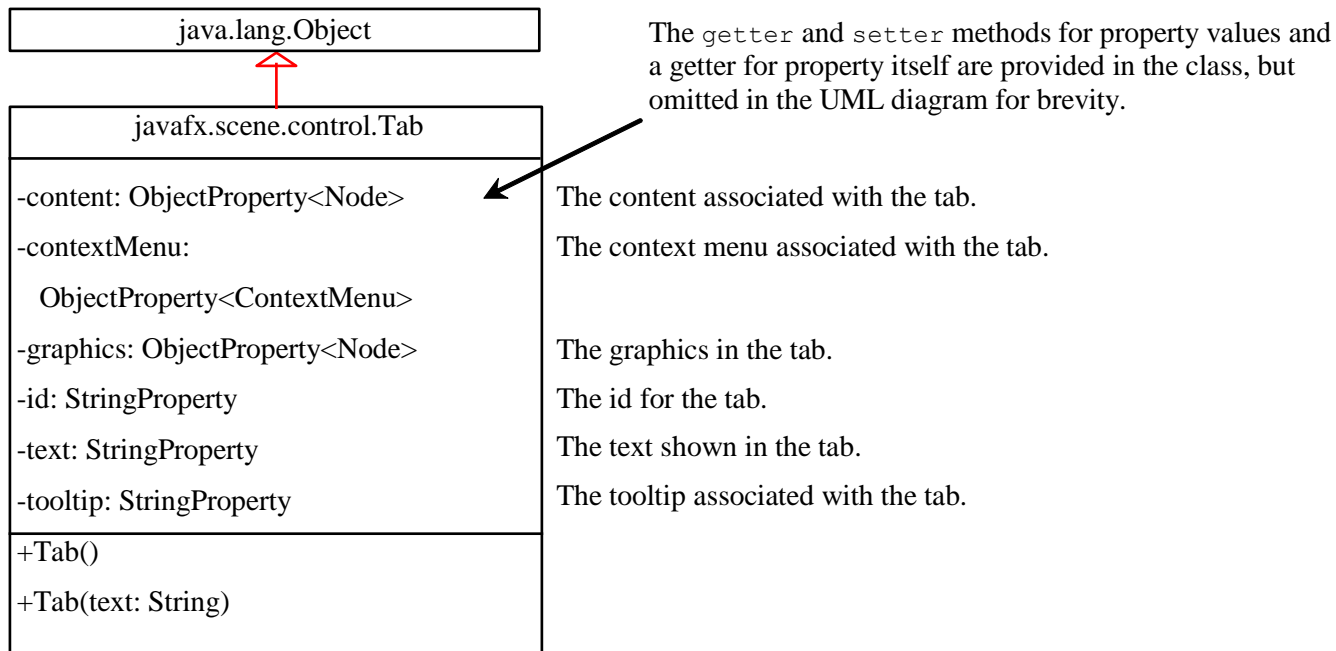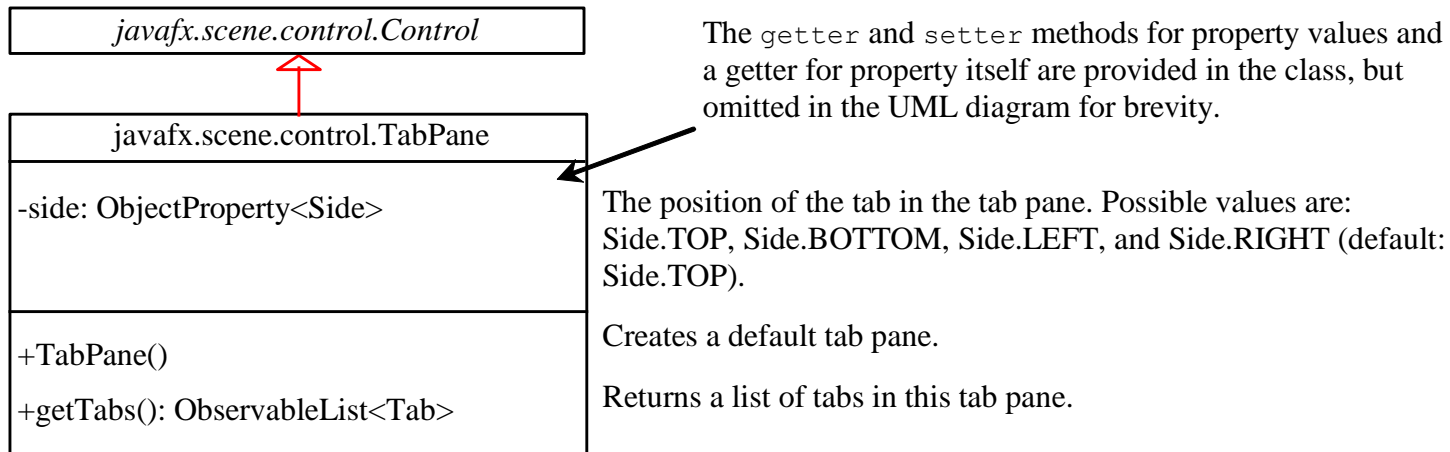
TabPane

- The TabPane class can be used to display multiple panes with tabs.

TabPane

javafx.scene.control.Control

The `getter` and `setter` methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.TabPane
-side: ObjectProperty<Side>
+TabPane()
+getTabs(): ObservableList<Tab>

The position of the tab in the tab pane. Possible values are: Side.TOP, Side.BOTTOM, Side.LEFT, and Side.RIGHT (default: Side.TOP).

Creates a default tab pane.

Returns a list of tabs in this tab pane.

java.lang.Object

The `getter` and `setter` methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.Tab
-content: ObjectProperty<Node>
-contextMenu: ObjectProperty<ContextMenu>
-graphics: ObjectProperty<Node>
-id: StringProperty
-text: StringProperty
-tooltip: StringProperty
+Tab()
+Tab(text: String)

The content associated with the tab.

The context menu associated with the tab.

The graphics in the tab.

The id for the tab.

The text shown in the tab.

The tooltip associated with the tab.

47

TableView

- You can display tables using the TableView class.

TableViewDemo			
Country	**Capital**	**Population (million)**	**Is Democratic?**
USA	Washington DC	280.0	true
Canada	Ottawa	32.0	true
United Kingdom	London	60.0	true
Germany	Berlin	83.0	true
France	Paris	60.0	true
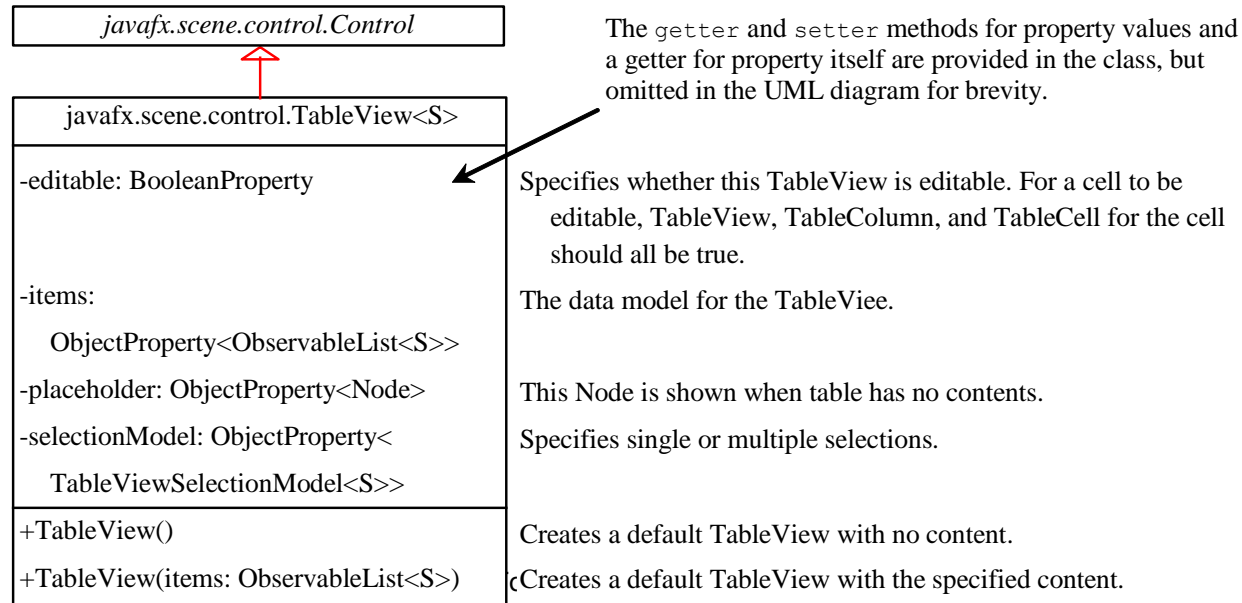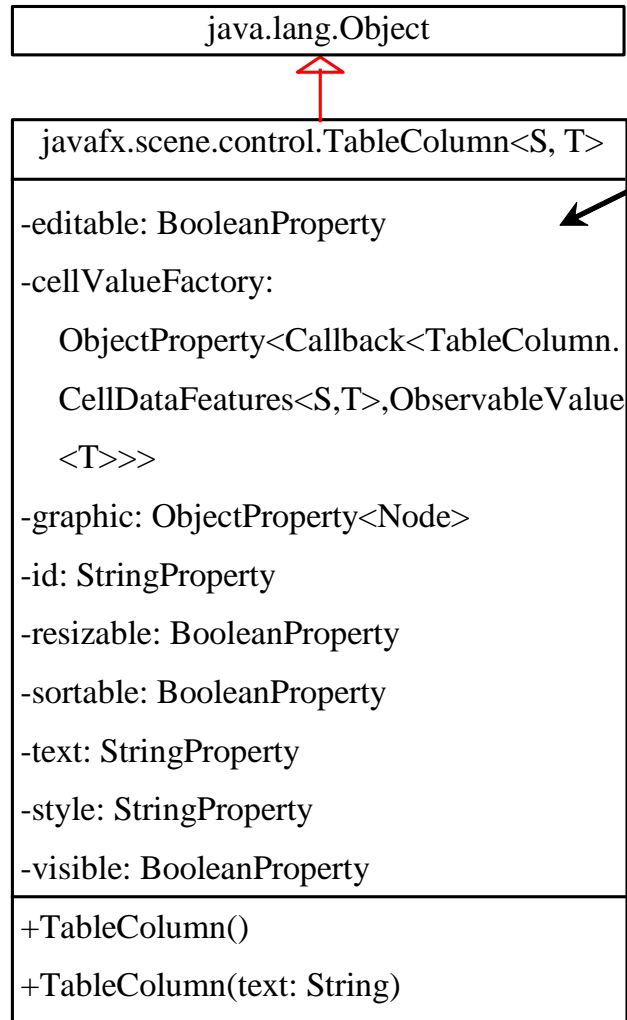
---

javafx.scene.control.Control

↑

javafx.scene.control.TableView<S>

-editable: BooleanProperty

-items:
　ObjectProperty<ObservableList<S>>

-placeholder: ObjectProperty<Node>

-selectionModel: ObjectProperty<
　TableViewSelectionModel<S>>

+TableView()

+TableView(items: ObservableList<S>)

The `getter` and `setter` methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

Specifies whether this TableView is editable. For a cell to be editable, TableView, TableColumn, and TableCell for the cell should all be true.

The data model for the TableViee.

This Node is shown when table has no contents.

Specifies single or multiple selections.

Creates a default TableView with no content.

Creates a default TableView with the specified content.

48

The TableColumn Class

java.lang.Object

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.TableColumn<S, T>
-editable: BooleanProperty

-cellValueFactory:
  ObjectProperty<Callback<TableColumn.
  CellDataFeatures<S,T>,ObservableValue
  <T>>>

-graphic: ObjectProperty<Node>
-id: StringProperty
-resizable: BooleanProperty
-sortable: BooleanProperty
-text: StringProperty
-style: StringProperty
-visible: BooleanProperty
+TableColumn()
+TableColumn(text: String)

Specifies whether this TableColumn allows editing.

The cell value factory to specify how to populate all cells within a single column.

The graphic for this TableColumn.

The id for this TableColumn.

Indicates whether the column is resizable.

Indicates whether the column is sortable.

Text in the table column header.

Specify the CSS style for the column.

Specify whether the column is visible (default: true).

Creates a default TableColumn.

Creates a TableView with the specified header text.

FXML

- FXML is a declarative XML-based language created by Oracle Corporation for defining the user interface of a JavaFX 2.0 application.
  - It can be edited and created using the JavaFX Scene Builder 2 (downloaded separately from J2SE)
  - Create a new JavaFX project in Netbeans and you will get 3 files: an FXML file with the UI design, a main application .java file that loads the FXML and a controller for the event handlers for the UI Nodes.

FXML document:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>

<?import java.util.*?>

<?import javafx.scene.*?>

<?import javafx.scene.control.*?>

<?import javafx.scene.layout.*?>

<AnchorPane id="AnchorPane" prefHeight="200" prefWidth="200"
xmlns:fx="http://javafx.com/fxml/1" xmlns="http://javafx.com/javafx/8"
fx:controller="javafxapplication1.FXMLDocumentController">

    <children>

      <FlowPane prefHeight="200.0" prefWidth="200.0">

        <children>

            <Label fx:id="label" minHeight="16" minWidth="69"

                    text="Welcome to FXML" />

        </children>

      </FlowPane>

    </children>

</AnchorPane>
```

51
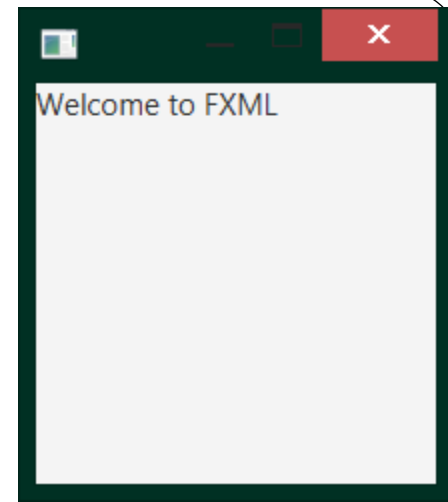
```java
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class JavaFXApplication5 extends Application {
    @Override
    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }
    public static void main(String[] args) {
        launch(args);
    }
}

import java.net.URL;
import java.util.ResourceBundle;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Label;
public class FXMLDocumentController implements Initializable {
    @FXML
    private Label label;
    @Override
    public void initialize(URL url, ResourceBundle rb) {
    }
```

(c) Paul Fodor and Pearson Inc.

HTML in JavaFX

- HTML intro.: the Internet Web pages format

- Example: **html_sample_01.html**

```
<!DOCTYPE html>
<html>
<body>


<h1>My First Heading</h1>


<p>My first paragraph.</p>


<span style="background-color:red; color:blue" >NOW?</span>


</body>
</html>
```

← This is the HTML tag. Every HTML page has one

← This is a heading

← This is a paragraph

(c) Paul Fodor and Pearson Inc.
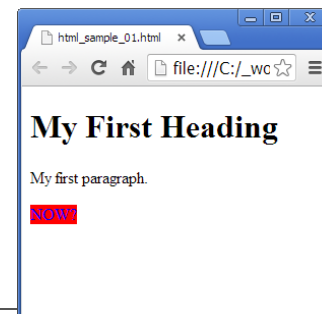
HTML

- HTML is a language for describing web pages.
- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is a **markup** language
- A markup language is a set of markup **tags**
- The tags **describe** document content
- HTML documents contain HTML **tags** and plain **text**
- HTML documents are also called **web pages**

HTML

- HTML markup tags are usually called HTML tags

- HTML tags are keywords (tag names) surrounded by **angle brackets** like <html>

- HTML tags normally **come in pairs** like and

- The first tag in a pair is the **start tag,** the second tag is the **end tag**

- The end tag is written like the start tag, with a **forward slash** before the tag name

- Start and end tags are also called **opening tags** and **closing tags**

<tagname>content</tagname>

<p>This is a paragraph.</p>

HTML by Examples

- http://www.w3schools.com/html/html_examples.asp

- HTML links:
  - `<a href="http://www.w3schools.com">This is a link</a>`
  - It appears as: This is a link

- HTML images:
  - `<img src="w3schools.jpg" width="104" height="142">`
  - It appears as:

JavaFX with HTML

- **You can put HTML code in JavaFX:**

```java
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;
import javafx.scene.web.WebEngine;
import javafx.scene.web.WebView;
public class HTMLDemo extends Application {
  @Override
  public void start(Stage primaryStage) {
    WebView browser = new WebView();
    WebEngine webEngine = browser.getEngine();
    webEngine.loadContent("<html><b><u>T</u>wo</b><br>lines</html>");
    StackPane root = new StackPane();
    root.getChildren().add(browser);
    Scene scene = new Scene(root, 100, 150);
    primaryStage.setTitle("Hello World!");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
  public static void main(String[] args) {
    launch(args);
```

```java
import org.w3c.dom.Document;
import org.w3c.dom.Element;

// ... get the document of the engine
Document doc = webEngine.getDocument();
// and the elements
... Element el = doc.getElementById("id1");
```

57

JavaFX with HTML

- **You can get the Document only when the asynchronized WebEngine had finished loading the page. That is,**
  ```
  Document doc = webEngine.getDocument();
  ```
  may be null if the page is not loaded yet.

  - **Solution: listen to the state of the WebEngine object to know when it is done loading:**
    ```
    engine.getLoadWorker().stateProperty().addListener(
        (ObservableValue<? extends State> observable,
            State oldValue, State newValue)
            -> {
            if (newValue == State.SUCCEEDED)
            docManager.setStatsDoc(engine.getDocument());
    });
    ```

javafx.scene.canvas.Canvas

- javafx.scene.canvas.Canvas is an image that can be drawn on using a set of graphics commands provided by a GraphicsContext.
- javafx.scene.canvas.GraphicsContext issues draw calls to a Canvas using a buffer:
  - each call pushes the necessary parameters onto the buffer where they will be later rendered onto the image of the Canvas node by the rendering thread at the end of a pulse.

```
Canvas canvas = new Canvas(250,250);
GraphicsContext gc =
    canvas.getGraphicsContext2D();
gc.fillRect(75,75,100,100);
```

59

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.stage.Screen;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.image.Image;
import javafx.geometry.Point2D;
import javafx.geometry.Rectangle2D;
import javafx.scene.Group;
import java.util.ArrayList;
import java.util.Iterator;
public class CanvasDemo extends Application {
    Stage primaryStage;
    Scene scene;
    Canvas canvas;
    GraphicsContext gc;
    Image logo1Image, logo2Image;
    ArrayList<Point2D> logo1Locations, logo2Locations;
    @Override
    public void start(Stage initPrimaryStage) {
        primaryStage = initPrimaryStage;
        initStage();
        initData();
        initGUI();
        initHandlers();
    }
```


Image Mosaic App

60

(c) Paul Fodor and Pearson Inc.
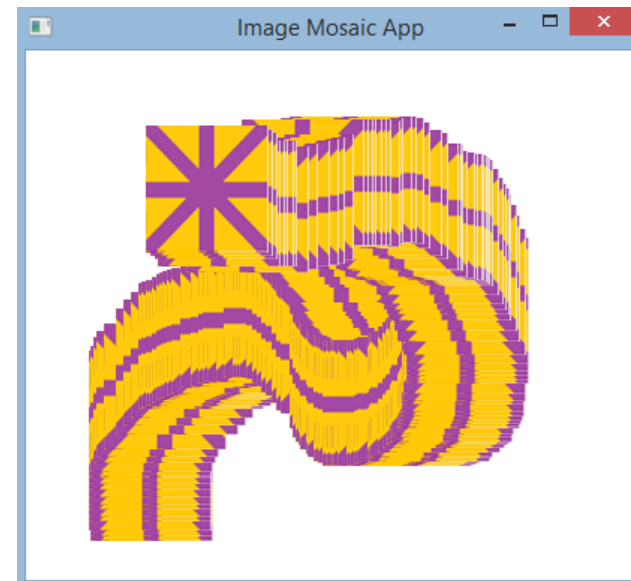
```java
public void initStage() {
    Screen screen = Screen.getPrimary();
    Rectangle2D bounds = screen.getVisualBounds();
    primaryStage.setX(bounds.getMinX());
    primaryStage.setY(bounds.getMinY());
    primaryStage.setWidth(bounds.getWidth());
    primaryStage.setHeight(bounds.getHeight());
}
public void initData() {
    logo1Locations = new ArrayList();
    logo2Locations = new ArrayList();
    logo1Image = new Image("file:images/logo1.png");
    logo2Image = new Image("file:images/logo2.png");
}
public void initGUI() {
    canvas = new Canvas();
    gc = canvas.getGraphicsContext2D(); // is graphics destination: monitor
    Group root = new Group();
    root.getChildren().add(canvas);
    scene = new Scene(root);
    primaryStage.setScene(scene);
    primaryStage.show();
    canvas.setWidth(scene.getWidth());
    canvas.setHeight(scene.getHeight());
}
```

61

```java
public void initHandlers() {
    canvas.setOnMouseClicked(mouseEvent -> {
        Point2D point = new Point2D(mouseEvent.getX(), mouseEvent.getY());
        if (!logo1Locations.contains(point)) {
            logo1Locations.add(point);
        }
        draw();
    });
    canvas.setOnMouseDragged(mouseEvent -> {
        Point2D point = new Point2D(mouseEvent.getX(), mouseEvent.getY());
        if (!logo2Locations.contains(point)) {
            logo2Locations.add(point);
        }
        draw();
    });
}
public void draw() {
    Iterator<Point2D> it = logo1Locations.iterator();
    while (it.hasNext()) {
        Point2D p = it.next();
        gc.drawImage(logo1Image, p.getX(), p.getY());
    }
    it = logo2Locations.iterator();
    while (it.hasNext()) {
        Point2D p = it.next();
        gc.drawImage(logo2Image, p.getX(), p.getY());
    }
}
public static void main(String[] args) {
    launch();
}}
```
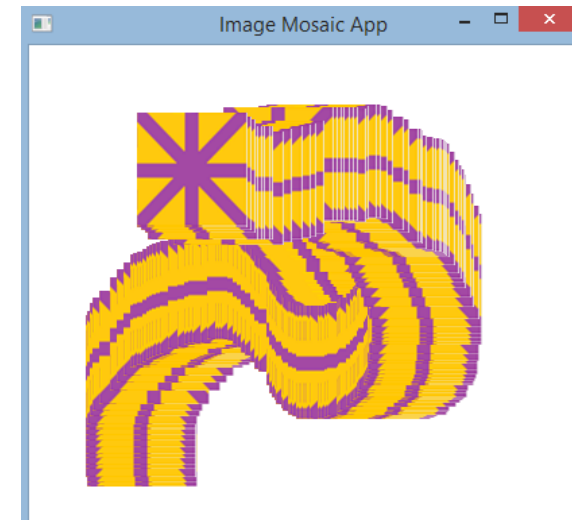

62

Penta App

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.stage.Screen;
import javafx.scene.Group;
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.geometry.Rectangle2D;
import javafx.scene.paint.Color;
import java.util.ArrayList;
public class PentaApp extends Application {
    private Stage primaryStage;
    private Scene scene;
    private Canvas canvas;
    private GraphicsContext gc;
    private ArrayList<double[]> xPoints;
    private ArrayList<double[]> yPoints;
    private ArrayList<Color> colors;

    @Override
    public void start(Stage initPrimaryStage) {
        primaryStage = initPrimaryStage;
        initStage();
        initData();
        initGUI();
        initHandlers();
    }
```

63
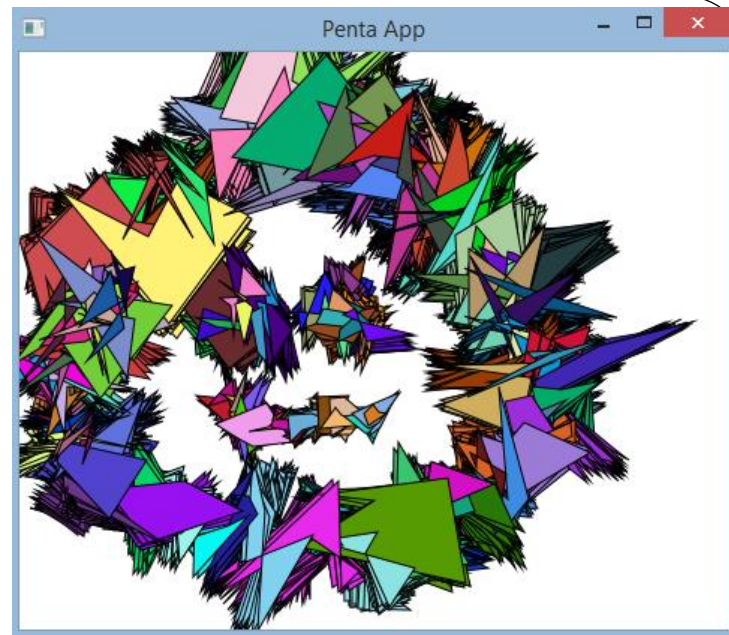
```java
public void initStage() {
    primaryStage.setTitle("Penta App");
    Screen screen = Screen.getPrimary(); // is graphics destination: monitor
    Rectangle2D bounds = screen.getVisualBounds();
    primaryStage.setX(bounds.getMinX());
    primaryStage.setY(bounds.getMinY());
    primaryStage.setWidth(bounds.getWidth());
    primaryStage.setHeight(bounds.getHeight());
}
public void initData() {
    xPoints = new ArrayList();
    yPoints = new ArrayList();
    colors = new ArrayList();
}
public void initGUI() {
    canvas = new Canvas();
    gc = canvas.getGraphicsContext2D();
    Group root = new Group();
    root.getChildren().add(canvas);
    scene = new Scene(root);
    primaryStage.setScene(scene);
    primaryStage.show();
    canvas.setWidth(scene.getWidth());
    canvas.setHeight(scene.getHeight());
}
 public void initHandlers() {
    canvas.setOnMouseClicked(mouseEvent -> {
        if (mouseEvent.getClickCount() == 2) {
            xPoints.clear();
            yPoints.clear();
```

(c) Paul Fodor and Pearson Inc.

```java
                colors.clear();
                gc.clearRect(0, 0, canvas.getWidth(), canvas.getHeight());
            }
        });
        canvas.setOnMouseDragged(mouseEvent -> {
            double x = mouseEvent.getX();
            double y = mouseEvent.getY();
            double[] xs = new double[5];
            double[] ys = new double[5];
            // CENTER
            xs[0] = x;
            ys[0] = y - (int) (Math.random() * 20) - 1;
            // TOP-RIGHT POINT
            xs[1] = x + (int) (Math.random() * 15) + 1;
            ys[1] = y - (int) (Math.random() * 10) - 1;
            // BOTTOM-RIGHT POINT
            xs[2] = x + (int) (Math.random() * 10) + 1;
            ys[2] = y + (int) (Math.random() * 15) + 1;
            // BOTTOM-LEFT POINT
            xs[3] = x - (int) (Math.random() * 10) - 1;
            ys[3] = y + (int) (Math.random() * 15) + 1;
            // TOP-LEFT POINT
            xs[4] = x - (int) (Math.random() * 15) - 1;
            ys[4] = y - (int) (Math.random() * 10) - 1;
            xPoints.add(xs);
            yPoints.add(ys);
            int r = (int) (Math.random() * 256);
            int g = (int) (Math.random() * 256);
            int b = (int) (Math.random() * 256);
            colors.add(Color.rgb(r, g, b));
            PentaApp.this.draw();
        });
    }
```

(c) Paul Fodor and Pearson Inc.

```java
public void draw() {
    for (int i = 0; i < xPoints.size(); i++) {
        double[] xVertices = xPoints.get(i);
        double[] yVertices = yPoints.get(i);
        for (int j = 0; j < 5; j++) {
            xVertices[j] += (int) (Math.random() * 9) - 4;
            yVertices[j] += (int) (Math.random() * 9) - 4;
        }
        Color color = colors.get(i);
        gc.setFill(color);
        gc.fillPolygon(xVertices, yVertices, 5);
        gc.setStroke(Color.BLACK);
        gc.strokePolygon(xVertices, yVertices, 5);
    }
}

public static void main(String[] args) {
    launch(args);
}
}
```
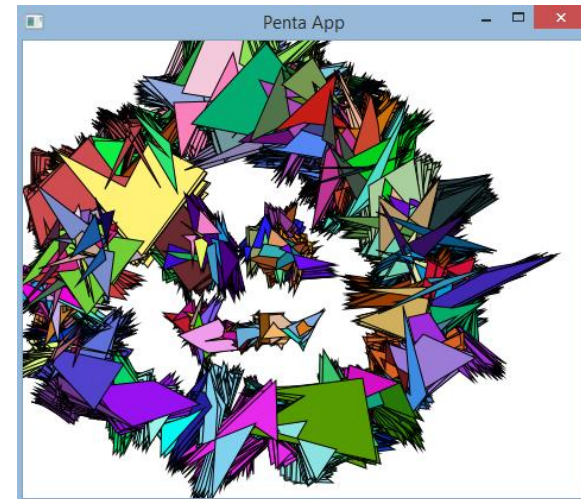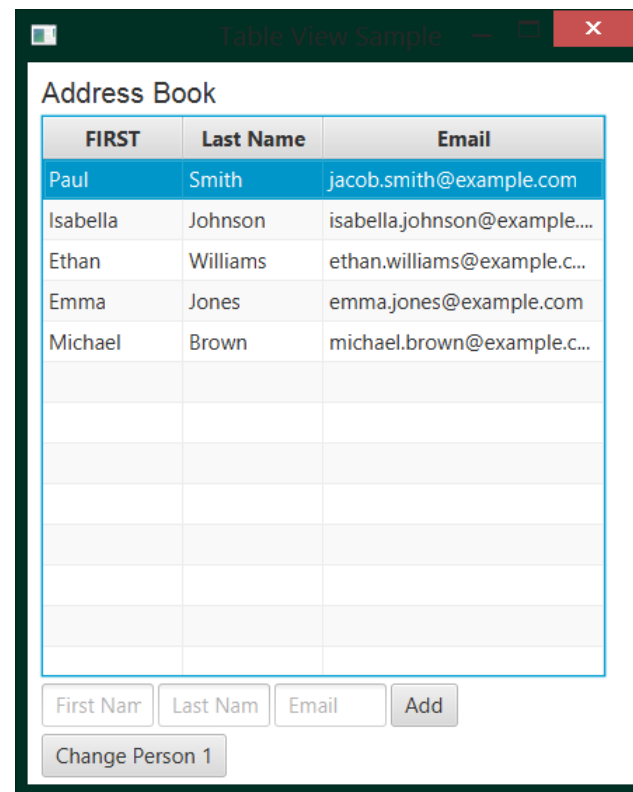
(c) Paul Fodor and Pearson Inc.

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;

class Person {
        private StringProperty firstName;
        private StringProperty lastName;
        private StringProperty email;
        public Person(String fName, String lName, String email) {
                this.firstName = new SimpleStringProperty(fName);
                this.lastName = new SimpleStringProperty(lName);
                this.email = new SimpleStringProperty(email);
        }
```

JavaFXTablesSample

Address Book

FIRST	Last Name	Email
Paul	Smith	jacob.smith@example.com
Isabella	Johnson	isabella.johnson@example....
Ethan	Williams	ethan.williams@example.c...
Emma	Jones	emma.jones@example.com
Michael	Brown	michael.brown@example.c...

First Nam Last Nam Email Add

Change Person 1

67

http://docs.oracle.com/javafx/2/ui_controls/table-view.htm

```java
            public String getFirstName() {
                    return firstName.get();
            }
            public void setFirstName(String fName) {
                    firstName.set(fName);
            }
            public String getLastName() {
                    return lastName.get();
            }
            public void setLastName(String fName) {
                    lastName.set(fName);
            }
            public String getEmail() {
                    return email.get();
            }
            public void setEmail(String fName) {
                    email.set(fName);
            }
    }
    public class JavaFXTablesSample extends Application {
            private TableView<Person> table = new TableView<Person>();
            private ObservableList<Person> data =
              FXCollections.observableArrayList(
                new Person("Jacob", "Smith", "jacob.smith@example.com"),
                new Person("Isabella", "Johnson", "isabella.johnson@example.com"),
                new Person("Ethan", "Williams", "ethan.williams@example.com"),
                new Person("Emma", "Jones", "emma.jones@example.com"),
                new Person("Michael", "Brown", "michael.brown@example.com"));
            HBox hb = new HBox();
```

(c) Paul Fodor and Pearson Inc.

```java
@Override
public void start(Stage stage) {
        Scene scene = new Scene(new Group());
        stage.setTitle("Table View Sample");
        stage.setWidth(450);
        stage.setHeight(550);
        Label label = new Label("Address Book");
        label.setFont(new Font("Arial", 20));
        table.setEditable(true);
        TableColumn firstNameCol = new TableColumn("First Name");
        firstNameCol.setText("FIRST");
        firstNameCol.setMinWidth(100);
        firstNameCol.setCellValueFactory(
                new PropertyValueFactory<Person, String>("firstName"));
        TableColumn lastNameCol = new TableColumn("Last Name");
        lastNameCol.setMinWidth(100);
        lastNameCol.setCellValueFactory(
                new PropertyValueFactory<Person, String>("lastName"));
        TableColumn emailCol = new TableColumn("Email");
        emailCol.setMinWidth(200);
        emailCol.setCellValueFactory(
                new PropertyValueFactory<Person, String>("email"));
        table.setItems(data);

        table.getColumns().addAll(firstNameCol, lastNameCol, emailCol);
        TextField addFirstName = new TextField();
        addFirstName.setPromptText("First Name");
        addFirstName.setMaxWidth(firstNameCol.getPrefWidth());
        TextField addLastName = new TextField();
        addLastName.setMaxWidth(lastNameCol.getPrefWidth());
```

```java
        addLastName.setPromptText("Last Name");
        TextField addEmail = new TextField();
        addEmail.setMaxWidth(emailCol.getPrefWidth());
        addEmail.setPromptText("Email");
        Button addButton = new Button("Add");
        addButton.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent e) {
                        data.add(new Person(
                                        addFirstName.getText(),
                                        addLastName.getText(),
                                        addEmail.getText()));
                        addFirstName.clear();
                        addLastName.clear();
                        addEmail.clear();
                }
        });
        Button changePersonButton = new Button("Change Person 1");
        changePersonButton.setOnAction(new EventHandler<ActionEvent>() {
                @Override
                public void handle(ActionEvent e) {
                        Person p = data.get(0);
                        p.setFirstName("Paul");
                        table.getColumns().get(0).setVisible(false);
                        table.getColumns().get(0).setVisible(true);
                }
        });
    hb.getChildren().addAll(addFirstName,addLastName,addEmail,addButton);
        hb.setSpacing(3);
        VBox vbox = new VBox();
```

(c) Paul Fodor and Pearson Inc.

```
                vbox.setSpacing(5);
                vbox.setPadding(new Insets(10, 0, 0, 10));
                vbox.getChildren().addAll(label, table, hb, changePersonButton);
                ((Group) scene.getRoot()).getChildren().addAll(vbox);
                stage.setScene(scene);
                stage.show();
        }
        public static void main(String[] args){
                launch(args);
        }
}
```