BIRZEIT UNIVERSITY

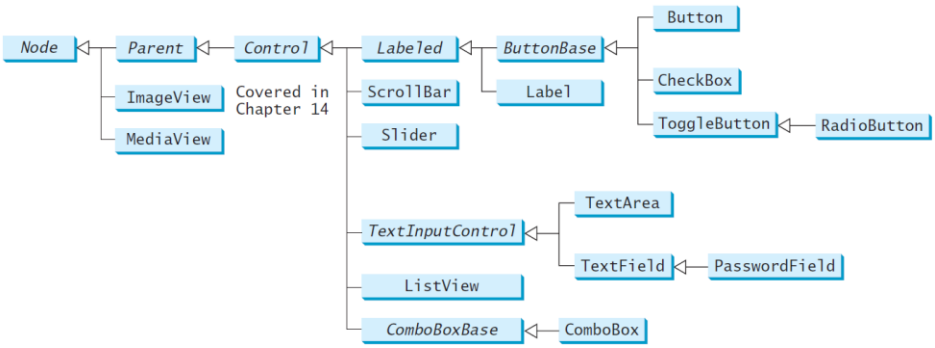
COMPUTER SCIENCE DEPARTMENT FACULTY OF ENGINEERING AND TECHNOLOGY

ADVANCED PROGRAMMING COMP231

Instructor :Murad Njoum
Office : Masri322

Chapter 16
JavaFX UI Controls

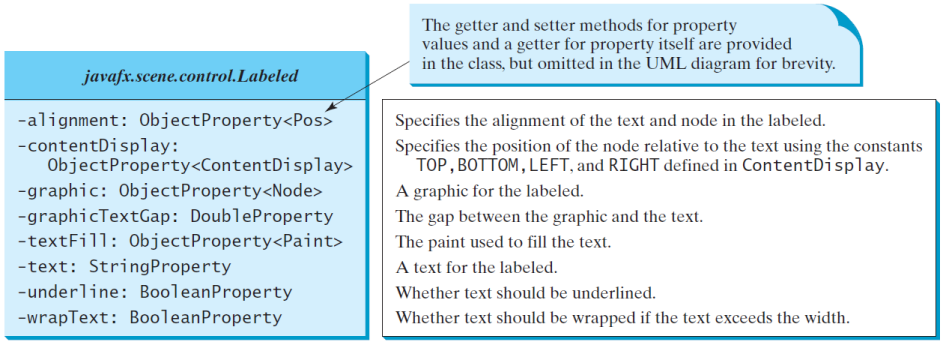
Frequently Used UI Controls



Throughout this book, the prefixes **lbl**, **bt**, **chk**, **rb**, **tf**, **pf**, **ta**, **cbo**, **lv**, **scb**, **sld**, and **mp** are used to name reference variables for **Label**, **Button**, **CheckBox**, **RadioButton**, **TextField**, **PasswordField**, **TextArea**, **ComboBox**, **ListView**, **ScrollBar**, **Slider**, and **MediaPlayer**.

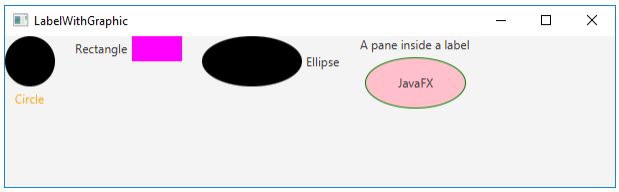
Labeled

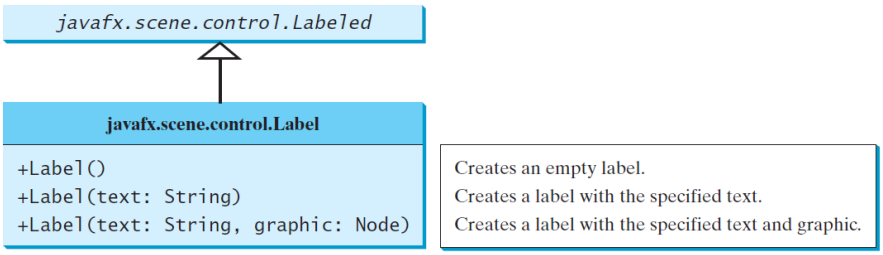
A *label* is a display area for a short text, a node, or both. It is often used to label other controls (usually text fields). Labels and buttons share many common properties. These common properties are defined in the **Labeled** class.



Label

The Label class defines labels.



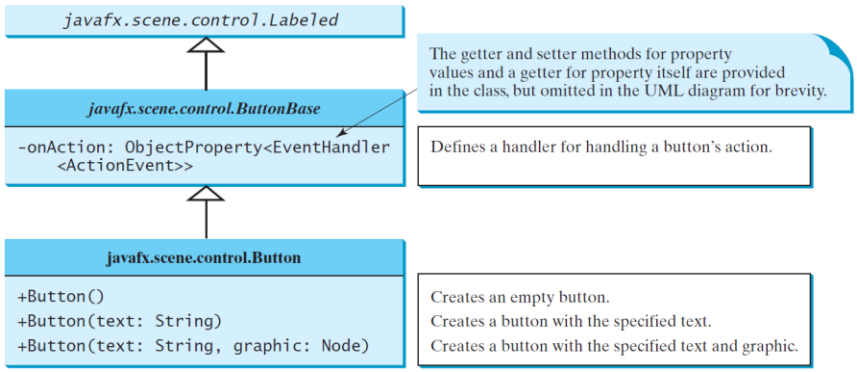


LabelWithGraphic Run

```
Label lb2 = new Label("Circle", new Circle(50, 50, 25));
    lb2.setContentDisplay(ContentDisplay.TOP);
    lb2.setTextFill(Color.ORANGE);
    Rectangle rectangle=new Rectangle(10, 10, 50, 25);
    rectangle.setFill(Color.MAGENTA);
    Label lb3 = new Label("Rectangle",rectangle);
    lb3.setContentDisplay(ContentDisplay.RIGHT);

    Label lb4 = new Label("Ellipse", new Ellipse(50, 50, 50, 25));
    lb4.setContentDisplay(ContentDisplay.LEFT);

    Ellipse ellipse = new Ellipse(50, 50, 50, 25);
    ellipse.setStroke(Color.GREEN);
    ellipse.setFill(Color.PINK);
    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( lb2, lb3, lb4, lb5);
```


ButtonBase and Button

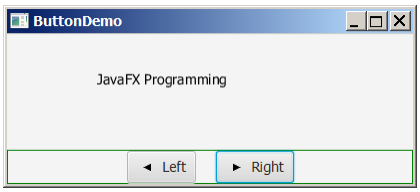
A *button* is a control that triggers an action event when clicked. JavaFX provides regular buttons, toggle buttons, check box buttons, and radio buttons. The common features of these buttons are defined in **ButtonBase** and **Labeled** classes.



Button Example

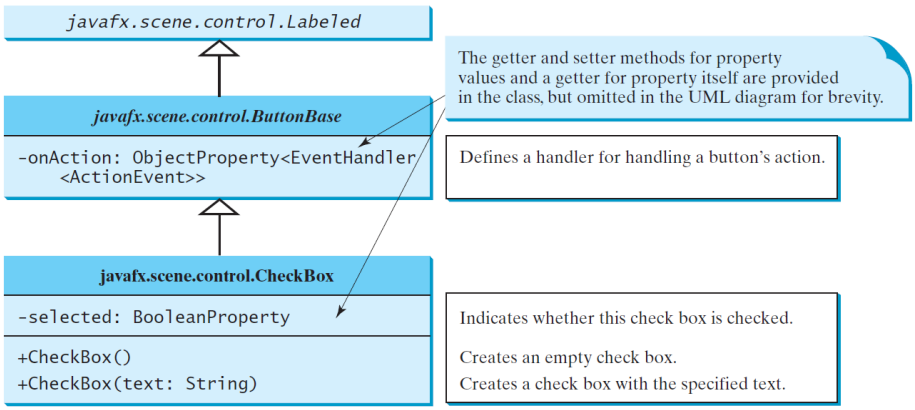
```
protected Text text = new Text(50, 50, "JavaFX Programming");
protected BorderPane getPane() {
    HBox paneForButtons = new HBox(20);
    Button btLeft = new Button("Left",
      new ImageView("image/left.gif"));
    Button btRight = new Button("Right",
      new ImageView("image/right.gif"));
    paneForButtons.getChildren().addAll(btLeft, btRight);
    paneForButtons.setAlignment(Pos.CENTER);


    BorderPane pane = new BorderPane();
    pane.setBottom(paneForButtons);

    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 void start(Stage primaryStage) {
Scene scene = new Scene(getPane(), 450, 200);
    primaryStage.setTitle("ButtonDemo");
primaryStage.setScene(scene);
primaryStage.show();
  }
```



CheckBox

A **CheckBox** is used for the user to make a selection. Like **Button**, **CheckBox** inherits all the properties such as **onAction**, **text**, **graphic**, **alignment**, **graphicTextGap**, **textFill**, **contentDisplay** from **ButtonBase** and **Labeled**.



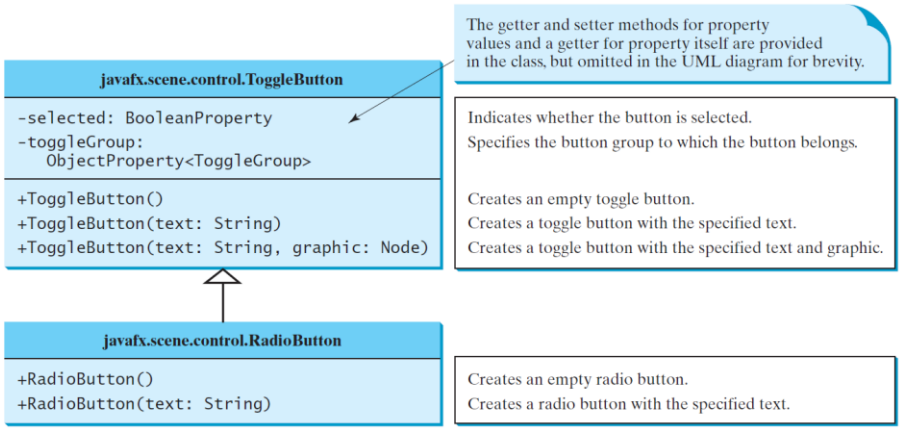
CheckBox Example


```
CheckBox chkBold = new CheckBox("Bold");
CheckBox chkItalic = new CheckBox("Italic");
```

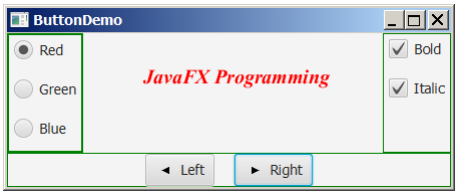
CheckBoxDemo Run

RadioButton

Radio buttons, also known as *option buttons*, enable you to choose a single item from a group of choices. In appearance radio buttons resemble check boxes, but check boxes display a square that is either checked or blank, whereas radio buttons display a circle that is either filled (if selected) or blank (if not selected).



RadioButton Example

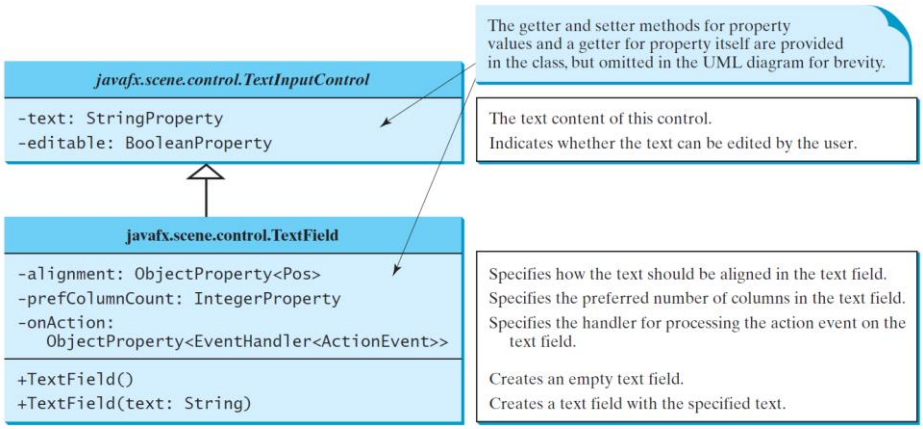



```
RadioButton rbRed = new RadioButton("Red");
RadioButton rbGreen = new RadioButton("Green");
RadioButton rbBlue = new RadioButton("Blue");
paneForRadioButtons.getChildren().addAll(rbRed, rbGreen, rbBlue);


ToggleGroup group = new ToggleGroup();

rbRed.setToggleGroup(group);
rbGreen.setToggleGroup(group);
rbBlue.setToggleGroup(group);
```

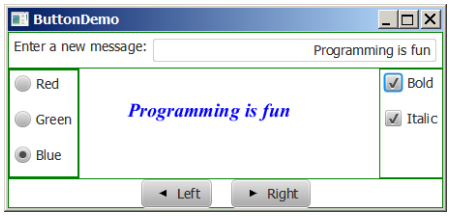
```
rbRed.setOnAction(e -> {
    if (rbRed.isSelected()) {
      text.setFill(Color.RED);
    }
  });

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

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


TextField

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



TextField Example

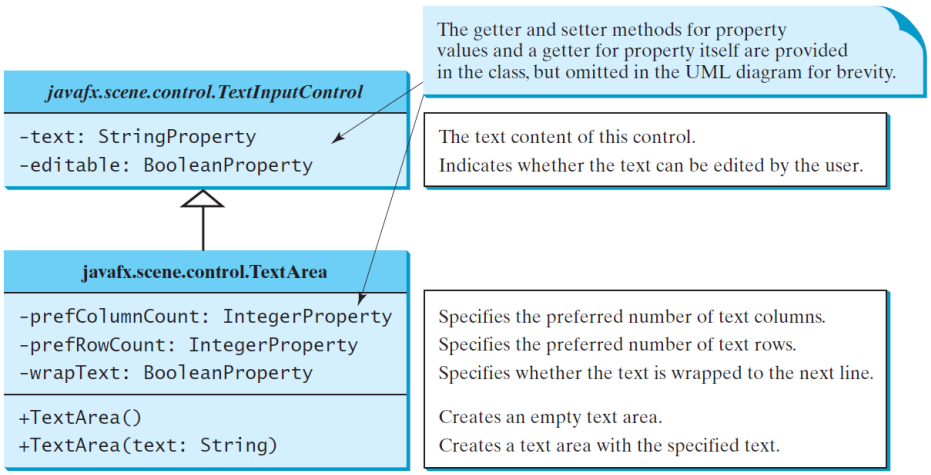



```
BorderPane paneForTextField = new BorderPane();
paneForTextField.setPadding(new Insets(5, 5, 5, 5));
paneForTextField.setStyle("-fx-border-color: green");
paneForTextField.setLeft(new Label("Enter a new message: "));

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

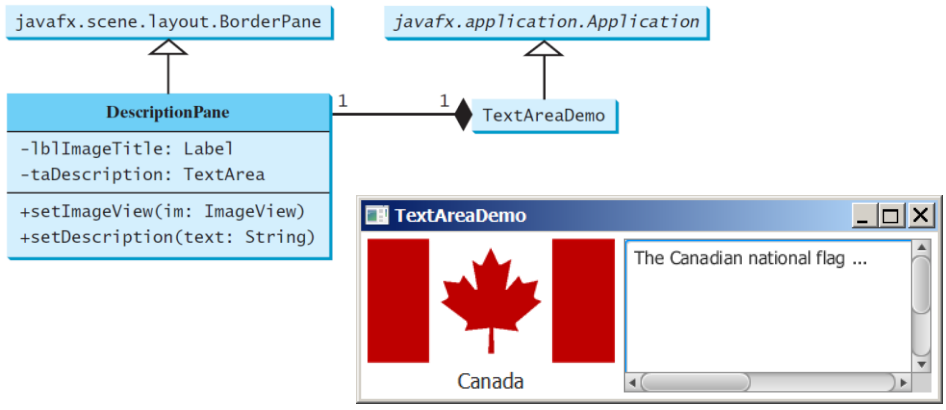
TextFieldDemo Run

TextArea

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



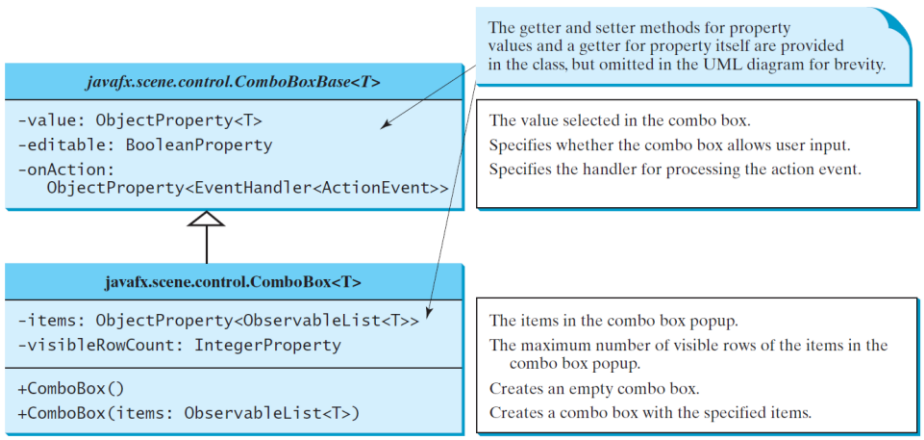
TextArea Example



DescriptionPane TextAreaDemo Run

ComboBox

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



ComboBox Example

This example lets users view an image and a description of a country's flag by selecting the country from a combo box.

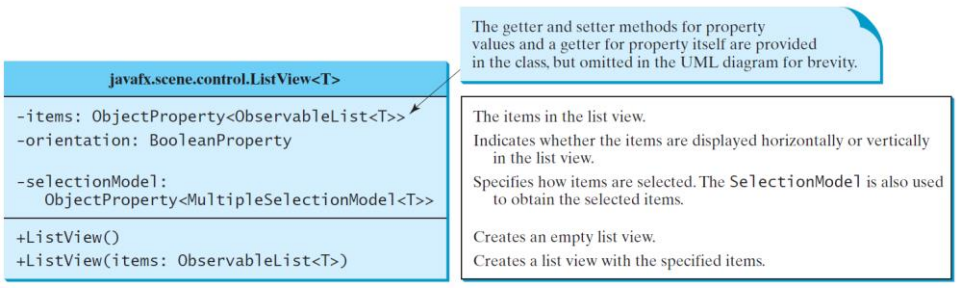
```
ComboBox<String> cbo = new ComboBox<>();
```


```
ObservableList<String> items = FXCollections.observableArrayList(flagTitles);
cbo.getItems().addAll(items);
```

ComboBoxDemo Run

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.



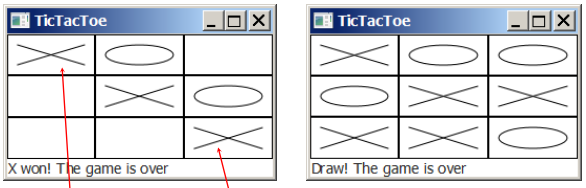
Example: Using ListView

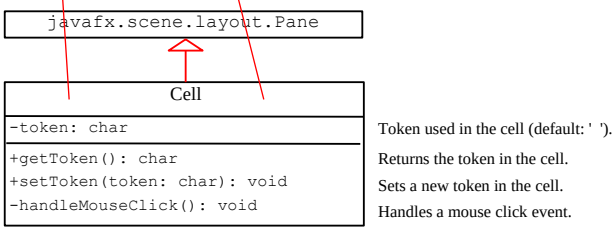
This example gives a program that lets users select countries in a list and display the flags of the selected countries in the labels.


```
ListView<String> lv = new ListView<> (FXCollections.observableArrayList(flagTitles));
```

ListViewDemo Run

Case Study: TicTacToe





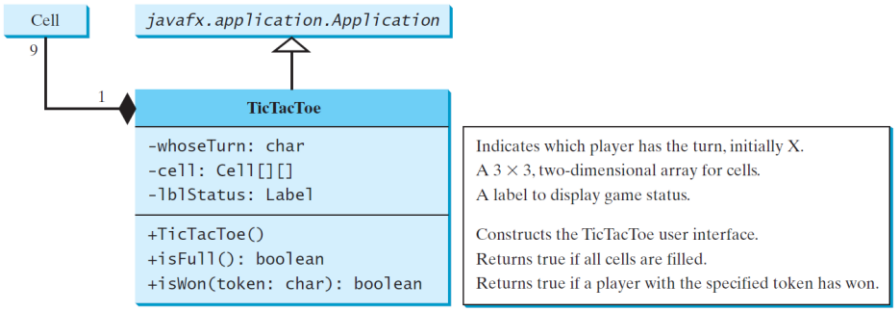
Token used in the cell (default: ' ').

Returns the token in the cell.

Sets a new token in the cell.

Handles a mouse click event.

Case Study: TicTacToe, cont.



TicTacToe Run