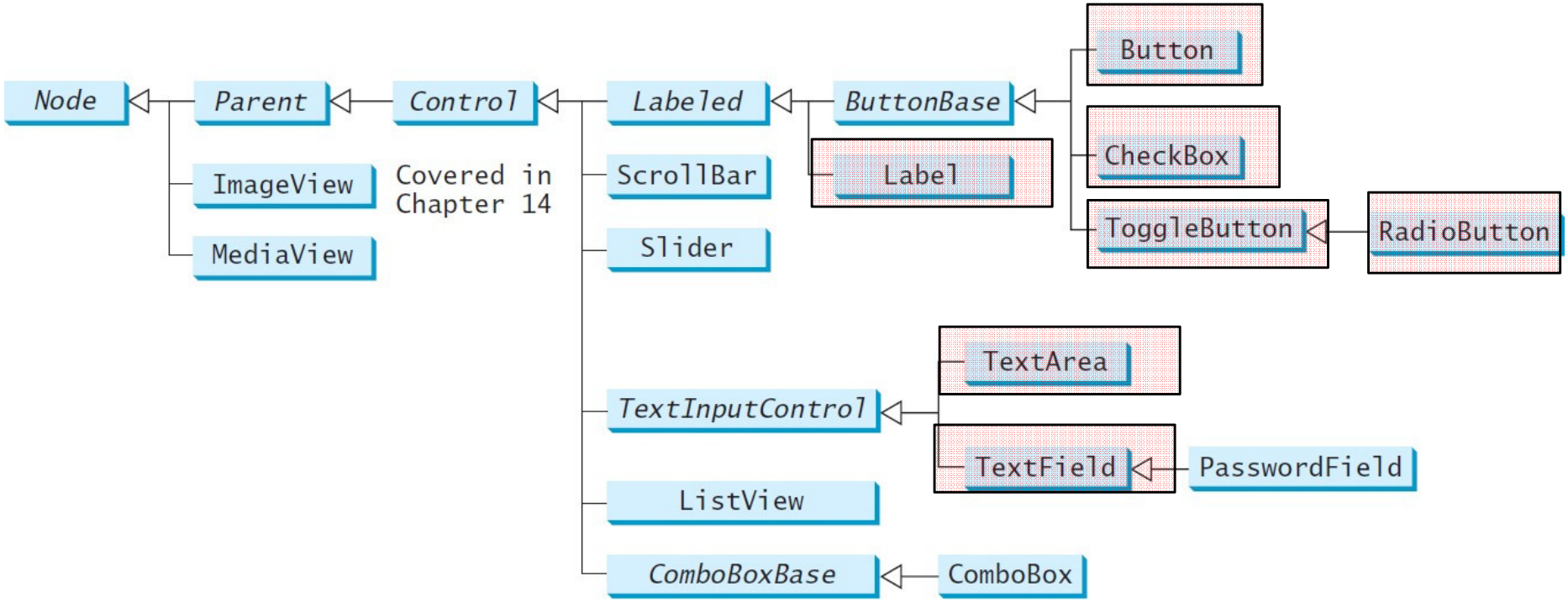
Chapter 16
JavaFX UI Controls

Dr. Asem Kitana
Dr. Abdallah Karakra

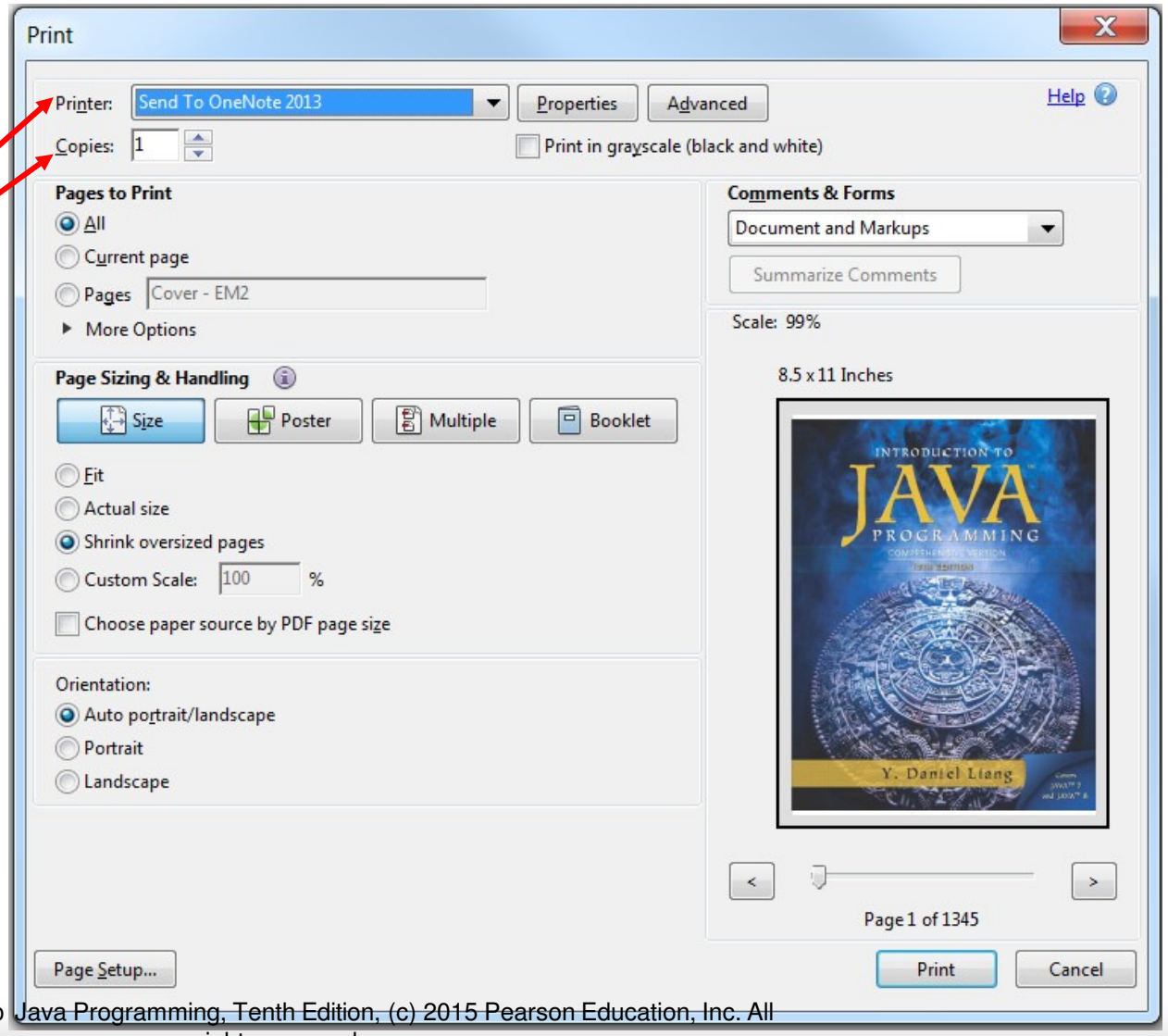
Frequently Used UI Controls



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**.

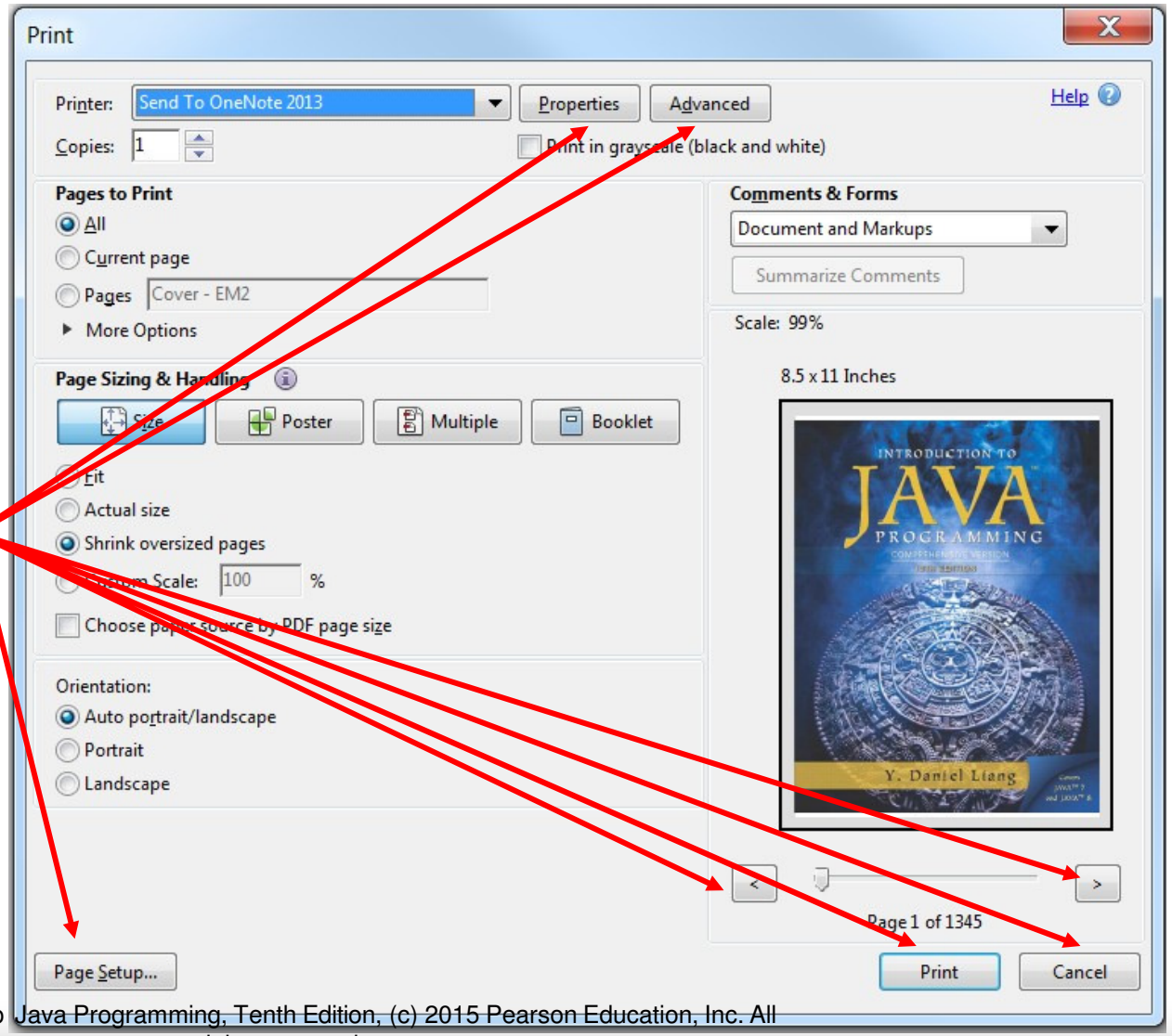
Introduction

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



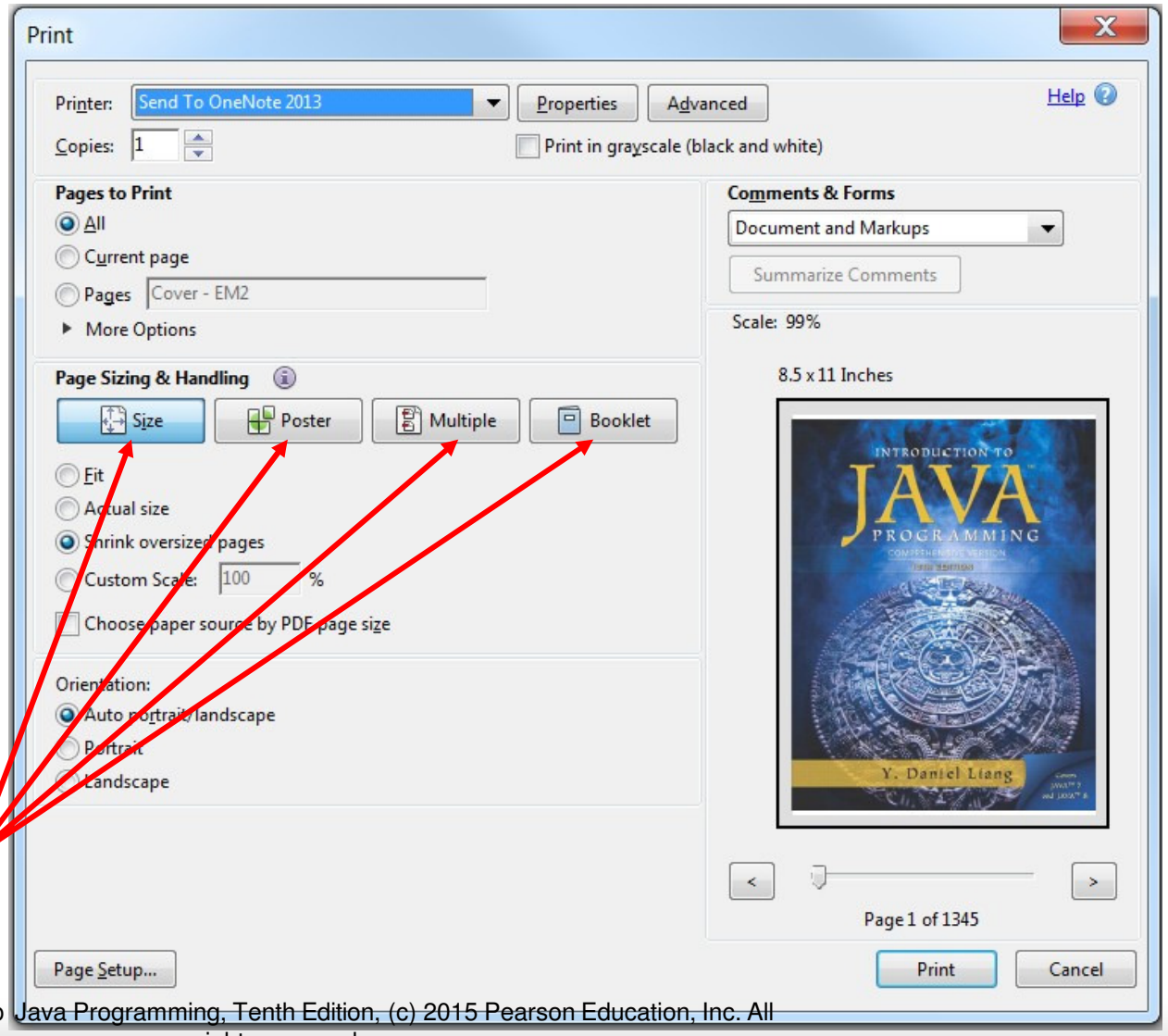
Introduction

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



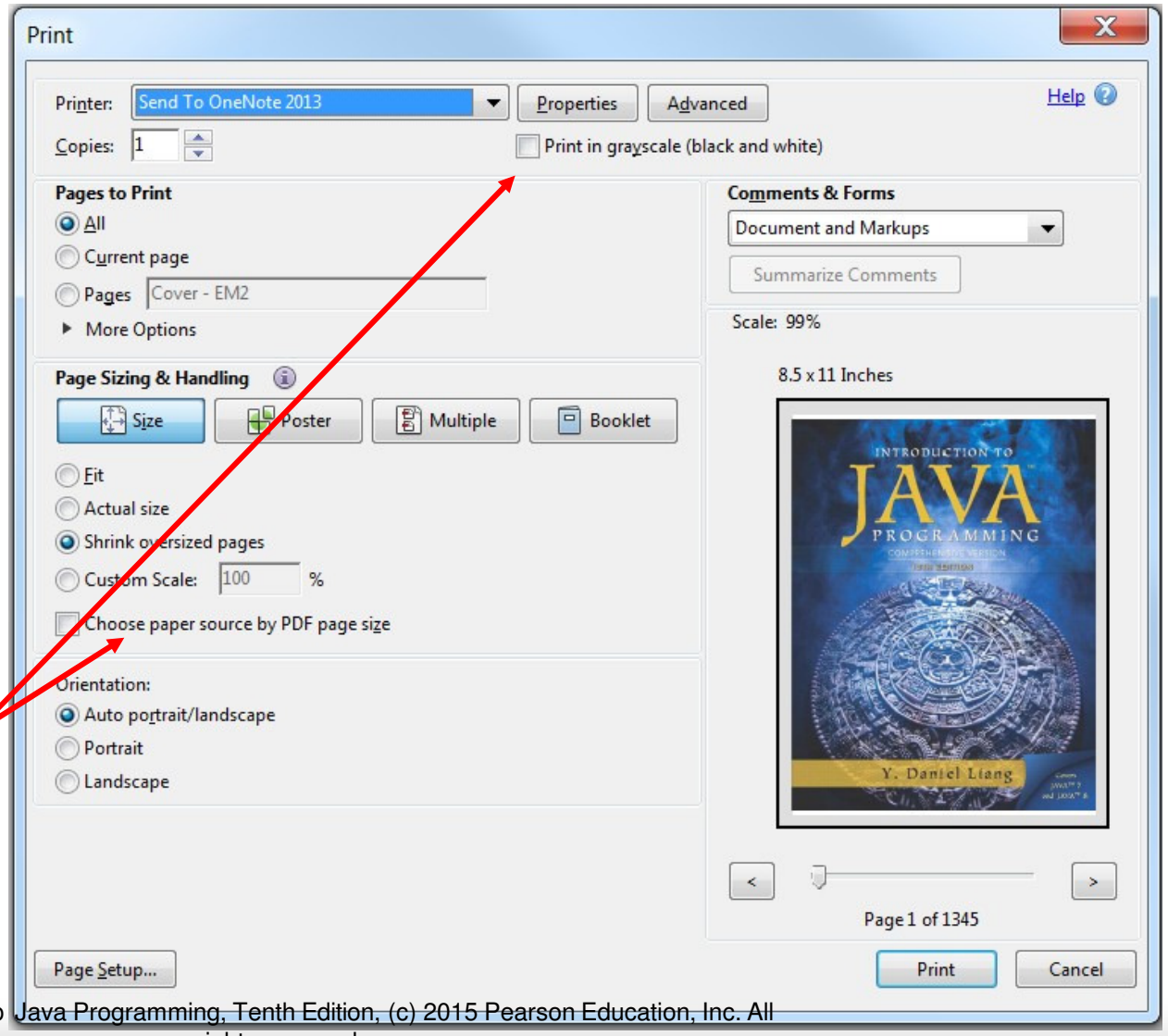
Introduction

Buttons with both a label *and* an image



Introduction

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

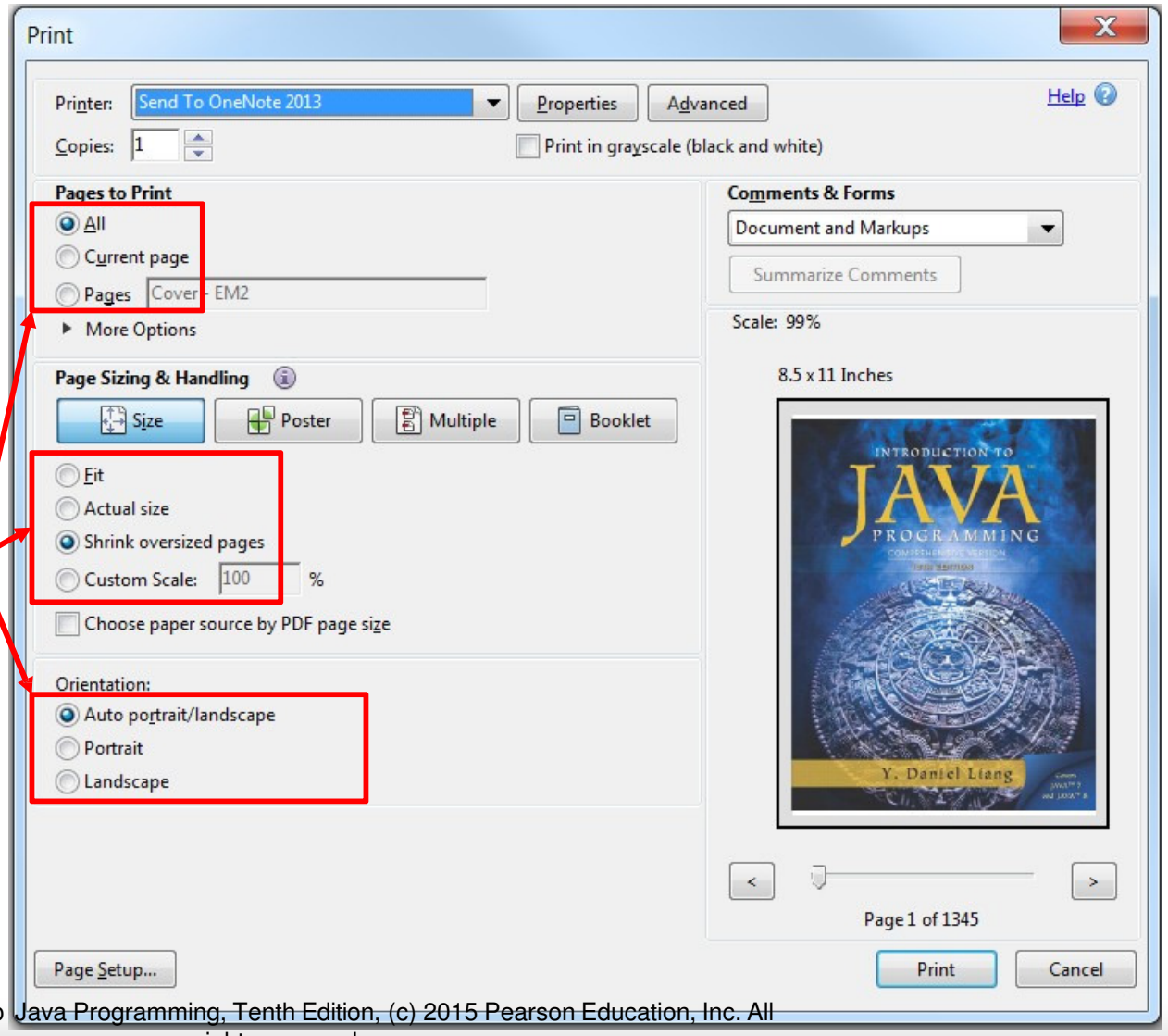


Introduction

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

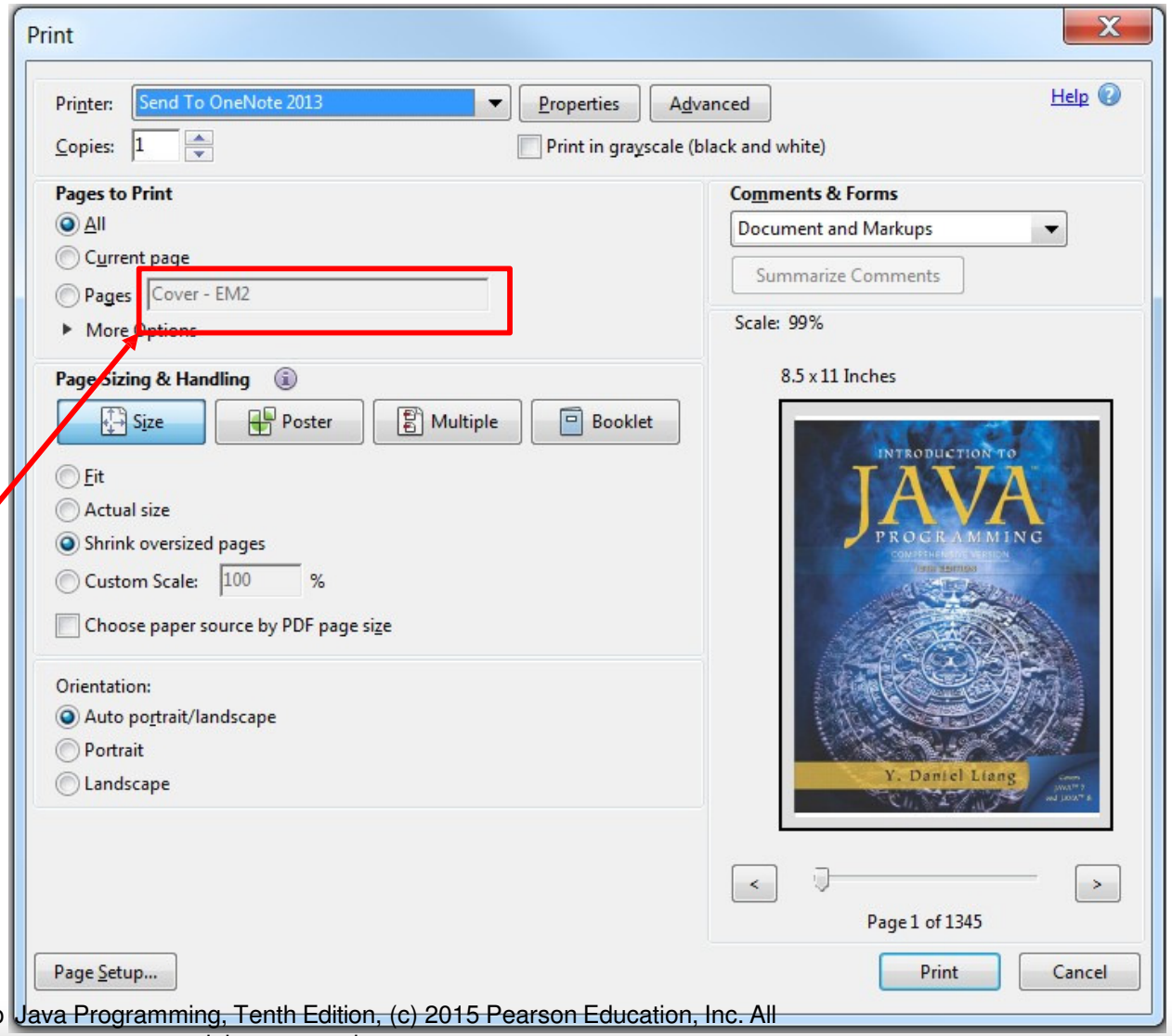
Each RadioButton has a *round* "selected" indicator and a label



Introduction

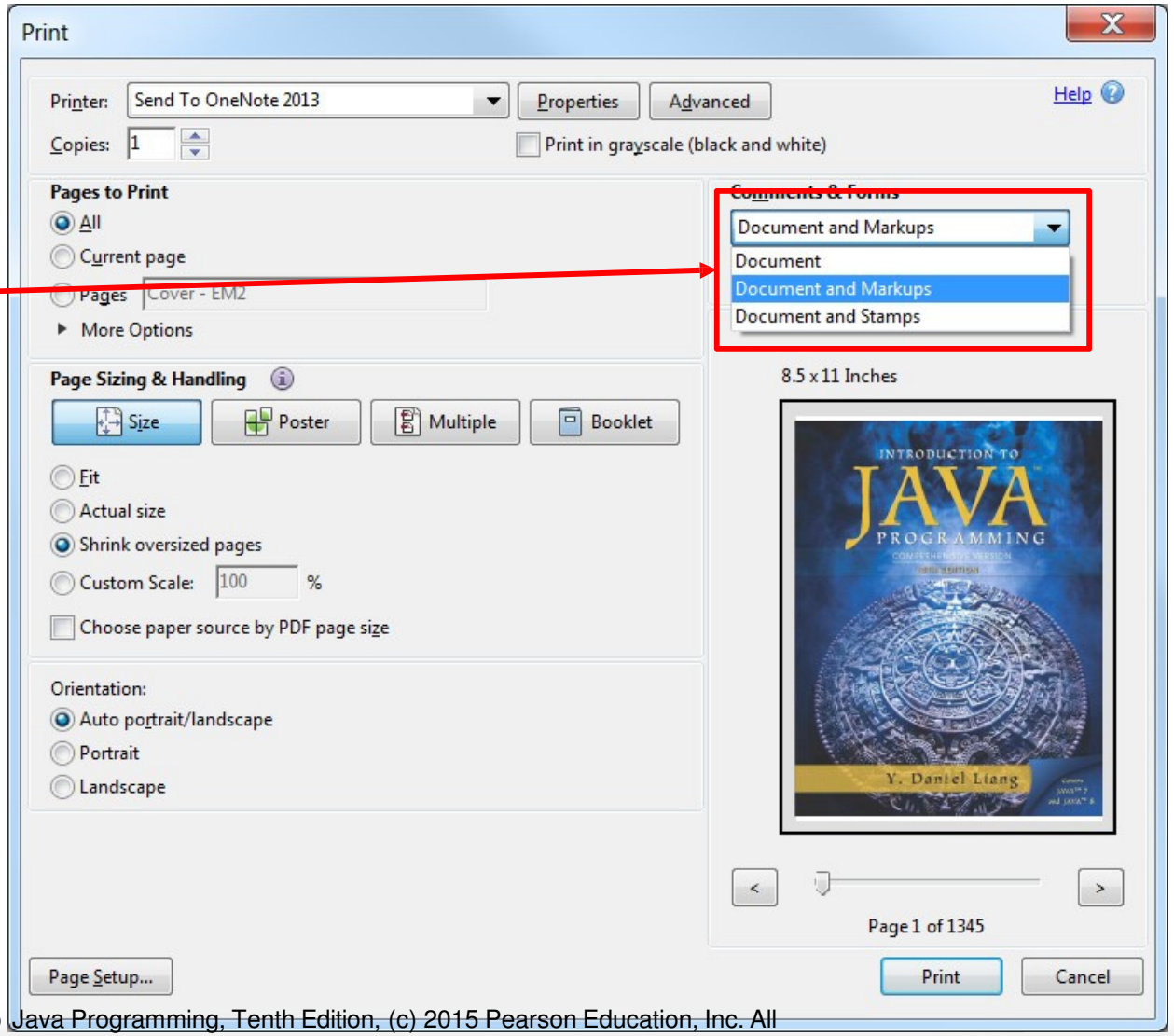
TextField, into which the user may type text.

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



Introduction

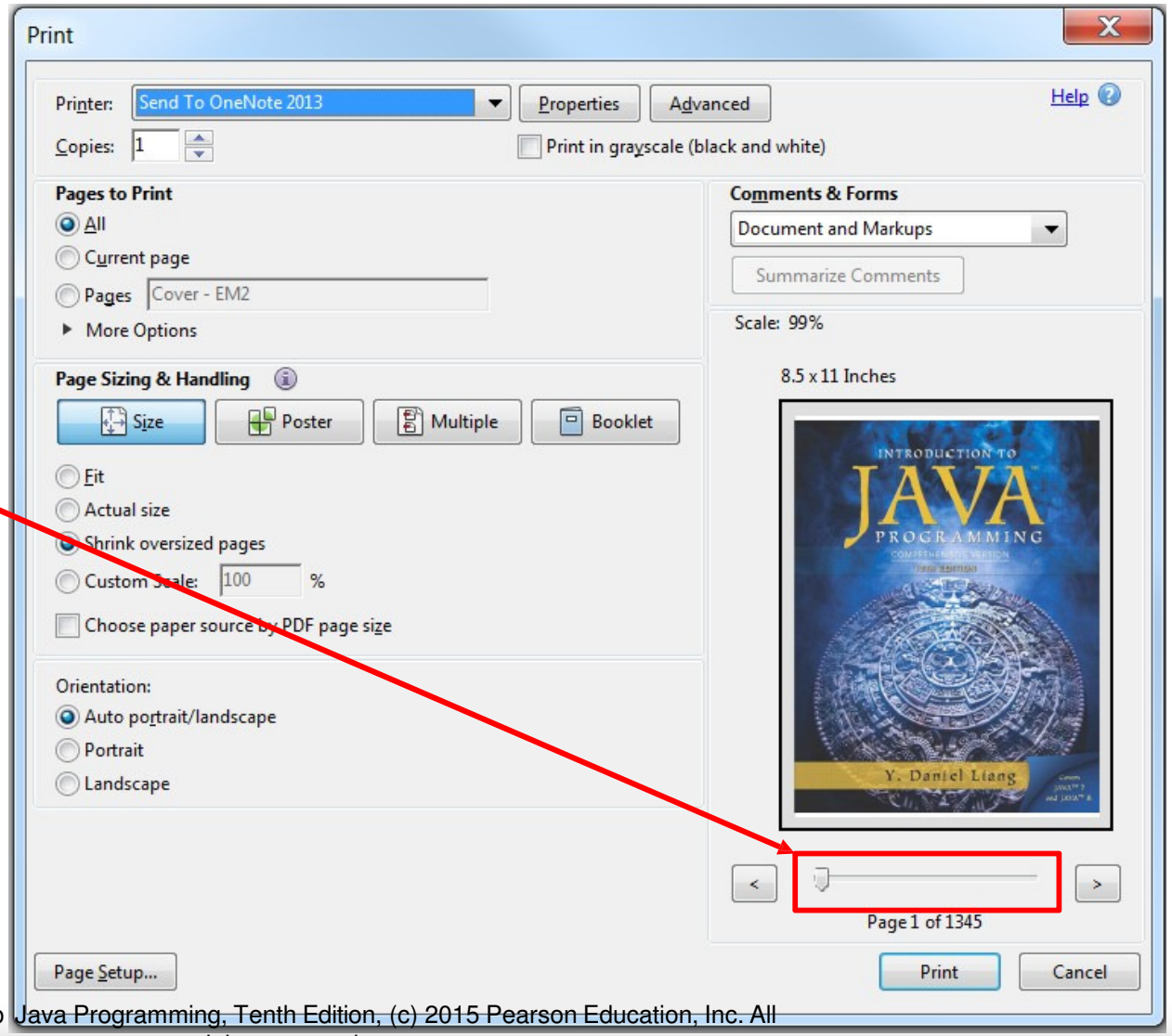
ComboBox, which lets the user drop-down a list of options from which to select



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).

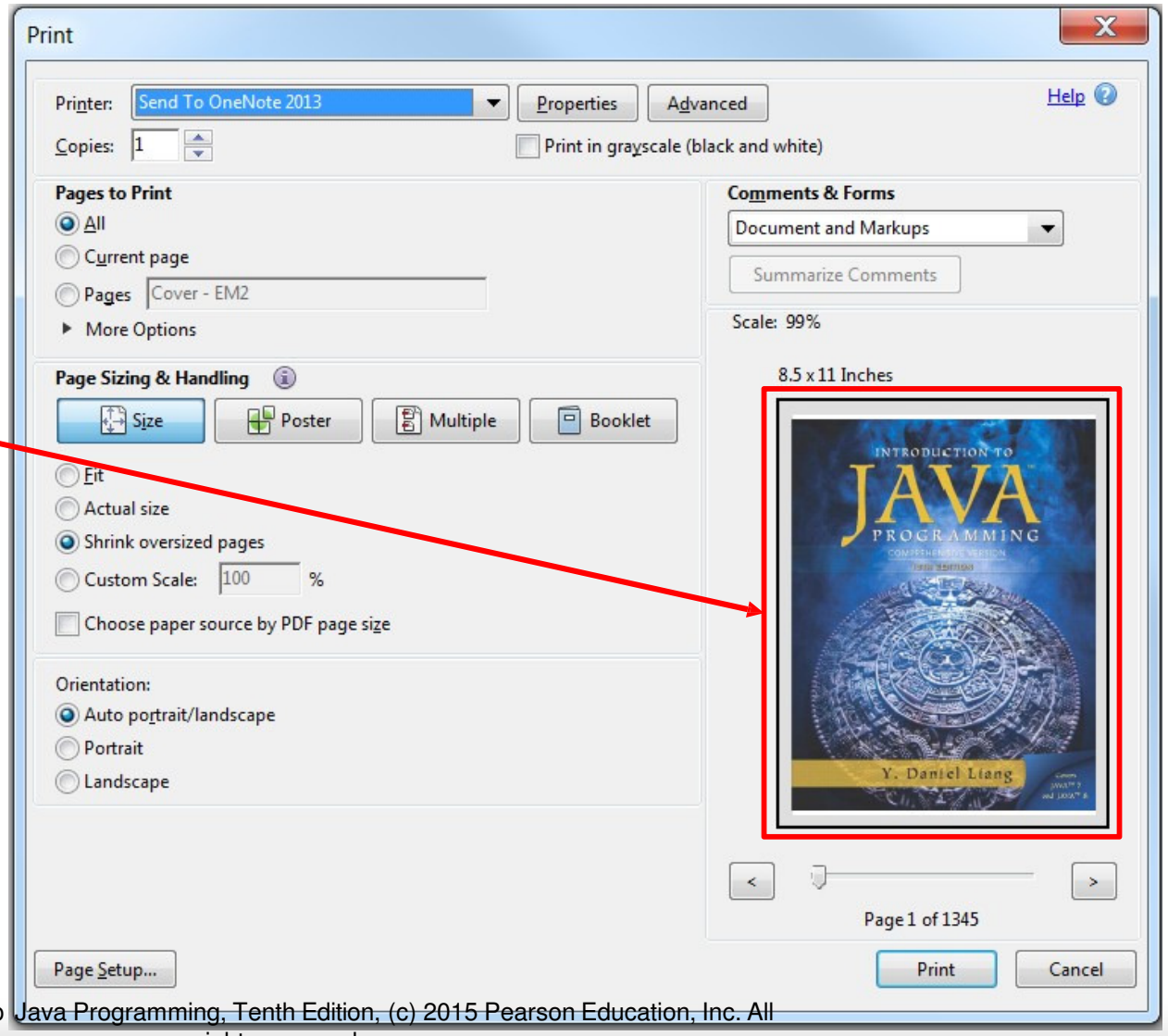
Sliders differ from ScrollBars, though, in that a Slider lets us select a value from a range, whereas a ScrollBar is usually used to let us scroll content that doesn't fit into its container



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

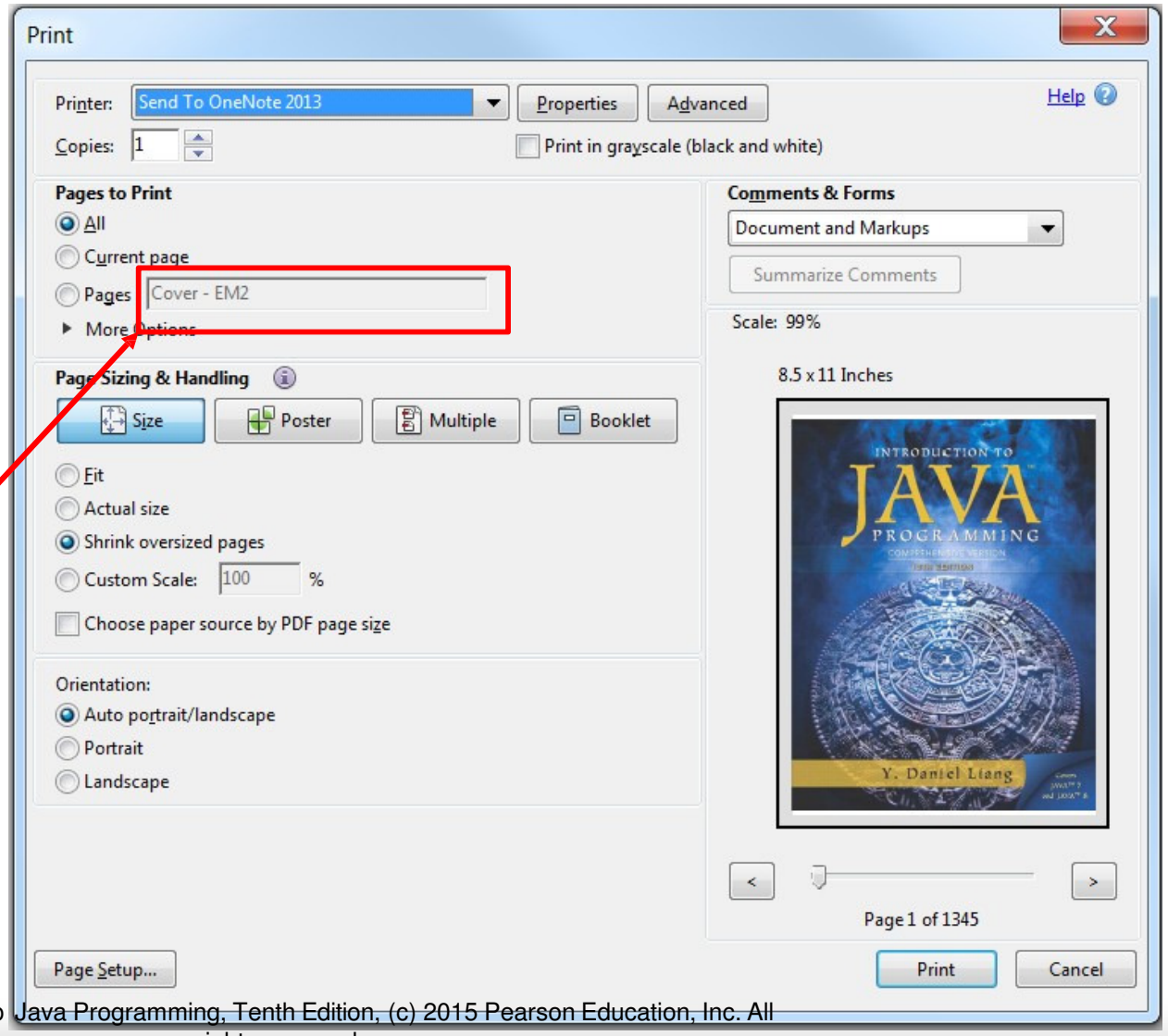
In this example, the image displayed is the one that corresponds to the page selected by the Slider below the ImageView



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")

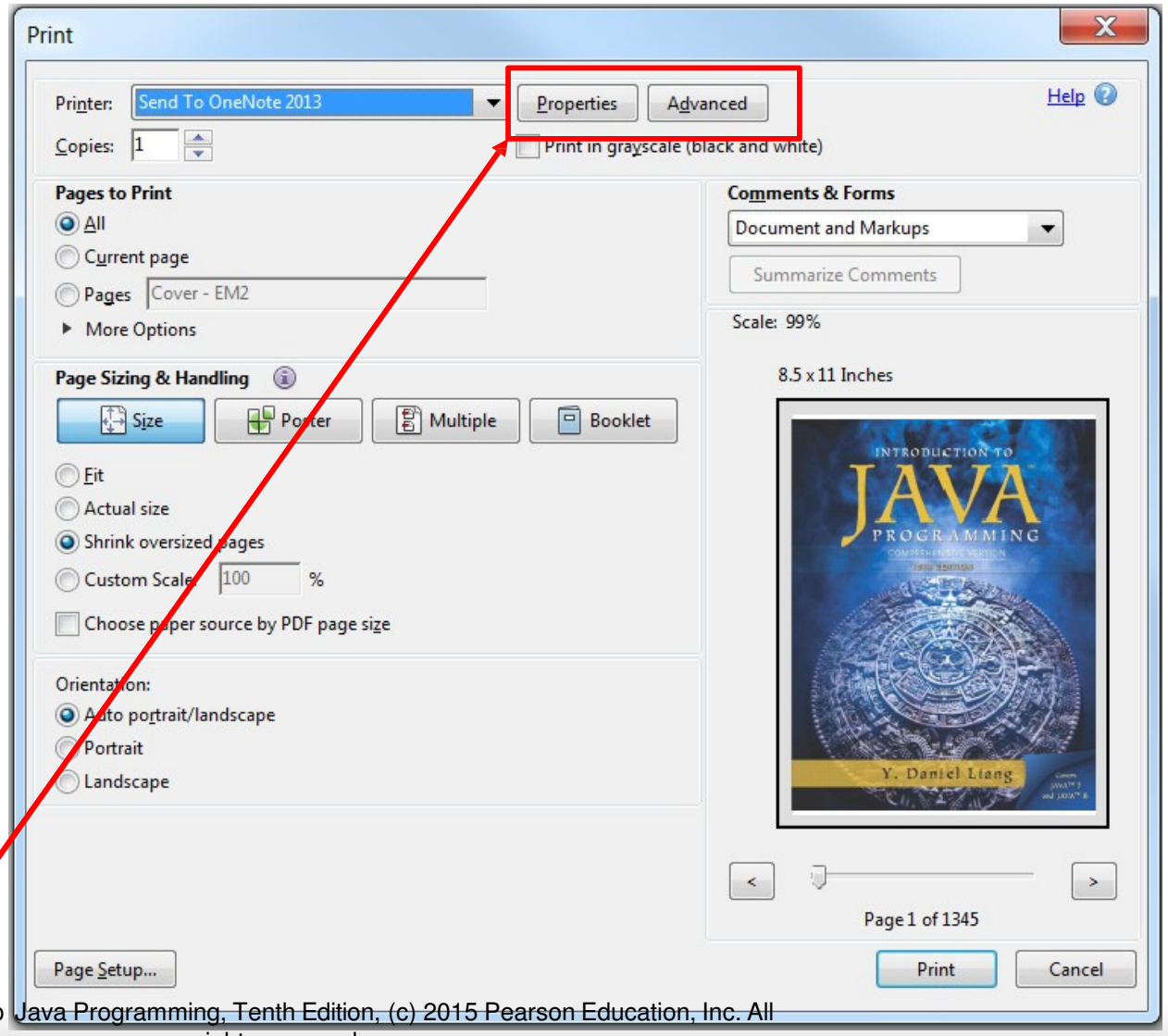
In this example, the page range TextBox is disabled when the "Pages" RadioButton is not selected – the TextBox is *there*, but we can't type in it



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)



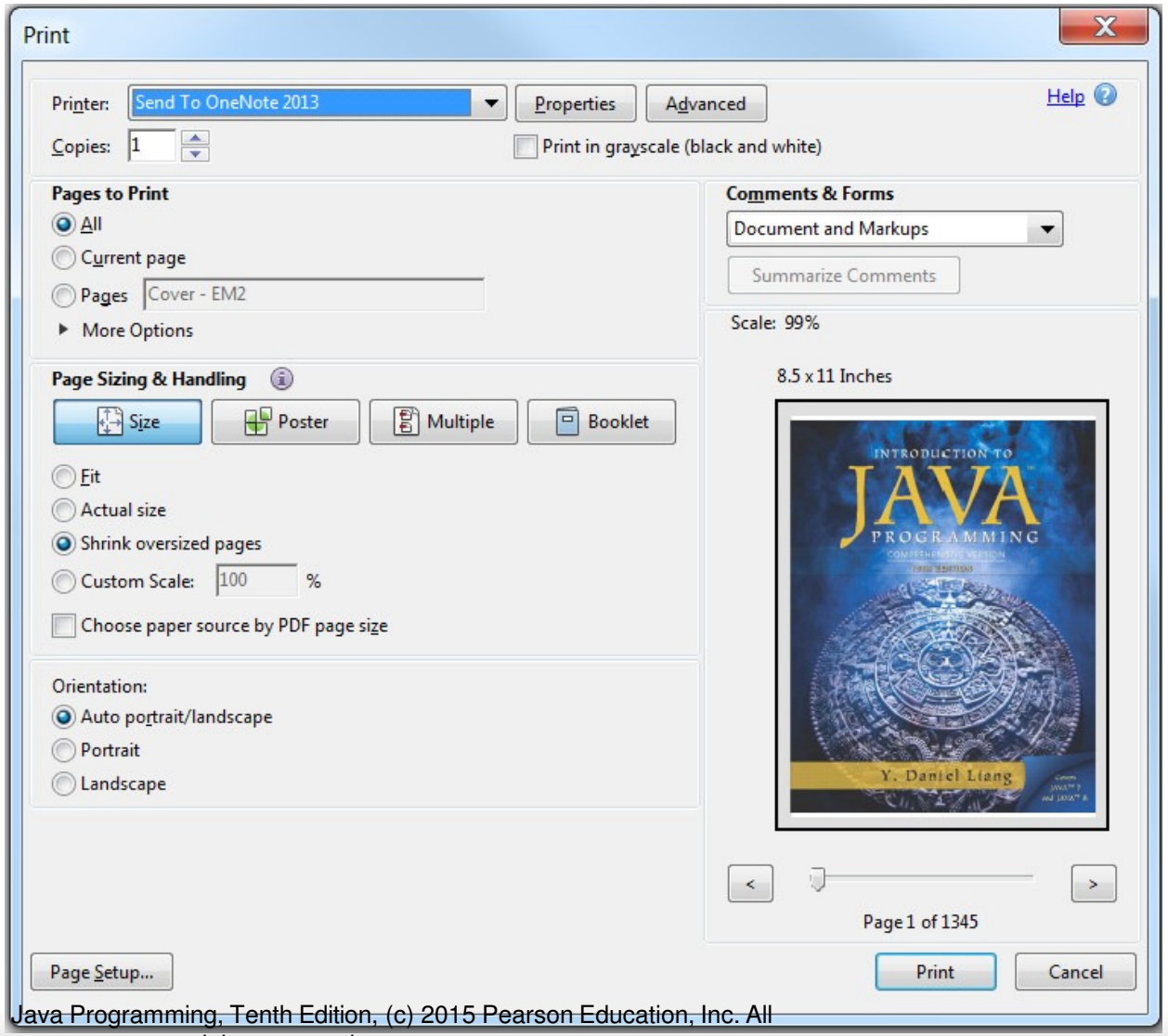
Introduction

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

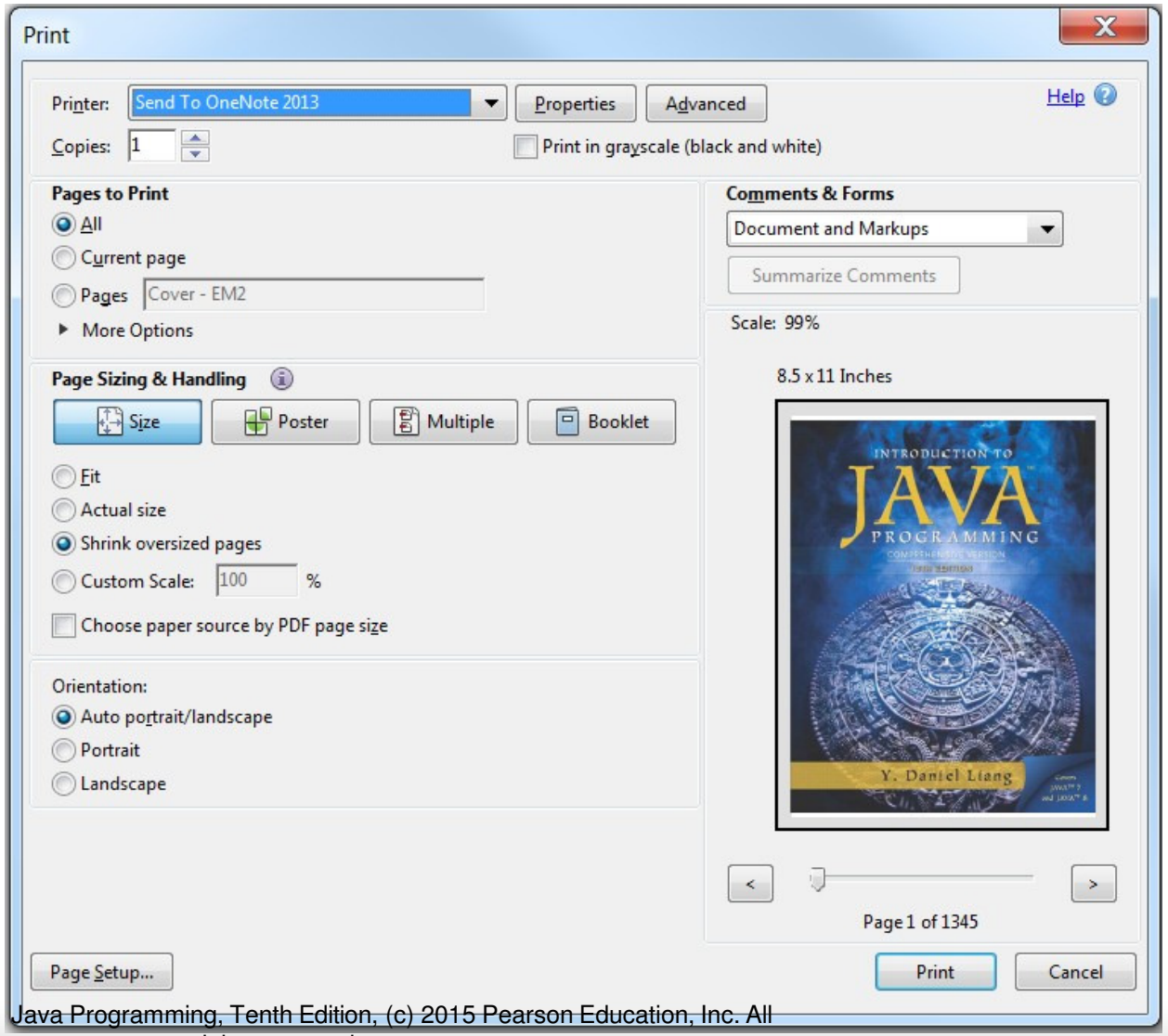


Introduction

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

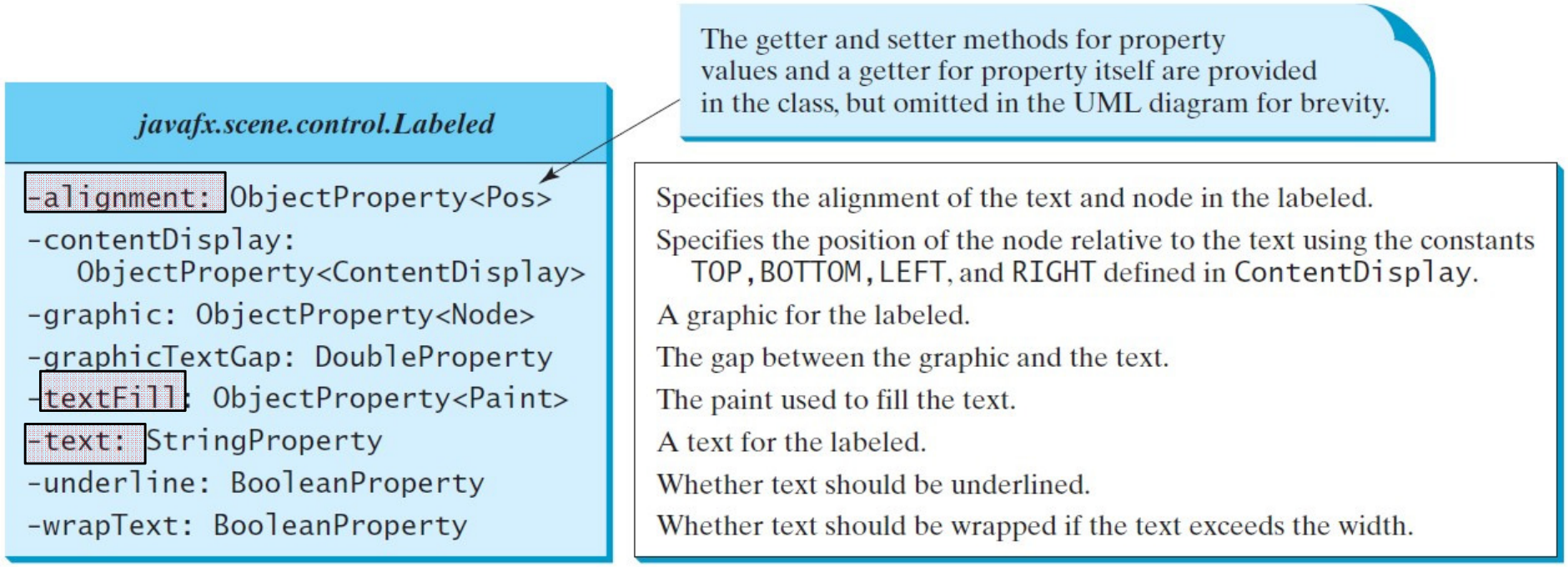
When ComboBox has the focus, the F4 key will make the box alternate between its dropped-down and collapsed views

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



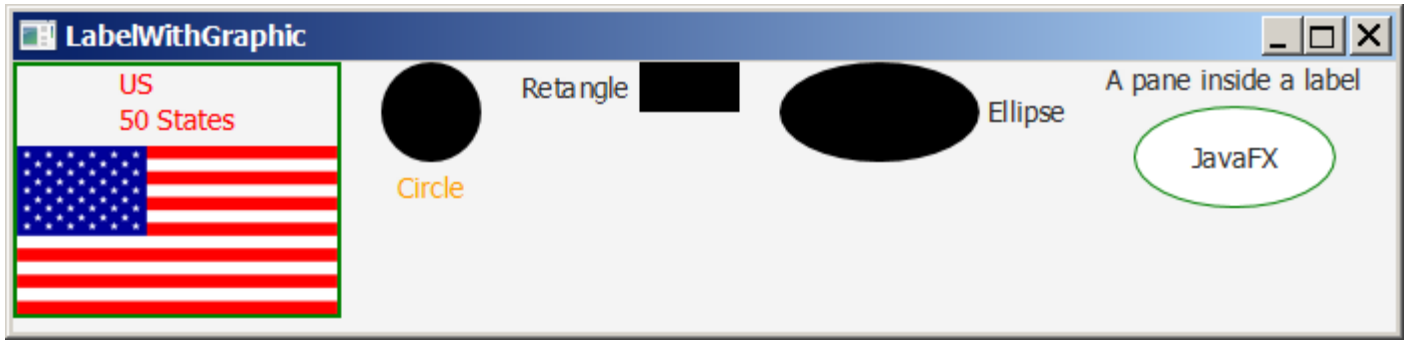
Labeled

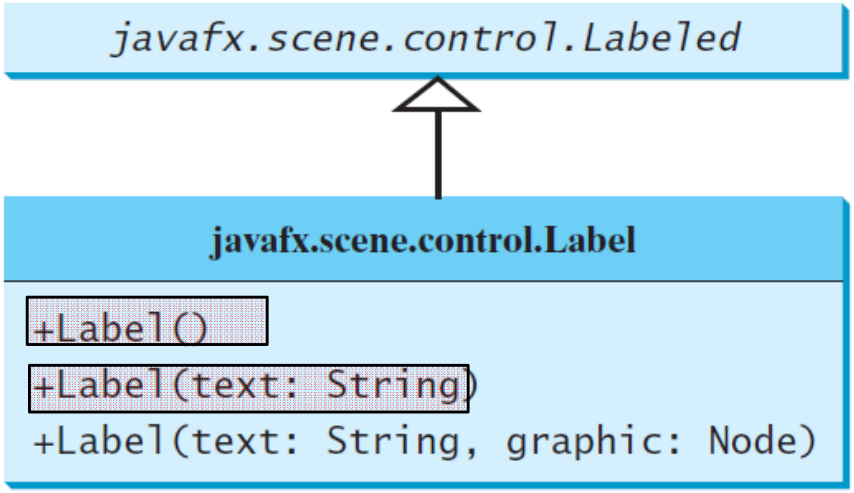
A *label* is a display area for a short text, a node, or both. It is often used to label other controls (usually text fields). Labels and buttons share many common properties. These common properties are defined in the **Labeled** class.



Label

The Label class defines labels.





Creates an empty label.

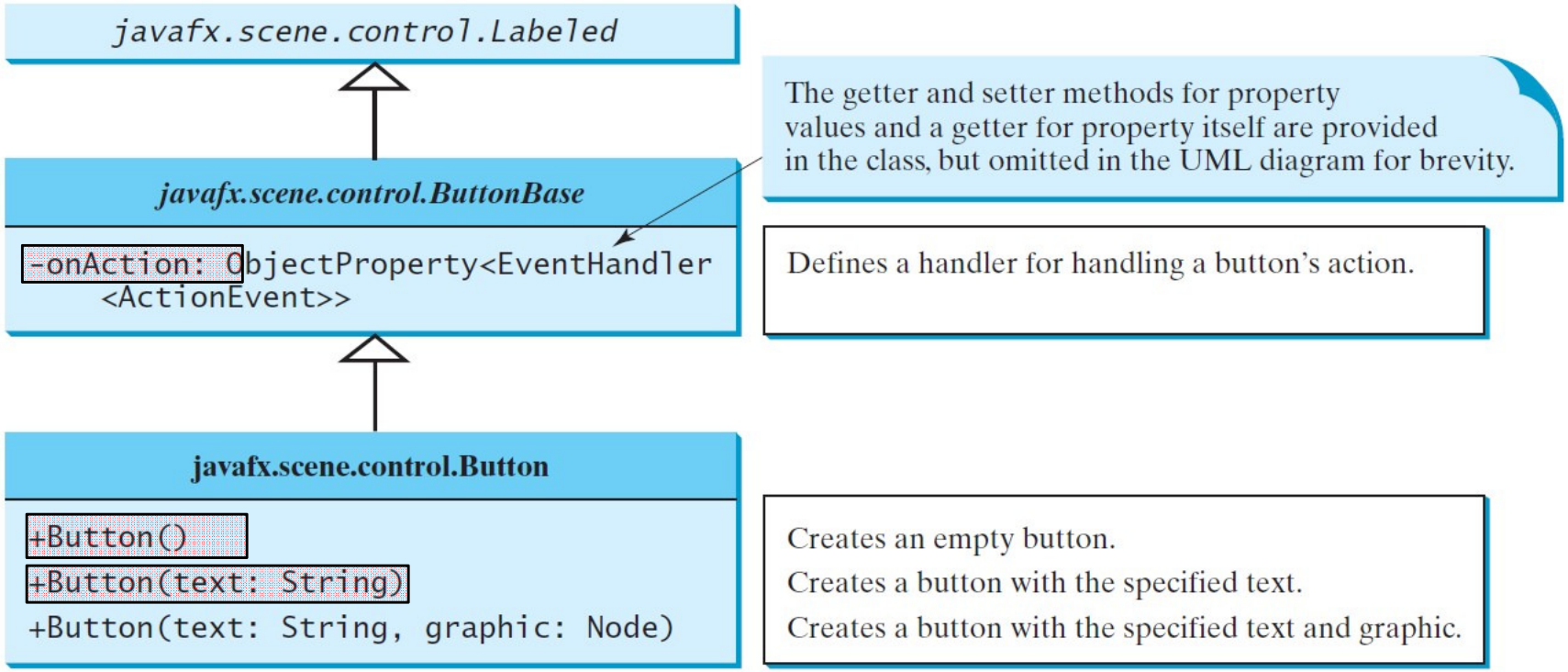
Creates a label with the specified text.

Creates a label with the specified text and graphic.

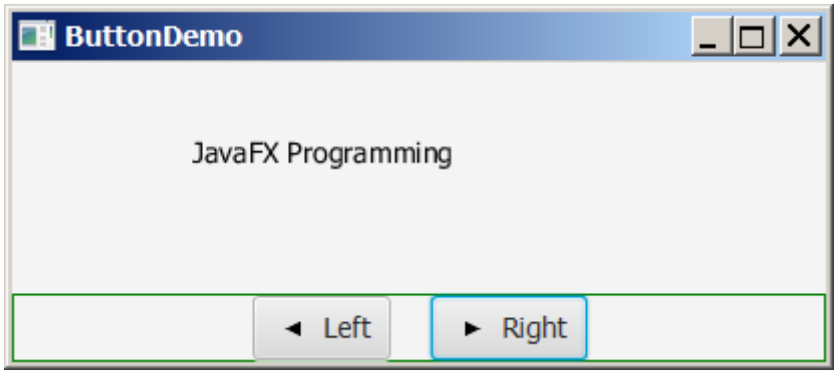
LabelWithGraphic Run

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.



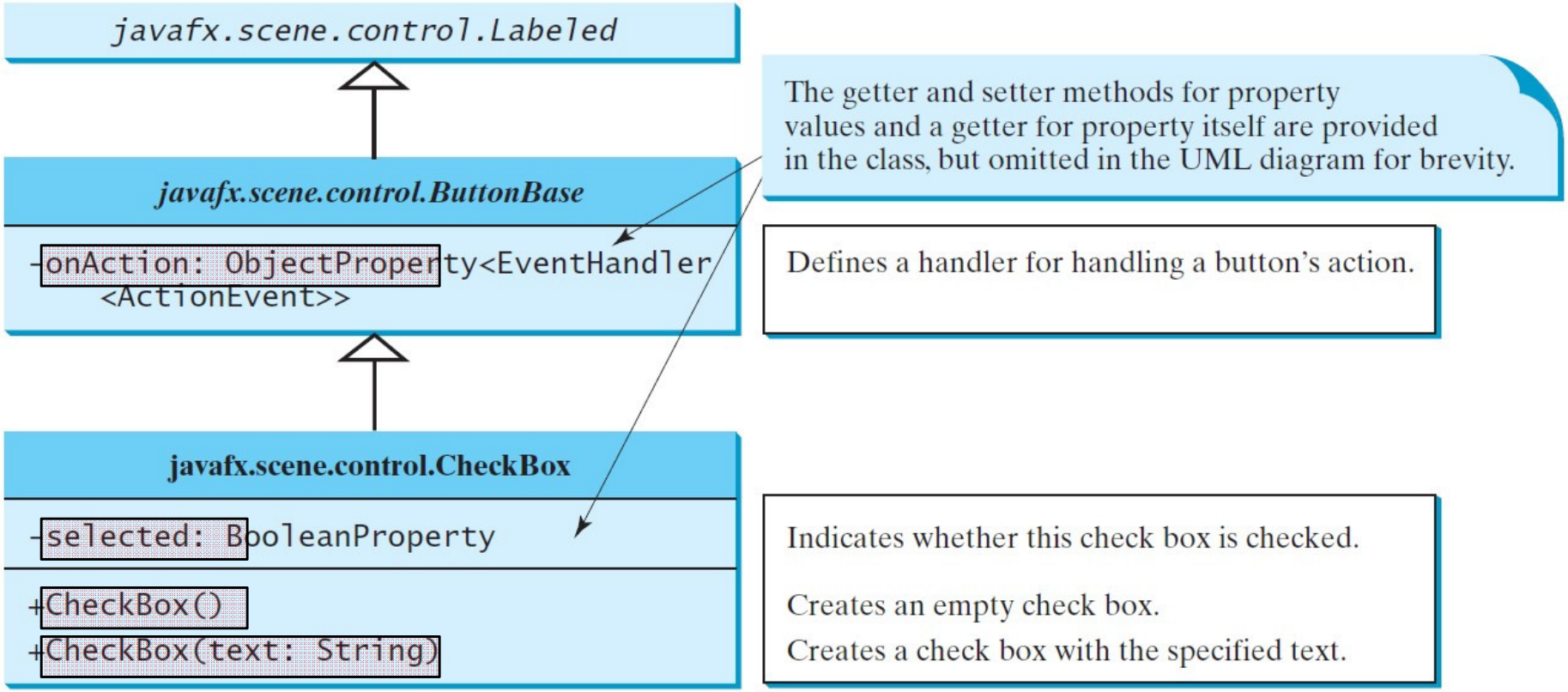
Button Example



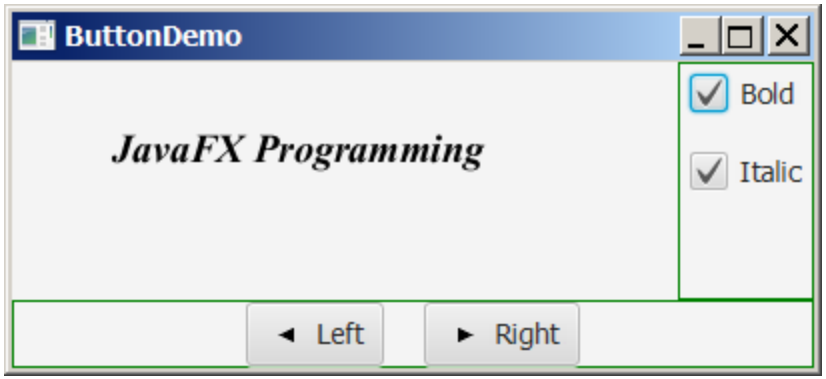
ButtonDemo Run

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**.



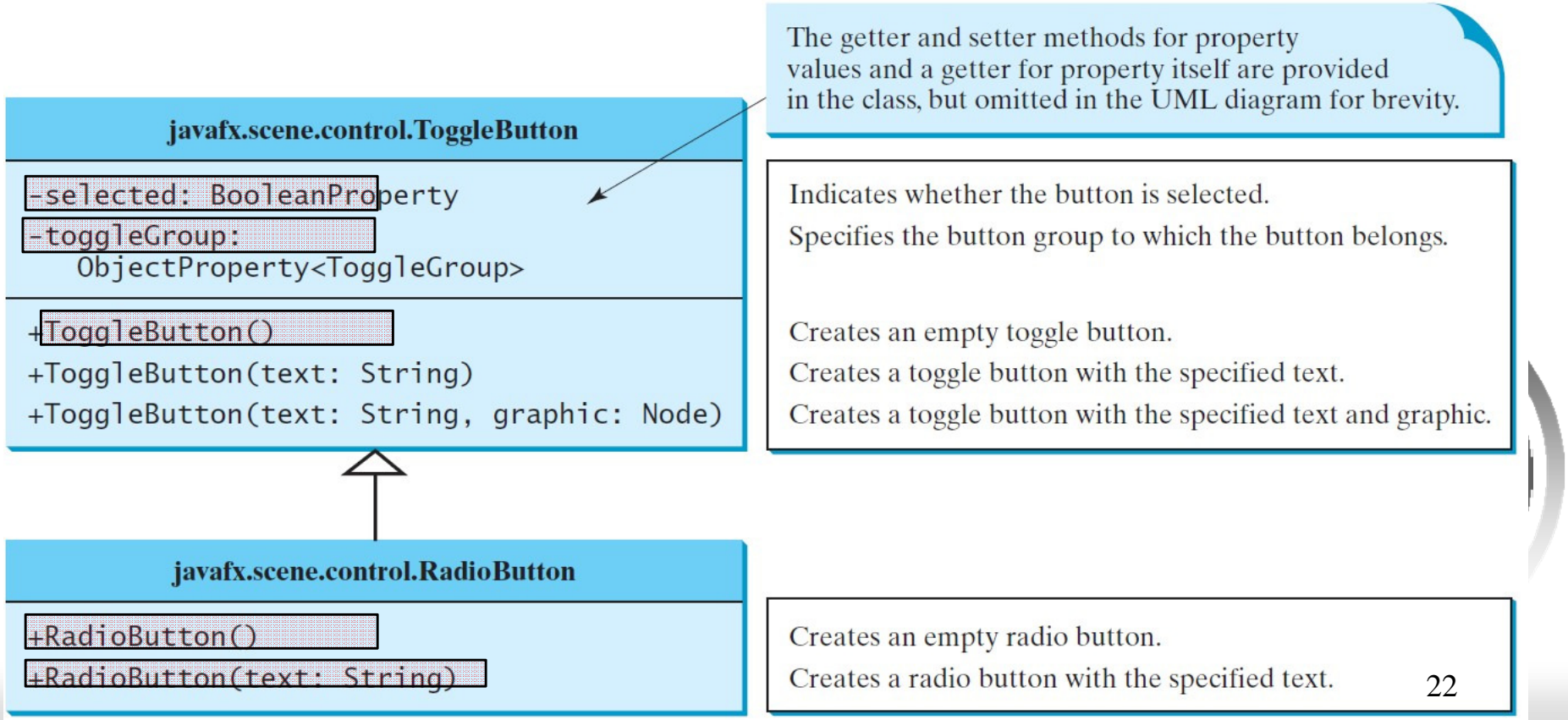
CheckBox Example



CheckBoxDemo Run

RadioButton

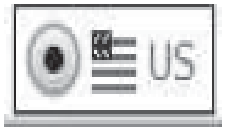
Radio buttons, also known as *option buttons*, enable you to choose a single item from a group of choices. In appearance radio buttons resemble check boxes, but check boxes display a square that is either checked or blank, whereas radio buttons display a circle that is either filled (if selected) or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



RadioButton

Here is an example of a radio button with text US, a graphic image, green text color, and black border, and initially selected.

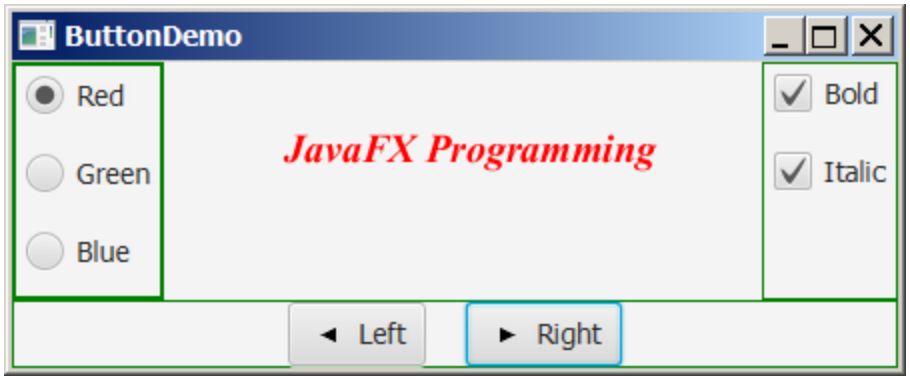
```
RadioButton rbUS = new RadioButton("US");
rbUS.setGraphic(new ImageView("image/usIcon.gif"));
rbUS.setTextFill(Color.GREEN);
rbUS.setContentDisplay(ContentDisplay.LEFT);
rbUS.setStyle("-fx-border-color: black");
rbUS.setSelected(true);
rbUS.setPadding(new Insets(5, 5, 5,));
```



To group radio buttons, you need to create an instance of ToggleGroup and set a radio button's toggleGroup property to join the group, as follows:

```
ToggleGroup group = new ToggleGroup();
rbRed.setToggleGroup(group);
rbGreen.setToggleGroup(group);
rbBlue.setToggleGroup(group);
```

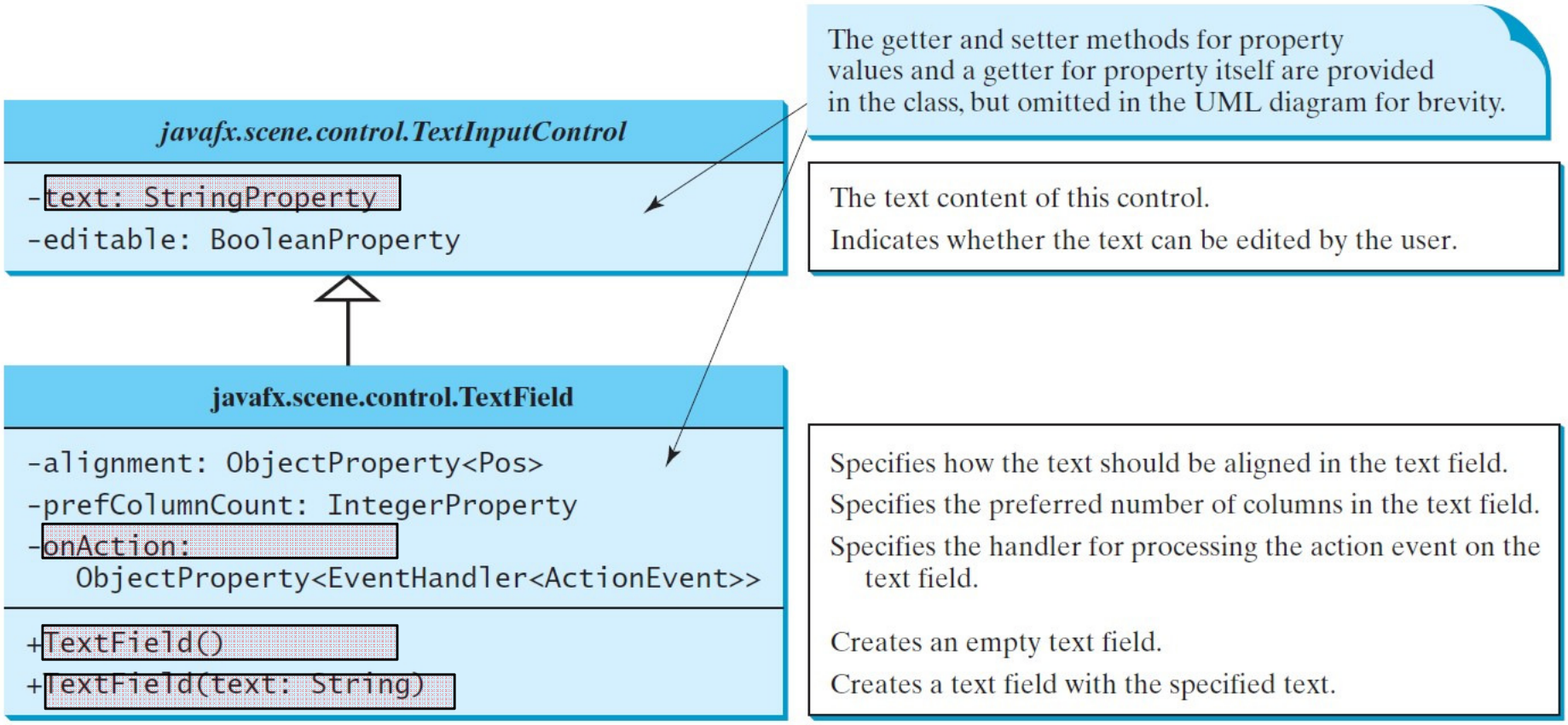
RadioButton Example



RadioButtonDemo Run

TextField

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



TextField

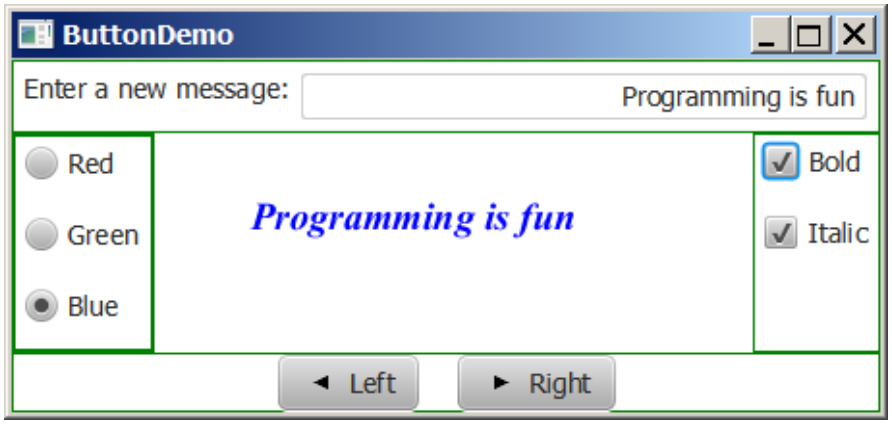
Here is an example of creating a noneditable text field with red text color, a specified font, and right horizontal alignment:

```
TextField tfMessage = new TextField("T-Strom");
tfMessage.setEditable(false);
tfMessage.setStyle("-fx-text-fill: red");
tfMessage.setFont(Font.font("Times", 20));
tfMessage.setAlignment(Pos.BASELINE_RIGHT);
```

TextField

If a text field is used for entering a password, use **PasswordField** to replace **TextField**. **PasswordField** extends **TextField** and hides the input text with echo characters ******.

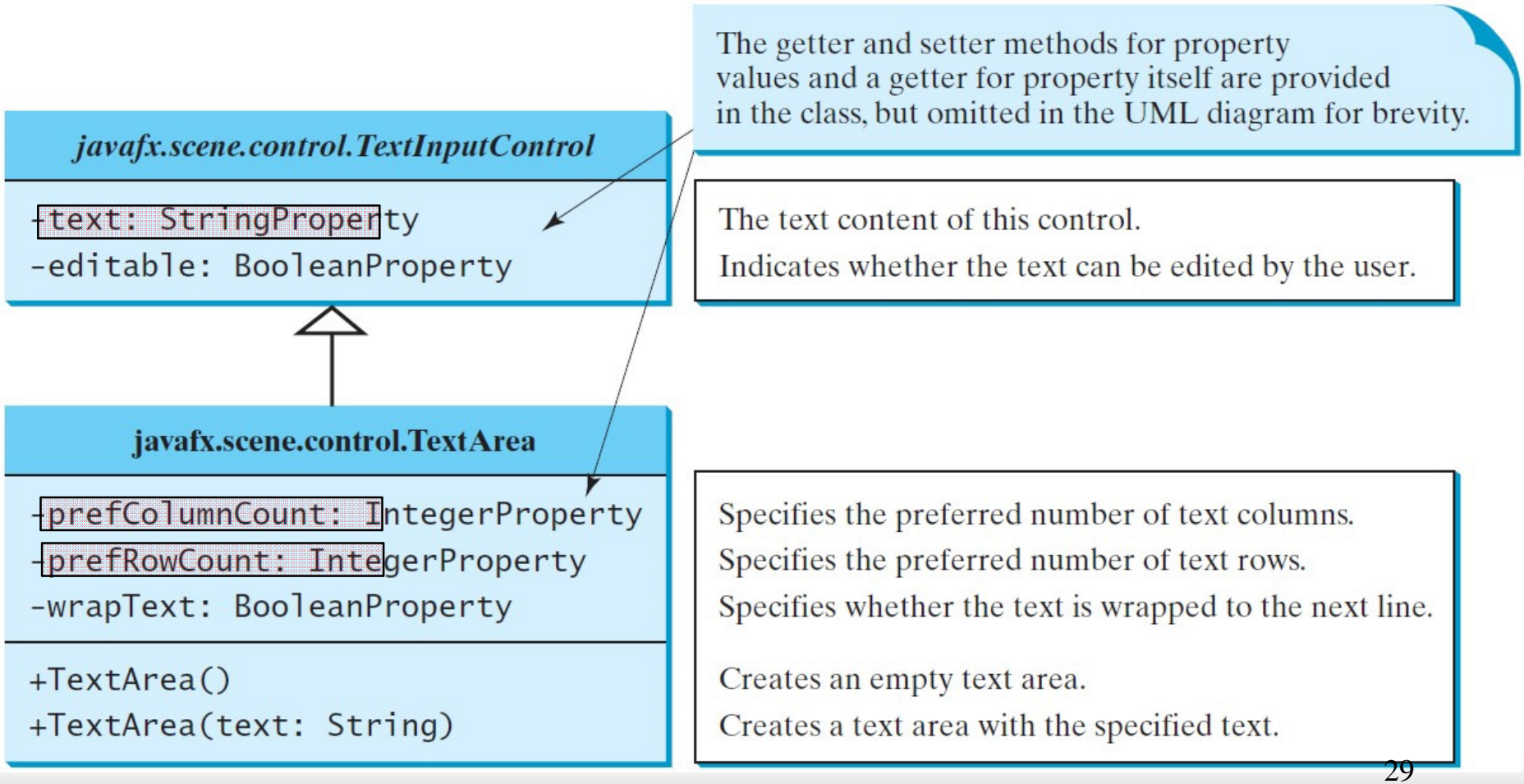
TextField Example



TextFieldDemo Run

TextArea

A **TextArea** enables the user to enter multiple lines of text.
If you want to let the user enter multiple lines of text, you may create several instances of **TextField**. A better alternative,is to use **TextArea**



TextArea

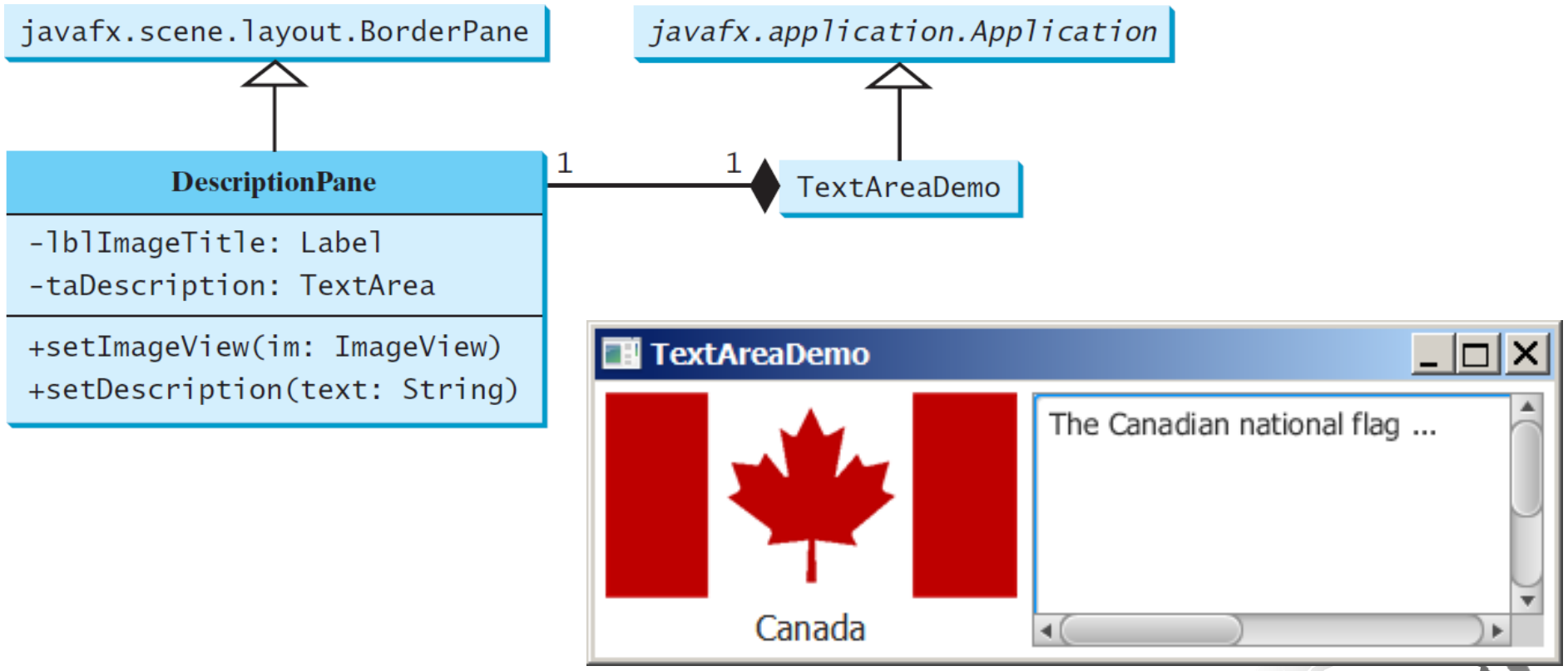
Here is an example of creating a text area with 5 rows and 20 columns, wrapped to the next line, red text color, and Courier font 20 pixels.

```
TextArea taNote = new TextArea("This is a text area");
taNote.setPrefColumnCount(20);
taNote.setPrefRowCount(5);
taNote.setWrapText(true);
taNote.setStyle("-fx-text-fill: red");
taNote.setFont(Font.font("Times", 20));
```

TextArea provides scrolling, but often it is useful to create a ScrollPane object to hold an instance of TextArea and let ScrollPane handle scrolling for TextArea, as follows:

```
// Create a scroll pane to hold text area
ScrollPane scrollPane = new ScrollPane(taNote);
```

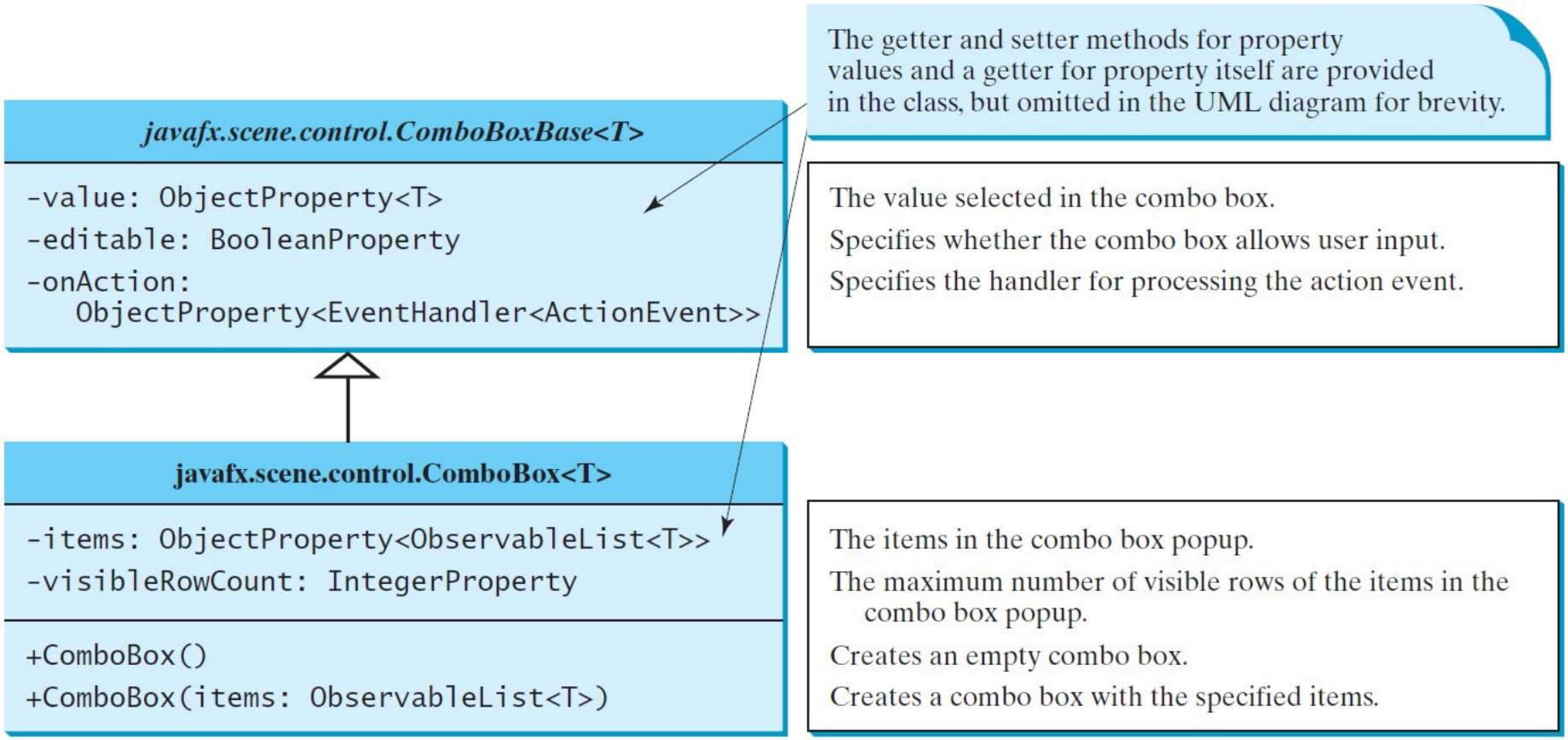
TextArea Example



DescriptionPane TextAreaDemo Run

ComboBox

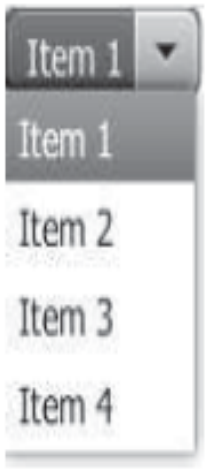
A combo box, also known as a choice list or drop-down list, contains a list of items from which the user can choose.



ComboBox

The following statements create a combo box with four items, red color, and value set to the first item.

```
ComboBox<String> cbo = new ComboBox<>();
cbo.getItems().addAll("Item 1", "Item 2",
  "Item 3", "Item 4");
cbo.setStyle("-fx-color: red");
cbo.setValue("Item 1");
```



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.

ComboBoxDemo Run