JavaFX UI Controls

Name Dr. Abdallah Karakra |Comp 2311 | Masri521

21/01/2023

جامعة بيرزيت
BIRZEIT UNIVERSITY

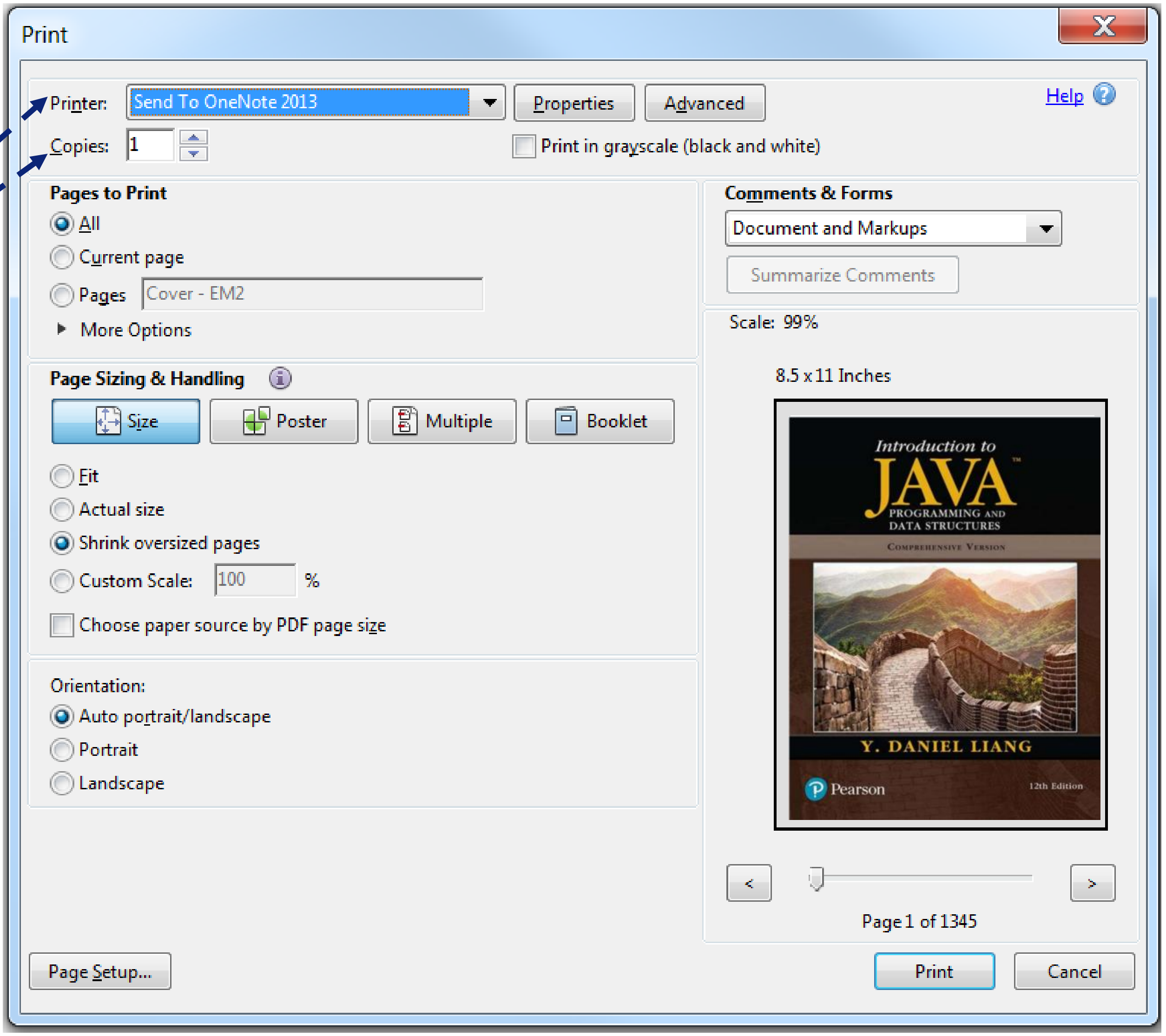
Index

CHAPTER 16

JavaFX UI Controls and Multimedia

Introduction

Introduction


Print

Printer: Send To OneNote 2013 | Properties | Advanced | Help

Copies: 1 | Print in grayscale (black and white)

Pages to Print

- (●) All
- ( ) Current page
- ( ) Pages: Cover - EM2
- More Options

Comments & Forms: Document and Markups | Summarize Comments

Scale: 99%

8.5 x 11 Inches

Page Sizing & Handling: Size | Poster | Multiple | Booklet

- ( ) Fit
- ( ) Actual size
- (●) Shrink oversized pages
- ( ) Custom Scale: 100 %
- [ ] Choose paper source by PDF page size

Orientation:

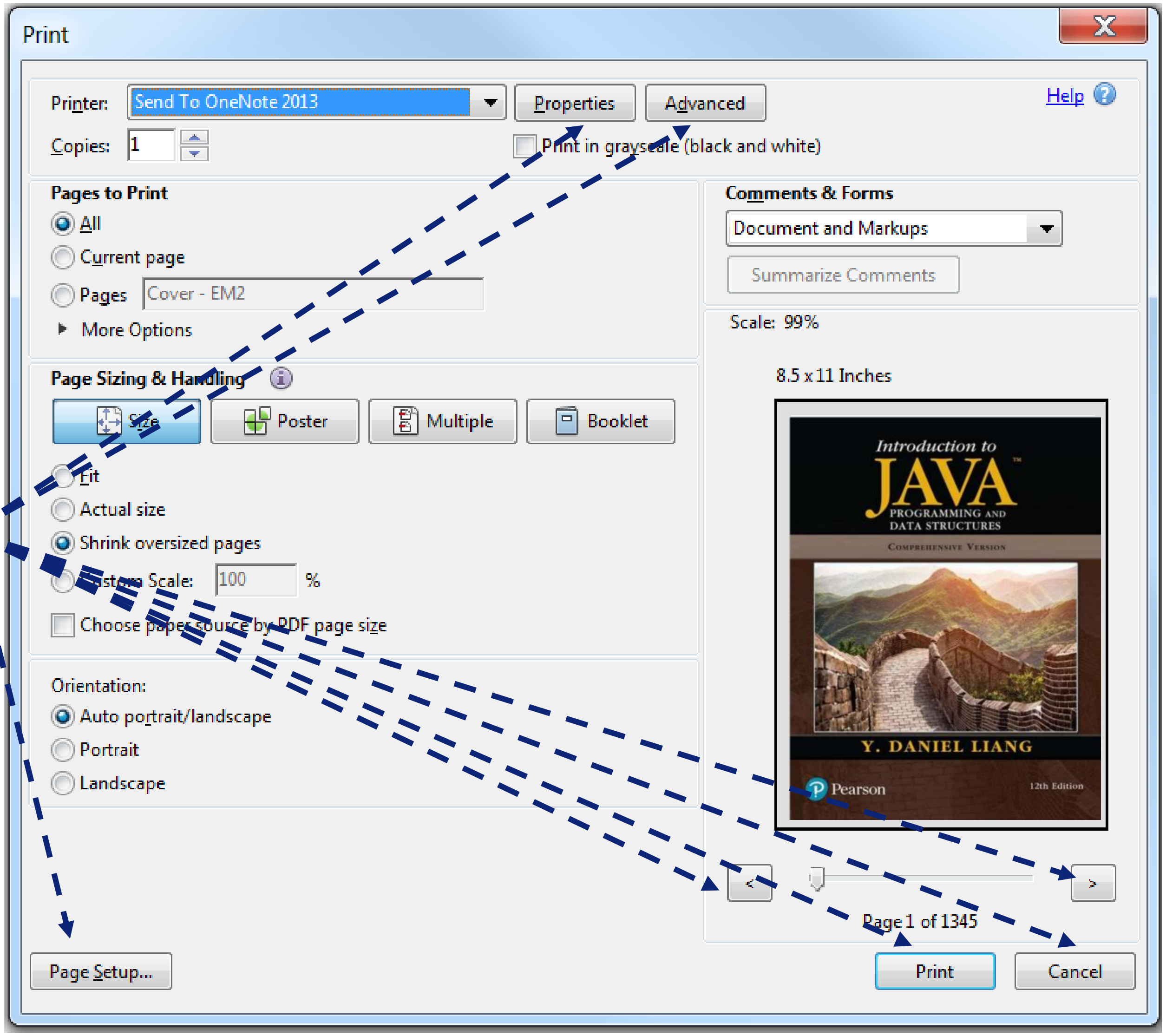
- (●) Auto portrait/landscape
- ( ) Portrait
- ( ) Landscape

Page 1 of 1345

Page Setup... | Print | Cancel

`Labels` – just a piece of text on the UI to show the user what the control NEXT to it is for

Introduction


Print

Printer: Send To OneNote 2013

Copies: 1

Print in grayscale (black and white): [ ]

Pages to Print: [✓ All] Current page Pages (Cover - EM2)

Page Sizing & Handling: [✓ Size]

Fit | Actual size | [✓ Shrink oversized pages] | Custom Scale: 100 %

Choose paper source by PDF page size: [ ]

Orientation: [✓ Auto portrait/landscape] Portrait Landscape

Comments & Forms: Document and Markups

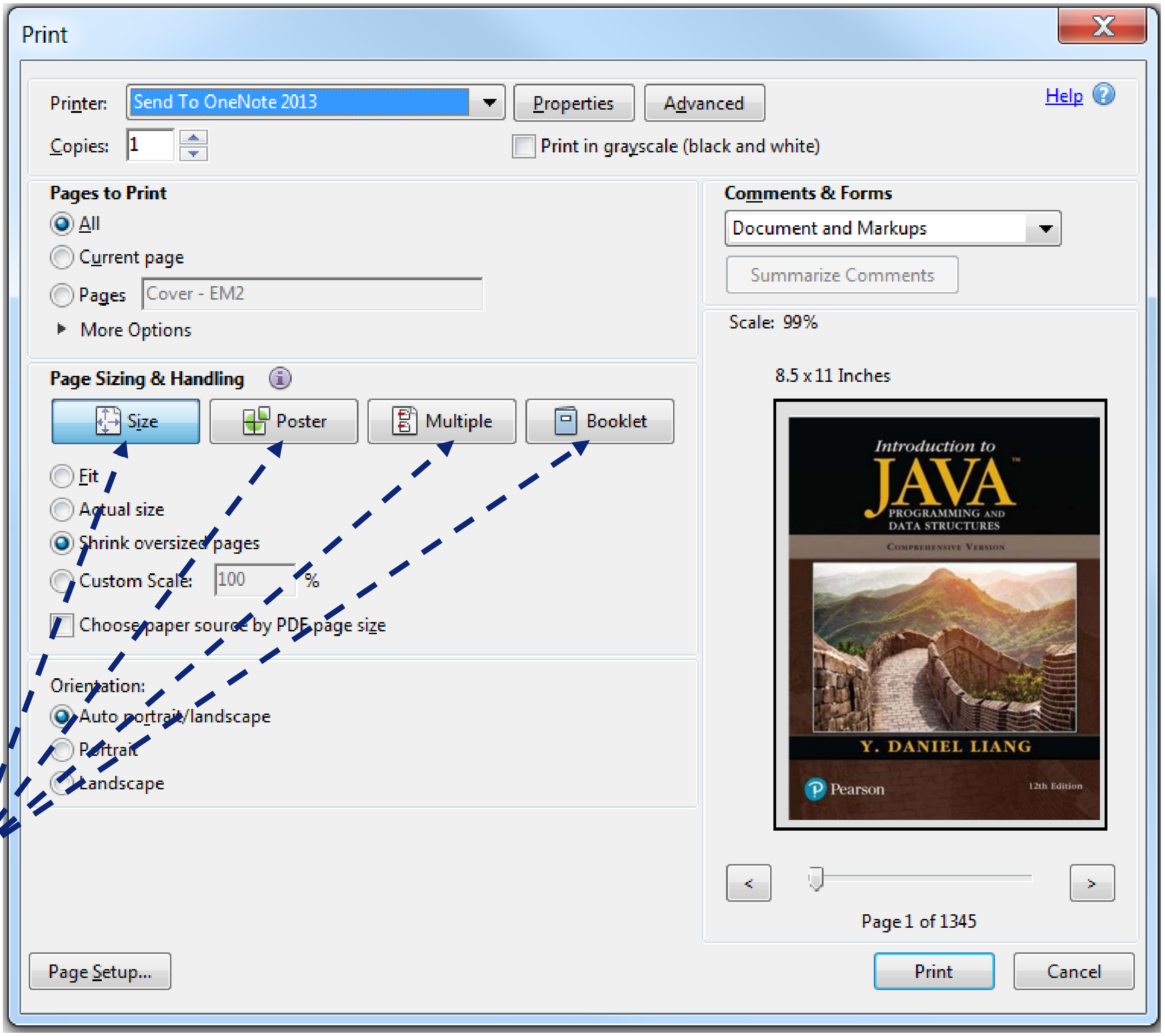
Scale: 99%

8.5 x 11 Inches

Page 1 of 1345

Buttons with a label to tell us what clicking on the button should do

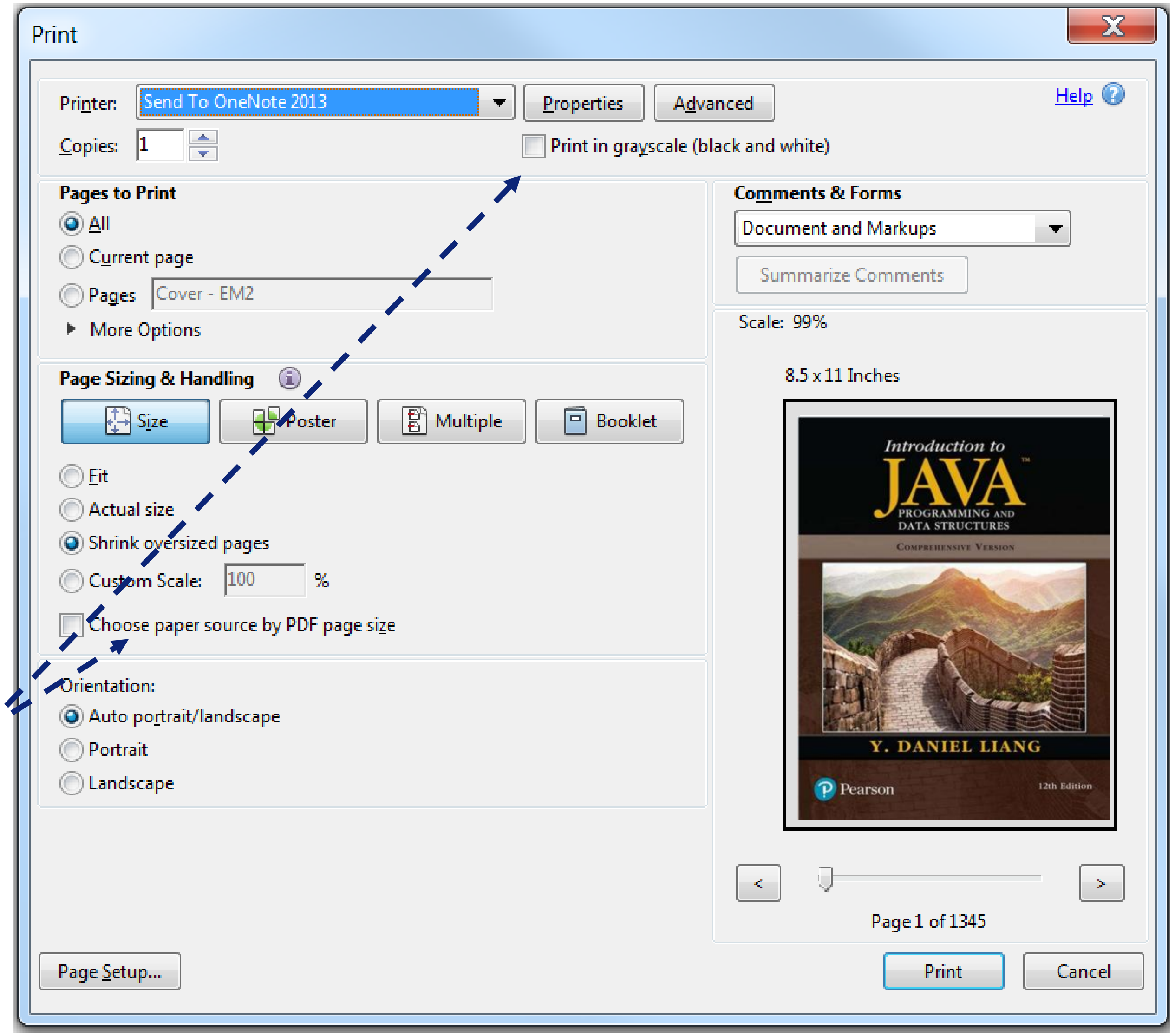
Introduction



Printer: Send To OneNote 2013
Copies: 1
Print in grayscale (black and white): [ ]
Pages to Print: [✓ All] Current page Pages (Cover - EM2)
Comments & Forms: Document and Markups
Scale: 99%
Page Sizing & Handling: [✓ Size] Poster Multiple Booklet
Fit / Actual size / [✓ Shrink oversized pages] / Custom Scale: 100 %
Choose paper source by PDF page size: [ ]
Orientation: [✓ Auto portrait/landscape] Portrait Landscape

`Button`s with both a label *and* an image

Introduction


Print

Printer: Send To OneNote 2013 | Properties | Advanced | Help

Copies: 1 | ☐ Print in grayscale (black and white)

Pages to Print
- [✓ All]
- Current page
- Pages: Cover - EM2
- More Options

Page Sizing & Handling

[✓ Size] | Poster | Multiple | Booklet

- Fit
- Actual size
- [✓ Shrink oversized pages]
- Custom Scale: 100 %

☐ Choose paper source by PDF page size

Orientation:
- [✓ Auto portrait/landscape]
- Portrait
- Landscape

Comments & Forms

Document and Markups

Summarize Comments

Scale: 99%

8.5 x 11 Inches

Page 1 of 1345

Page Setup... | Print | Cancel

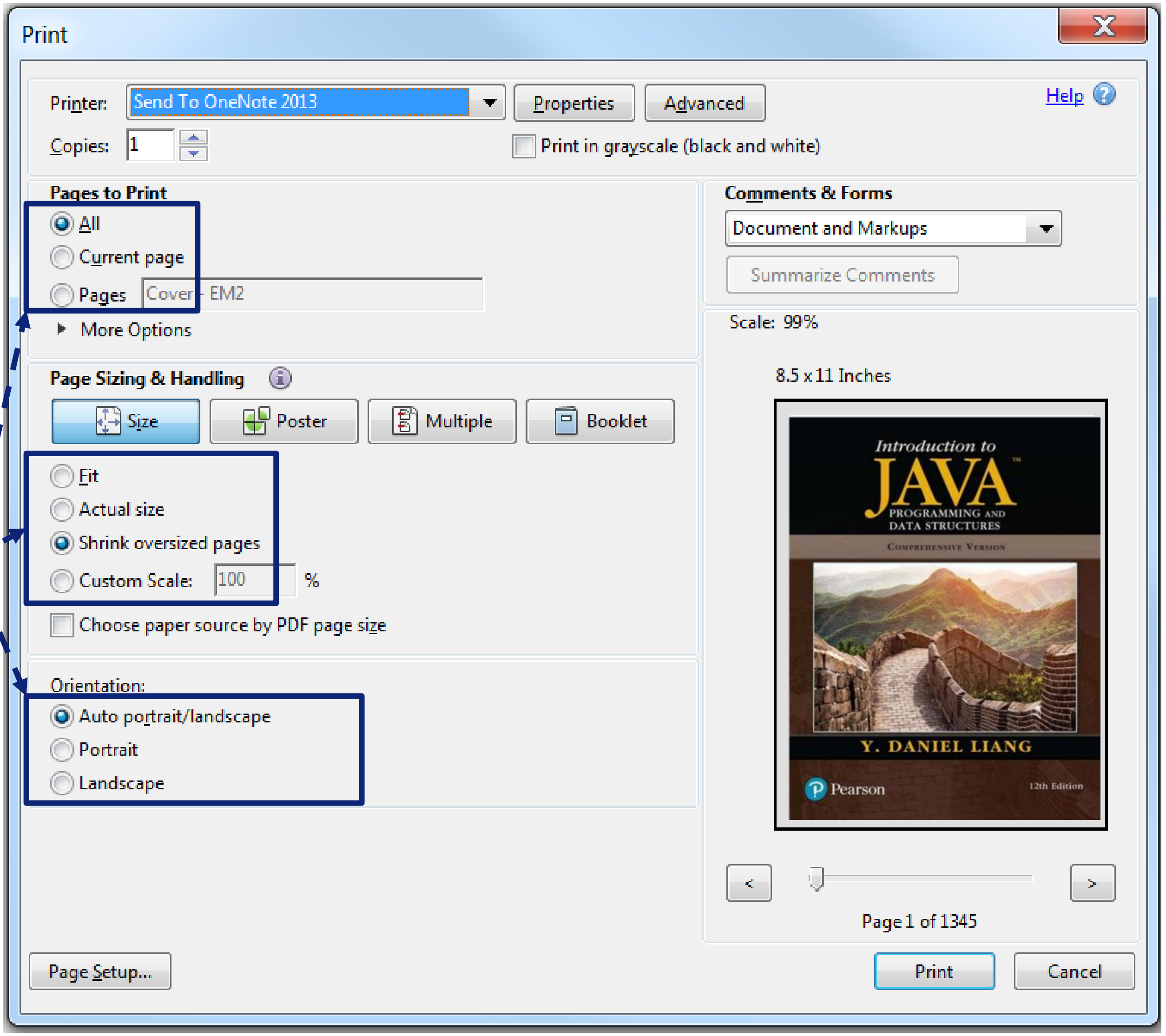
CheckBox with a *square* box to the left of its label that lets us turn on or off some Boolean value

Introduction

RadioButtons, arranged in groups.

Only one button in a group can be selected (marked) at any one time – selecting one *deselects* the others in the group

Each RadioButton has a *round* "selected" indicator and a label



Print

Printer: Send To OneNote 2013 | Properties | Advanced | Help

Copies: 1 | Print in grayscale (black and white)

Pages to Print
- (●) All
- ( ) Current page
- ( ) Pages: Cover - EM2
- More Options

Comments & Forms: Document and Markups | Summarize Comments

Scale: 99%

8.5 x 11 Inches

Page Sizing & Handling: Size | Poster | Multiple | Booklet
- ( ) Fit
- ( ) Actual size
- (●) Shrink oversized pages
- ( ) Custom Scale: 100 %
- Choose paper source by PDF page size

Orientation:
- (●) Auto portrait/landscape
- ( ) Portrait
- ( ) Landscape

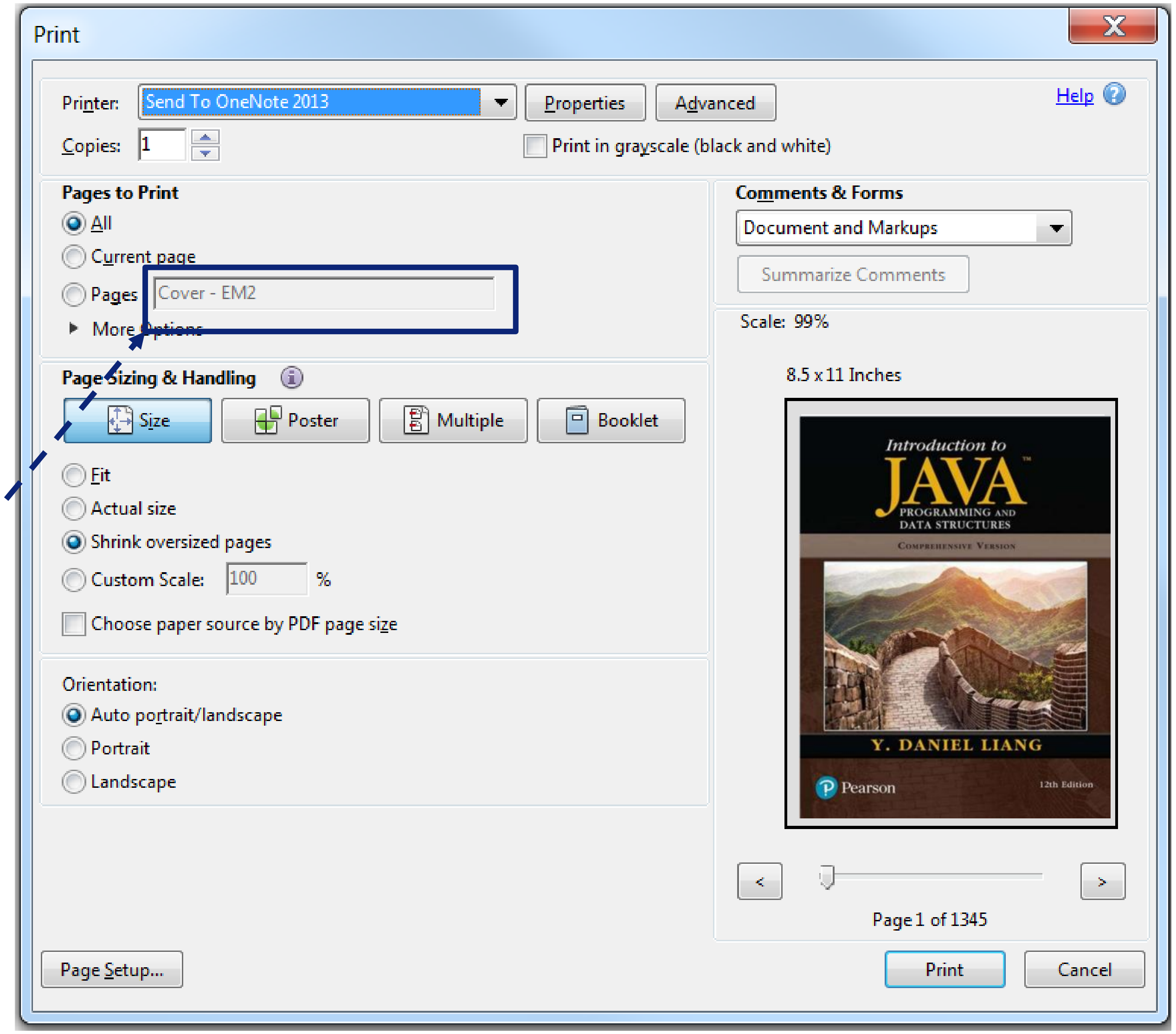
Page 1 of 1345

Page Setup... | Print | Cancel

Introduction

`TextField`, into which the user may type text.

If the text is inherently numeric in nature, we will have to parse the `TextField`'s contents from `String` to a numeric type before we can use it in a calculation



Print

Printer: Send To OneNote 2013 | Properties | Advanced | Help

Copies: 1 | Print in grayscale (black and white)

Pages to Print
- [✓ All]
- Current page
- Pages: Cover - EM2
- More Options

Page Sizing & Handling: [✓ Size] | Poster | Multiple | Booklet
- Fit
- Actual size
- [✓ Shrink oversized pages]
- Custom Scale: 100 %
- Choose paper source by PDF page size

Orientation:
- [✓ Auto portrait/landscape]
- Portrait
- Landscape

Comments & Forms: Document and Markups | Summarize Comments

Scale: 99%

8.5 x 11 Inches

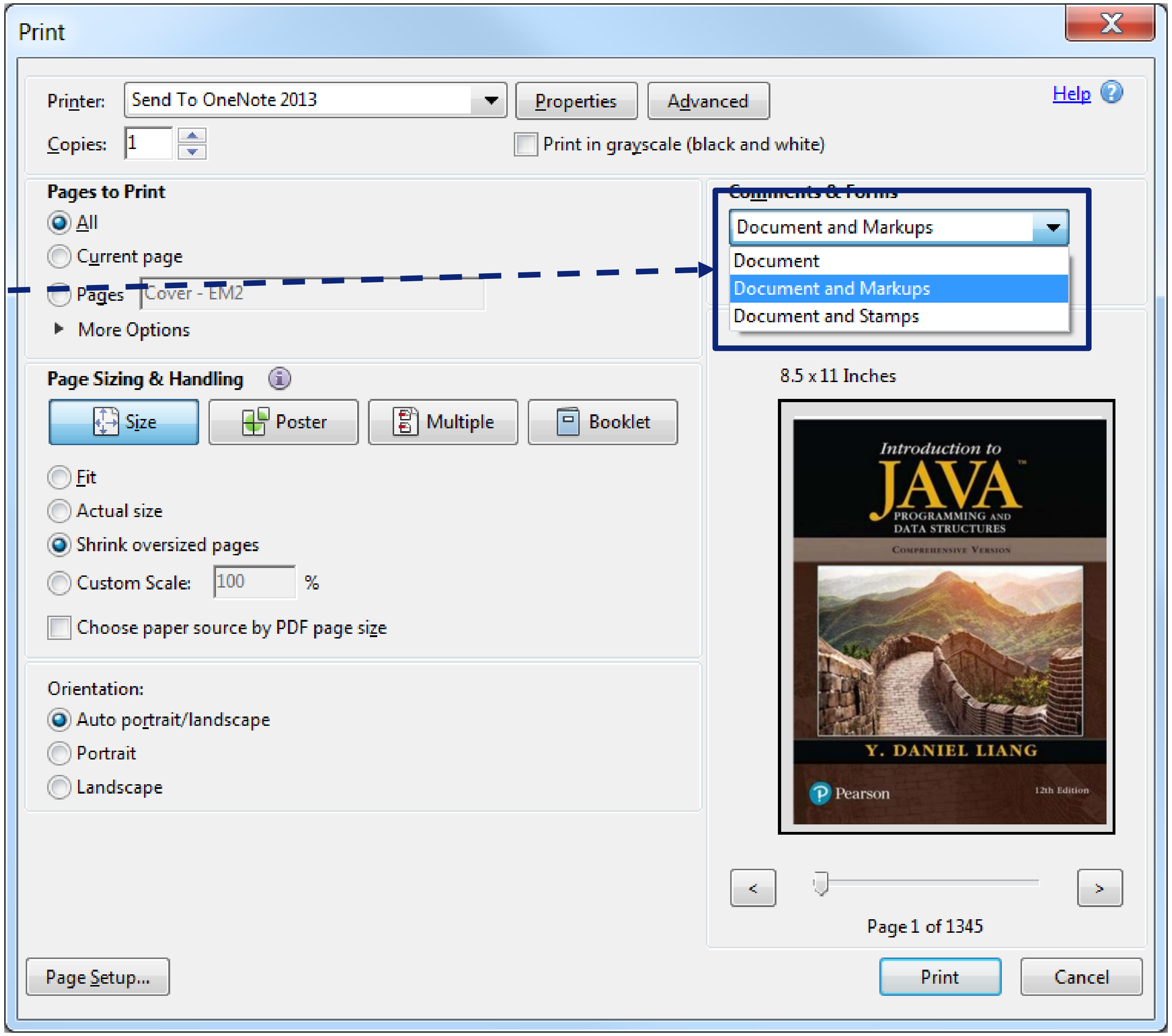
Introduction to JAVA™ Programming and Data Structures, Comprehensive Version, Y. Daniel Liang, Pearson, 12th Edition

Page 1 of 1345

Page Setup... | Print | Cancel

Introduction

ComboBox, which lets the user drop-down a list of options from which to select


Print

Printer: Send To OneNote 2013 | Properties | Advanced | Help

Copies: 1 | Print in grayscale (black and white)

Pages to Print
- [✓ All]
- Current page
- Pages: Cover - EM2
- More Options

Comments & Forms
Document and Markups
- Document
- Document and Markups
- Document and Stamps

Page Sizing & Handling
Size | Poster | Multiple | Booklet
- Fit
- Actual size
- [✓ Shrink oversized pages]
- Custom Scale: 100 %
- Choose paper source by PDF page size

Orientation:
- [✓ Auto portrait/landscape]
- Portrait
- Landscape

8.5 x 11 Inches

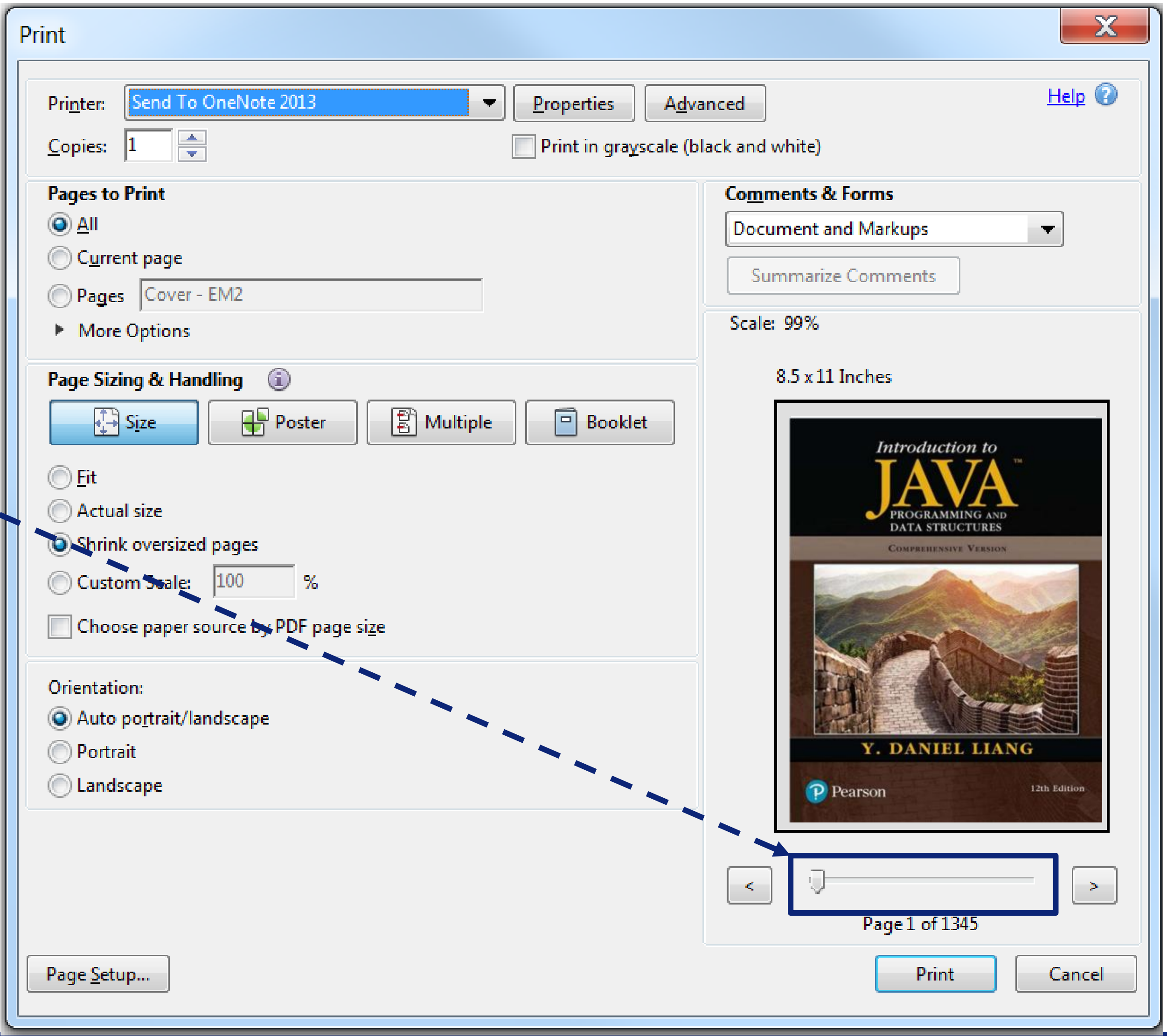
Page 1 of 1345

Page Setup... | Print | Cancel

Introduction

Sliders are similar to ScrollBars (which this UI doesn't happen to have, but which you are, no doubt, already familiar with for scrolling the screen).

Sliders differ from ScrollBars, though, in that a Slider lets us select a value from a range, whereas a ScrollBar is usually used to let us scroll content that doesn't fit into its container



Print

Printer: Send To OneNote 2013 | Properties | Advanced | Help

Copies: 1 | Print in grayscale (black and white)

Pages to Print

[✓ All]
Current page
Pages: Cover - EM2
More Options

Comments & Forms

Document and Markups

Summarize Comments

Scale: 99%

8.5 x 11 Inches

Page Sizing & Handling

[✓ Size] | Poster | Multiple | Booklet

Fit
Actual size
[✓ Shrink oversized pages]
Custom Scale: 100 %
Choose paper source by PDF page size

Orientation:
[✓ Auto portrait/landscape]
Portrait
Landscape

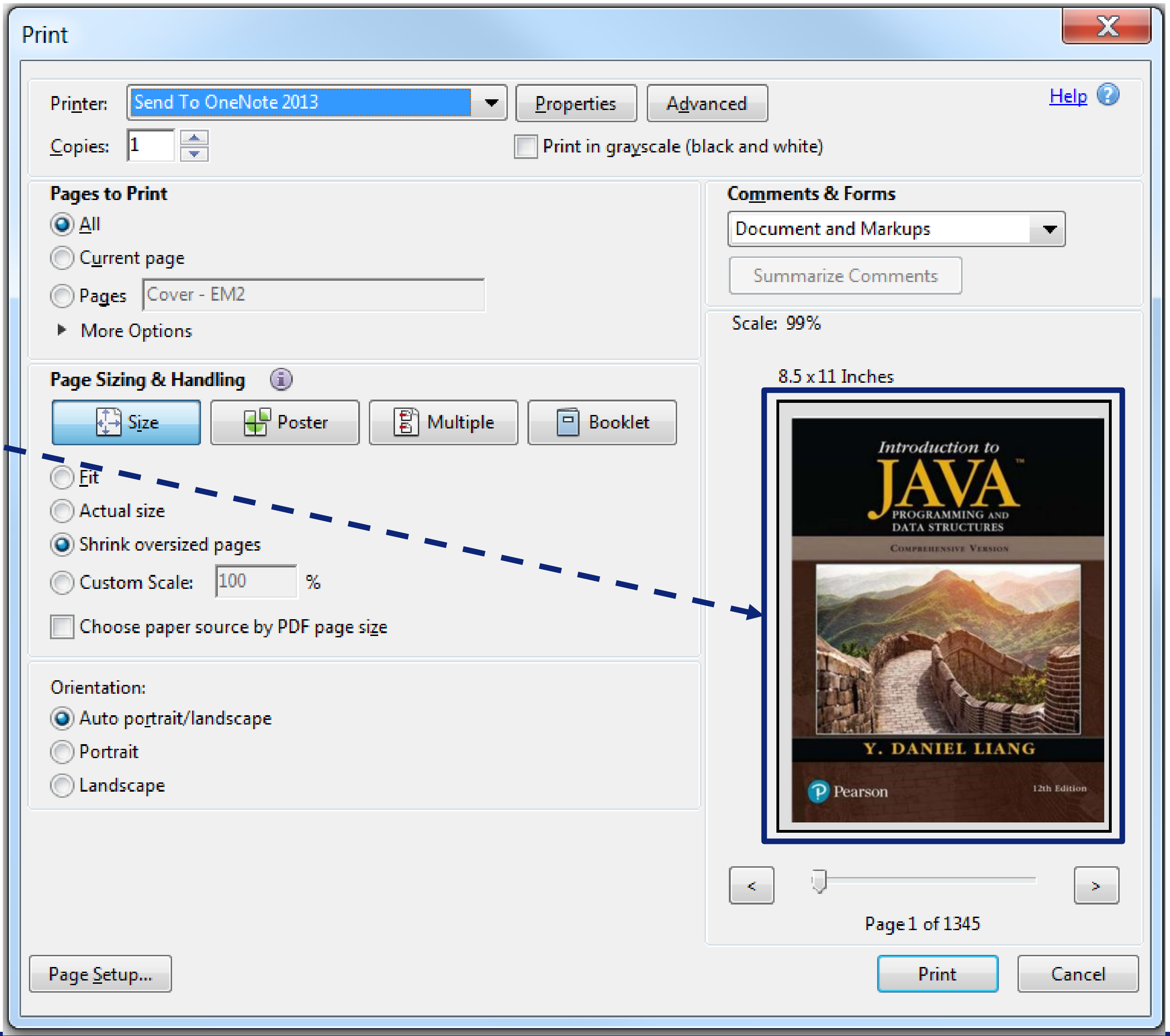
Page 1 of 1345

Page Setup... | Print | Cancel

Introduction

The `ImageView` can be used to either provide static information, or the image it displays can change depending on what the user does while in the interface

In this example, the image displayed is the one that corresponds to the page selected by the `Slider` below the `ImageView`


Print

Printer: Send To OneNote 2013 | Properties | Advanced | Help

Copies: 1 | Print in grayscale (black and white)

Pages to Print
- [✓ All]
- Current page
- Pages: Cover - EM2
- More Options

Comments & Forms: Document and Markups | Summarize Comments

Scale: 99%

8.5 x 11 Inches

Page Sizing & Handling: [✓ Size] | Poster | Multiple | Booklet

- Fit
- Actual size
- [✓ Shrink oversized pages]
- Custom Scale: 100 %
- Choose paper source by PDF page size

Orientation:
- [✓ Auto portrait/landscape]
- Portrait
- Landscape

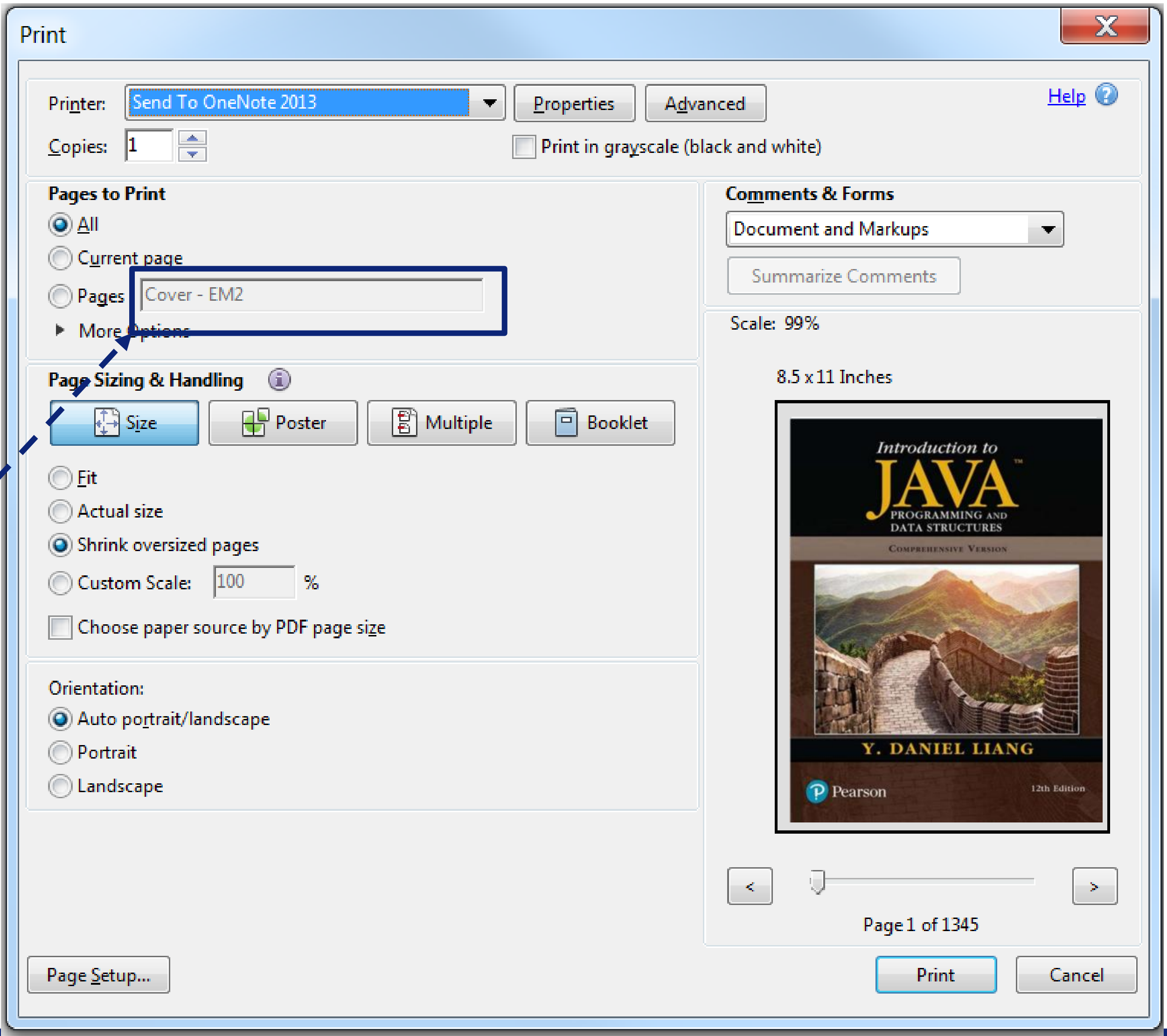
Page 1 of 1345

Page Setup... | Print | Cancel

Introduction

UI elements can be either *enabled* (allowing the user to interact with them), or *disabled*, in which case they're present and visible, but "deactivated" or ("grayed out")

In this example, the page range TextBox is disabled when the "Pages" RadioButton is not selected – the TextBox is *there*, but we can't type in it


Print

Printer: Send To OneNote 2013

Copies: 1

☐ Print in grayscale (black and white)

Pages to Print
● All
○ Current page
○ Pages Cover - EM2

Page Sizing & Handling
○ Fit
○ Actual size
● Shrink oversized pages
○ Custom Scale: 100 %
☐ Choose paper source by PDF page size

Orientation:
● Auto portrait/landscape
○ Portrait
○ Landscape

Comments & Forms
Document and Markups

Scale: 99%

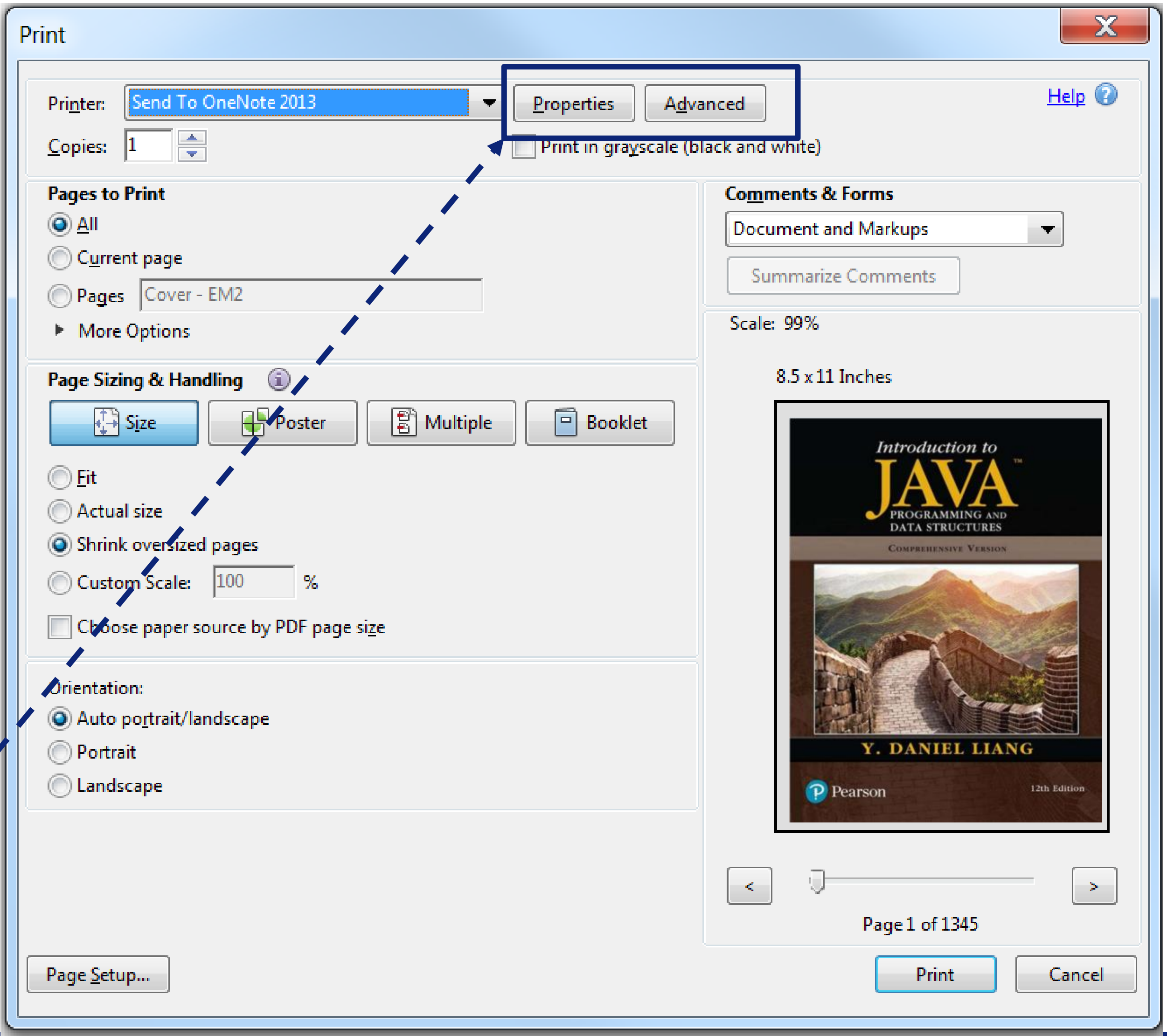
8.5 x 11 Inches

Page 1 of 1345

Introduction

It's usually *easier* to navigate a GUI with the mouse, but it's typically much *faster* to do so it with the keyboard, and there are many keyboard shortcuts available.

First, controls with an underlined letter in their label can be selected by using ALT and the underlined letter (ALT+P from the keyboard is the same as clicking on the "Properties" `Button`)


Print

Printer: Send To OneNote 2013
Copies: 1
Print in grayscale (black and white)

Pages to Print
[✓ All]
Current page
Pages: Cover - EM2
More Options

Comments & Forms: Document and Markups

Scale: 99%
8.5 x 11 Inches

Page Sizing & Handling
[✓ Size] Poster Multiple Booklet
Fit
Actual size
[✓ Shrink oversized pages]
Custom Scale: 100 %
Choose paper source by PDF page size

Orientation:
[✓ Auto portrait/landscape]
Portrait
Landscape

Page 1 of 1345

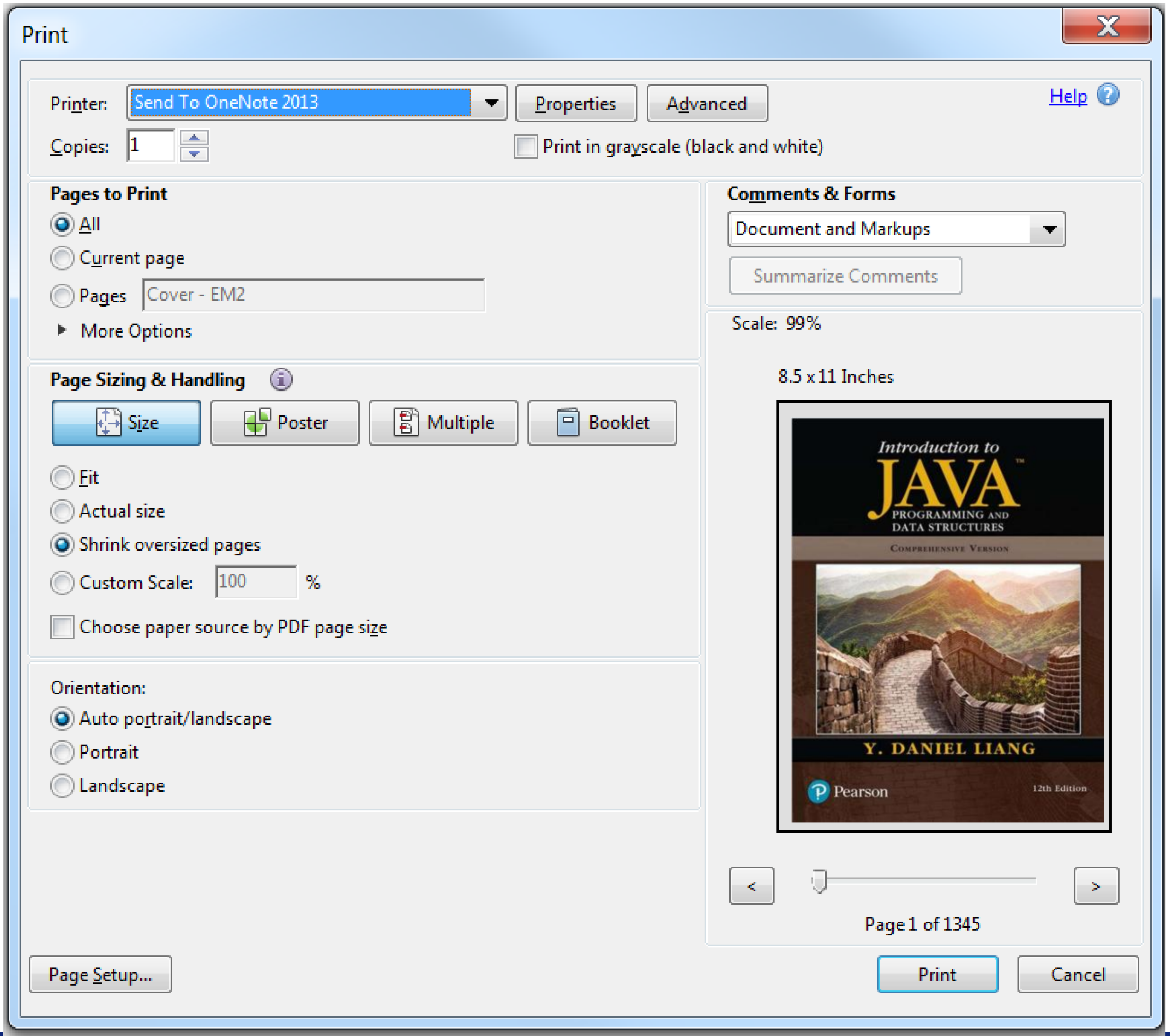
Introduction

What about the controls that we can't select with ALT?

The Tab key shifts the focus from control to control in a pre-set order

When a `Button` or a `CheckBox` has the focus, pressing the Space Bar generates a click event

When a `RadioButton` or a `ComboBox` has the focus, UP / DOWN selects a different item



Print

Printer: Send To OneNote 2013

Copies: 1

☐ Print in grayscale (black and white)

Pages to Print
- [✓ All]
- Current page
- Pages: Cover - EM2

Comments & Forms: Document and Markups

Page Sizing & Handling: Size
- Fit
- Actual size
- [✓ Shrink oversized pages]
- Custom Scale: 100 %

☐ Choose paper source by PDF page size

Orientation:
- [✓ Auto portrait/landscape]
- Portrait
- Landscape

Scale: 99%

8.5 x 11 Inches

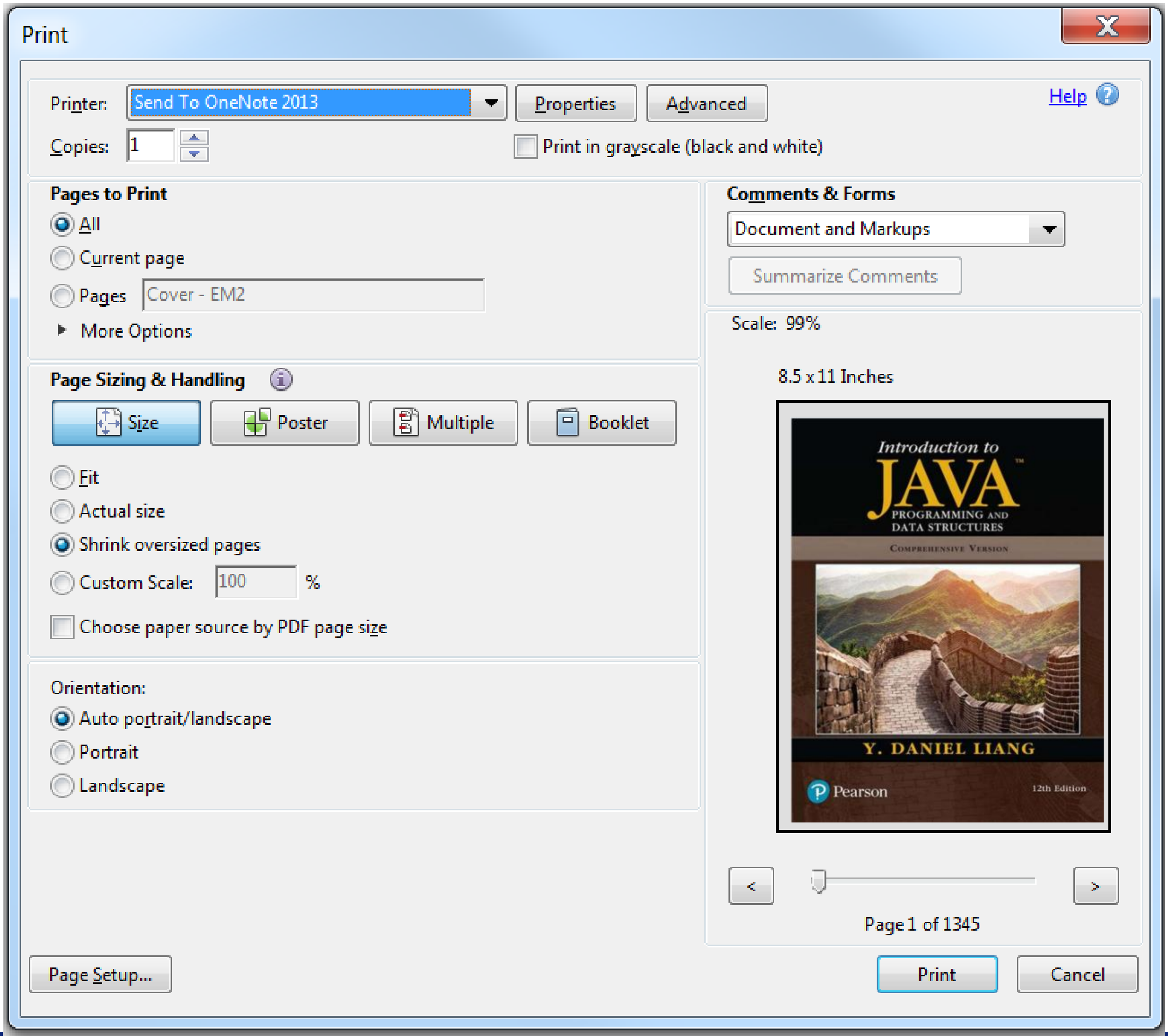
Page 1 of 1345

Introduction

When a horizontal `Slider` has the focus, the LEFT / RIGHT arrow keys change its value.

When `ComboBox` has the focus, the F4 key will make the box alternate between its dropped-down and collapsed views

The more you can rely on the keyboard (and less on the mouse), the more productive you will be on the computer!



Print

Printer: Send To OneNote 2013

Copies: 1

Print in grayscale (black and white): [ ]

Pages to Print

[✓ All] / Current page / Pages: Cover - EM2

More Options

Comments & Forms: Document and Markups

Scale: 99%

8.5 x 11 Inches

Page Sizing & Handling: [✓ Size] / Poster / Multiple / Booklet

Fit / Actual size / [✓ Shrink oversized pages] / Custom Scale: 100 %

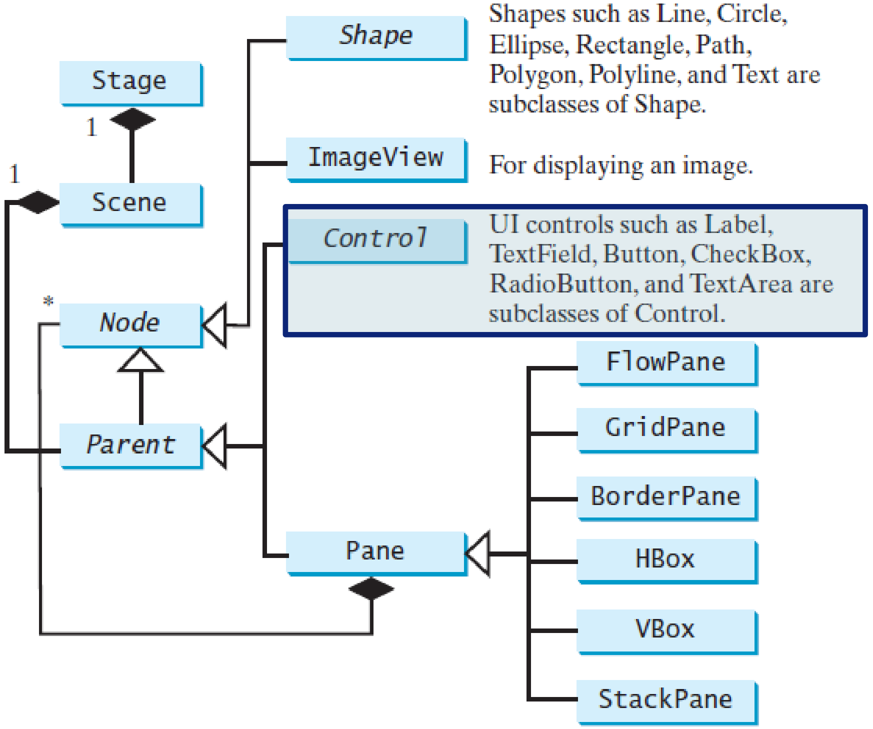
Choose paper source by PDF page size: [ ]

Orientation: [✓ Auto portrait/landscape] / Portrait / Landscape

Page 1 of 1345

Introduction

- We started Chapter 14 with this diagram:
- This is great, but that whole branch that just says *Control* has a lot of missing pieces
- This chapter goes into the details of what the primary controls *are*, and how we can use them to build a richer UI.
- The next slide shows us what "lies beyond"…


Stage
1
Scene
1
*
Node
Parent
Shape
Shapes such as Line, Circle, Ellipse, Rectangle, Path, Polygon, Polyline, and Text are subclasses of Shape.
ImageView
For displaying an image.
Control
UI controls such as Label, TextField, Button, CheckBox, RadioButton, and TextArea are subclasses of Control.
Pane
FlowPane
GridPane
BorderPane
HBox
VBox
StackPane

A Notational Note

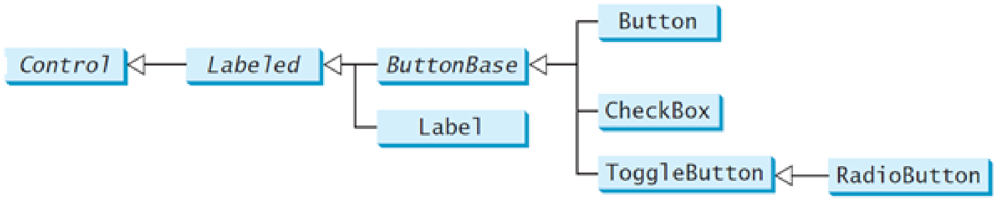
- The author notes that the book will consistently use the following prefixes for the various node types, making it easy to tell by looking at a node's variable name, what type of node it is:

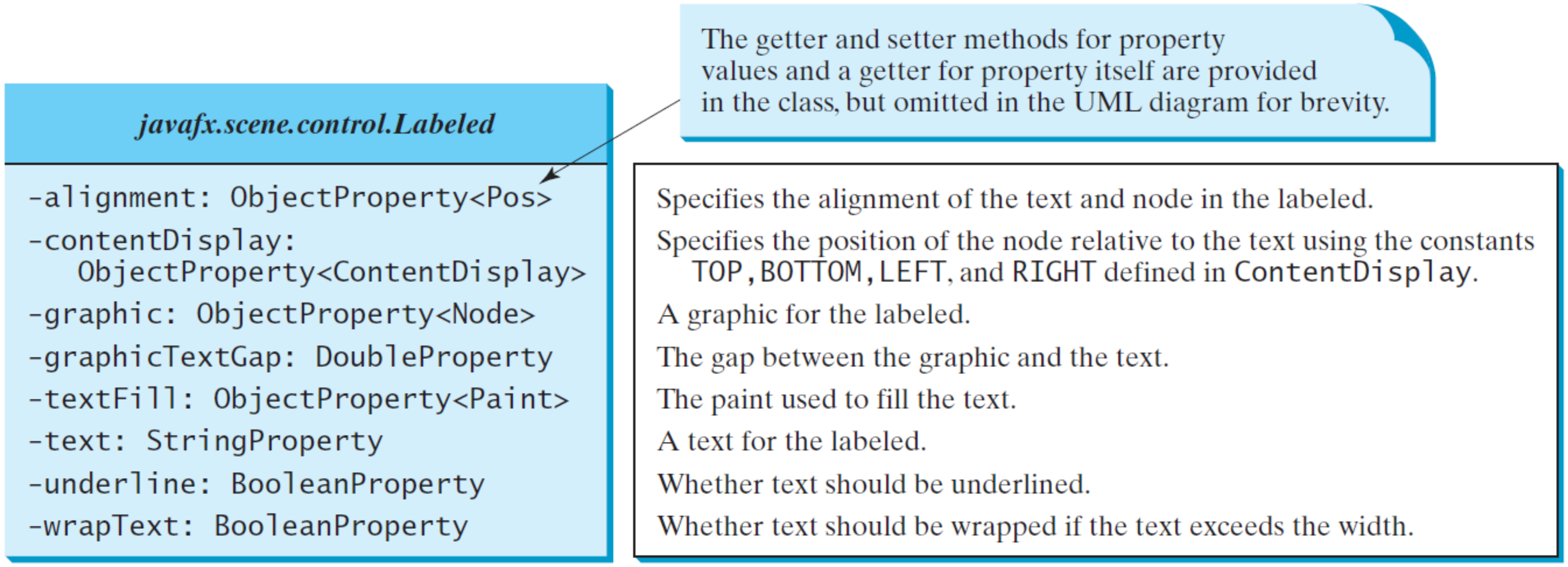
lbl	Label	bt	Button
chk	CheckBox	rb	RadioButton
tf	TextField	pf	PasswordField
ta	TextArea	cbo	ComboBox
lv	ListView	scb	ScrollBar
Sld	Slider	mp	MediaPlayer

Labeled and Label

Labeled and Label

- 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.

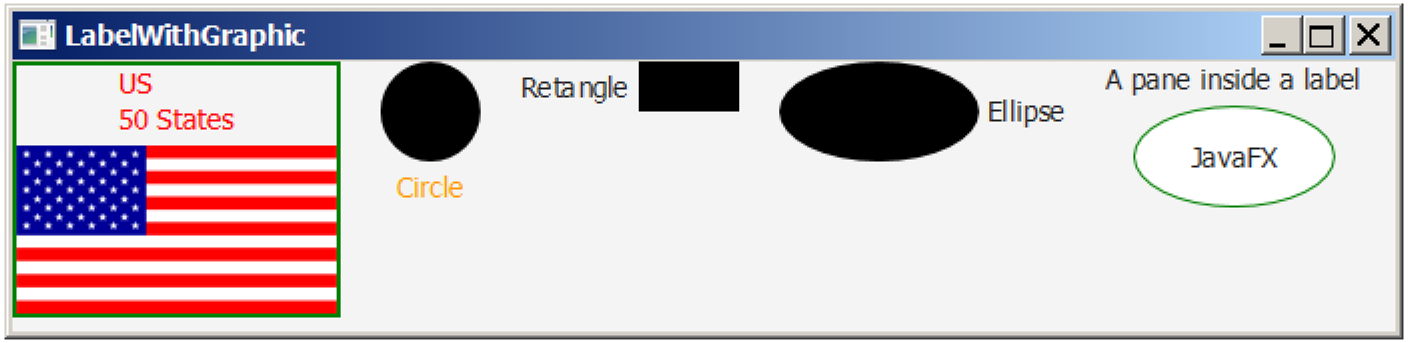


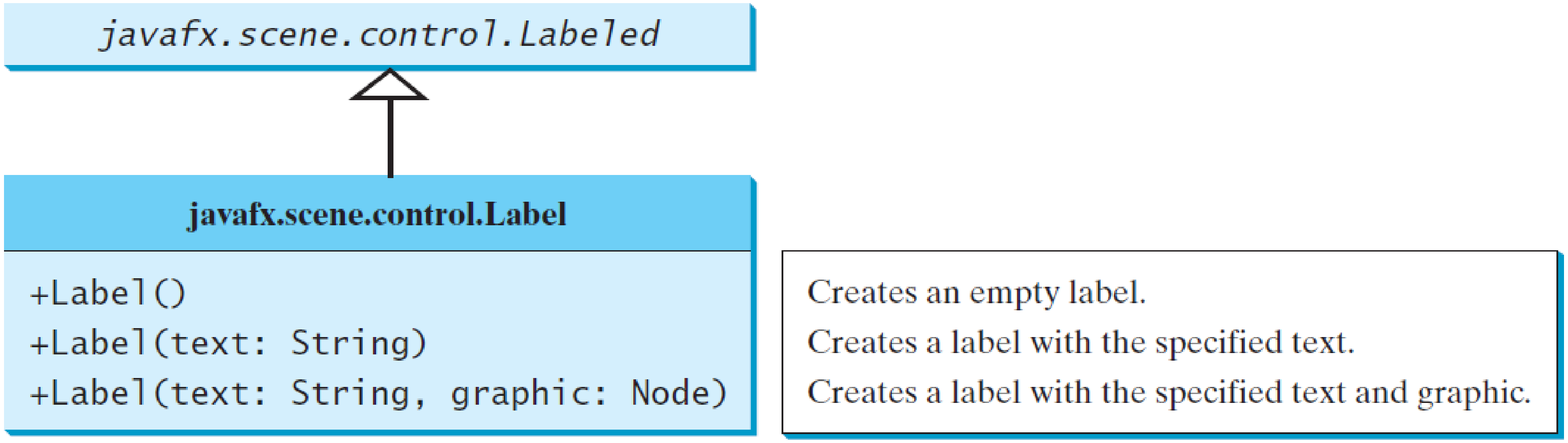

The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

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.

Label

The Label class defines labels.

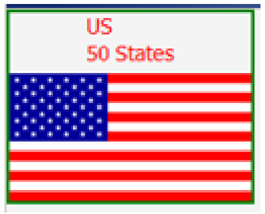


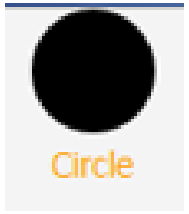

javafx.scene.control.Labeled
javafx.scene.control.Label
+Label()
+Label(text: String)
+Label(text: String, graphic: Node)
Creates an empty label.
Creates a label with the specified text.
Creates a label with the specified text and graphic.

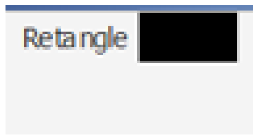
LabelWithGraphic

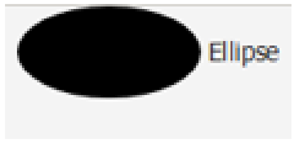
Label

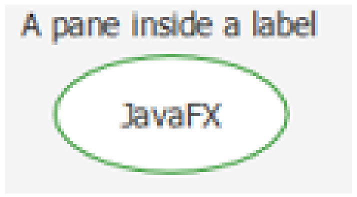
```
ImageView us = new ImageView(new Image("image/us.gif"));
Label lb1 = new Label("US\n50 States", us);
lb1.setStyle("-fx-border-color: green; -fx-border-width: 2");
lb1.setContentDisplay(ContentDisplay.BOTTOM);
lb1.setTextFill(Color.RED);

Label lb2 = new Label("Circle", new Circle(50, 50, 25));
lb2.setContentDisplay(ContentDisplay.TOP);
lb2.setTextFill(Color.ORANGE);

Label lb3 = new Label("Rectangle", 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);

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);
```







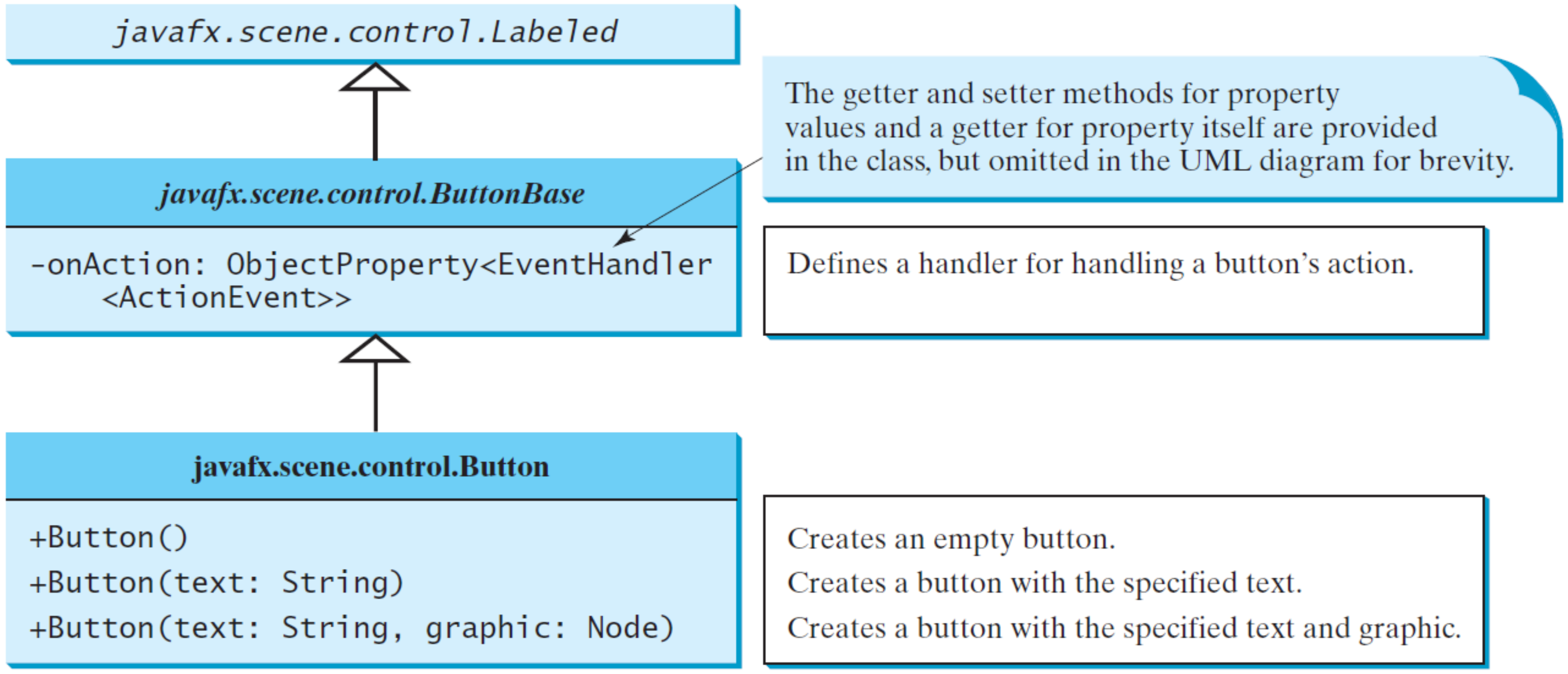




Button

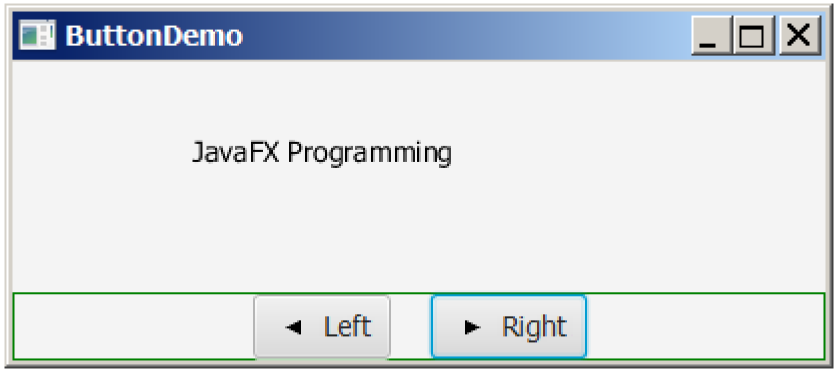
ButtonBase and Button

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
javafx.scene.control.ButtonBase
-onAction: ObjectProperty<EventHandler<ActionEvent>>
javafx.scene.control.Button
+Button()
+Button(text: String)
+Button(text: String, graphic: Node)
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.
Creates an empty button.
Creates a button with the specified text.
Creates a button with the specified text and graphic.

Button Example

```
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);

  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;
}
```

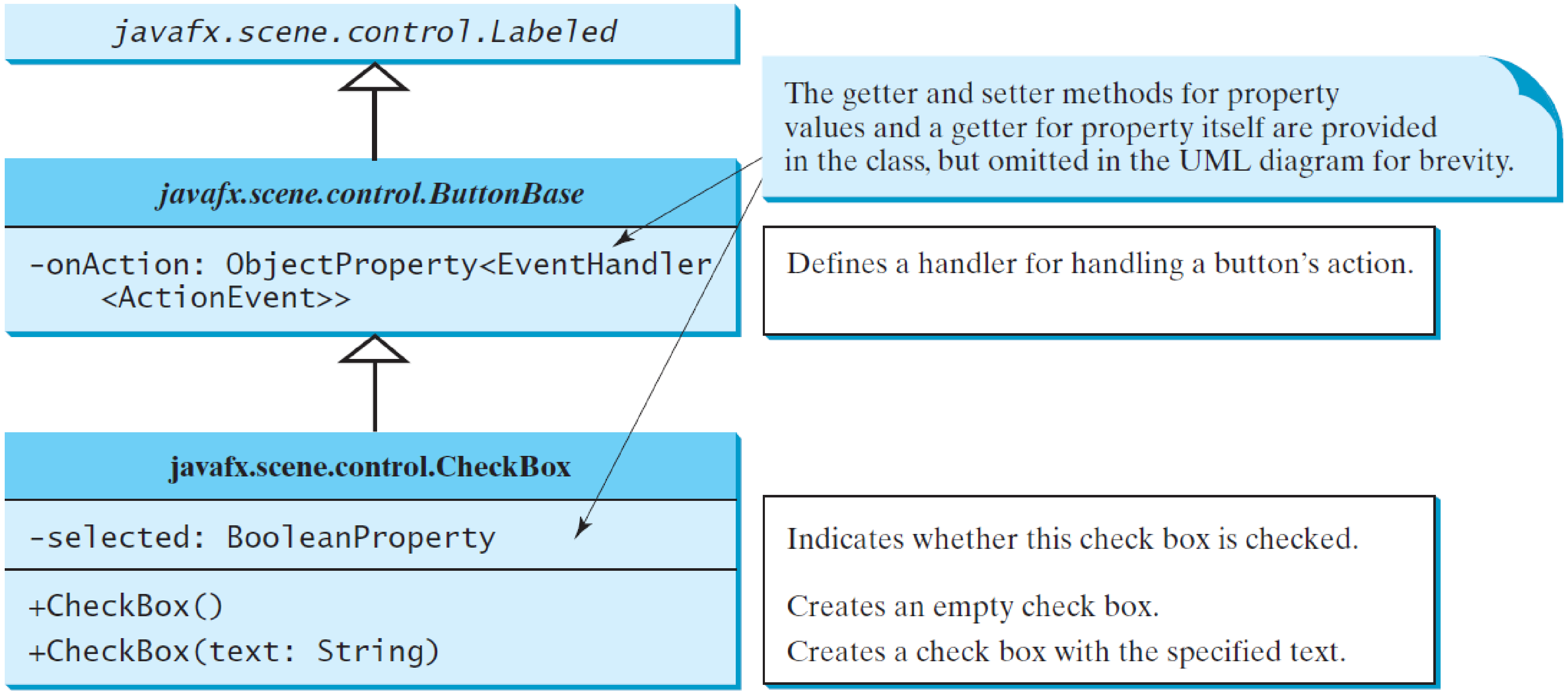


ButtonDemo

CheckBox

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**.

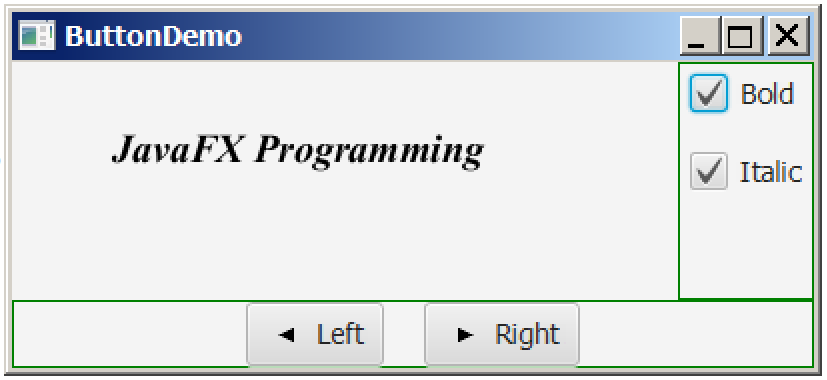


javafx.scene.control.Labeled
javafx.scene.control.ButtonBase
-onAction: ObjectProperty<EventHandler<ActionEvent>>
javafx.scene.control.CheckBox
-selected: BooleanProperty
+CheckBox()
+CheckBox(text: String)
The getter and setter methods for property values and a getter for property itself are provided 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.

CheckBox Example

```
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");
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);
```

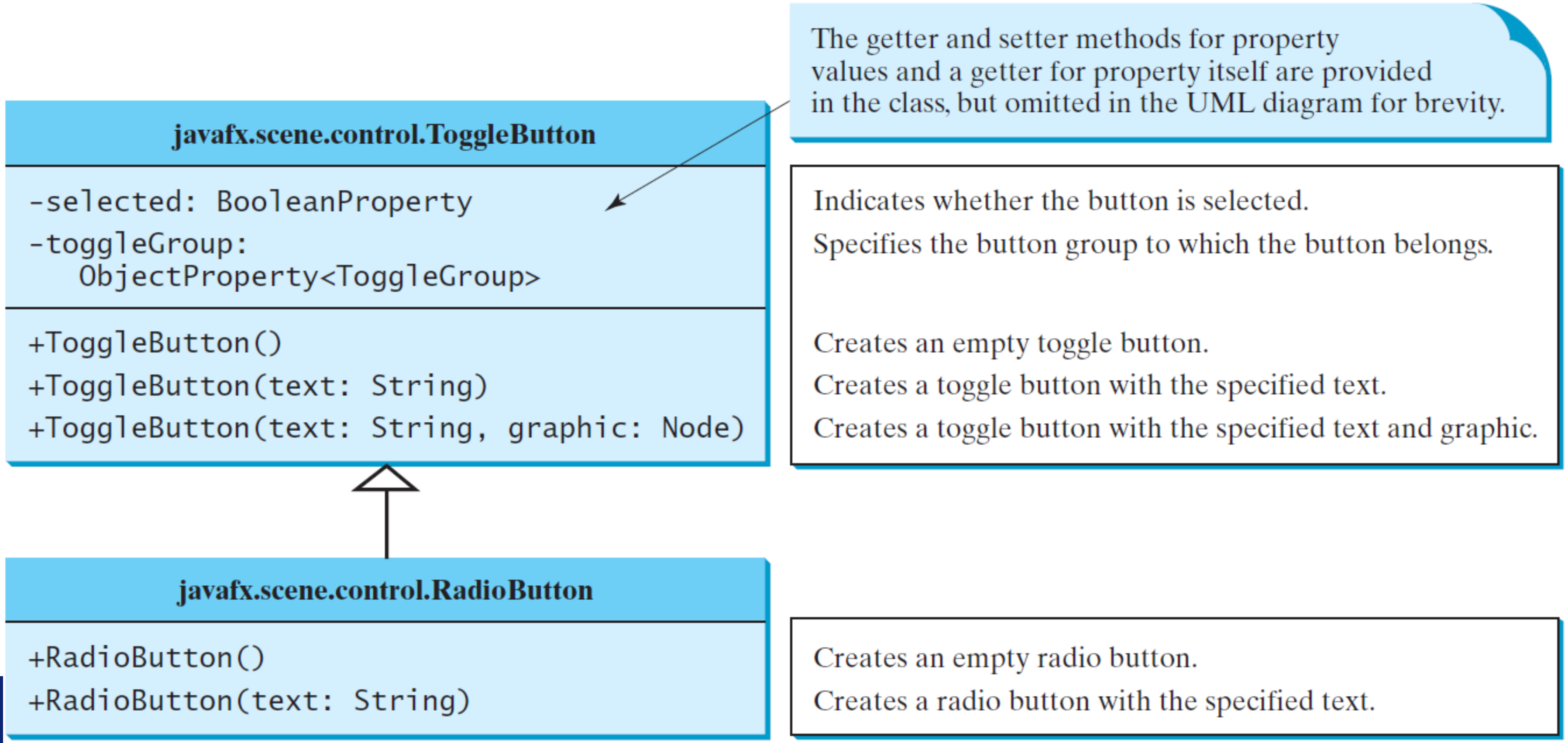


CheckBoxDemo

RadioButton

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).



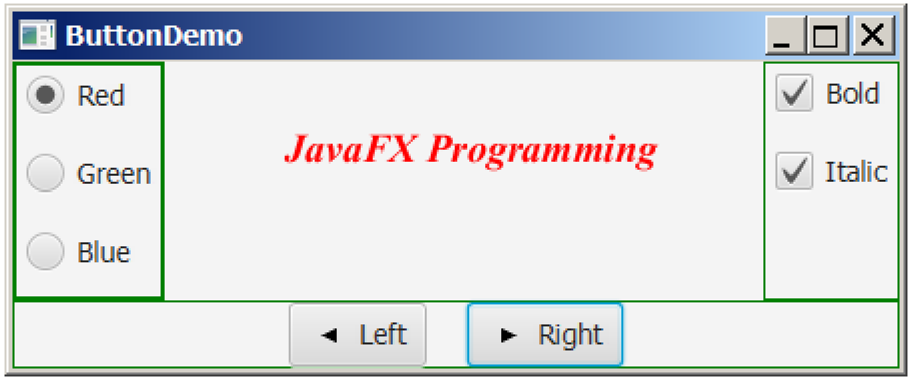
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.
javafx.scene.control.ToggleButton
-selected: BooleanProperty
-toggleGroup: ObjectProperty<ToggleGroup>
+ToggleButton()
+ToggleButton(text: String)
+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.

RadioButton Example

```
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);
  }
});

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

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

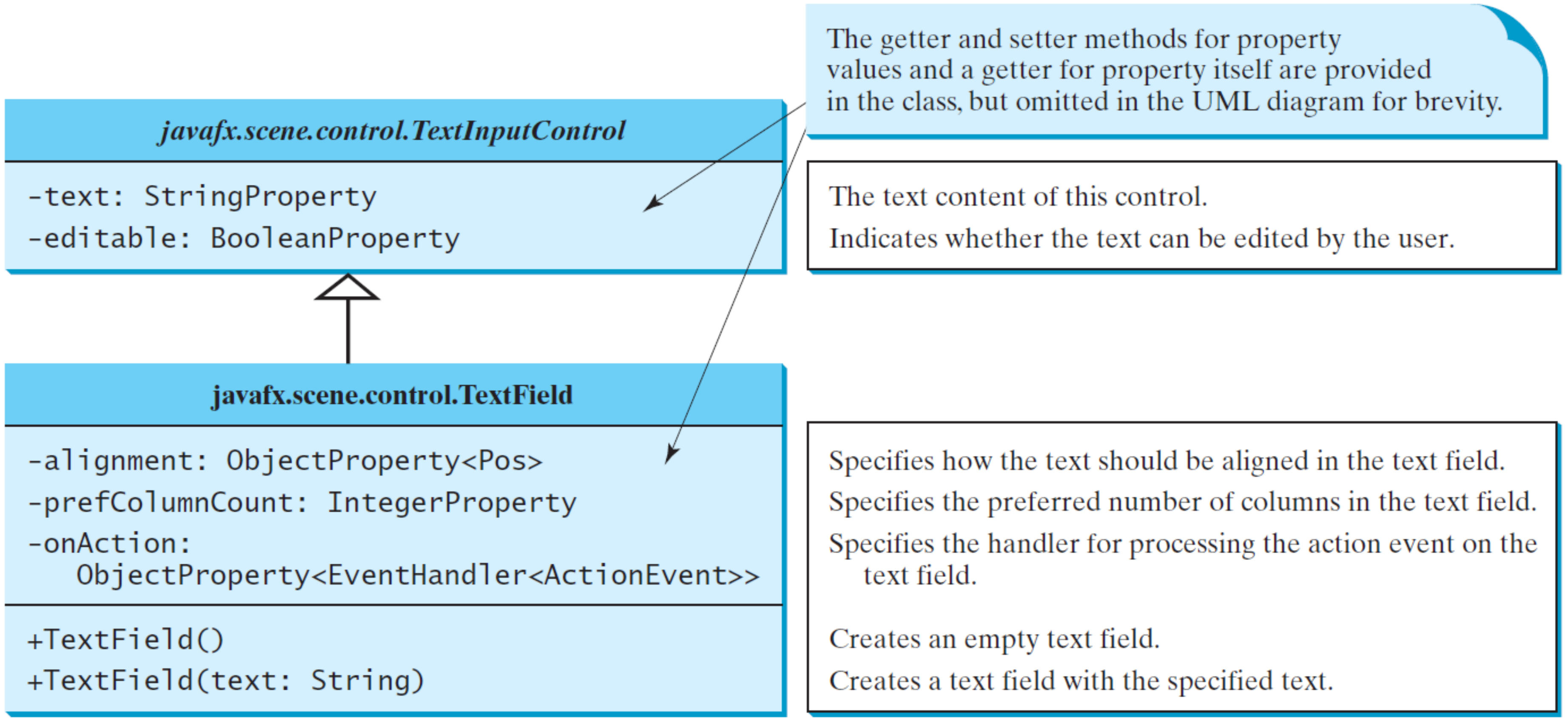


RadioButtonDemo

TextField

TextField

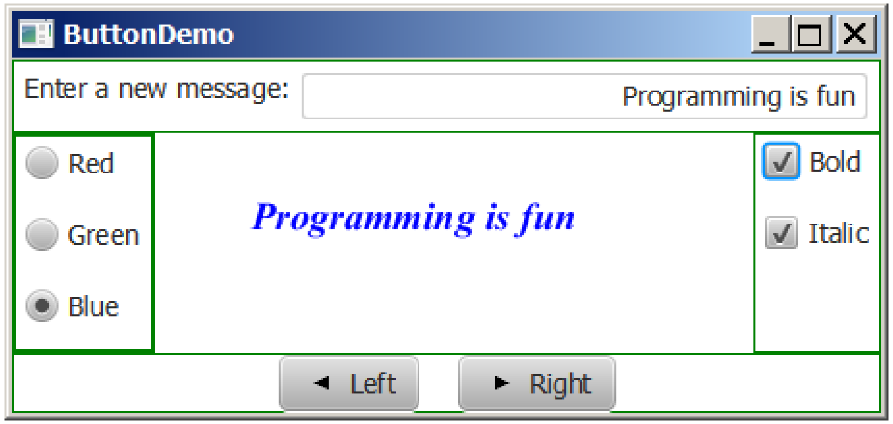
A text field can be used to enter or display a string. **TextField** is a subclass of **TextInputControl**.


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.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.

TextField Example

A TextField can be used to either display (uneditable) text, or to create a place the user can type textual information

Pressing the Enter key inside a TextField fires an ActionEvent



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

TextFieldDemo

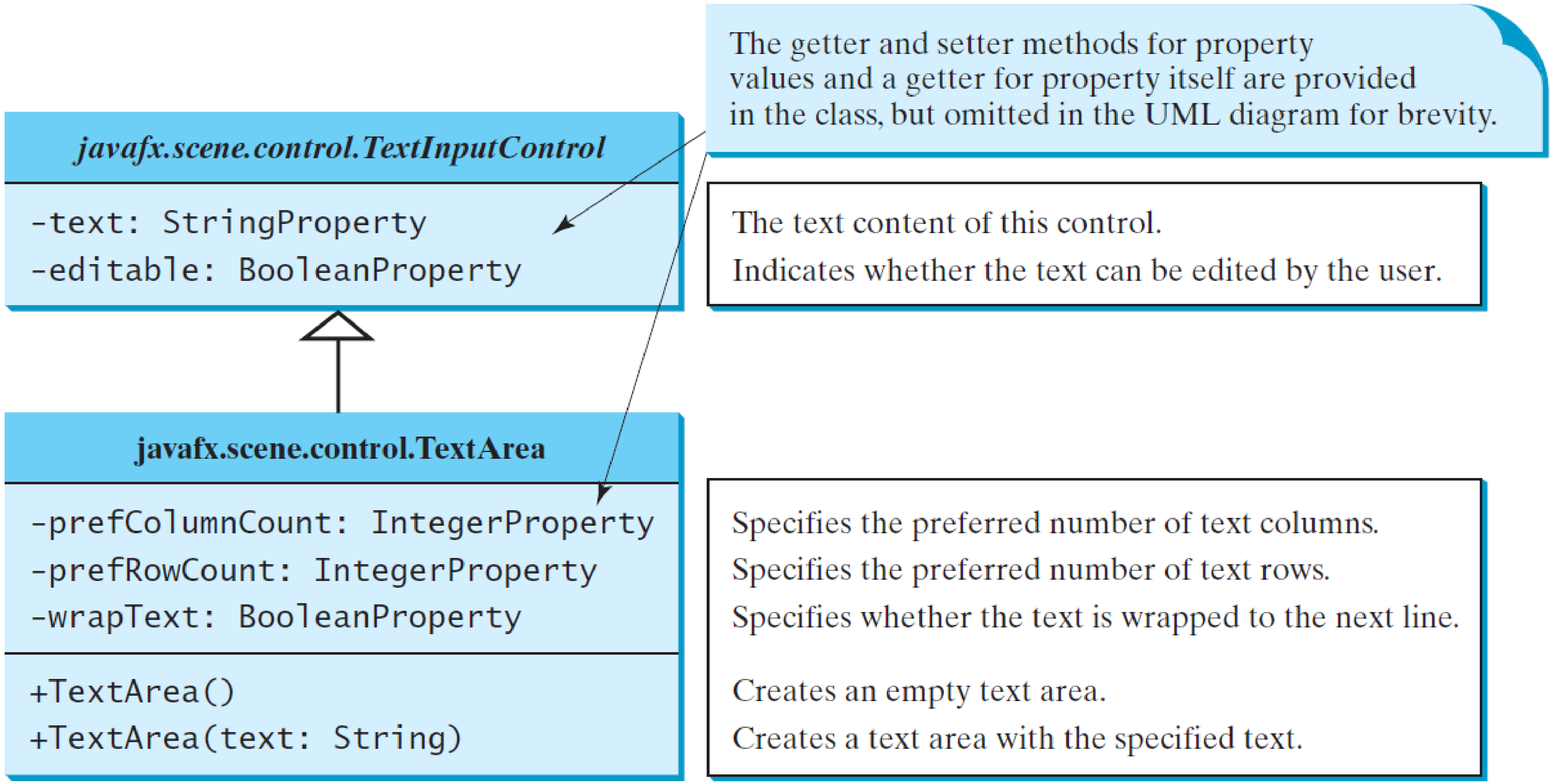
TextField

- The only difference between a `TextField` and a `PasswordField` is that, as the user is typing text nto a `PasswordField`, rather than showing the characters the user typed, the system displays an asterisk for each character, effectively hiding the text (but not its length)

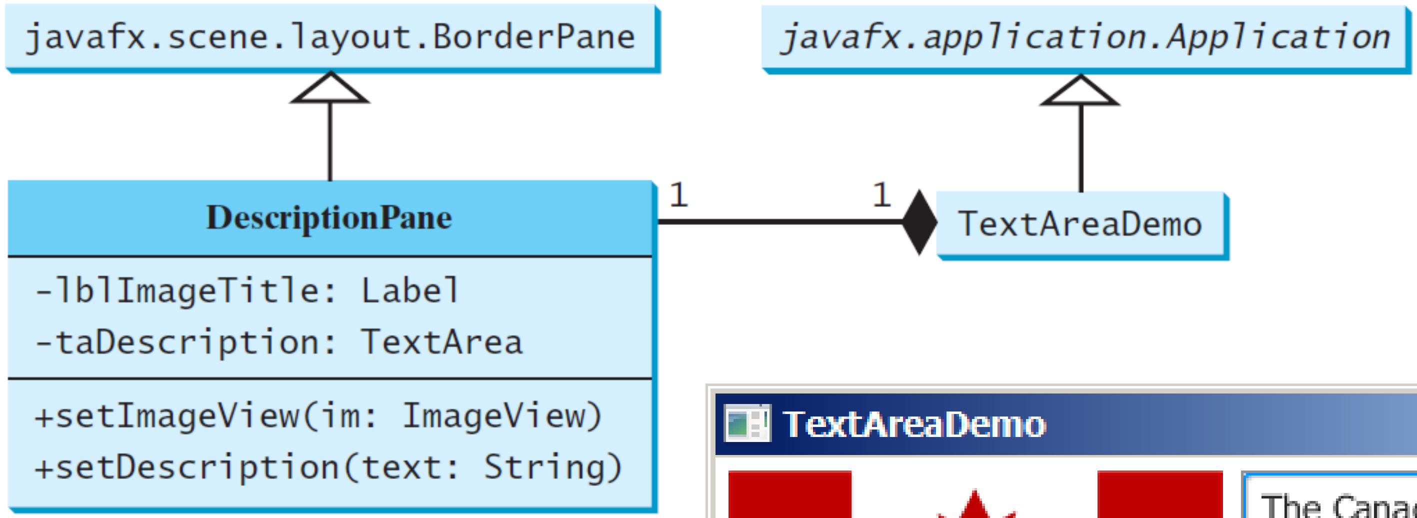
TextArea

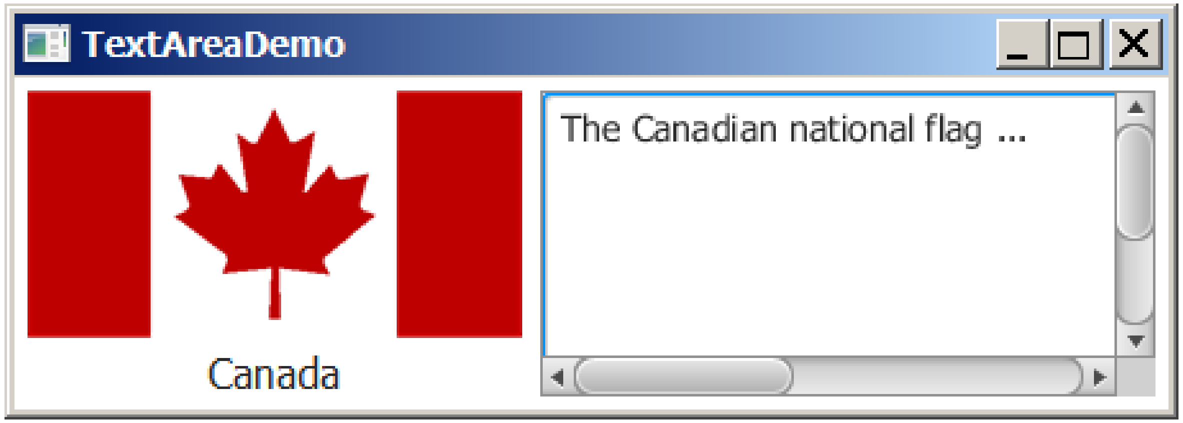
TextArea

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.

TextArea Example


javafx.scene.layout.BorderPane
javafx.application.Application
DescriptionPane
-lblImageTitle: Label
-taDescription: TextArea
+setImageView(im: ImageView)
+setDescription(text: String)
1
1
TextAreaDemo

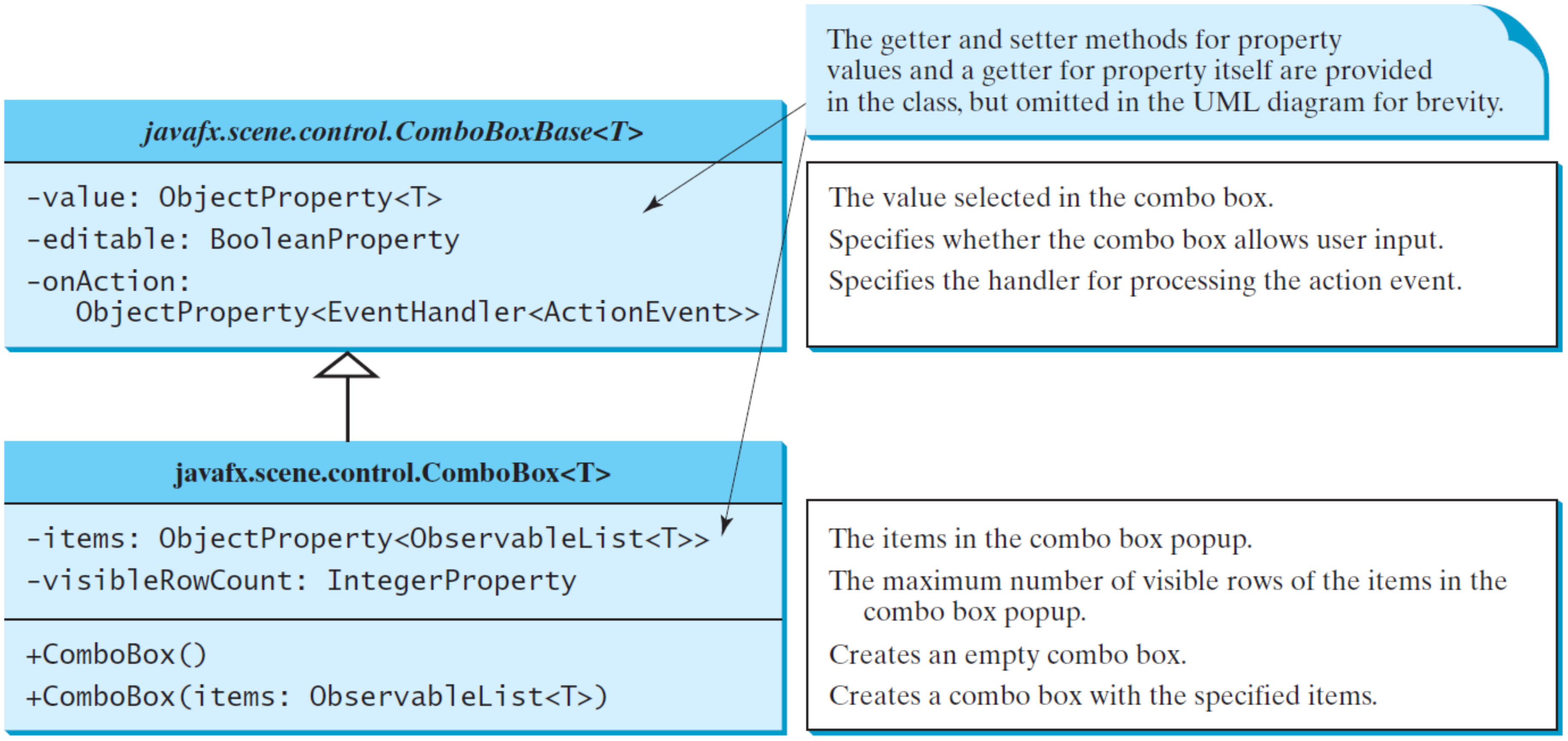


TextAreaDemo

ComboBox

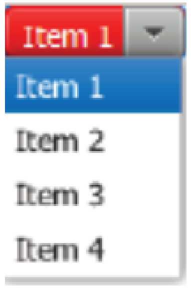
ComboBox

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


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.

ComboBox Example

```
ComboBox<String> cbo = new ComboBox<>();
cbo.getItems().addAll("Item 1", "Item 2",
  "Item 3", "Item 4");
```



- A combo box is useful for limiting a user's range of choices
- and avoids the cumbersome validation of data input.

FXCollections.observableArrayList(arrayOfElements) for creating an ObservableList from an array of element

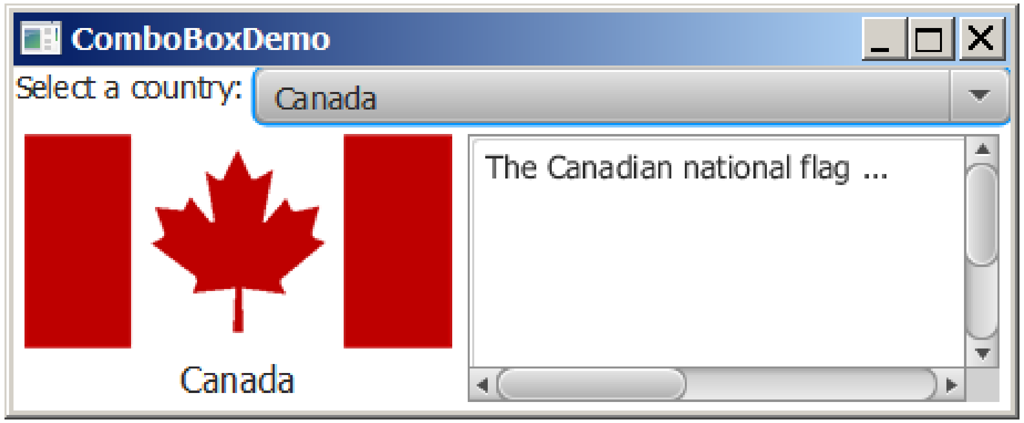
Example:

```
private String[] myStringArray = {"Hello", "Hi", "Welcome"};
ObservableList<String> items = FXCollections.observableArrayList(myStringArray);
cbo.getItems().addAll(items); // Add items to combo box
```

ComboBoxDemo

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.



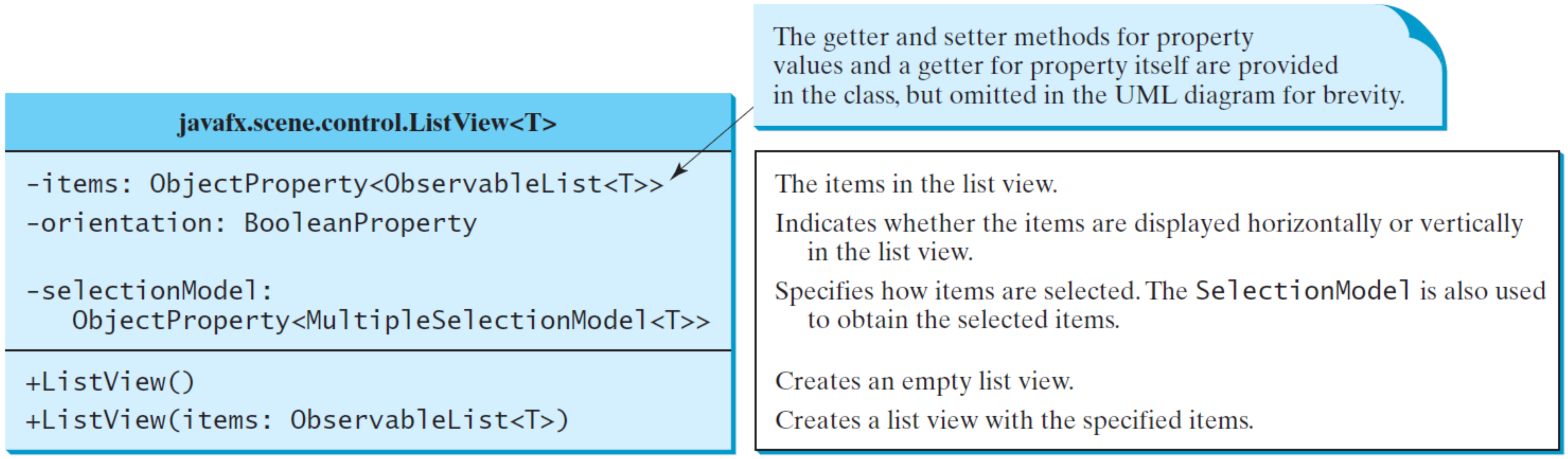
The Canadian national flag ...

ComboBoxDemo

ListView

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.



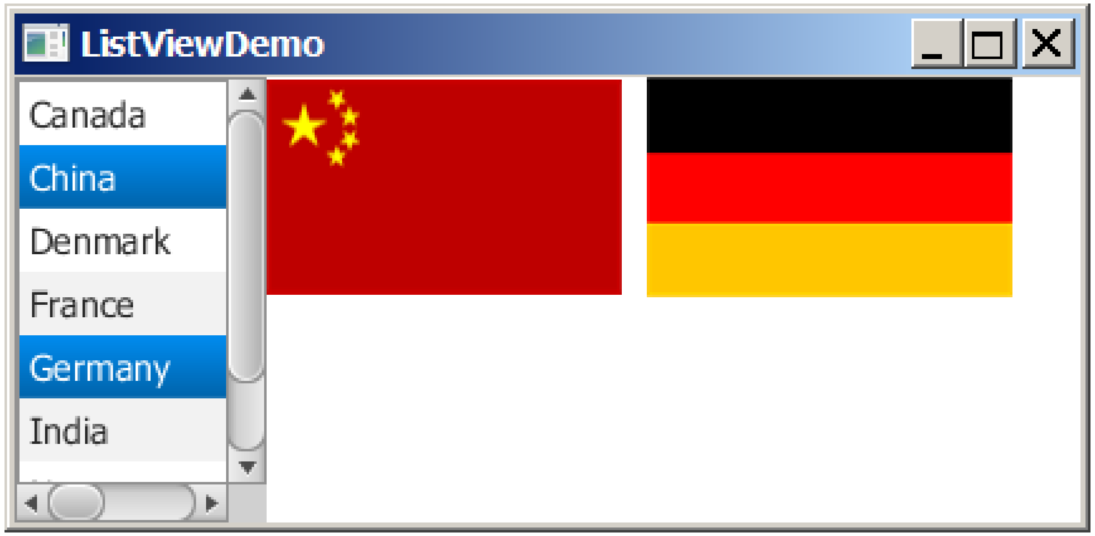
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

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.

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.

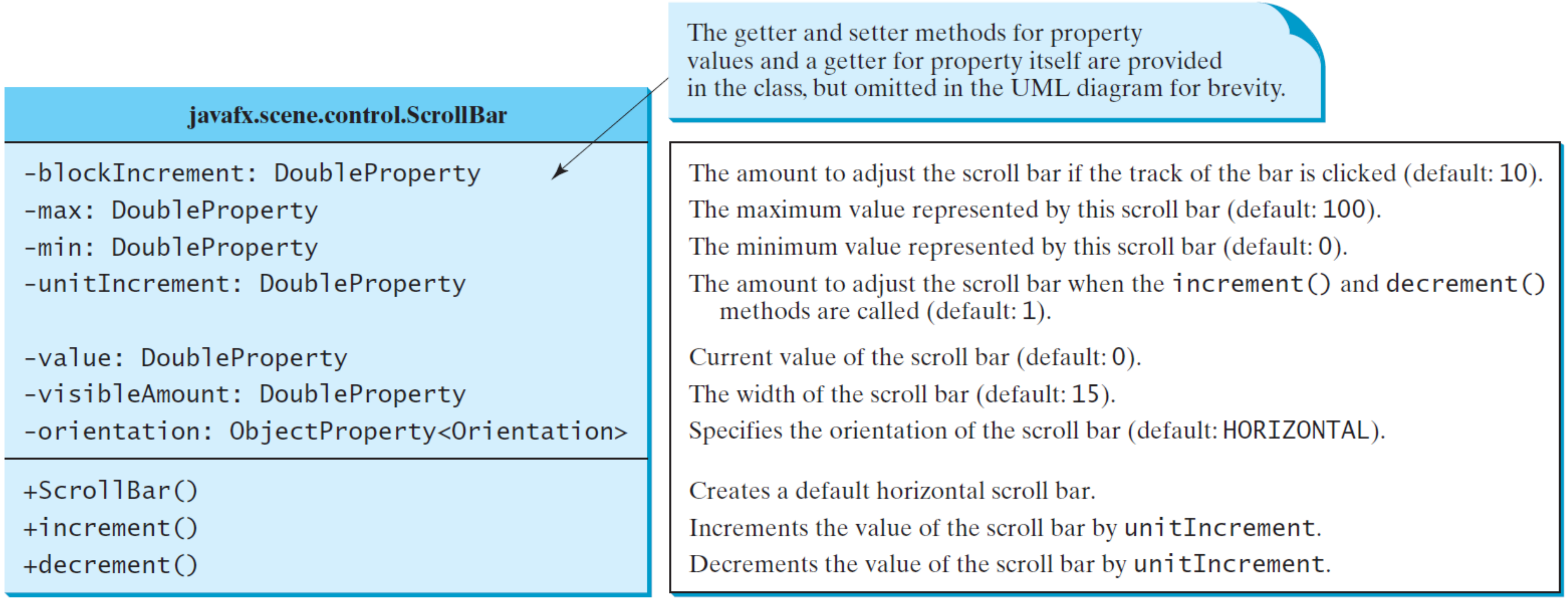


ListViewDemo Run

ScrollBar

ScrollBar

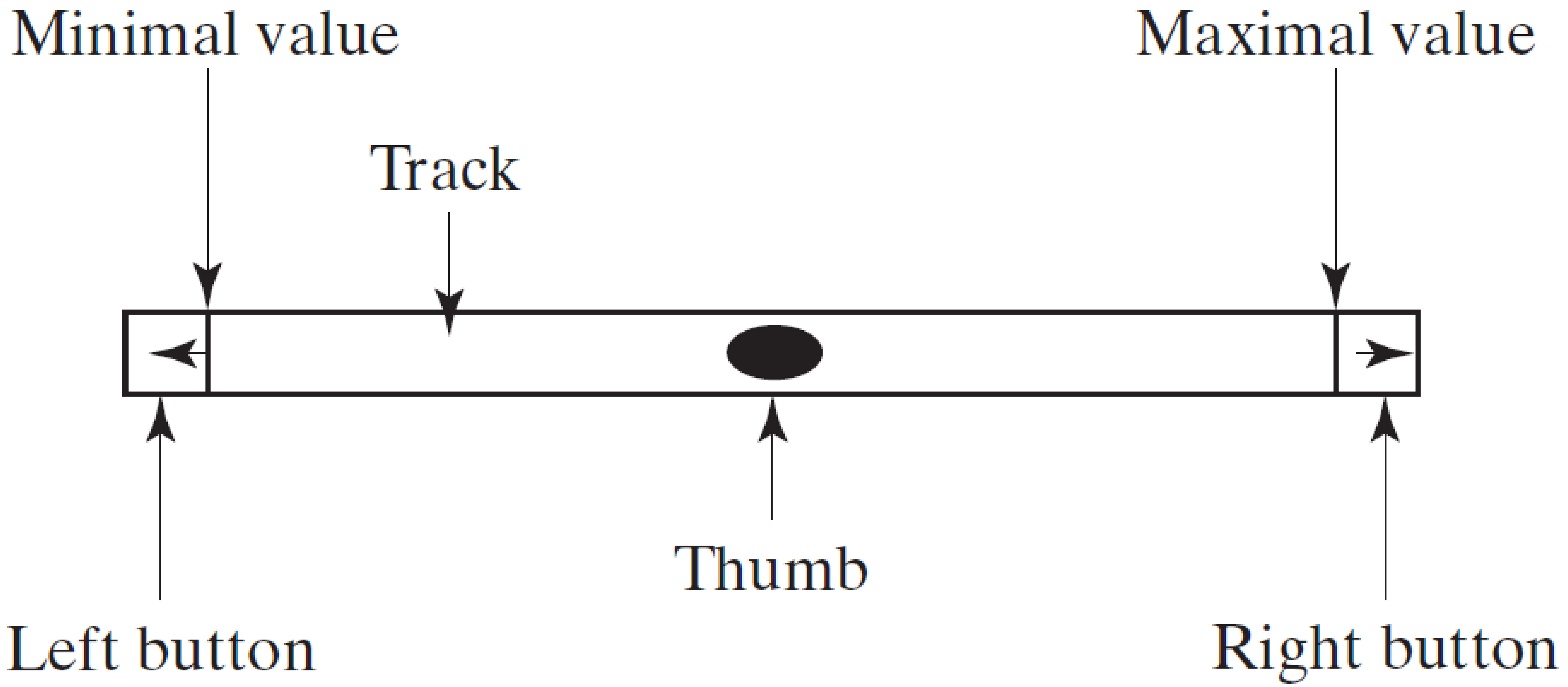
A *scroll bar* is a control that enables the user to select from a range of values. The scrollbar appears in two styles: *horizontal* and *vertical*.



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.

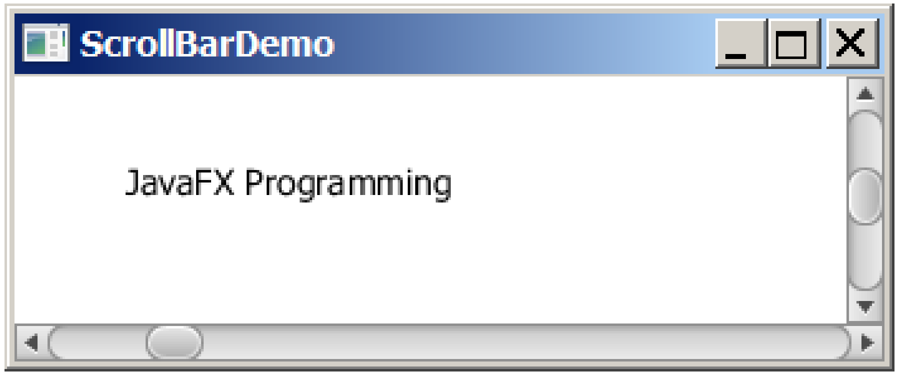
javafx.scene.control.ScrollBar	
`-blockIncrement: DoubleProperty`	The amount to adjust the scroll bar if the track of the bar is clicked (default: `10`).
`-max: DoubleProperty`	The maximum value represented by this scroll bar (default: `100`).
`-min: DoubleProperty`	The minimum value represented by this scroll bar (default: `0`).
`-unitIncrement: DoubleProperty`	The amount to adjust the scroll bar when the `increment()` and `decrement()` methods are called (default: `1`).
`-value: DoubleProperty`	Current value of the scroll bar (default: `0`).
`-visibleAmount: DoubleProperty`	The width of the scroll bar (default: `15`).
`-orientation: ObjectProperty<Orientation>`	Specifies the orientation of the scroll bar (default: `HORIZONTAL`).
`+ScrollBar()`	Creates a default horizontal scroll bar.
`+increment()`	Increments the value of the scroll bar by `unitIncrement`.
`+decrement()`	Decrements the value of the scroll bar by `unitIncrement`.

Scroll Bar Properties



Example: Using Scrollbars

This example uses horizontal and vertical scrollbars to control a message displayed on a panel. The horizontal scrollbar is used to move the message to the left or the right, and the vertical scrollbar to move it up and down.

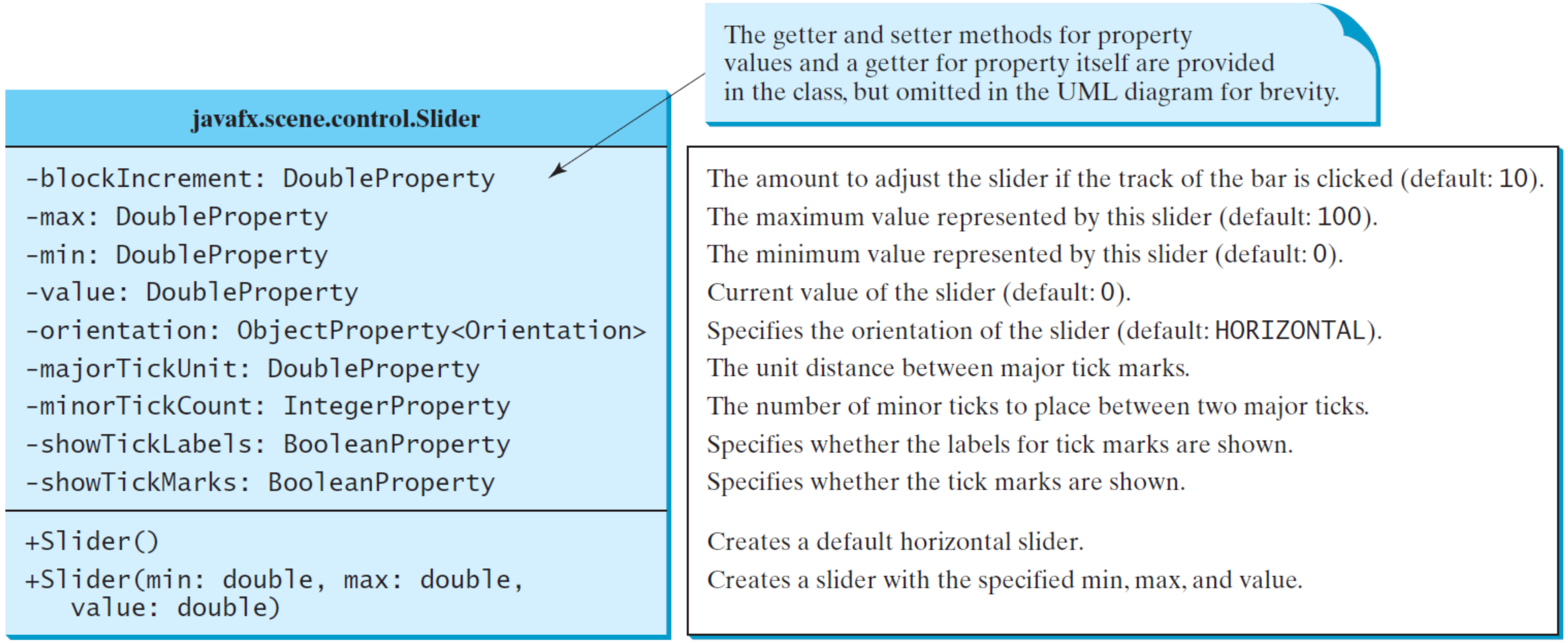


ScrollBarDemo Run

Slider

Slider

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



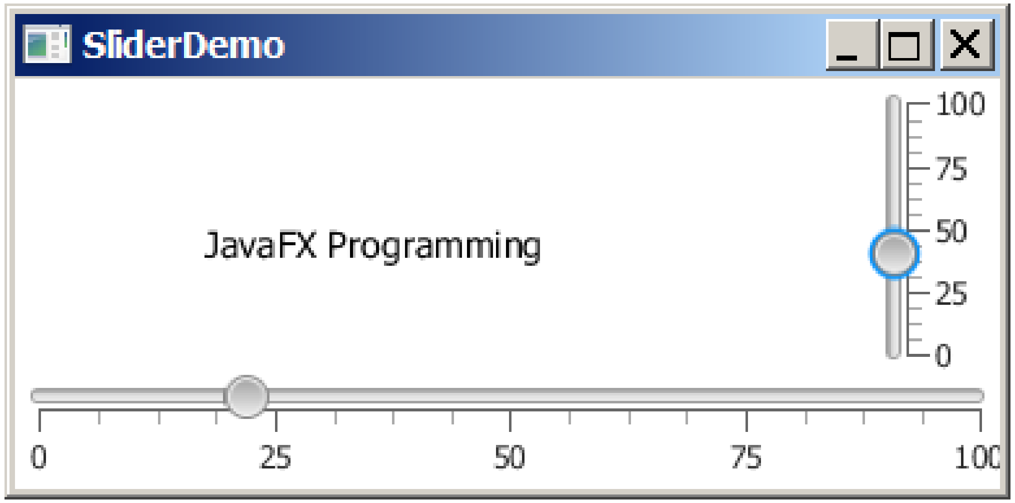
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

javafx.scene.control.Slider

`-blockIncrement: DoubleProperty`	The amount to adjust the slider if the track of the bar is clicked (default: `10`).
`-max: DoubleProperty`	The maximum value represented by this slider (default: `100`).
`-min: DoubleProperty`	The minimum value represented by this slider (default: `0`).
`-value: DoubleProperty`	Current value of the slider (default: `0`).
`-orientation: ObjectProperty<Orientation>`	Specifies the orientation of the slider (default: `HORIZONTAL`).
`-majorTickUnit: DoubleProperty`	The unit distance between major tick marks.
`-minorTickCount: IntegerProperty`	The number of minor ticks to place between two major ticks.
`-showTickLabels: BooleanProperty`	Specifies whether the labels for tick marks are shown.
`-showTickMarks: BooleanProperty`	Specifies whether the tick marks are shown.
`+Slider()`	Creates a default horizontal slider.
`+Slider(min: double, max: double, value: double)`	Creates a slider with the specified min, max, and value.

Example: Using Sliders

Rewrite the preceding program using the sliders to control a message displayed on a panel instead of using scroll bars.



GUI development and JavaFX Basics