Middlesex University London

Lecture 14-15

JavaFX

TODAY:

What is JavaFX

JavaFX Architecture

Stage

Scene

Node

Layouts - Panes and groups

setOnAction – event

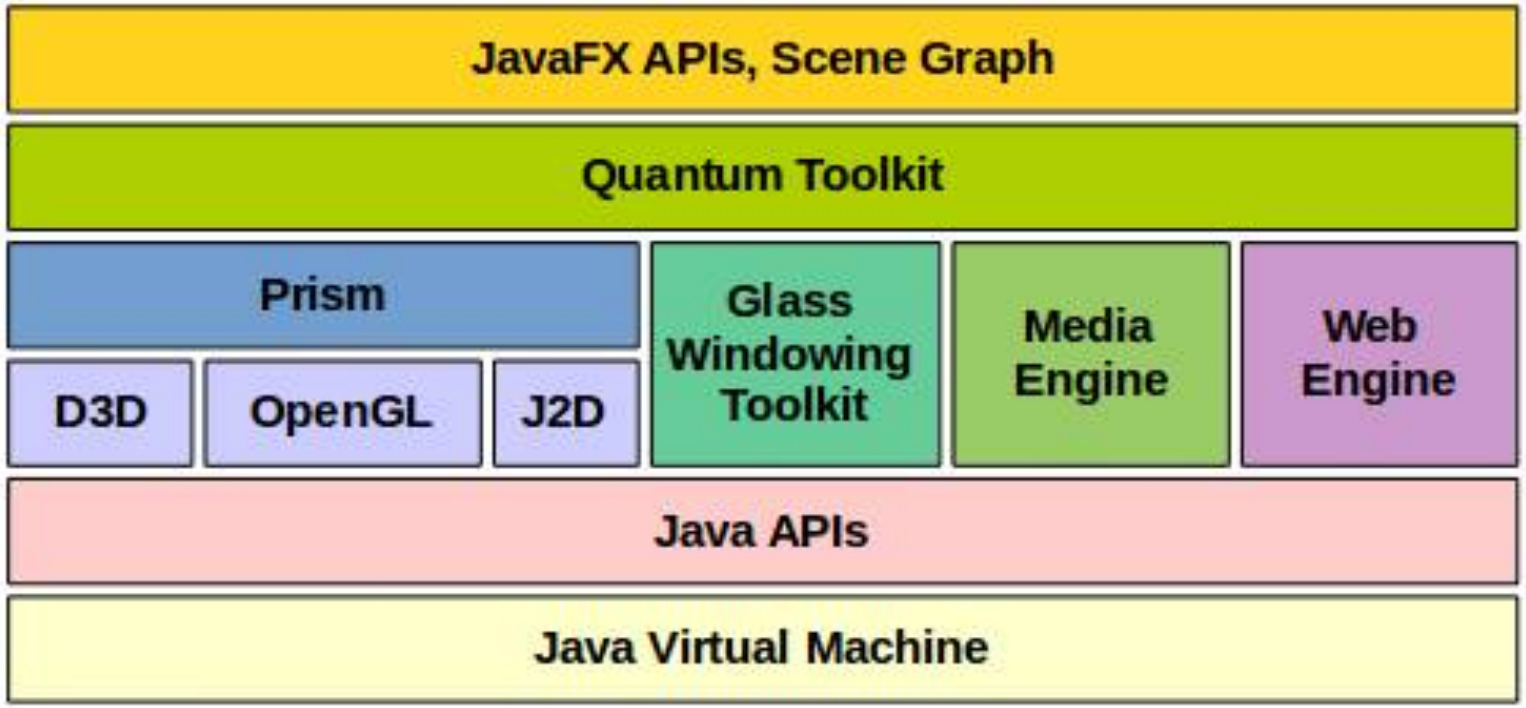
setOnKeyPressed – event

UI controls: Button, Label, TextField, CheckBox, RadioBox …etc

WHAT IS JAVAFX

- New GUI architecture
- JavaFX is a Java library used for purpose of developing graphical application
- Has a library written as a Java API
- Automation of the repaint process
- Uses the MVC
- Supports CSS and and FXML
- Can use CSS to add style to the UI

JAVAFX ARCHITECTURE



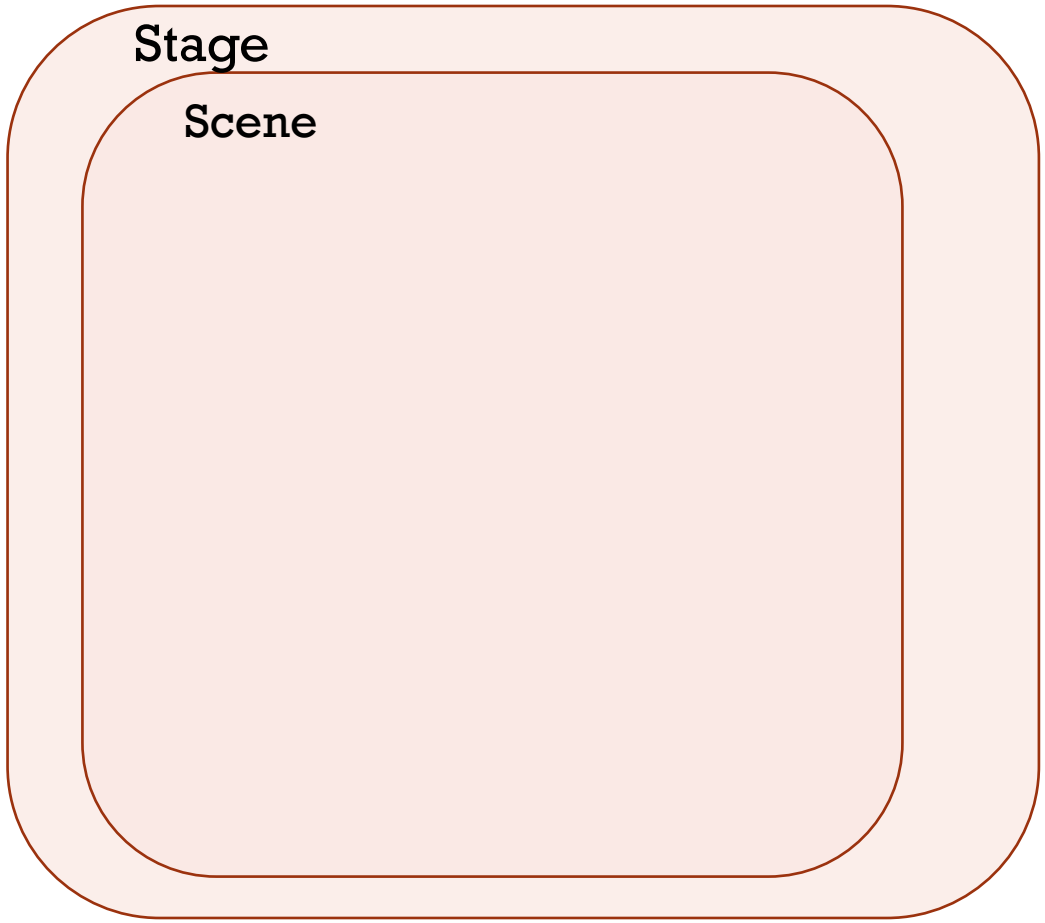
JAVAFX APPLICATION STRUCTURE - STAGE

Stage

A Stage in JavaFX is a top-level container that hosts a Scene, which consists of visual elements. It contains all the objects of a JavaFX application **javafx.stage**.
The created stage object is passed as an argument to the **start()**


```
3
4   import javafx.application.Application;
5   import javafx.stage.Stage;
6
7   public class JavaFXSample extends Application{
8
9       @Override
10      public void start(Stage primaryStage) {
11
12          primaryStage.setTitle("JavaFX");
13          primaryStage.show();
14
15      }
16
17      public static void main(String[] args) {
18          launch(args);
19      }
20
21
22  }
```

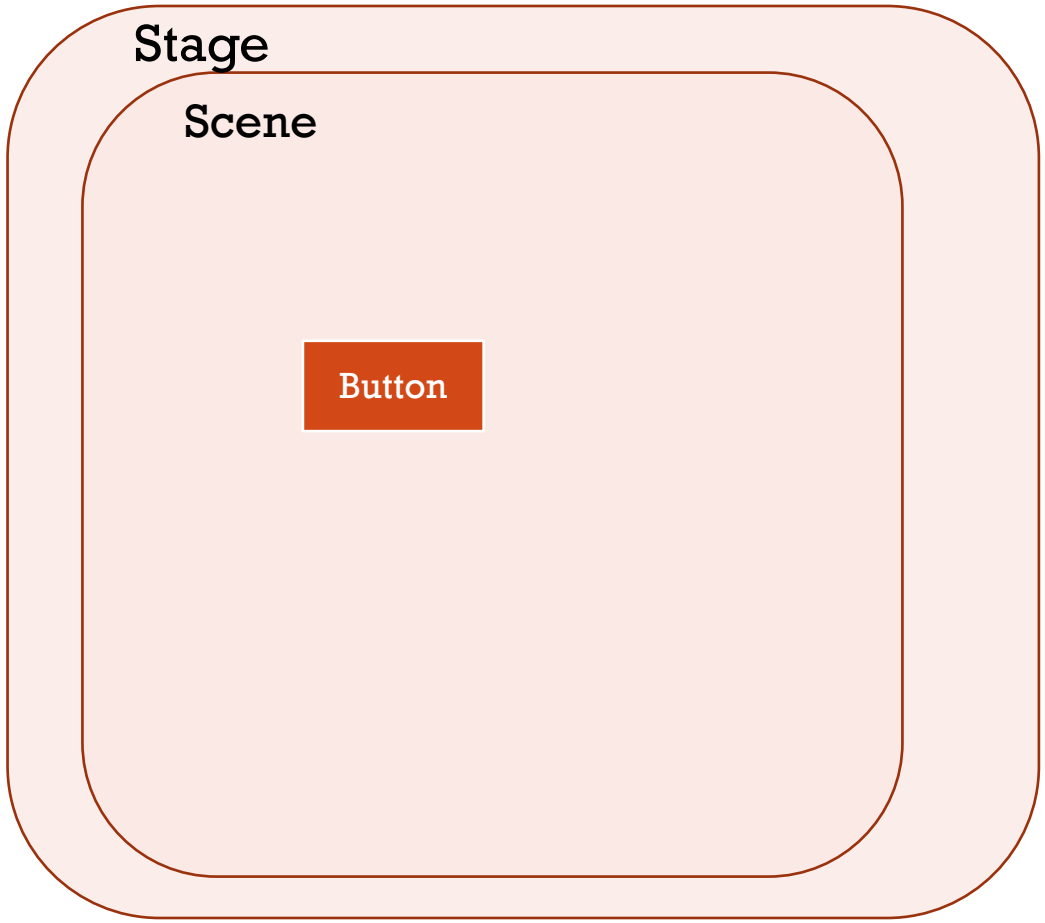
JAVAFX APPLICATION STRUCTURE - SCENE

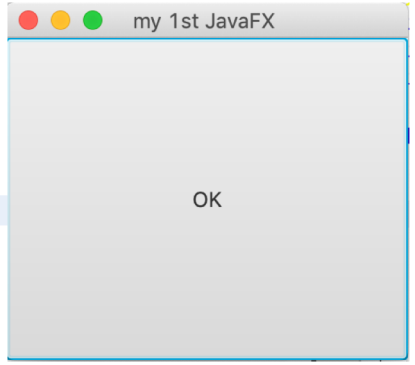


Scene

represents the physical contents of a JavaFX application
javafx.scene represents the scene object.
You can create a scene by instantiating the Scene Class.
You can opt for the size of the scene by passing its dimensions (height and width) along with the **root node** to its constructor.

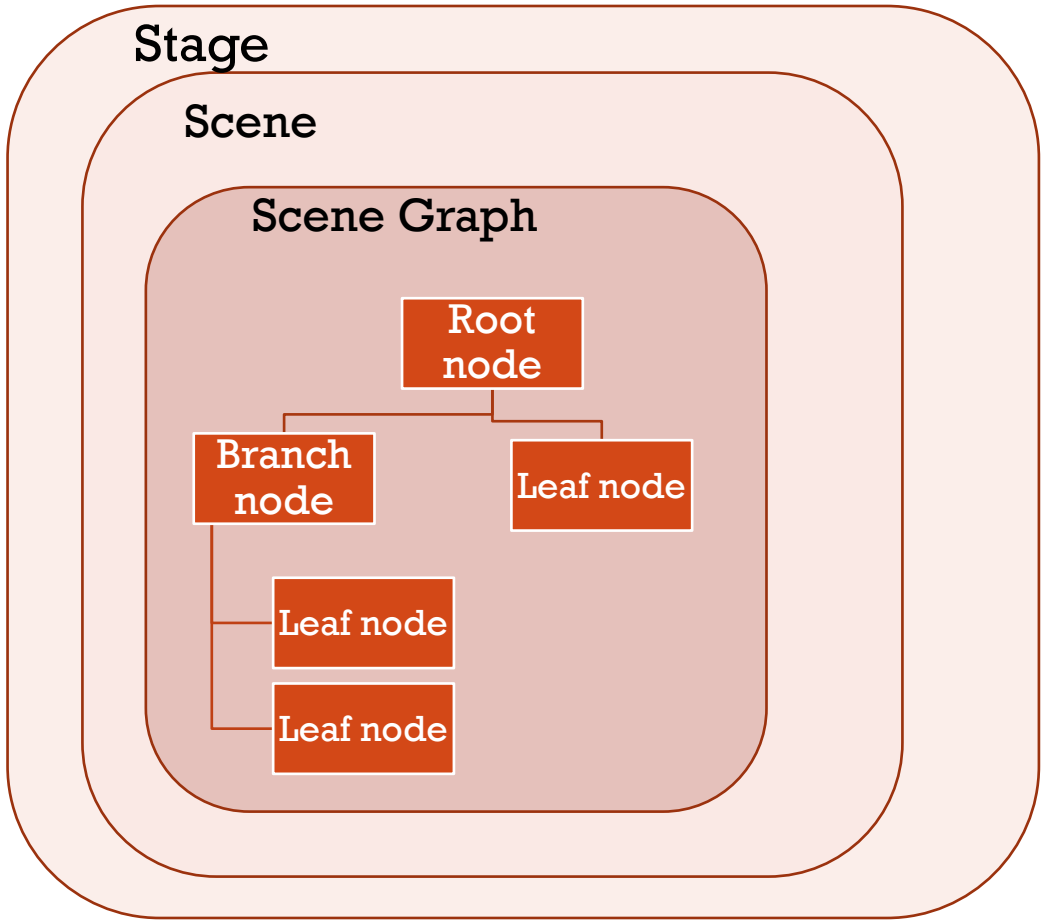
NODE






```
3
4
5   import javafx.application.Application;
7   import javafx.scene.Group;
8   import javafx.scene.Scene;
9   import javafx.scene.control.Button;
10  import javafx.stage.Stage;
11
12  public class JavaFXSample extends Application{
13
15      @Override
16      public void start(Stage primaryStage) {
17
18          Button btOK = new Button ("OK");
19          Scene scene = new Scene (btOK,250,200);
20
21          primaryStage.setTitle("my 1st JavaFX ");
22          primaryStage.setScene(scene);
23          primaryStage.show();
24
25      }
26
27      public static void main(String[] args) {
28          launch(args);
29      }
30
31
32  }
```

JAVAFX APPLICATION STRUCTURE - NODE



Scene Graph and Nodes

A **scene graph** is a tree-like data structure (hierarchical) representing the contents of a scene.

In contrast, a **node** is a visual/graphical object of a scene graph.

A node may include –

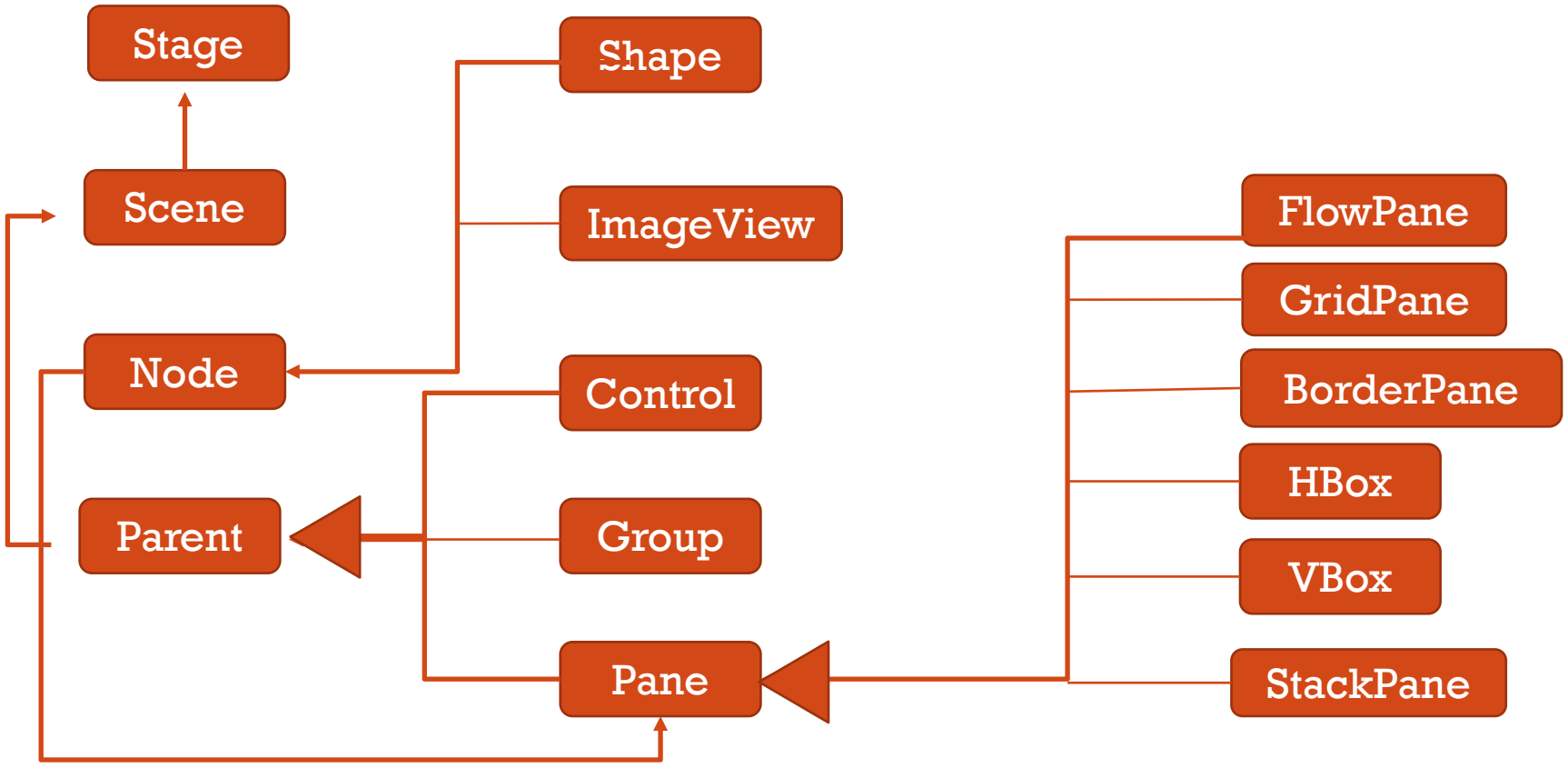
Geometrical (Graphical) objects (2D and 3D) such as – Circle, Rectangle, Polygon, etc.

UI Controls such as – Button, Checkbox, Choice Box, Text Area, etc.

Containers (Layout Panes) such as Border Pane, Grid Pane, Flow Pane, etc.

Media elements such as Audio, Video and Image Objects.

Panes and groups are used to hold nodes
nodes can be shapes, images views, UI controls, groups and panes



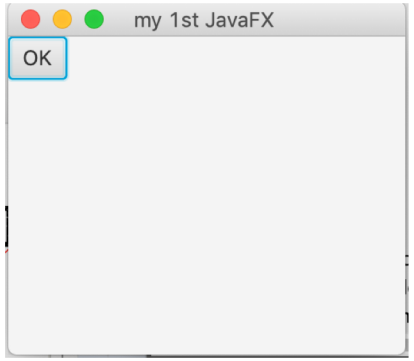
LAYOUT PANES AND GROUPS

- Allowing automatically laying out the nodes in desired location.
- **Group** class is often used to group nodes and perform transformation and scale as a group
- Panes and UI control objects are resizable. Group, shape and text are not.

Class:	Description:
Pane	Base class for layout panes. It contains the getChildren() method for returning a list of nodes in the pane
StackPane	Place the nodes on top of each other in the center of the pane
FlowPane	Place the nodes row-by-row horizontally or column-by-column vertically
GridPane	Place the nodes in the cell in two- dimensional grid
BorderPane	Place the nodes in the TOP, RIGHT, BOTTOM, LEFT and CENTER regions
HBox	Place the nodes in a single row
VBox	Place the nodes in a single column

EXAMPLE OF FLOWPANE:

```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class JavaFXSample extends Application{

    @Override
    public void start(Stage primaryStage) {
        Button btOK = new Button ("OK");

        FlowPane root = new FlowPane();
        Scene scene = new Scene (root,250,200);


        root.getChildren().add(btOK);


        primaryStage.setTitle("my 1st JavaFX ");
        primaryStage.setScene(scene);
        primaryStage.show();

    }

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


}
```

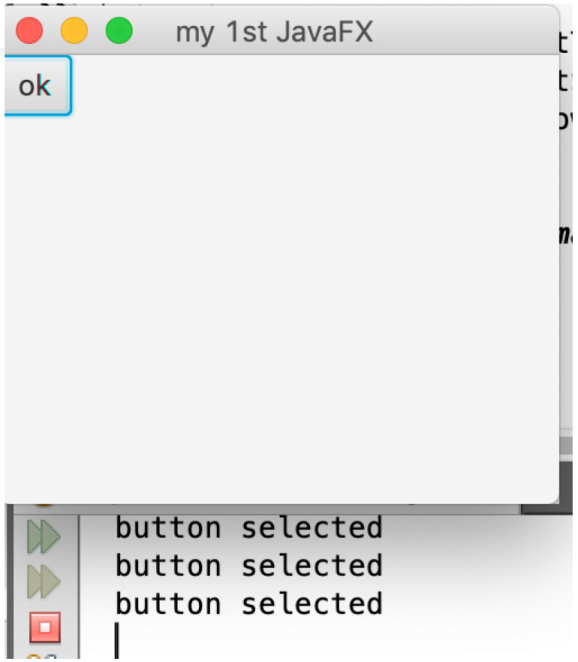


```
x.application.Application;
x.event.ActionEvent;
x.event.EventHandler;
x.scene.Scene;
x.scene.control.Button;
x.scene.control.Label;
x.scene.layout.FlowPane;
x.stage.Stage;

 JavaFXSample extends Application{

e
oid start(Stage primaryStage) {
on btOK = new Button ("ok");
Pane root = new FlowPane();
e scene = new Scene (root,250,200);
.getChildren().add(btOK);

ction event
tHandler<ActionEvent> event1 = new EventHandler<ActionEvent>() {
ublic void handle(ActionEvent e)
{
    System.out.println("button selected    ");
}


.setOnAction(event1);

aryStage.setTitle("my 1st JavaFX ");
aryStage.setScene(scene);
aryStage.show();


tatic void main(String[] args) {
ch(args);
```

EventHandlers are used for managing user's actions

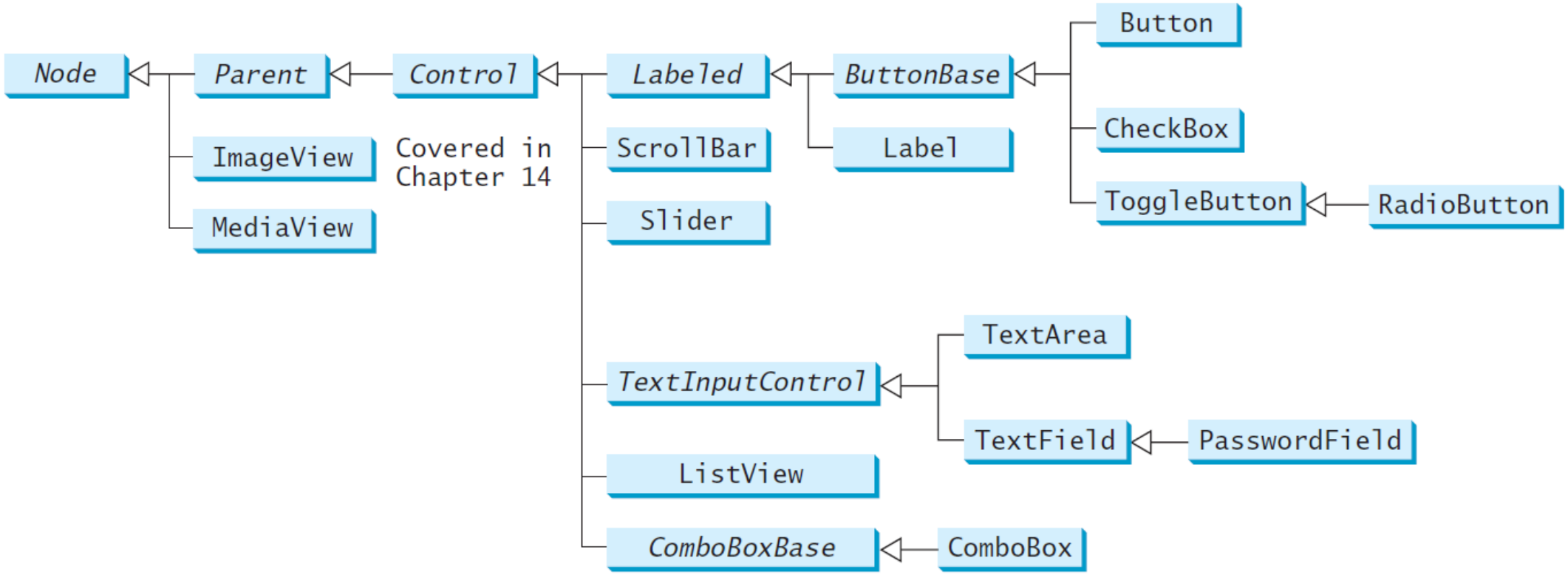
They are responsible for performing actions when user is interacting with an actionable nodes such as buttons and text fields

Example:
setOnAction for a button node



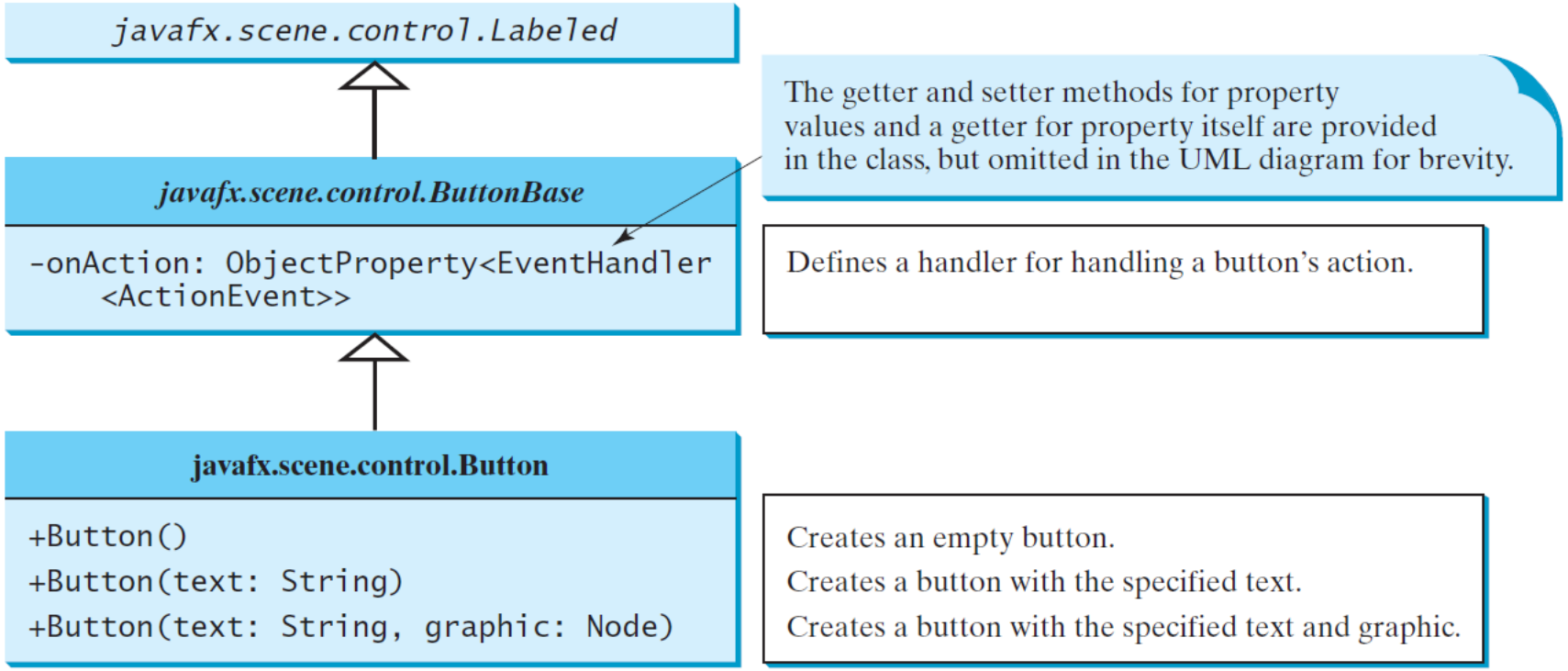
MORE ABOUT EVENTS LATER..

FREQUENTLY USED UI CONTROLS



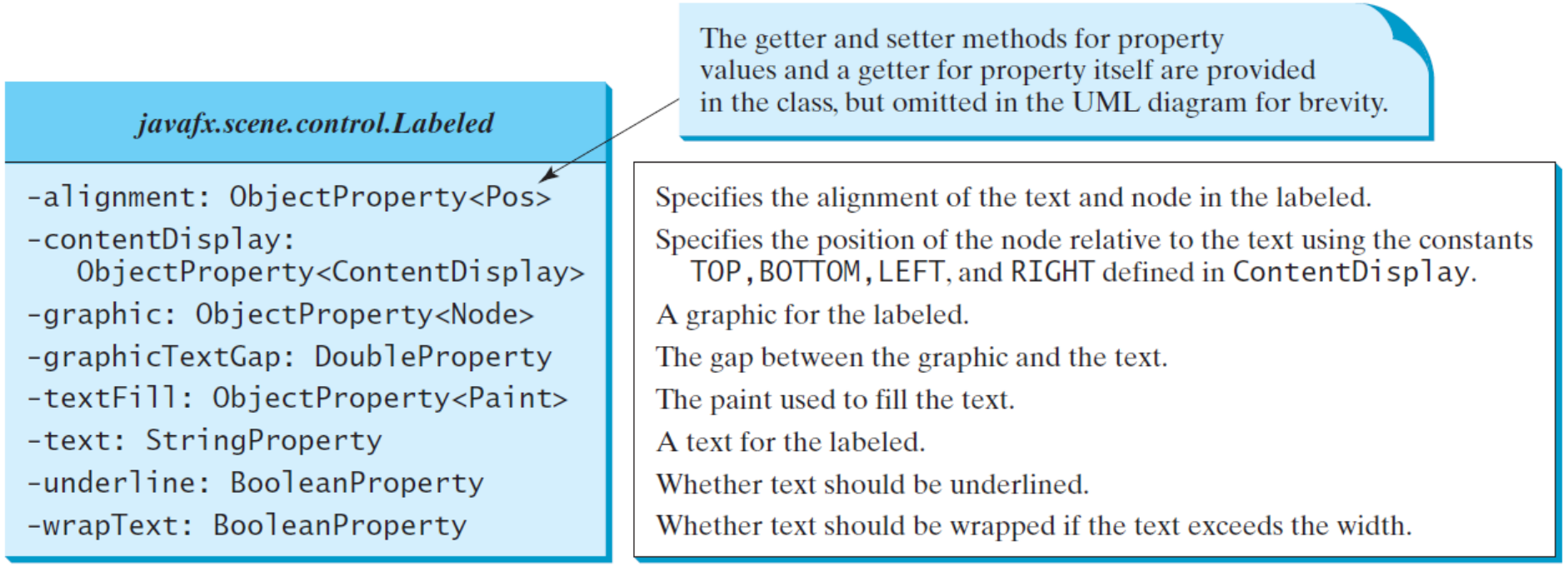
BUTTONBASE AND BUTTON

- 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.

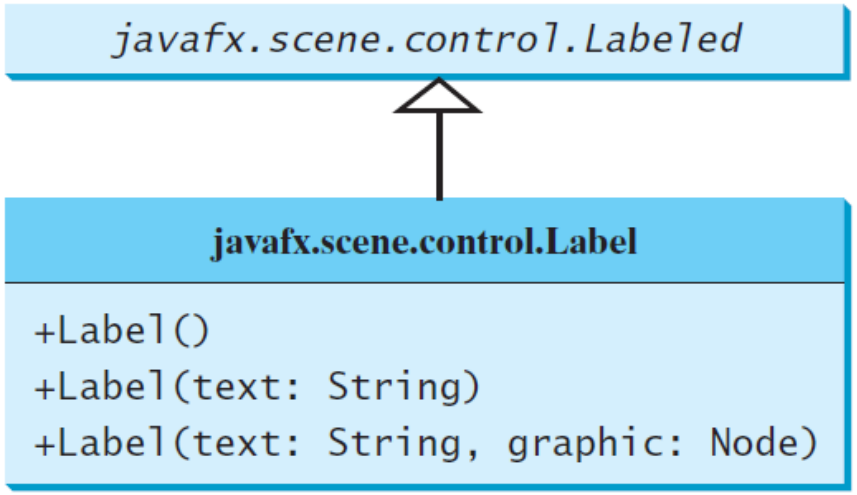


LABELED

- 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.



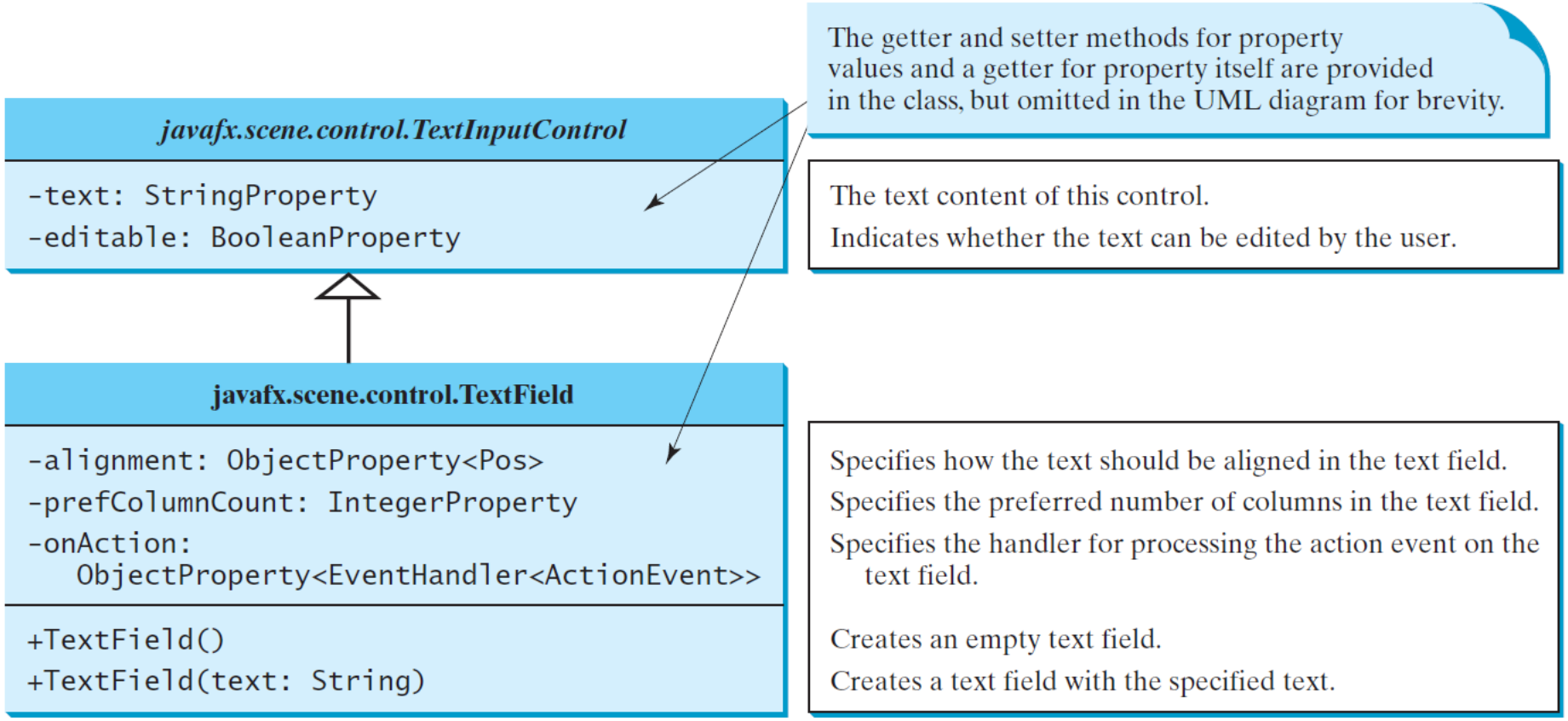
LABEL



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

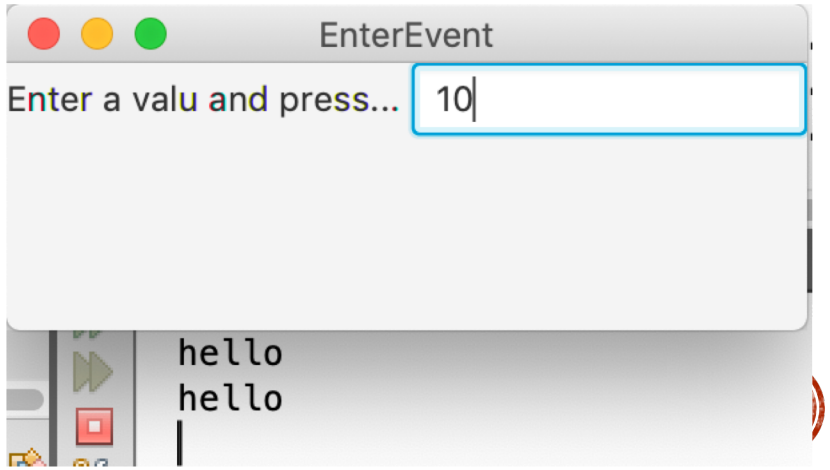
TEXTFIELD

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



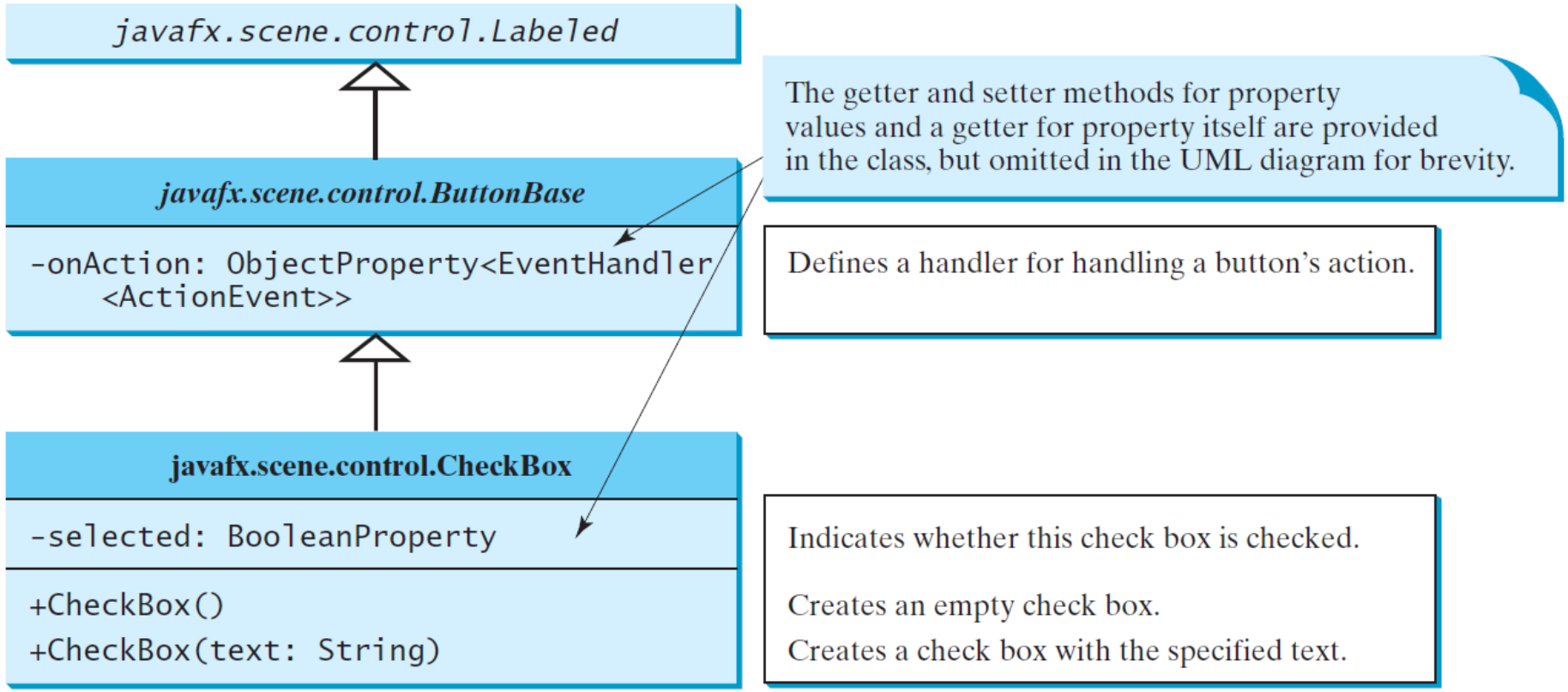
```
21  */
22  ublic class EnterEvent extends Application{
23
24      @Override
25      public void start(Stage primaryStage){
26              GridPane pane = new GridPane();
27
28              Label l = new Label();
29              l.setText("Enter a valu and press enter");
30              TextField tf = new TextField();
31
32
33              pane.add(l, 0, 0);
34              pane.add(tf, 1, 0);
35
36              EventHandler<KeyEvent> event = new EventHandler<KeyEvent>(){
37
38                  public void handle(KeyEvent event) {
39                      if (event.getCode()== KeyCode.ENTER){
40                          System.out.println("hello");
41                      }
42                  }
43
44              };
45
46              tf.setOnKeyPressed(event);
47
48              Scene scene = new Scene(pane, 300, 100);
49              primaryStage.setTitle("EnterEvent"); // Set the stage title
50              primaryStage.setScene(scene); // Place the scene in the stage
51              primaryStage.show(); // Display the stage
52      }
53
54
55      public static void main(String[] args) {
56          launch(args);
57      }
58
```

Example:
setOnKEyPressed for a textField node



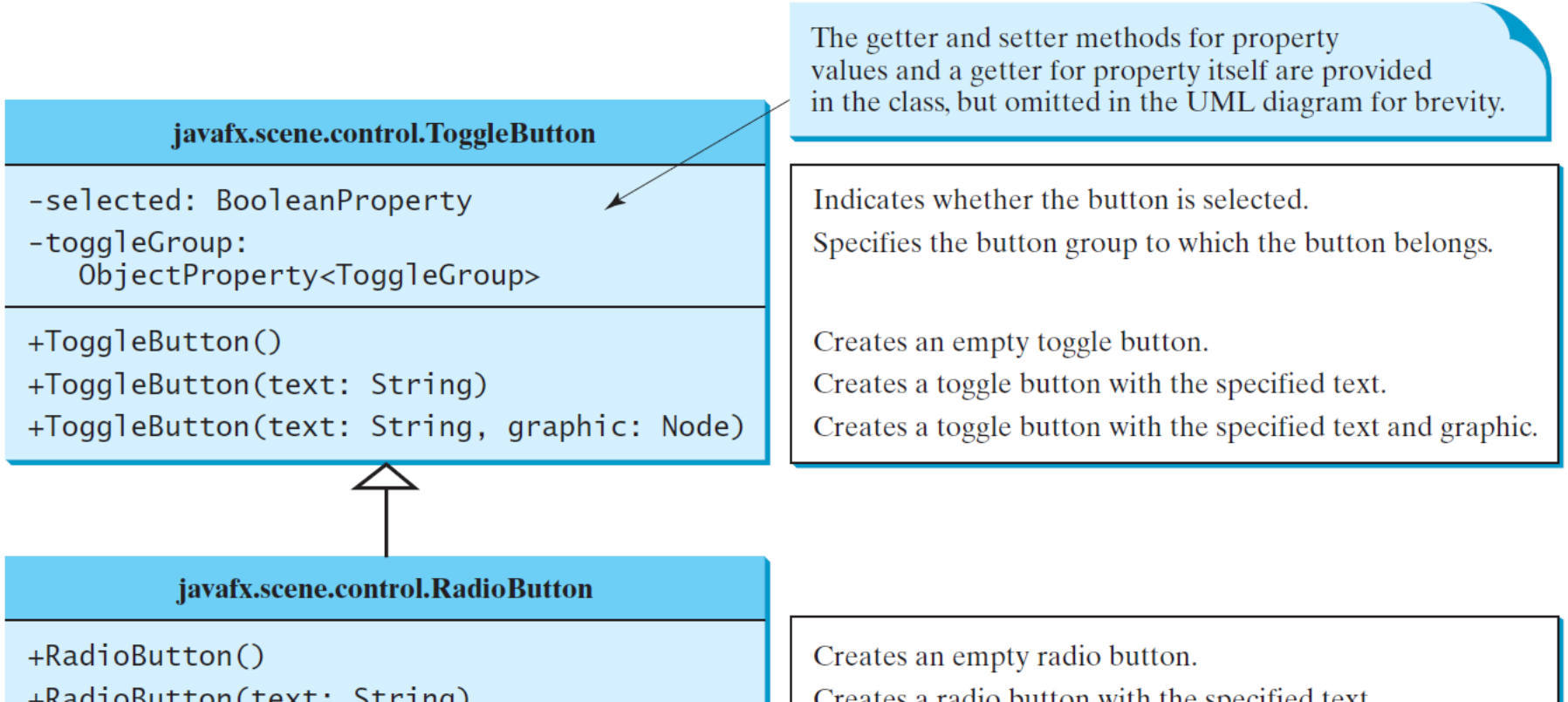
CHECKBOX

- 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**.

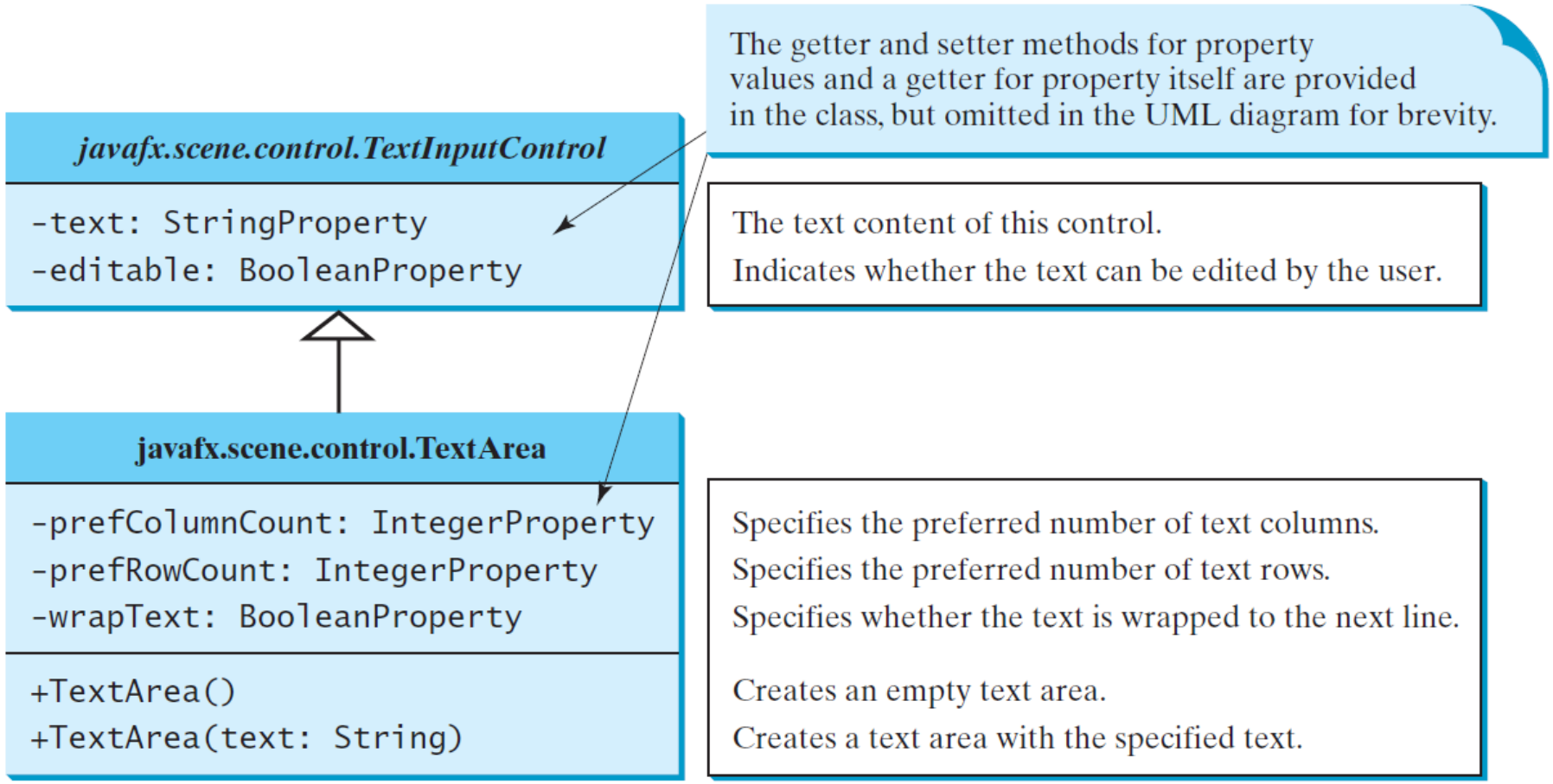


RADIOBUTTON

- 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).

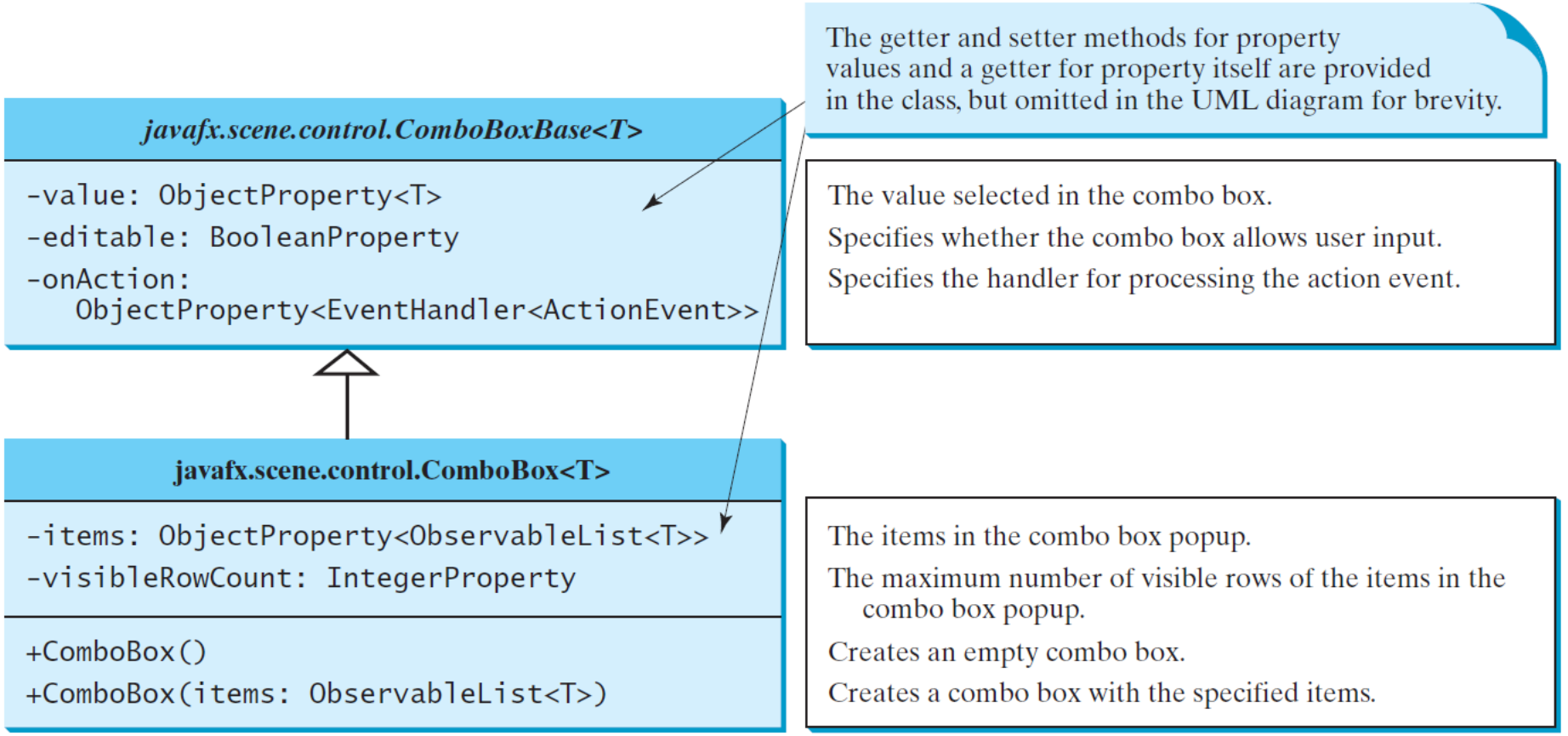


TEXTAREA


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.

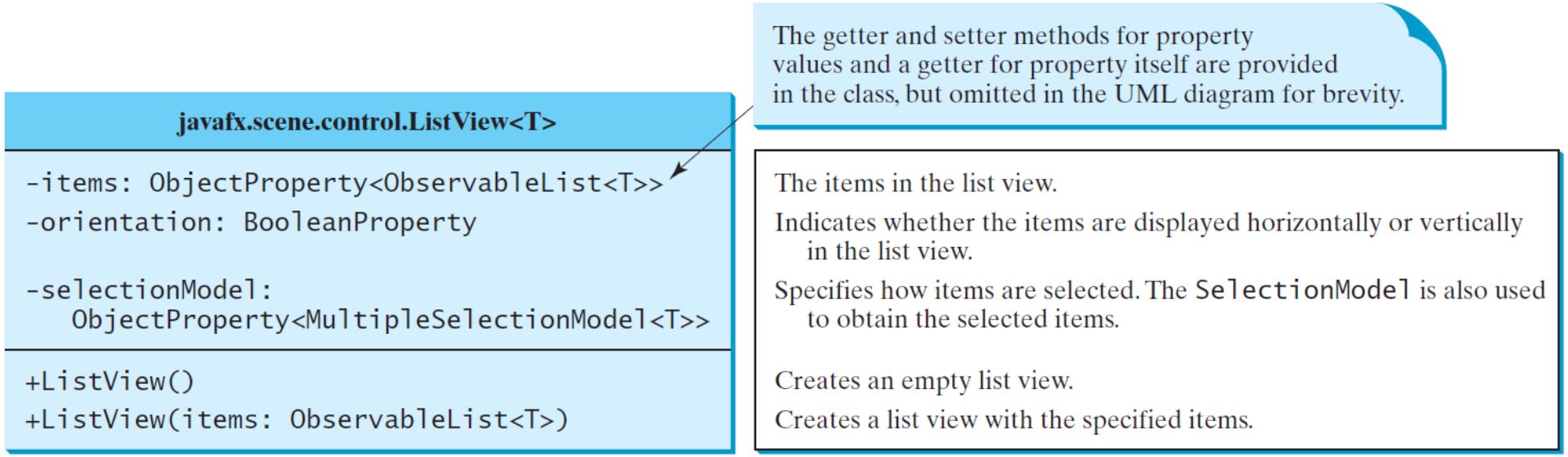
COMBOBOX

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



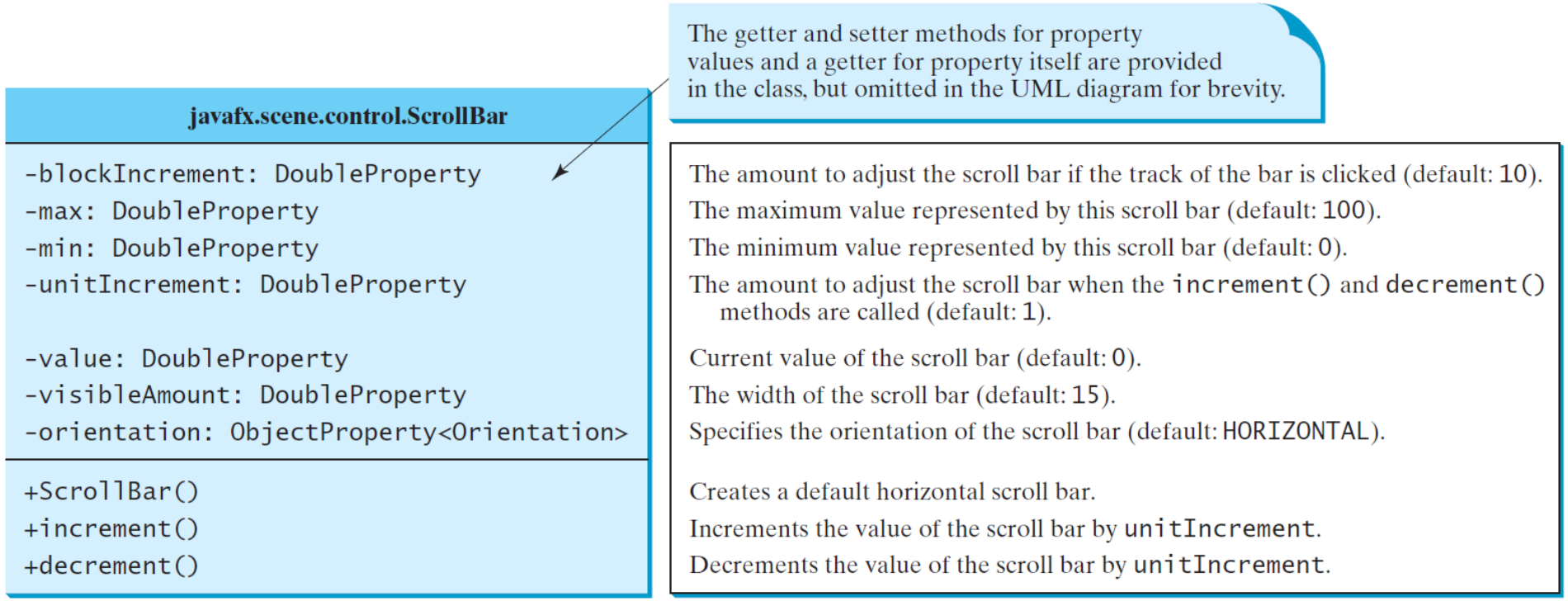
LIST VIEW

- 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.

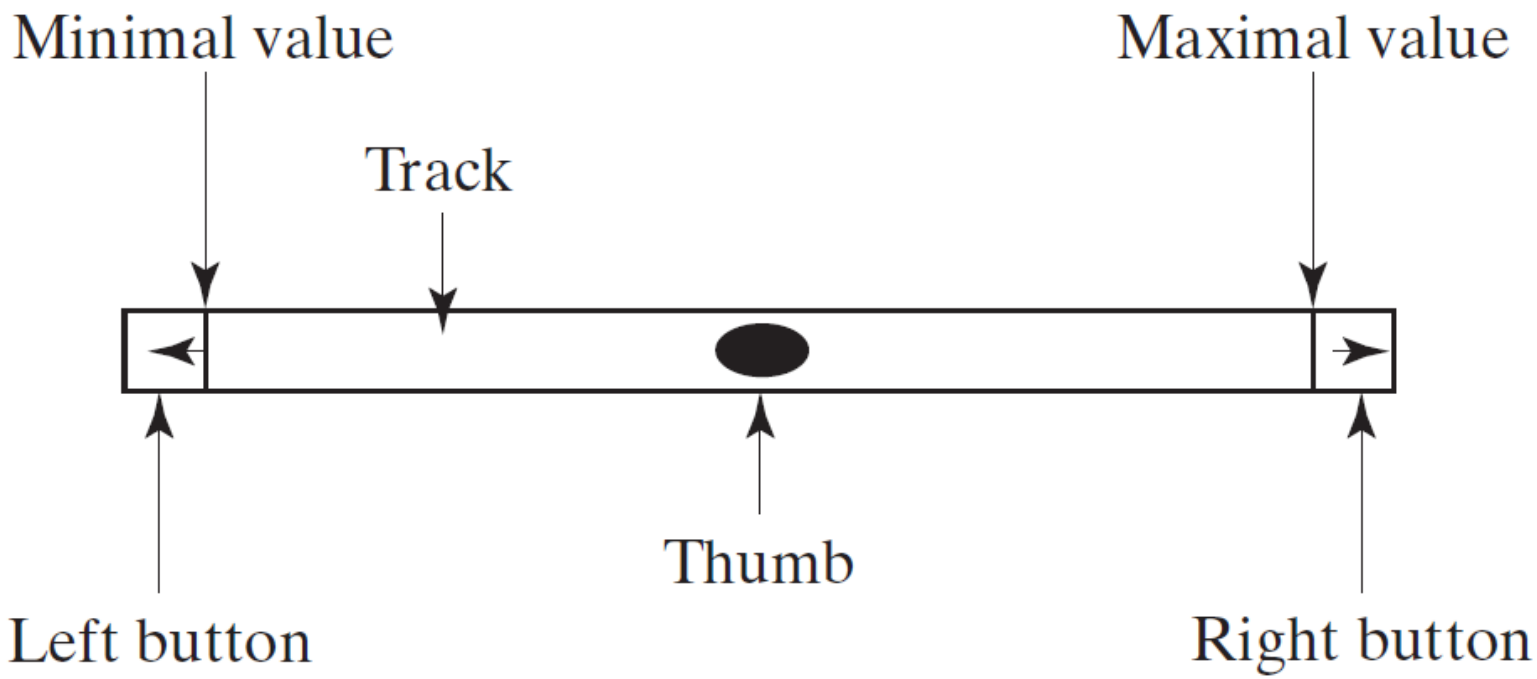


SCROLL BAR

- 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*.

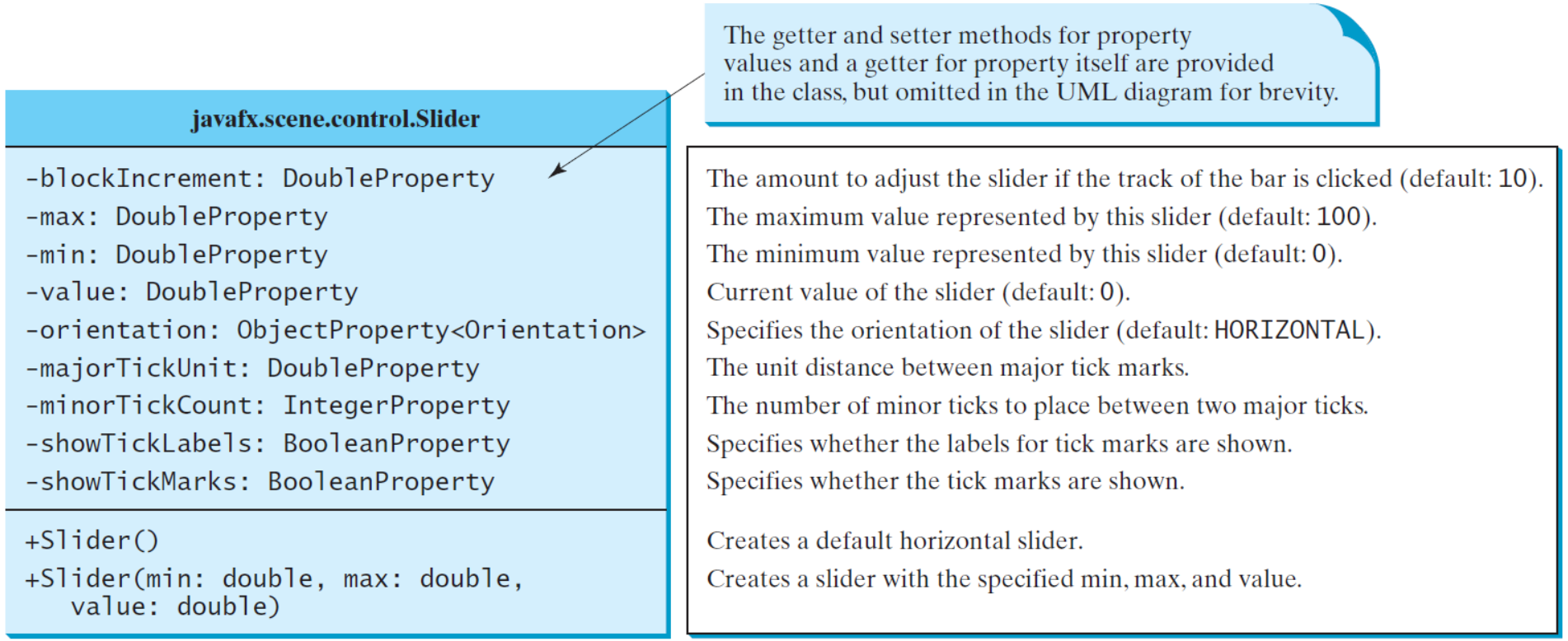


SCROLL BAR PROPERTIES



SLIDER

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



LINKS AND ADDITIONAL RESOURCES:

- http://tutorials.jenkov.com/javafx/index.html
- https://www.tutorialspoint.com/javafx/javafx_quick_guide.htm
- http://tutorials.jenkov.com/javafx/overview.html
- https://www.tutorialspoint.com/javafx/javafx_layout_panes.htm
- https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm
- https://www.dummies.com/programming/java/need-know-scene-class-javafx/
- https://examples.javacodegeeks.com/desktop-java/javafx/javafx-stage-example/
- http://tutorials.jenkov.com/javafx/vbox.html
- https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/GridPane.html
- https://www.tutorialspoint.com/javafx/javafx_ui_controls.htm
- https://www.tutorialspoint.com/javafx/javafx_overview.htm
- https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm

MASTERING YOUR SKILLS

JavaFX it is a big area of study on them on. Please create a task and learn based on your needs. The foundations are in this lecture. Enjoy!