Unit: 8
JAVAFX UI Controls and Multimedia

Prof. Ravikumar R Natarajan

Dept. of CE

Contents

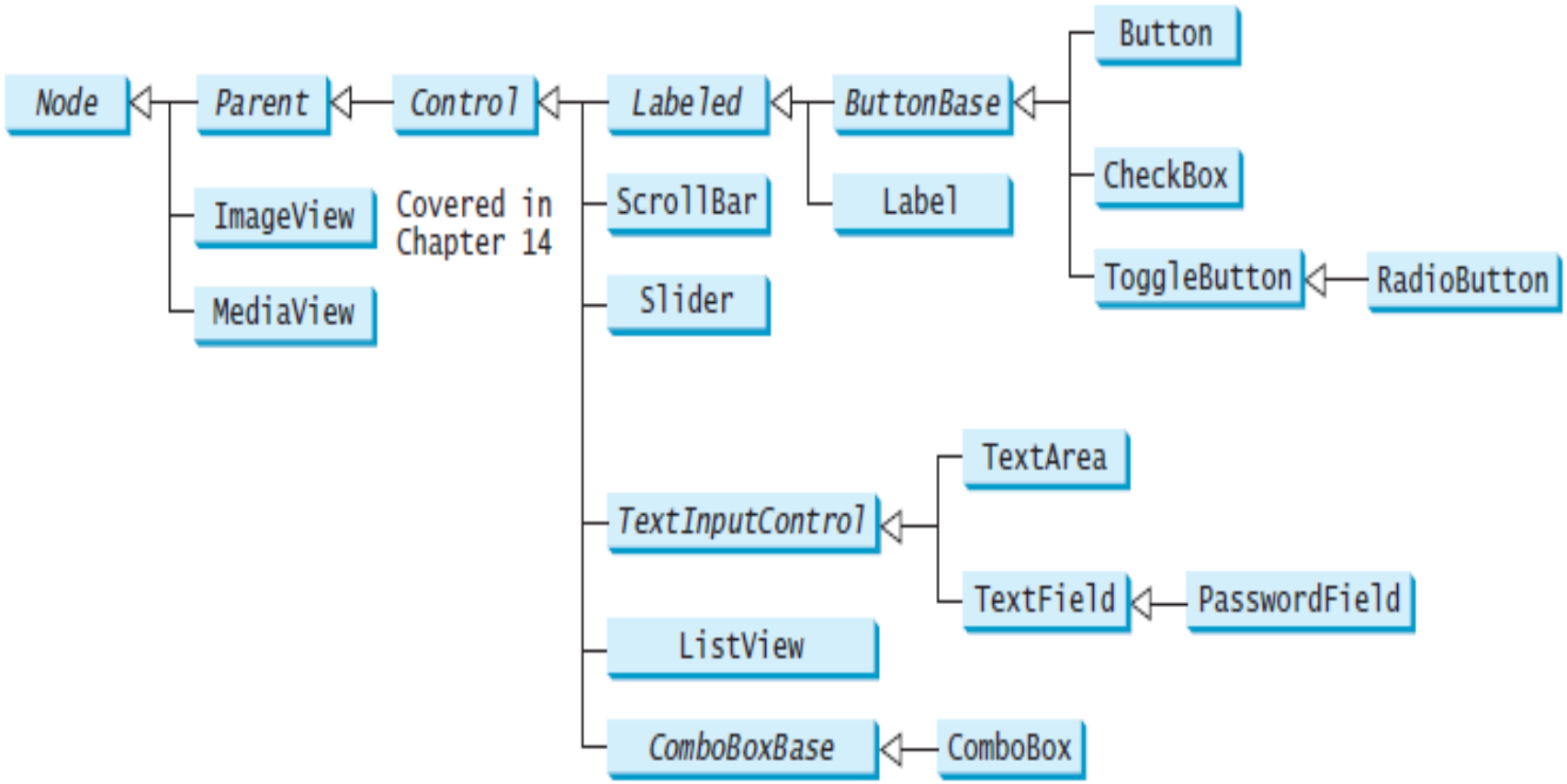
- Labeled and Label
- Button
- Checkbox
- RadioButton
- Textfield
- TextArea
- Combo Box
- ListView
- Scrollbar
- Slider
- Video and Audio

Introduction

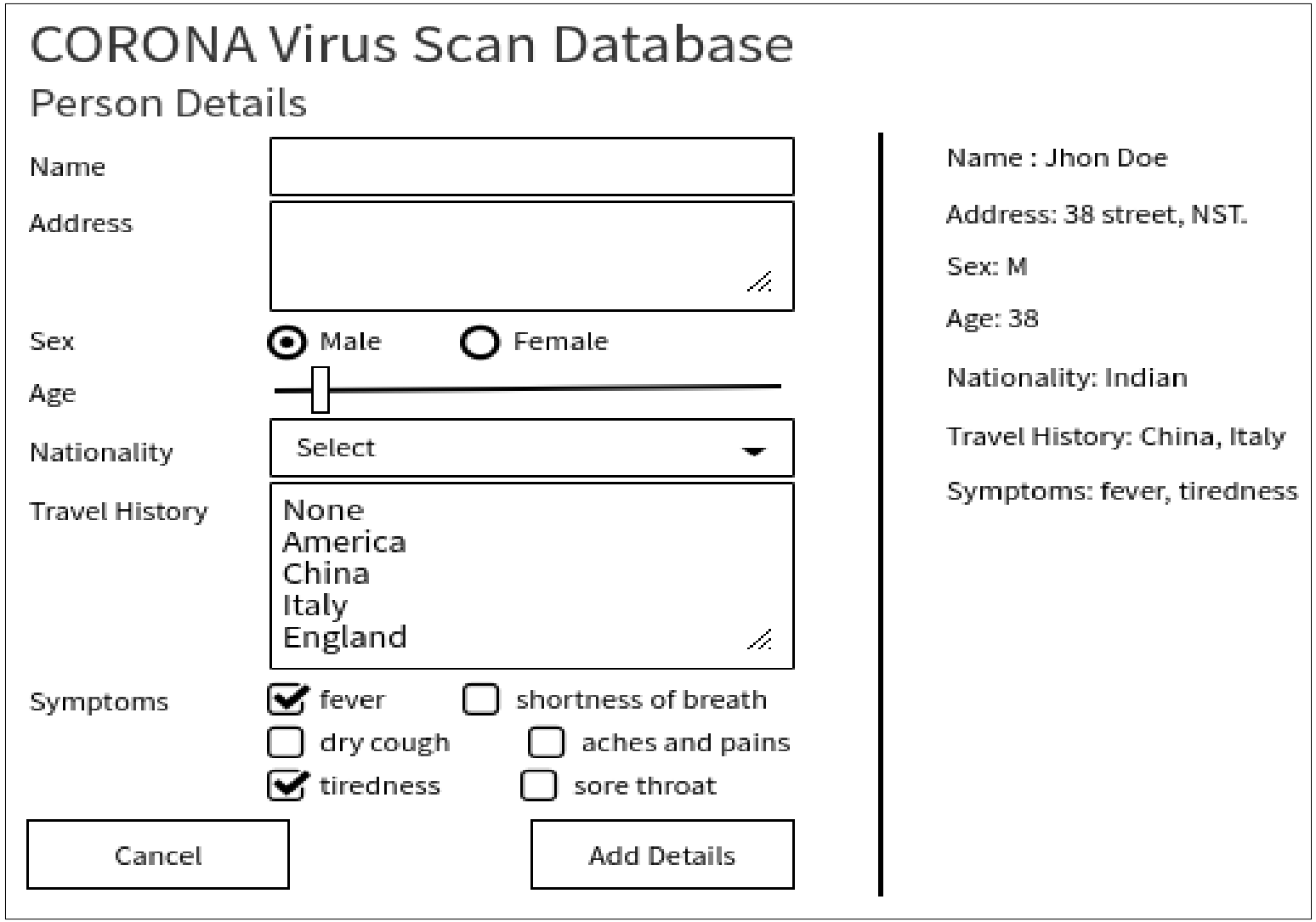
- **JavaFX provides many UI controls for developing a comprehensive user interface.**
- A graphical user interface (GUI) makes a system user-friendly and easy to use. Creating a GUI requires creativity and knowledge of how UI controls work.
- Since the UI controls in JavaFX are very flexible and versatile, you can create a wide assortment of useful user interfaces for rich Internet applications.

UI Controls

These UI controls are frequently used to create user interfaces.



A Sample UI Form

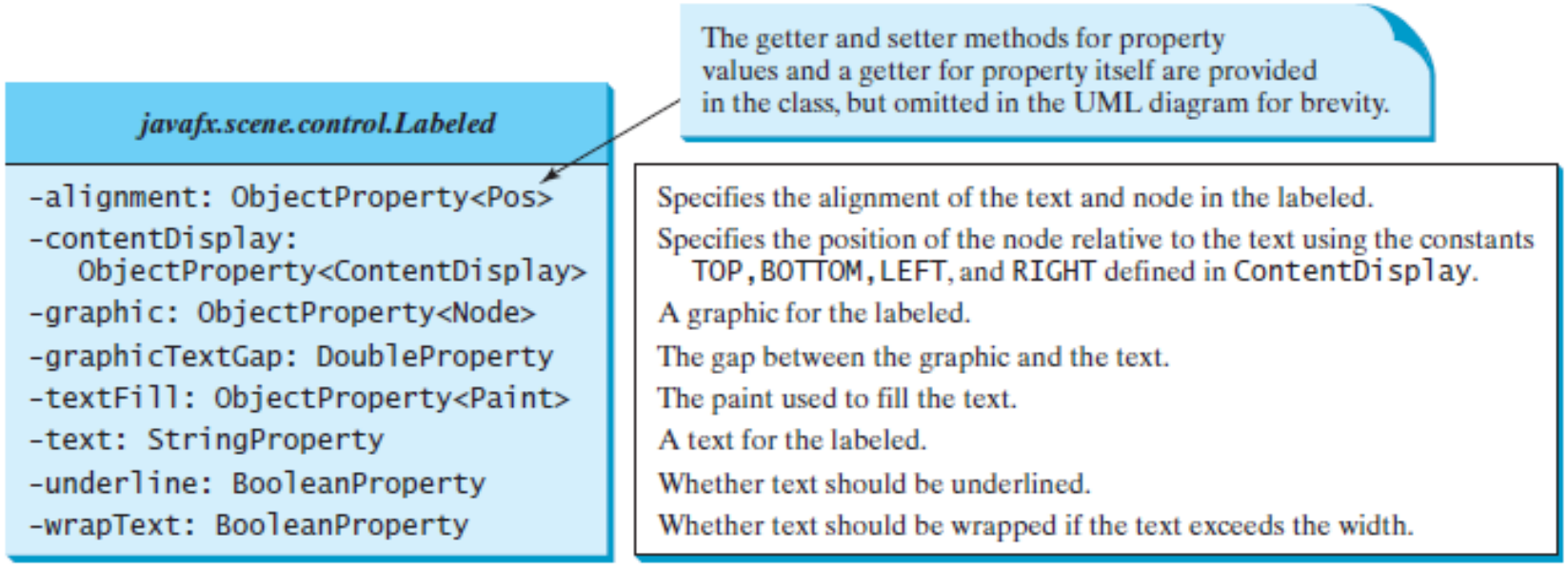


Labeled and Label

- 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.
- Labeled nodes can have the graphic property also, which can be any node such as a shape, an image, or a control.

Labeled and Label

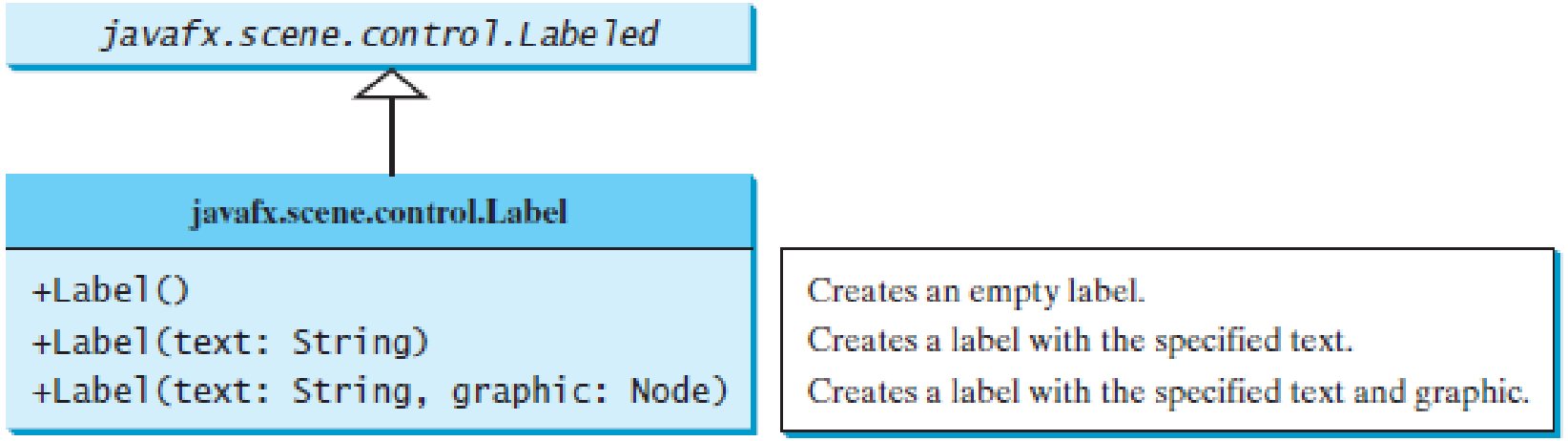
- Label is created to display a text or a node, or both.



*For Example Refer Unit 8 Demo Programs Folder

Labeled and Label

- Labeled defines common properties for Label, Button, CheckBox, and RadioButton



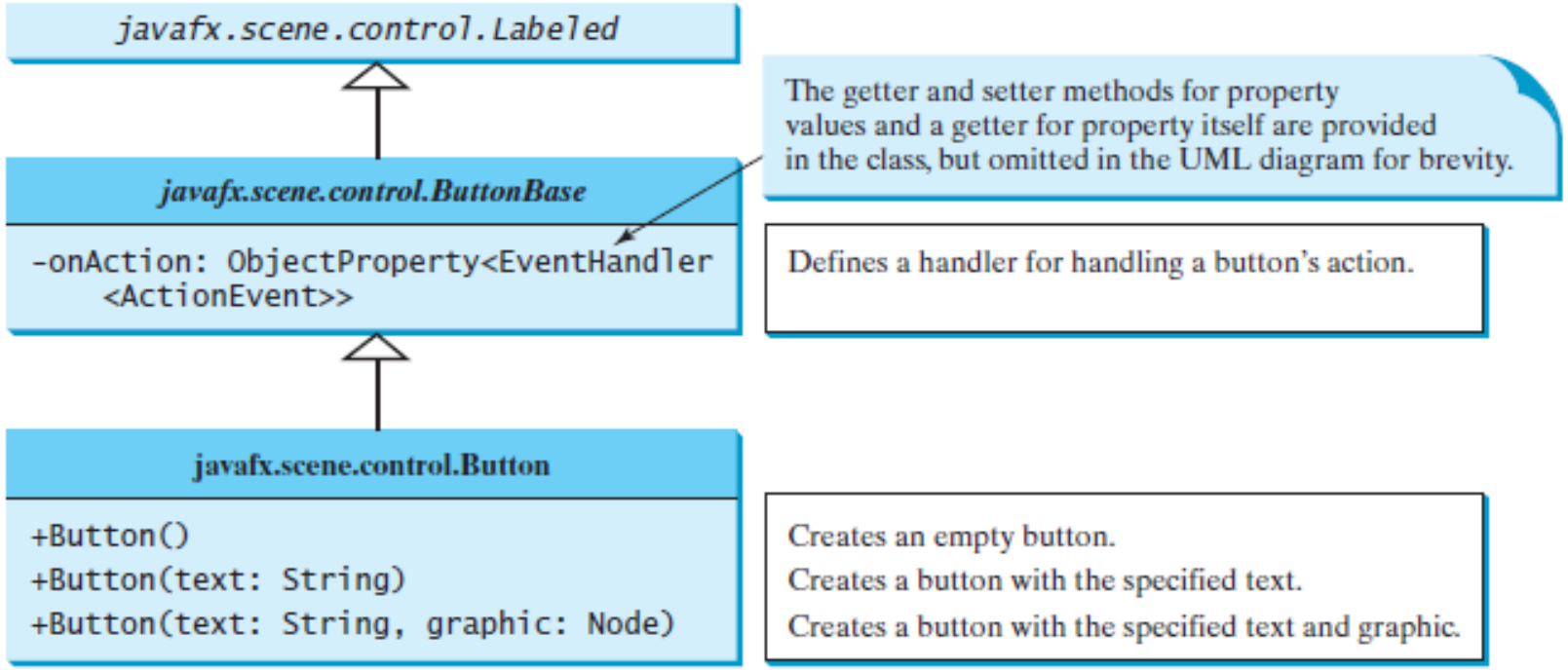
Labeled and Label

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*; import javafx.scene.layout.*;
import javafx.event.ActionEvent; import javafx.event.EventHandler;
import javafx.scene.control.Label; import javafx.stage.Stage;
public class labelDemo extends Application {
        public void start(Stage s)      {
                s.setTitle("Calculator");
                Label b = new Label("This is a Label");
                Pane r = new Pane();
                r.getChildren().addAll(b);
                Scene sc = new Scene(r, 600, 600);
                s.setScene(sc);
                s.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.
- The Labeled class defines the common properties for labels and buttons.
- A button is just like a label except that the button has the onAction property defined in the ButtonBase class, which sets a handler for handling a button's action.

Button

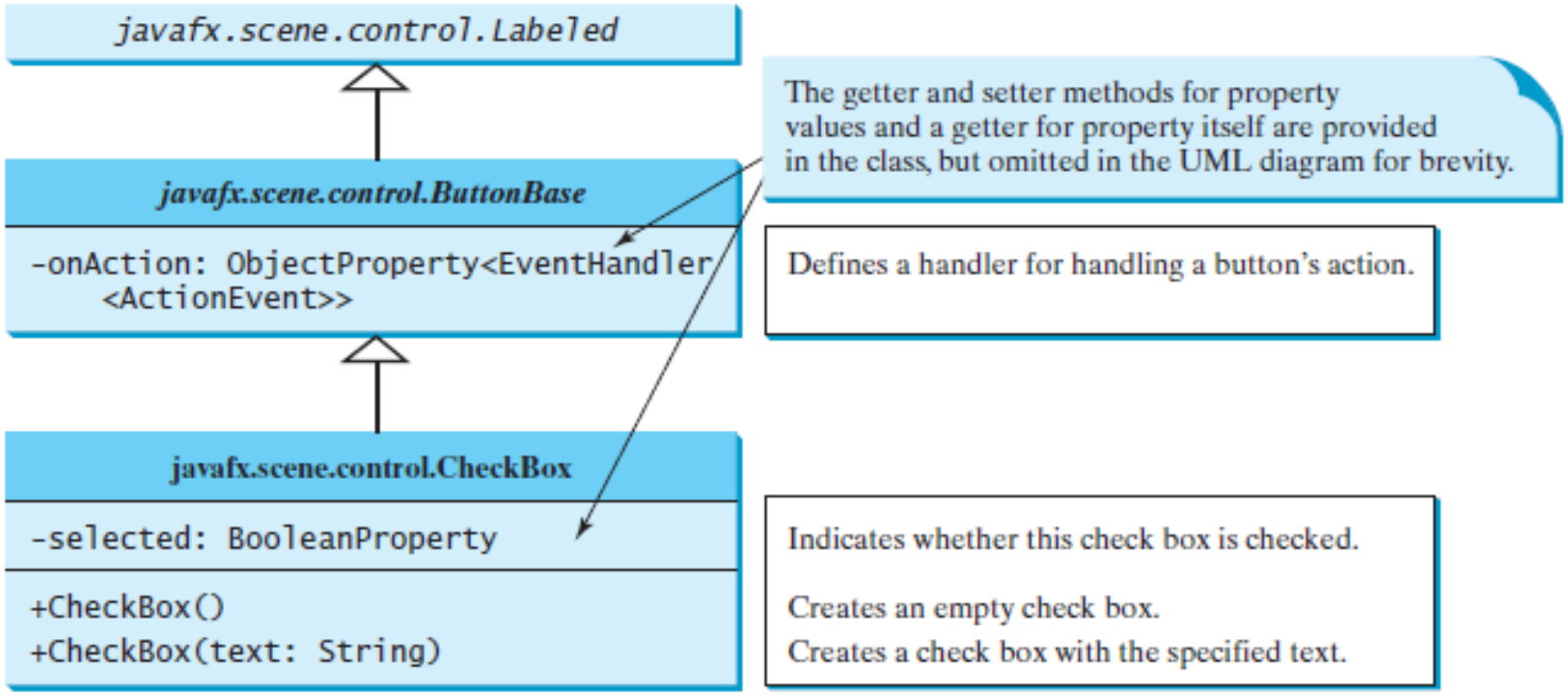


*For Example Refer Unit 8 Demo Programs Folder

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.
- Additionally, it provides the selection property to indicate whether a check box is selected.
- When a check box is clicked (checked or unchecked), it fires an ActionEvent.
- To see if a check box is selected, use the isSelected() method.

CheckBox



*For Example Refer Unit 8 Demo Programs Folder

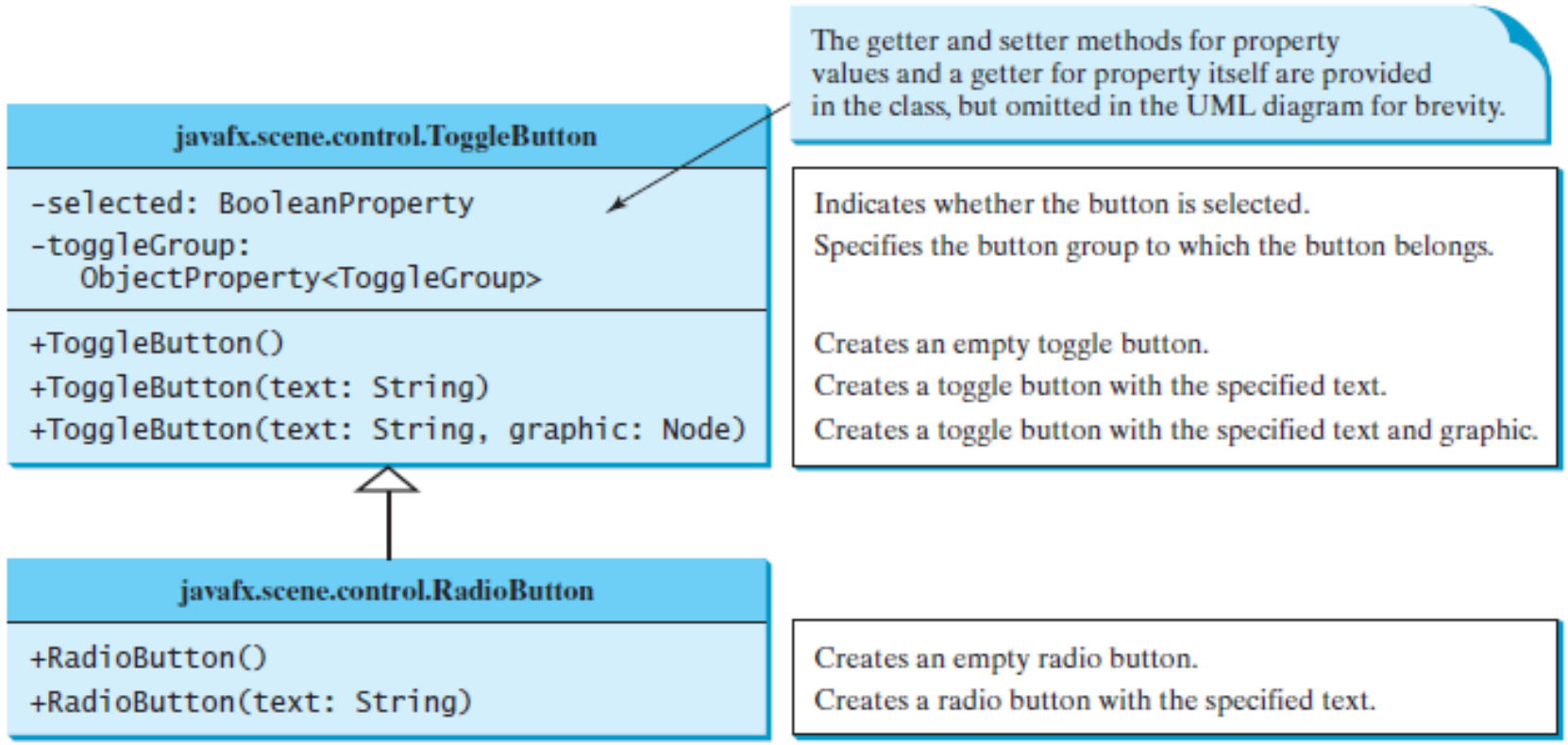
Radio Button

- Radio buttons, also known as option buttons, enable the user 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 is a subclass of ToggleButton.
- The difference between a radio button and a toggle button is that a radio button displays a circle, but a toggle button is rendered similar to a button.

Radio Button

- To group radio buttons, you need to create an instance of ToggleGroup and set a radio button's toggleGroup property to join the group.
- When a radio button is changed (selected or deselected), it fires an ActionEvent . To see if a radio button is selected, use the isSelected() method.

RadioButton



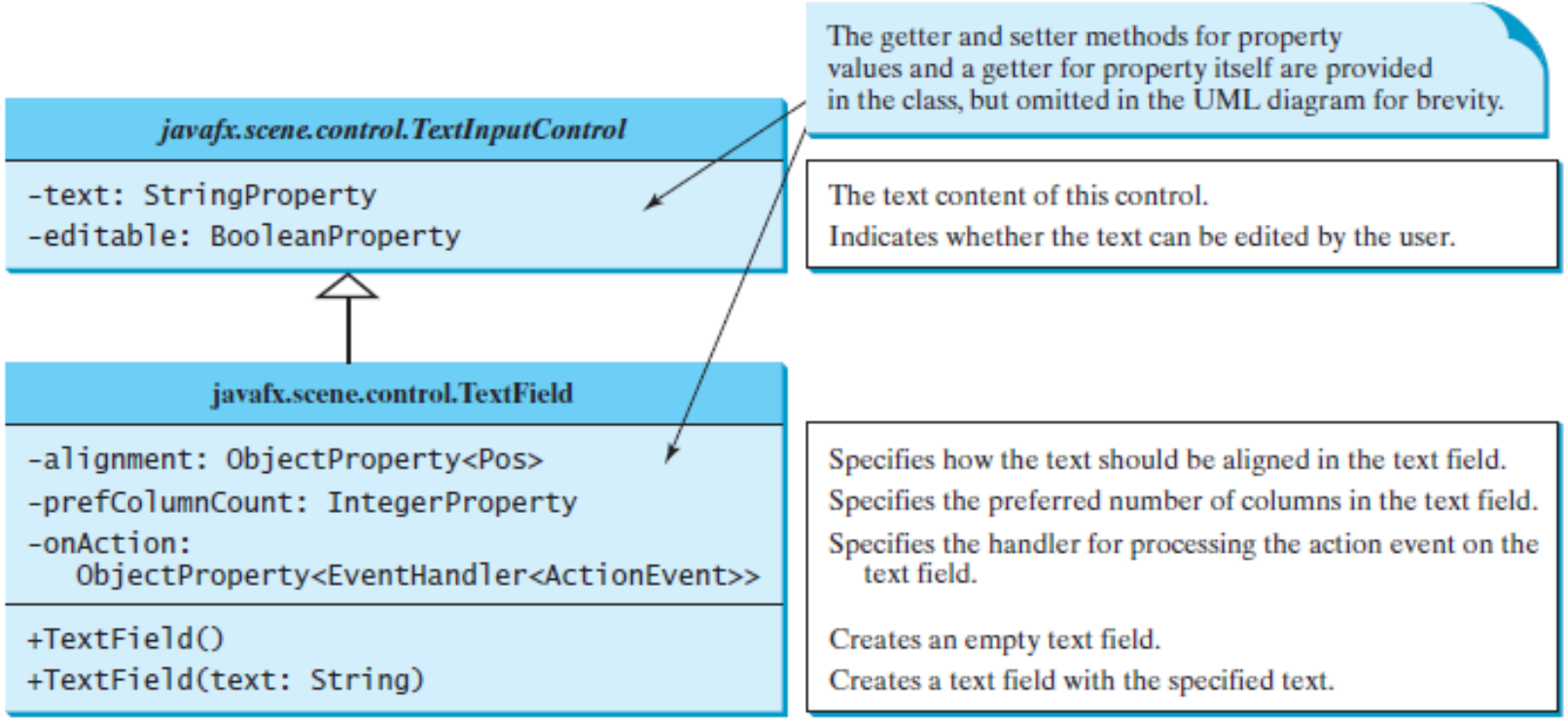
*For Example Refer Unit 8 Demo Programs Folder

RadioButton, CheckBox, Button, Vbox, Hbox

TextField

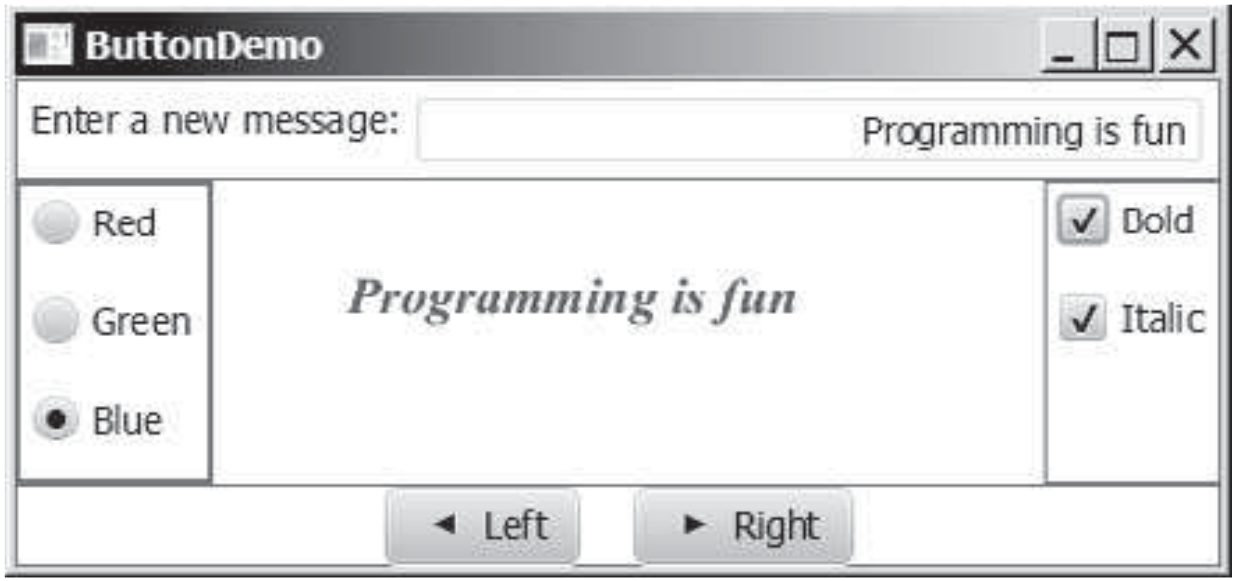
- A text field can be used to enter or display a string.
- TextField is a subclass of TextInputControl.
- When you move the cursor in the text field and press the Enter key, it fires an ActionEvent.

TextField



*For Example Refer Unit 8 Demo Programs Folder

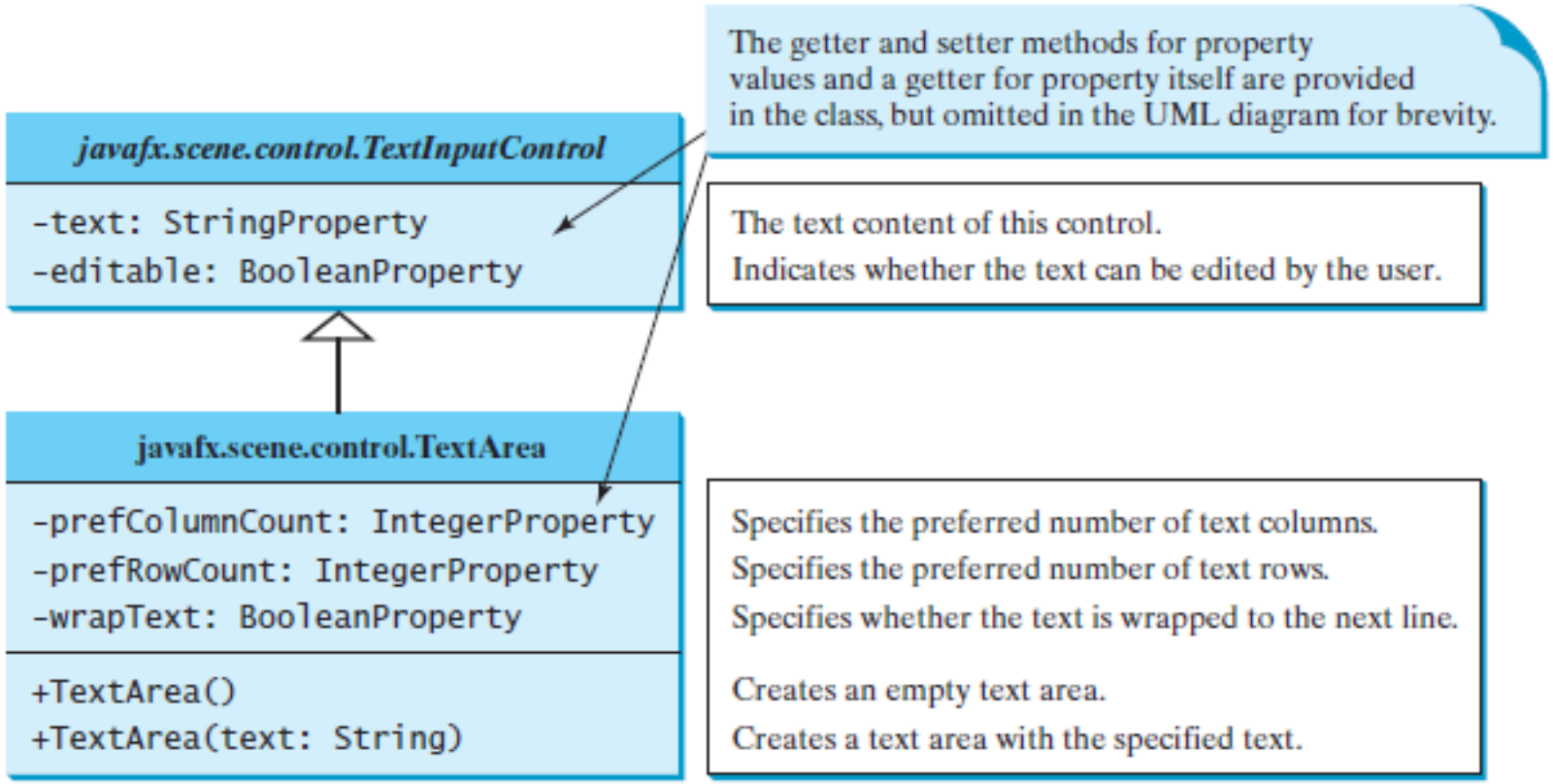
Identify TextField



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, however, is to use TextArea, which enables the user to enter multiple lines of text.
- TextArea also 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.

TextArea



*For Example Refer Unit 8 Demo Programs Folder

ComboBox

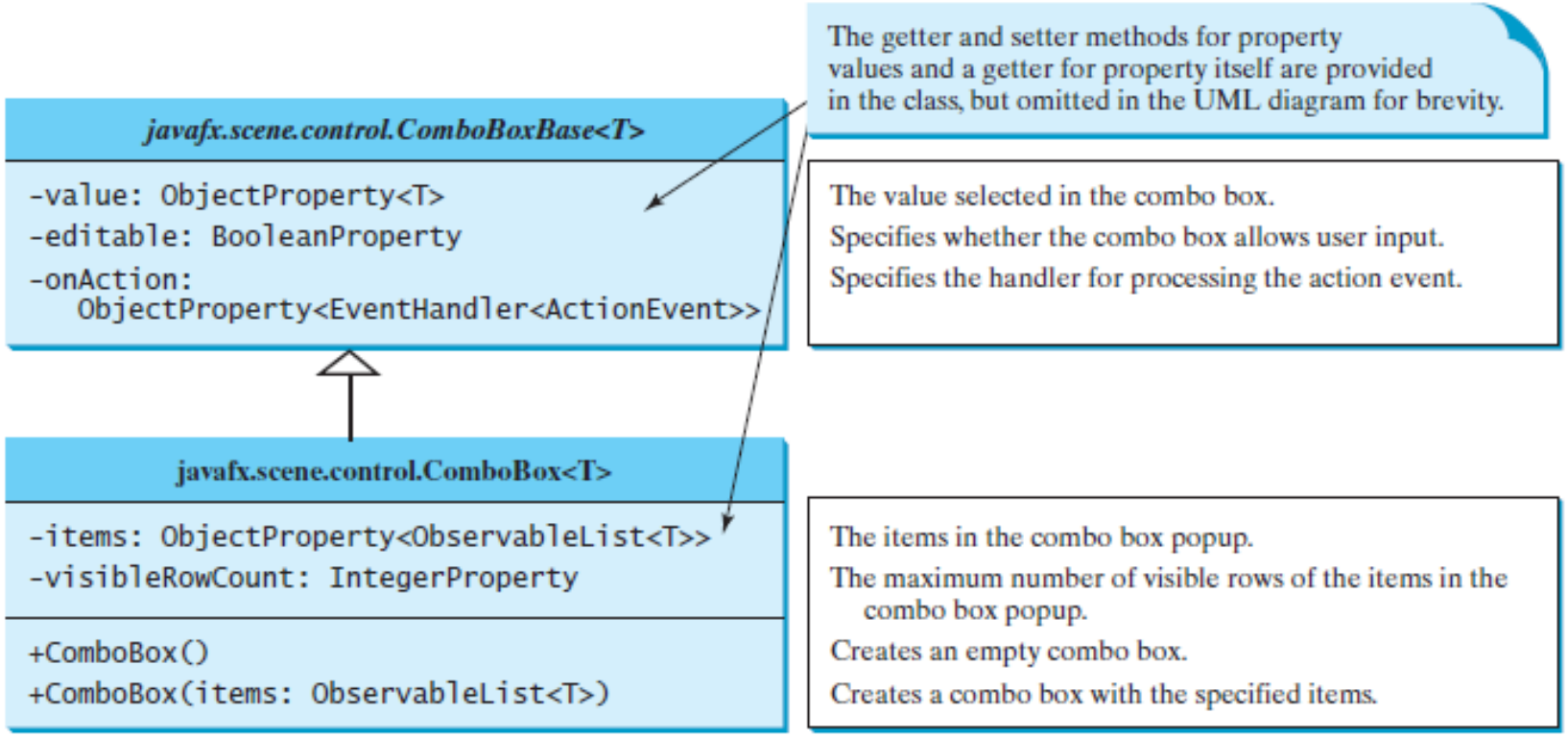
- A combo box, also known as a choice list or drop-down list, contains a list of items from which the user can choose.
- A combo box is useful for limiting a user's range of choices and avoids the cumbersome validation of data input.
- ComboBox inherits from ComboBoxBase.
- ComboBox is defined as a generic class.
- The generic type T specifies the element type for the elements stored in a combo box.

```
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

- ComboBox can fire an ActionEvent.
- Whenever an item is selected, an ActionEvent is fired.
- ObservableList is a subinterface of java.util.List. So you can apply all the methods defined in List for an ObservableList.
- For convenience, JavaFX provides the static method FXCollections.observableArrayList(arrayOfElements) for creating an ObservableList from an array of elements.

ComboBox



*For Example Refer Unit 8 Demo Programs Folder

ListView

- A list view is a control that basically performs the same function as a combo box, but it enables the user to choose a single value or multiple values.
- ListView is defined as a generic class.
- The generic type T specifies the element type for the elements stored in a list view.
- The getSelectionModel() method returns an instance of SelectionModel, which contains the methods for setting a selection mode and obtaining selected indices and items.
- The selection mode is defined in one of the two constants SelectionMode.MULTIPLE and
- SelectionMode.SINGLE, which indicates whether a single item or multiple items can be selected.
- The default value is SelectionMode.SINGLE.

ListView

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>>
-orientation: BooleanProperty

-selectionModel:
   ObjectProperty<MultipleSelectionModel<T>>

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

The items in the list view.

Indicates whether the items are displayed horizontally or vertically in the list view.

Specifies how items are selected. The `SelectionModel` is also used to obtain the selected items.

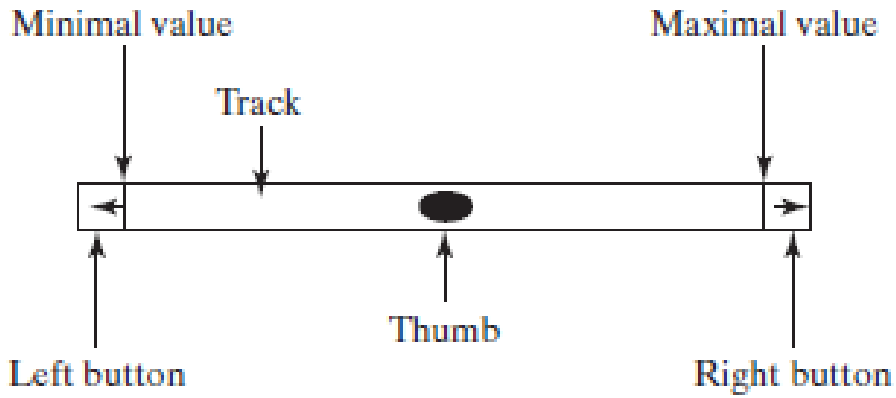
Creates an empty list view.

Creates a list view with the specified items.

*For Example Refer Unit 8 Demo Programs Folder

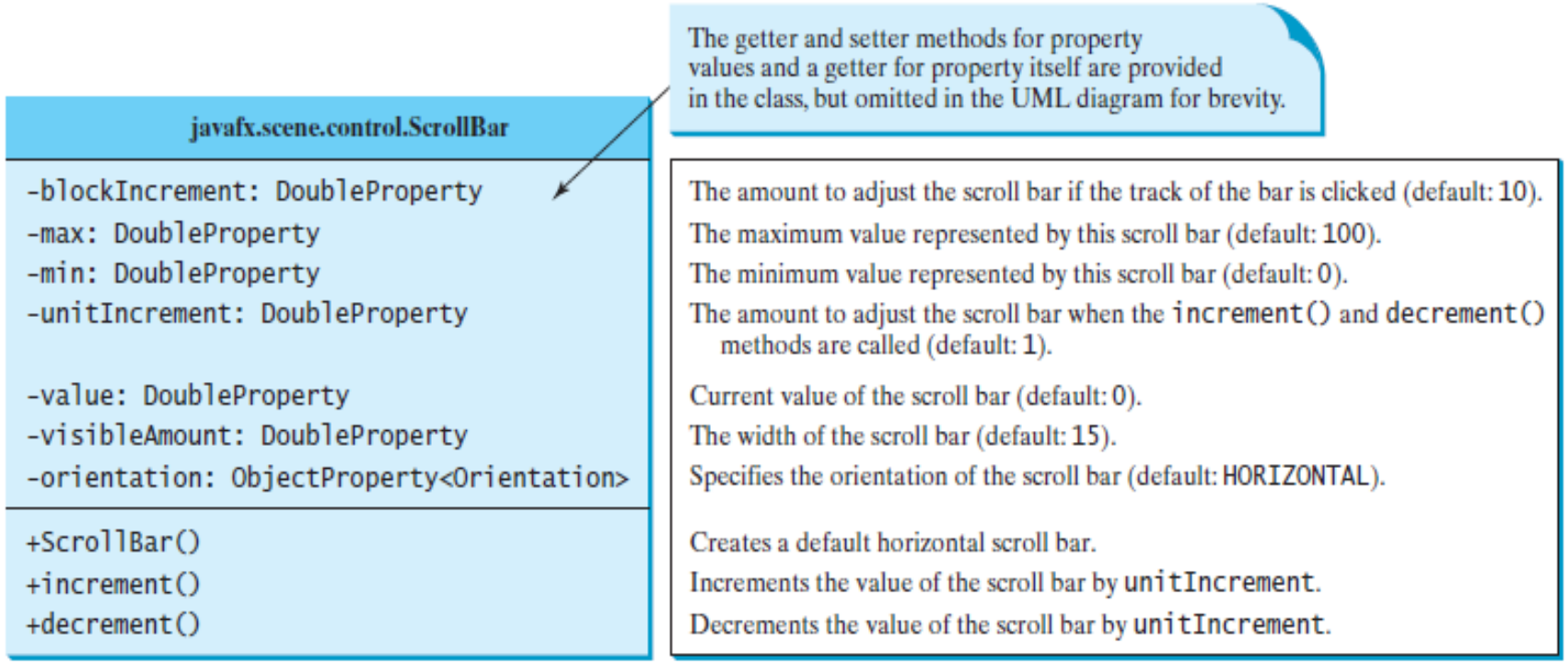
ScrollBar

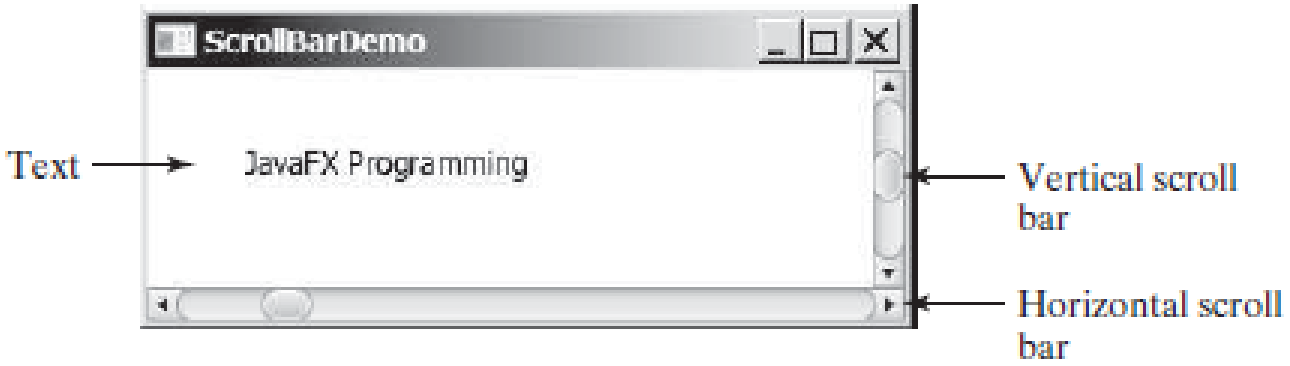
- ScrollBar is a control that enables the user to select from a range of values.
- Normally, the user changes the value of a scroll bar by making a gesture with the mouse.
- For example, the user can drag the scroll bar's thumb, click on the scroll bar track, or the scroll bar's left or right buttons.
- When the user changes the value of the scroll bar, it notifies the listener of the change.
- You can register a listener on the scroll bar's valueProperty for responding to this change.



A scroll bar represents a range of values graphically.

ScrollBar



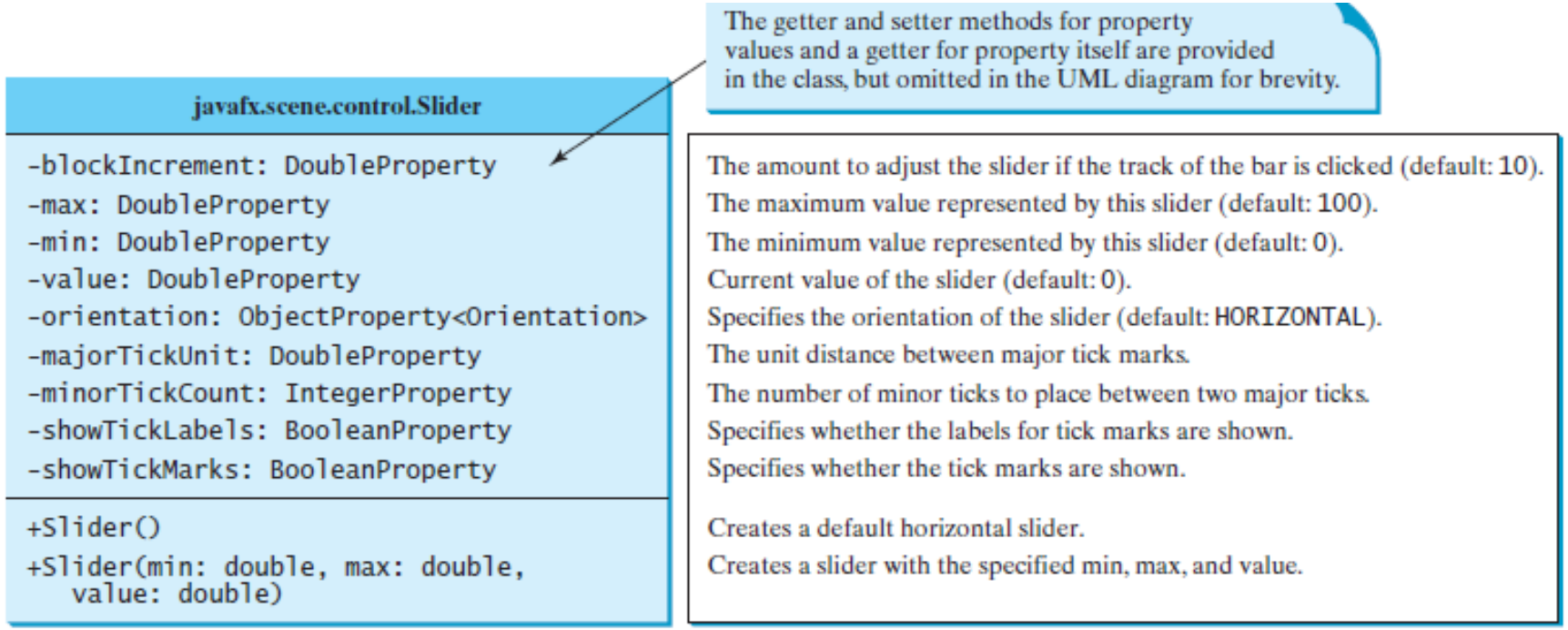


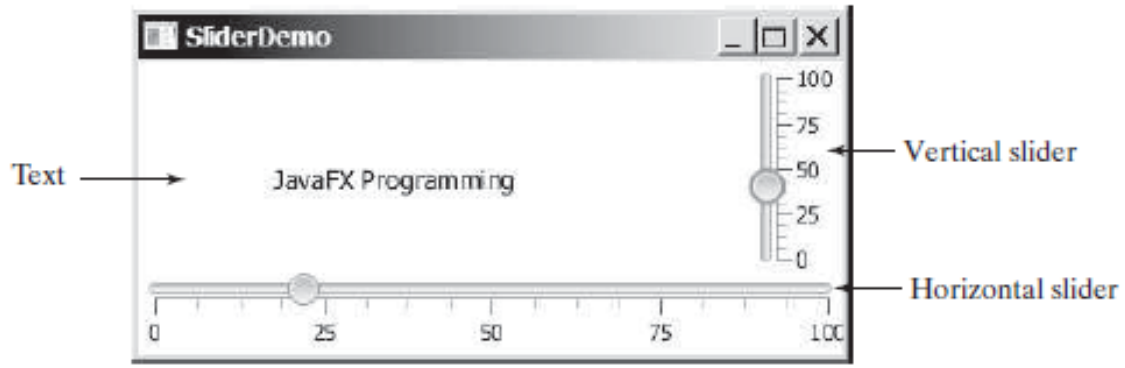
*For Example Refer Unit 8 Demo Programs Folder

Slider

- Slider is similar to ScrollBar, but Slider has more properties and can appear in many forms.
- Slider lets the user graphically select a value by sliding a knob within a bounded interval.
- The slider can show both major tick marks and minor tick marks between them.
- The number of pixels between the tick marks is specified by the majorTickUnit and minorTickUnit properties.
- Sliders can be displayed horizontally or vertically, with or without ticks, and with or without labels.
- You can add a listener to listen for the value property change in a slider in the same way as in a scroll bar.

Slider

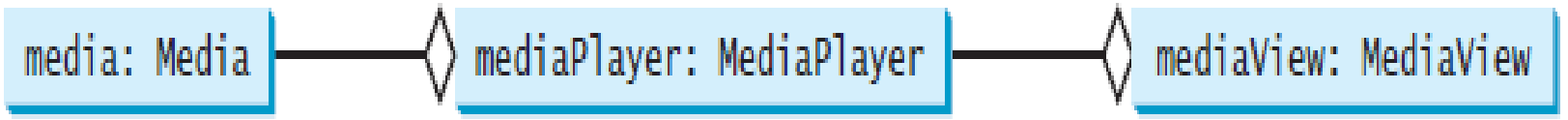




*For Example Refer Unit 8 Demo Programs Folder

Video and Audio

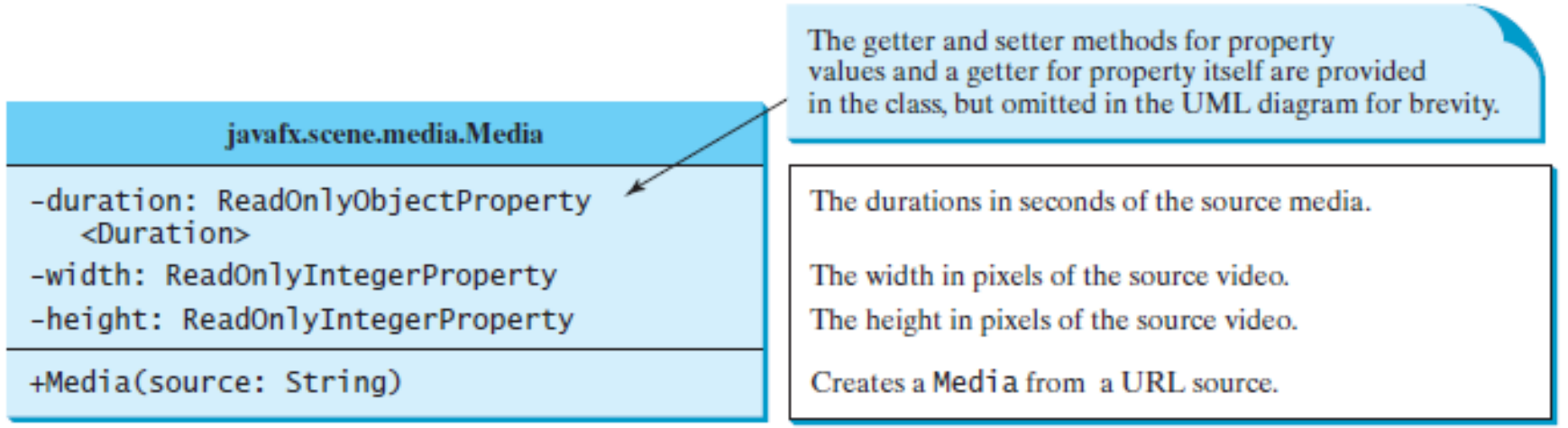
- You can use the Media class to obtain the source of the media, the MediaPlayer class to play and control the media, and the MediaView class to display the video.
- Media (video and audio) is essential in developing rich Internet applications.
- Currently, JavaFX supports MP3, AIFF, WAV, and MPEG-4 audio formats and FLV and MPEG-4 video formats.



Video and Audio

- The Media class represents a media source with properties duration, width, and height.
- You can construct a Media object from an Internet URL string.
- A Media object supports live streaming.
- A Media object can be shared by multiple media players and different views can use the same MediaPlayer object.

Media Class

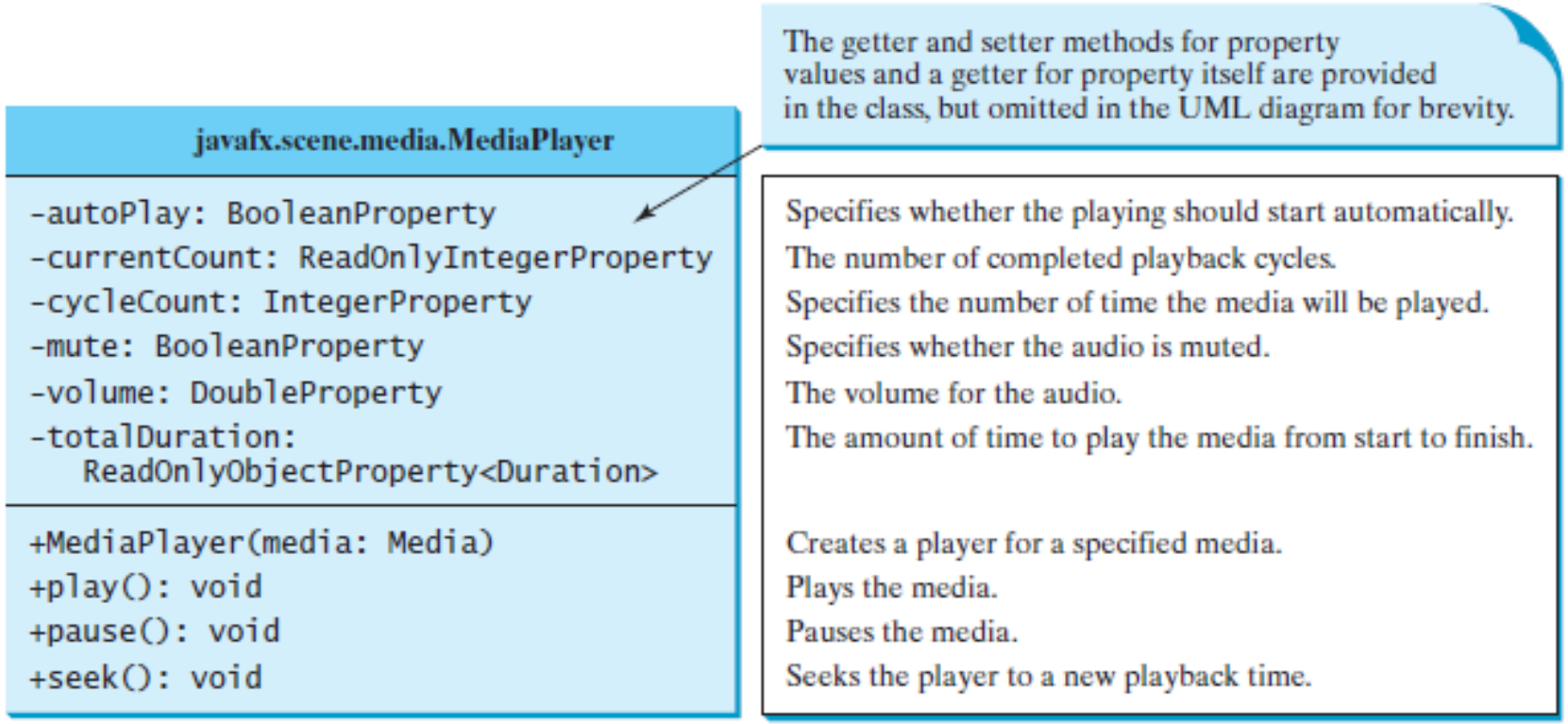


*For Example Refer Unit 8 Demo Programs Folder

MediaPlayer class

- The MediaPlayer class plays and controls the media with properties such as autoPlay, currentCount, cycleCount, mute, volume, and totalDuration.
- You can construct a MediaPlayer object from a media and use the pause() and play() method to pause and resume playing.

MediaPlayer Class

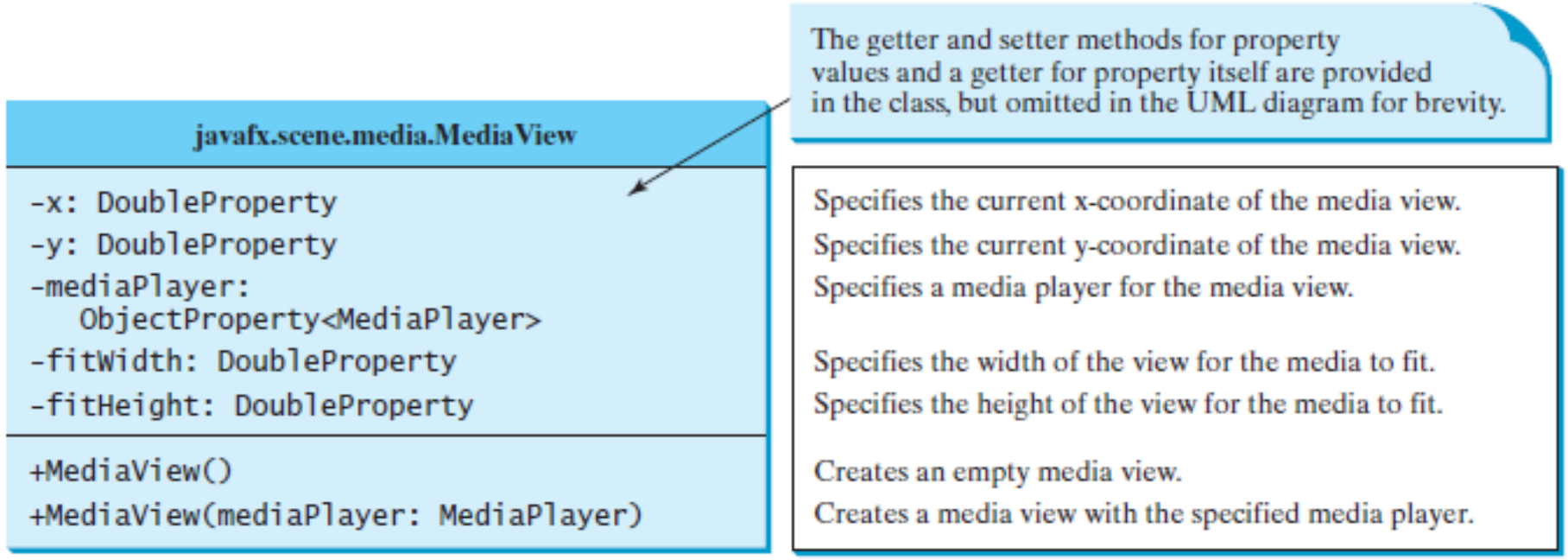


*For Example Refer Unit 8 Demo Programs Folder

MediaView class

- 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.

MediaView class



*For Example Refer Unit 8 Demo Programs Folder

Summary

- Labeled and Label
- Button
- Checkbox
- RadioButton
- Textfield
- TextArea
- Combo Box
- ListView
- Scrollbar
- Slider
- Video and Audio

Queries??

Thank You