JavaFX kontroller

- Mouse Events (fra kap.15)
- Key Events (fra kap.15)
- Kontroller:
  - ✓ Label
  - ✓ Button
  - ✓ CheckBox
  - ✓ RadioButton
  - ✓ TextField
  - ✓ TextArea
  - ✓ ComboBox
  - ✓ ListView
  - ✓ ScrollBar
  - ✓ Slider
- Case: TicTacToe

Mouse Events

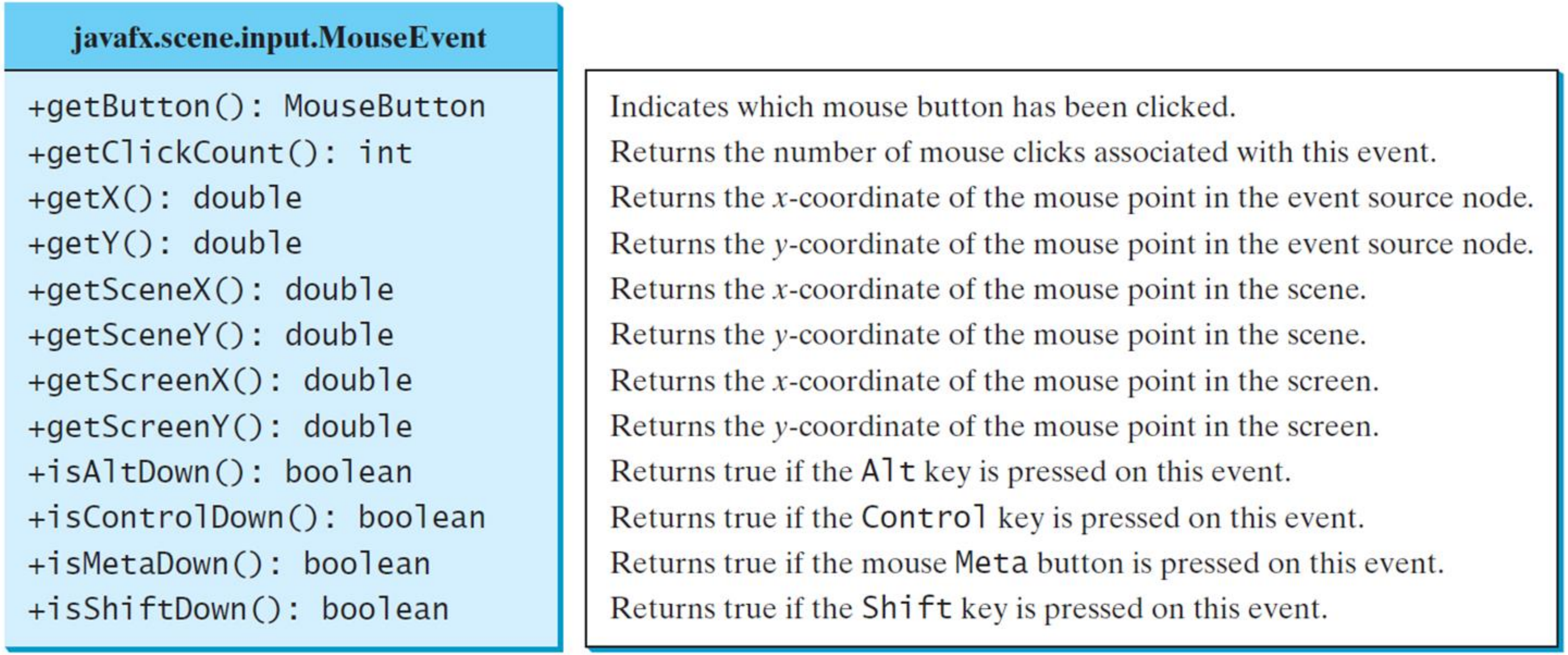
Muse-event genereres når museknapp –

- trykkes
- slippes
- klikkes (trykk og slipp)

-eller når hele musa

- flyttes
- «dragges» - flyttes med knapp trykket

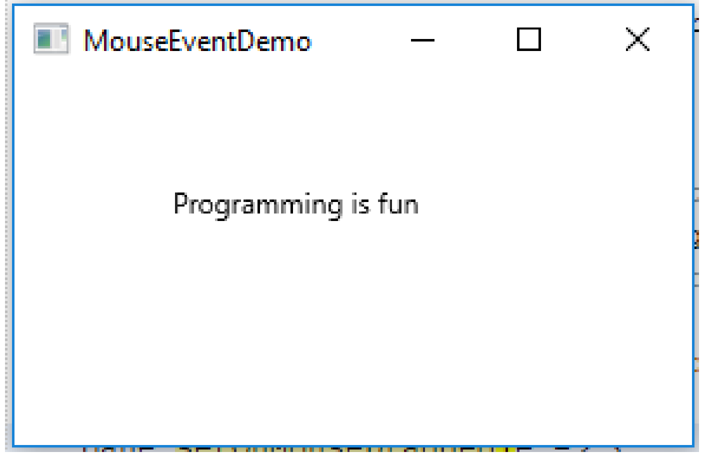
class MouseEvent



javafx.scene.input.MouseEvent	
+getButton(): MouseButton	Indicates which mouse button has been clicked.
+getClickCount(): int	Returns the number of mouse clicks associated with this event.
+getX(): double	Returns the *x*-coordinate of the mouse point in the event source node.
+getY(): double	Returns the *y*-coordinate of the mouse point in the event source node.
+getSceneX(): double	Returns the *x*-coordinate of the mouse point in the scene.
+getSceneY(): double	Returns the *y*-coordinate of the mouse point in the scene.
+getScreenX(): double	Returns the *x*-coordinate of the mouse point in the screen.
+getScreenY(): double	Returns the *y*-coordinate of the mouse point in the screen.
+isAltDown(): boolean	Returns true if the `Alt` key is pressed on this event.
+isControlDown(): boolean	Returns true if the `Control` key is pressed on this event.
+isMetaDown(): boolean	Returns true if the mouse `Meta` button is pressed on this event.
+isShiftDown(): boolean	Returns true if the `Shift` key is pressed on this event.

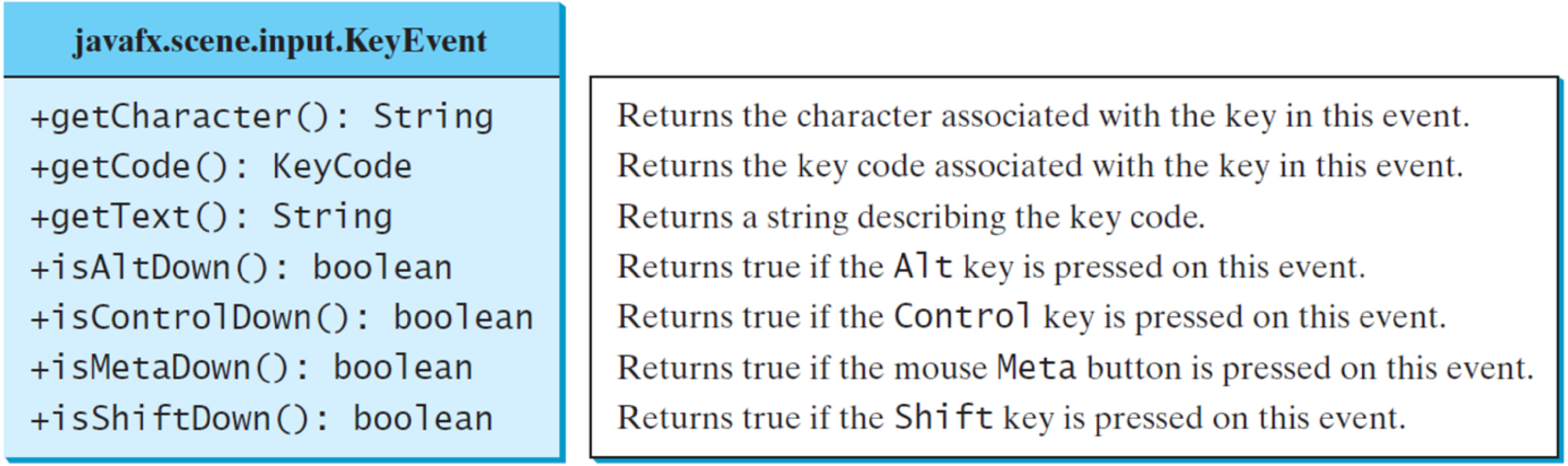
MouseEventDemo

```
public class MouseEventDemo extends Application {
  @Override
  public void start(Stage primaryStage) {
    Pane pane = new Pane();
    Text text = new Text(20, 20, "Programming is fun");
    pane.getChildren().addAll(text); // eller .add()
    text.setOnMouseDragged(e -> {
      text.setX(e.getX());
      text.setY(e.getY());
    });

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 300, 100);
    primaryStage.setTitle("MouseEventDemo");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
  // main-metode som vanlig
}
```


MouseEventDemo

Programming is fun

class KeyEvent



javafx.scene.input.KeyEvent

`+getCharacter(): String`	Returns the character associated with the key in this event.
`+getCode(): KeyCode`	Returns the key code associated with the key in this event.
`+getText(): String`	Returns a string describing the key code.
`+isAltDown(): boolean`	Returns true if the `Alt` key is pressed on this event.
`+isControlDown(): boolean`	Returns true if the `Control` key is pressed on this event.
`+isMetaDown(): boolean`	Returns true if the mouse `Meta` button is pressed on this event.
`+isShiftDown(): boolean`	Returns true if the `Shift` key is pressed on this event.

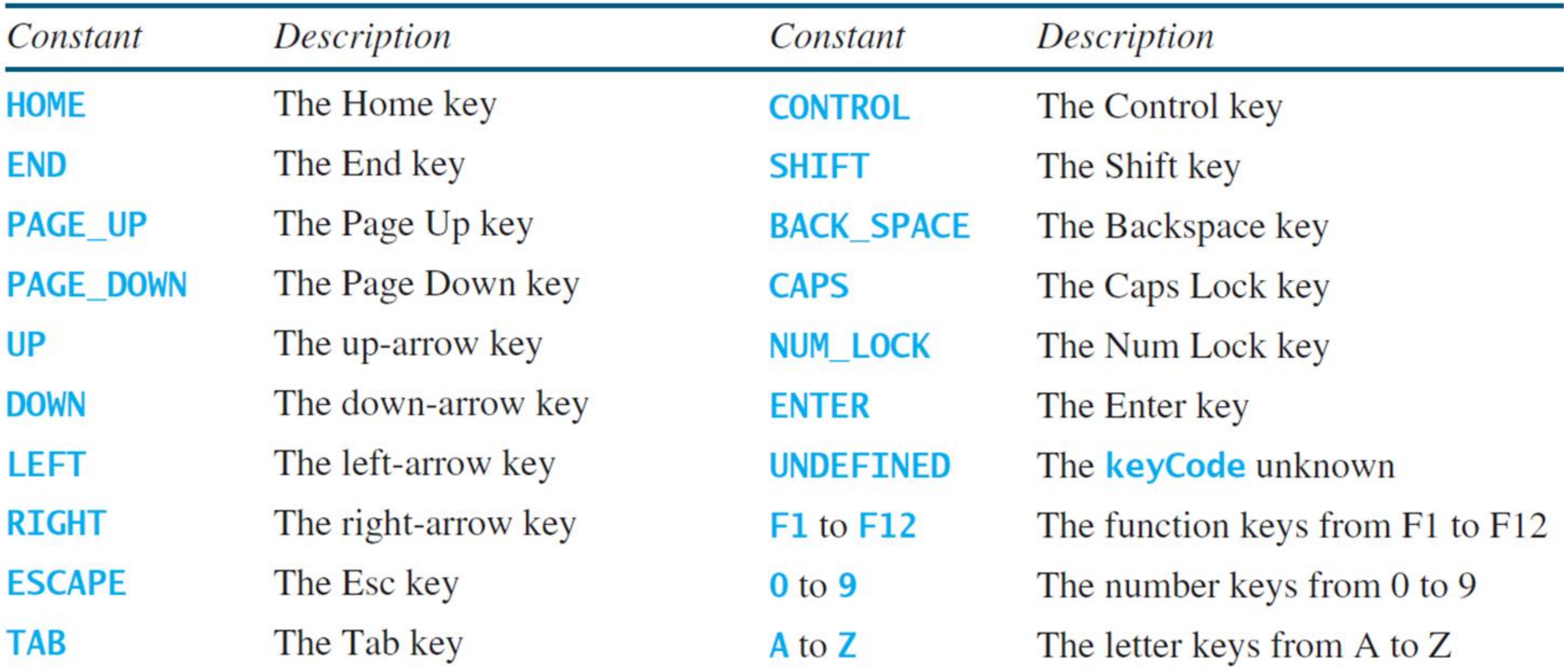
KeyEventDemo

```java
public class KeyEventDemo extends Application {
  @Override
  public void start(Stage primaryStage) {
    Pane pane = new Pane();
    Text text = new Text(20, 20, "A");

    pane.getChildren().add(text);
    text.setOnKeyPressed(e -> {
      switch (e.getCode()) {
        case DOWN: text.setY(text.getY() + 10); break;
        case UP:  text.setY(text.getY() - 10); break;
        case LEFT: text.setX(text.getX() - 10); break;
        case RIGHT: text.setX(text.getX() + 10); break;
        default:
          if (e.getText().length() > 0)
            text.setText(e.getText());
      }
    });
```

Disse og mange flere er definert i KeyCode

```java
    Scene scene = new Scene(pane);
    primaryStage.setTitle("KeyEventDemo");
    primaryStage.setScene(scene);
    primaryStage.show();

    text.requestFocus(); // to receive key input
  }

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

Noen KeyCode konstanter



Constant	*Description*	*Constant*	*Description*
HOME	The Home key	CONTROL	The Control key
END	The End key	SHIFT	The Shift key
PAGE_UP	The Page Up key	BACK_SPACE	The Backspace key
PAGE_DOWN	The Page Down key	CAPS	The Caps Lock key
UP	The up-arrow key	NUM_LOCK	The Num Lock key
DOWN	The down-arrow key	ENTER	The Enter key
LEFT	The left-arrow key	UNDEFINED	The keyCode unknown
RIGHT	The right-arrow key	F1 to F12	The function keys from F1 to F12
ESCAPE	The Esc key	0 to 9	The number keys from 0 to 9
TAB	The Tab key	A to Z	The letter keys from A to Z

class ControlCircleWithMouseAndKey

```
public class ControlCircleWithMouseAndKey extends
Application {
  private CirclePane circlePane = new CirclePane();

  @Override
  public void start(Stage primaryStage) {
    HBox hBox = new HBox();
    hBox.setSpacing(10);
    hBox.setAlignment(Pos.CENTER);
    Button btEnlarge = new Button("Enlarge");
    Button btShrink = new Button("Shrink");
    hBox.getChildren().add(btEnlarge);
    hBox.getChildren().add(btShrink);

    // Create and register the handler
    btEnlarge.setOnAction(e -> circlePane.enlarge());
    btShrink.setOnAction(e -> circlePane.shrink());

    BorderPane borderPane = new BorderPane();
    borderPane.setCenter(circlePane);
    borderPane.setBottom(hBox);
    BorderPane.setAlignment(hBox, Pos.CENTER);

    Scene scene = new Scene(borderPane, 200, 150);
    primaryStage.setTitle("ControlCircle");
    primaryStage.setScene(scene);
    primaryStage.show();

    circlePane.setOnMouseClicked(e -> {
      if (e.getButton() == MouseButton.PRIMARY) {
        circlePane.enlarge();
      }
      else if (e.getButton() == MouseButton.SECONDARY) {
        circlePane.shrink();
      }
    });

    scene.setOnKeyPressed(e -> {
      if (e.getCode() == KeyCode.UP) {
        circlePane.enlarge();
      }
      else if (e.getCode() == KeyCode.DOWN) {
        circlePane.shrink();
      }
    });
   }
```

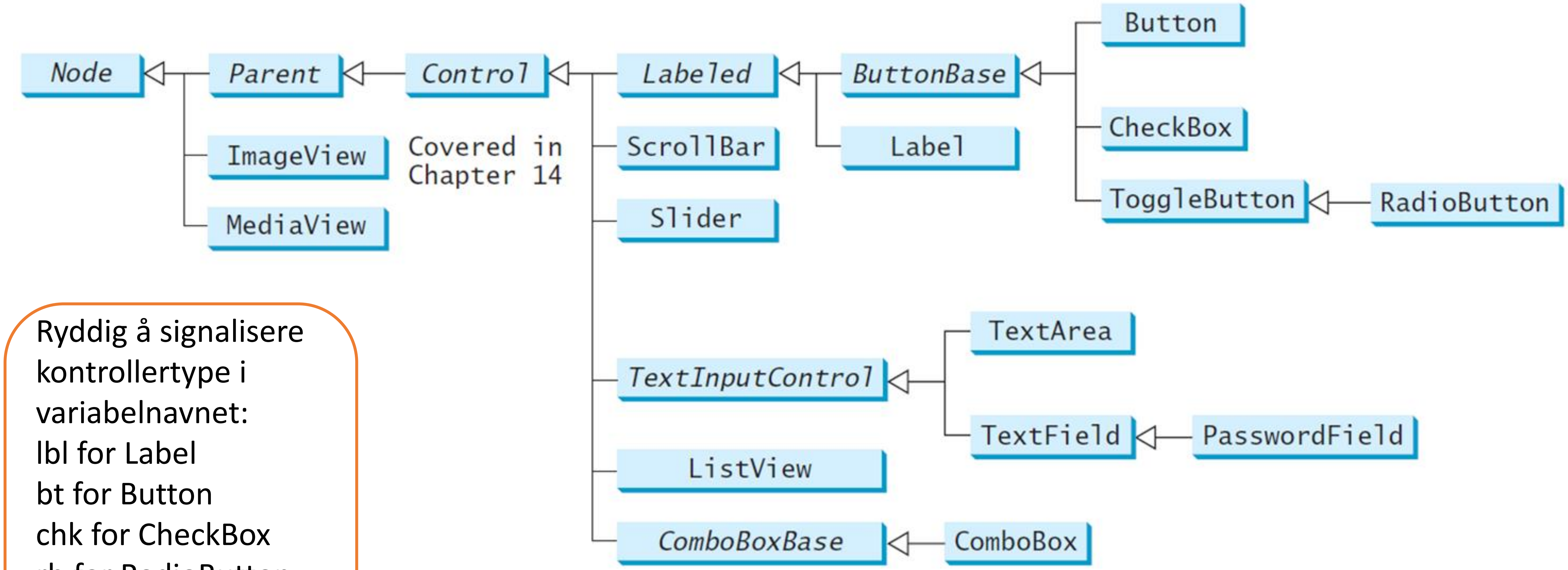
15.13 Case Study: US Map

Blir ikke gjennomgått, men bruker bare kjent stoff.

Les koden og se om du har forstått alt. Nyttig repetisjon!

Illustrerer nyttig bruk av Group: skalerer alle polygonene likt.

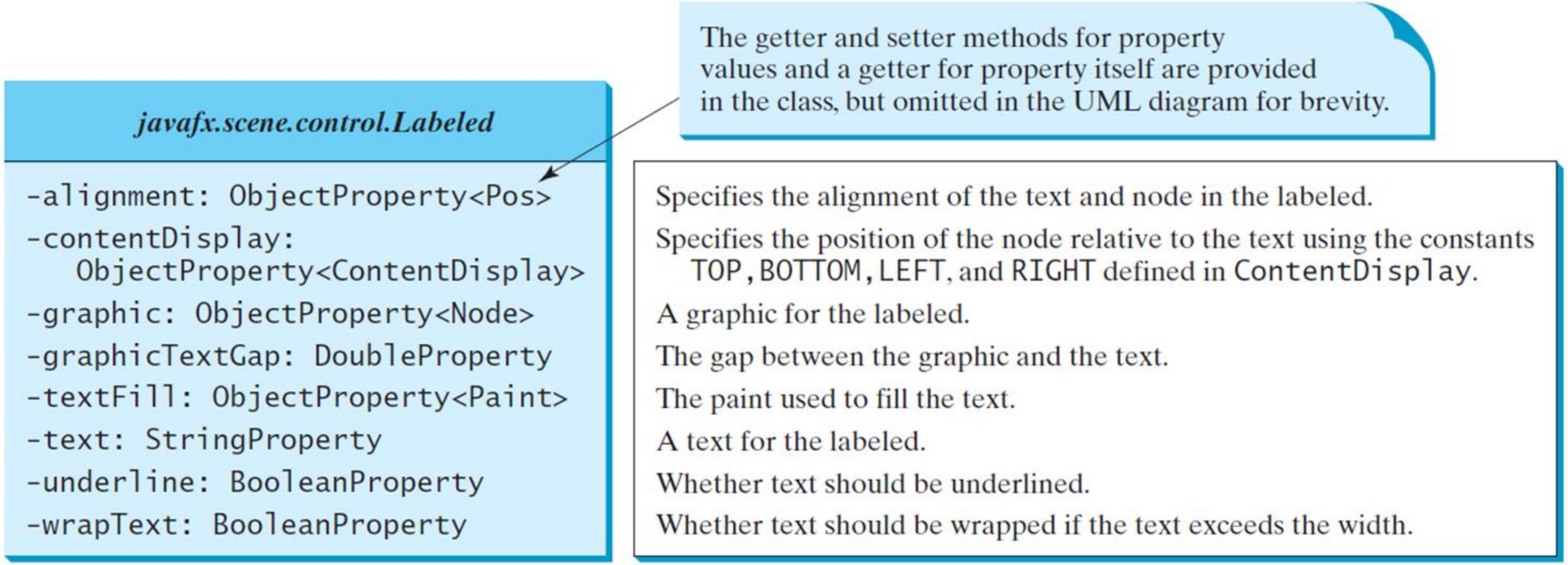
JavaFX UI Controls



Klassehierarkiet for sentrale kontroller-klasser

Ryddig å signalisere kontrollertype i variabelnavnet:
lbl for Label
bt for Button
chk for CheckBox
rb for RadioButton
tf for TextField o.s.v.

class Labeled



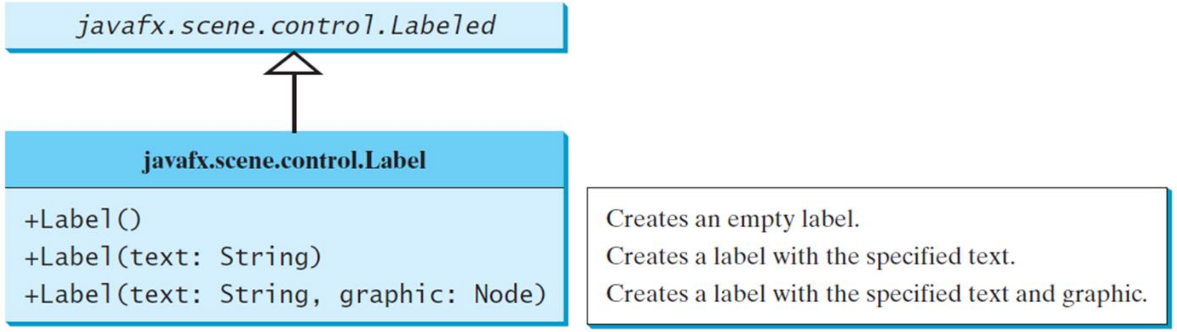
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

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.

Superklasse for (beskriver felles egenskaper for) bl.a.
Label, Button, CheckBox og RadioButton

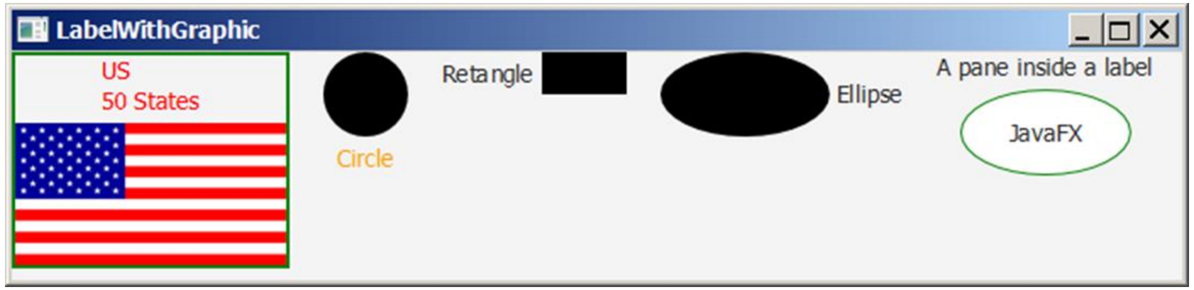
class Label



javafx.scene.control.Labeled

javafx.scene.control.Label

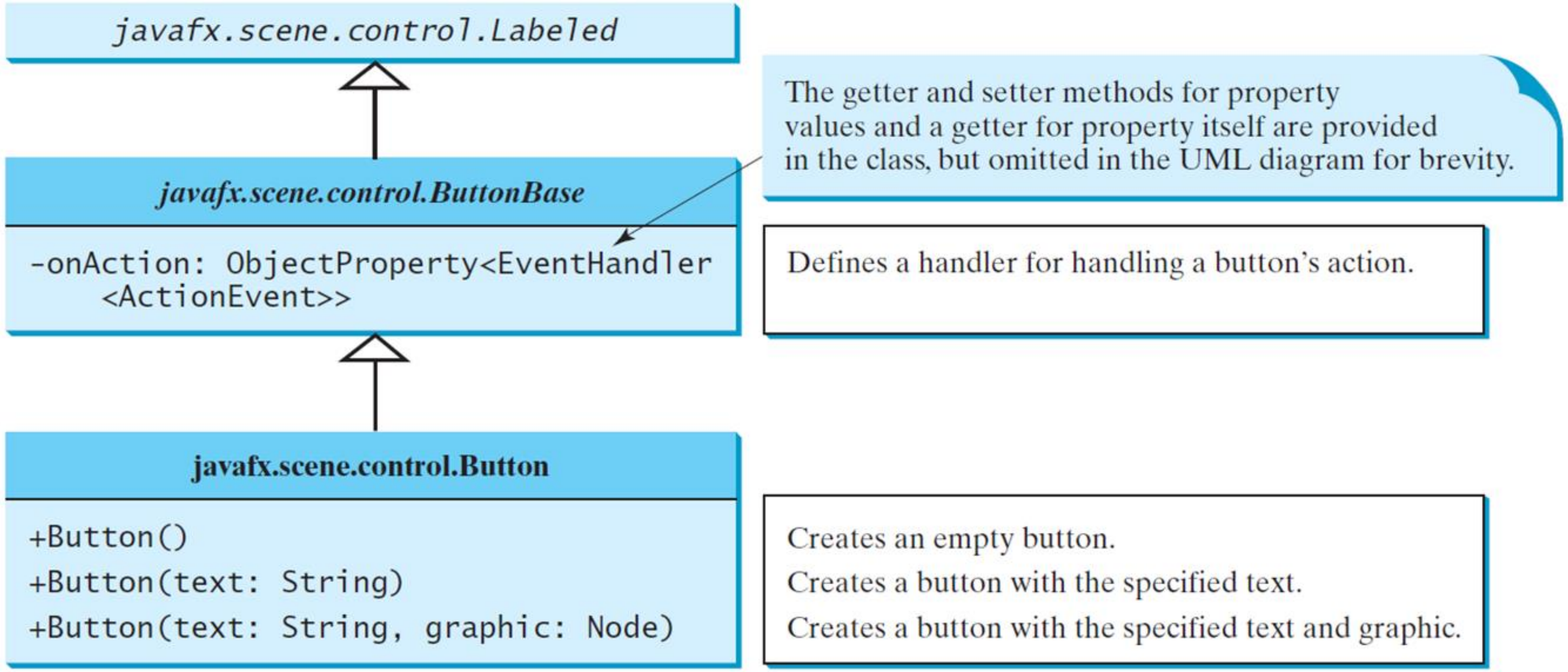
+Label() — Creates an empty label.
+Label(text: String) — Creates a label with the specified text.
+Label(text: String, graphic: Node) — Creates a label with the specified text and graphic.



Oftest har vi bare en tekst i en Label. Men den kan inneholde mye rart!

```
public void start(Stage primaryStage) {
    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);

    HBox pane = new HBox(20);
    pane.getChildren().addAll(lb1, lb2, lb3, lb4, lb5);

    // Create a scene and place it in the stage ...
```

ButtonBase og Button



javafx.scene.control.Labeled

javafx.scene.control.ButtonBase

```
-onAction: ObjectProperty<EventHandler
    <ActionEvent>>
```

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.

javafx.scene.control.Button

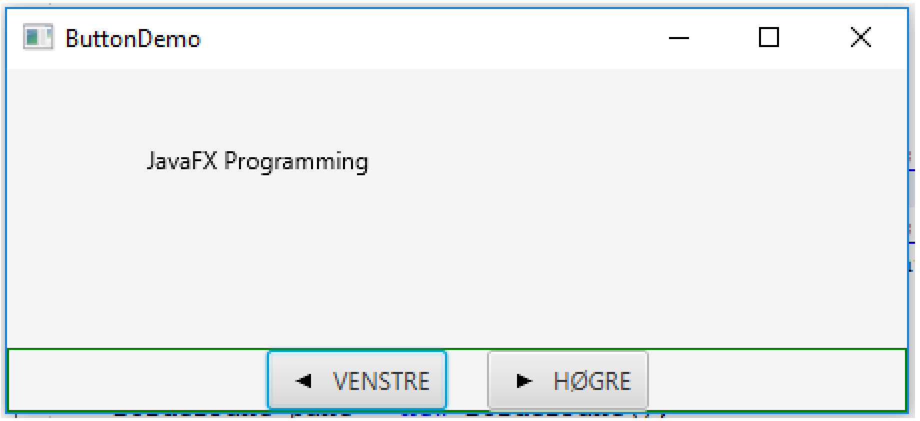
```
+Button()
+Button(text: String)
+Button(text: String, graphic: Node)
```

Creates an empty button.

Creates a button with the specified text.

Creates a button with the specified text and graphic.

class ButtonDemo

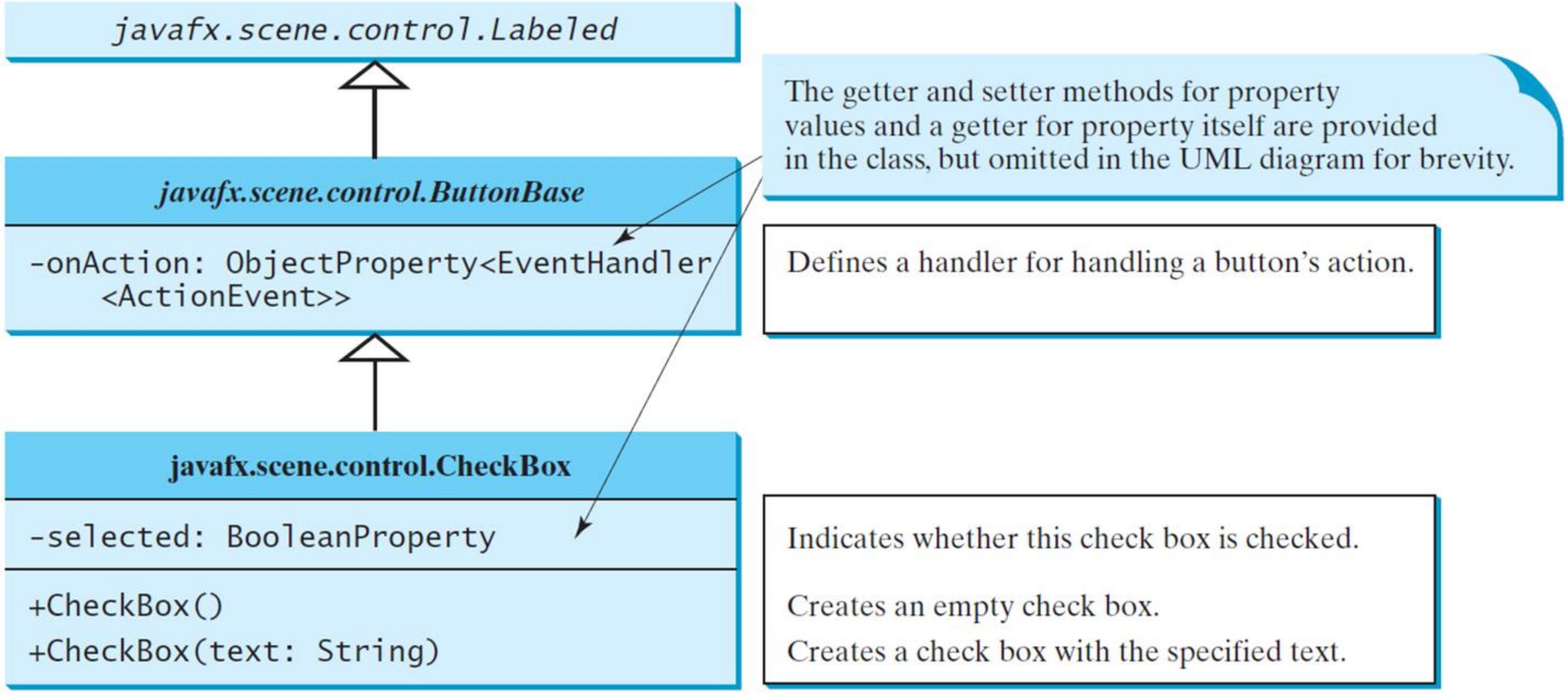


ButtonDemo

JavaFX Programming

```
public class ButtonDemo extends Application {
  protected Text text = new Text(50, 50, "JavaFX Programming");

  protected BorderPane getPane() {
    HBox paneForButtons = new HBox(20);
    Button btLeft = new Button("VENSTRE",
      new ImageView("image/left.gif"));
    Button btRight = new Button("HØGRE",
      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;
  }

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

class CheckBox



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.

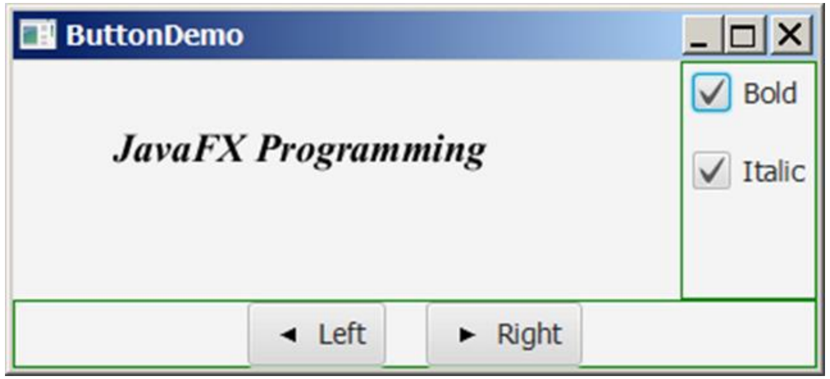
Indicates whether this check box is checked.

Creates an empty check box.

Creates a check box with the specified text.

Brukes typisk til Ja/Nei, Av/På og lignende.

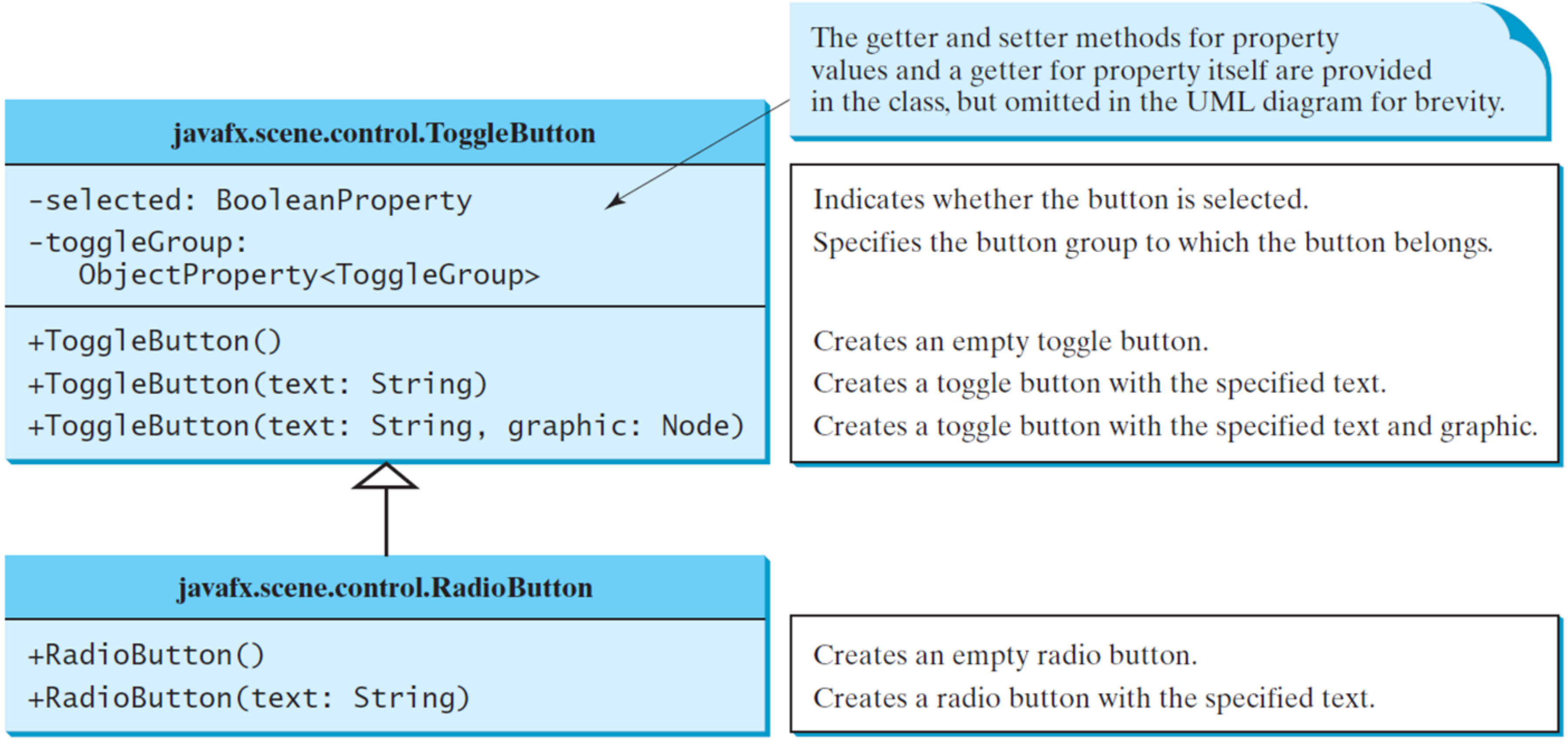
CheckBoxDemo



```java
public class CheckBoxDemo extends ButtonDemo {
  @Override
  protected BorderPane getPane() {
    BorderPane pane = super.getPane();

    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);
```

```java
EventHandler<ActionEvent> handler = e -> {
      if (chkBold.isSelected() && chkItalic.isSelected()) {
        text.setFont(fontBoldItalic);
      }
      else if (chkBold.isSelected()) {
        text.setFont(fontBold);
      }
      else if (chkItalic.isSelected()) {
        text.setFont(fontItalic);
      }
      else {
        text.setFont(fontNormal);
      }
    };

    chkBold.setOnAction(handler);
    chkItalic.setOnAction(handler);

    return pane;
  }
```

class RadioButton



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

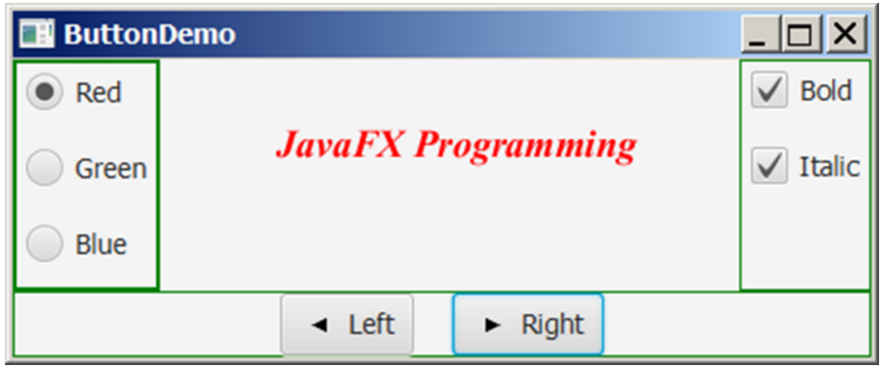
```
+RadioButton()
+RadioButton(text: String)
```

Creates an empty radio button.

Creates a radio button with the specified text.

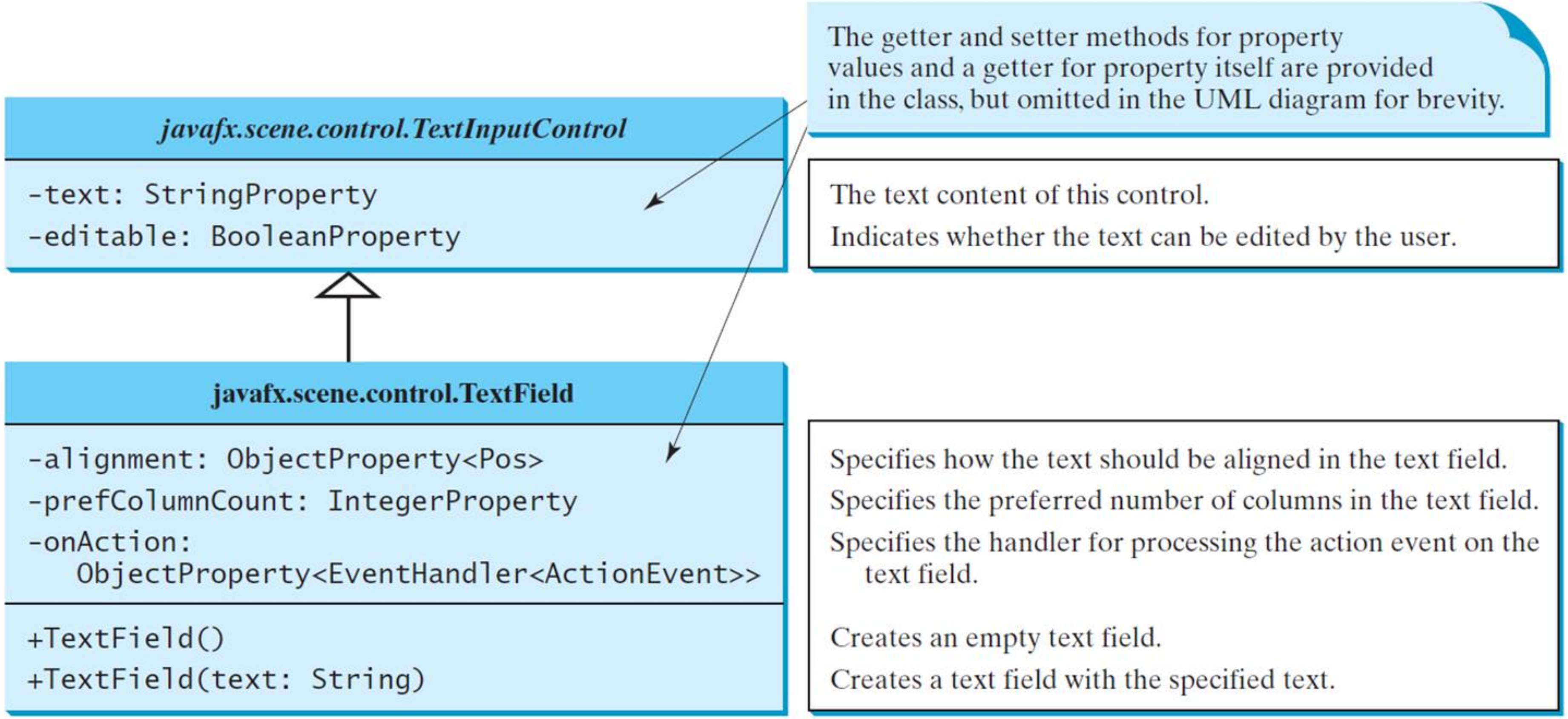
Hver enkelt RadioButton ligner på en CheckBox, men gruppe av RadioButtons kobles sammen slik at bare én kan velges.

RadioButtonDemo



```
public class RadioButtonDemo extends CheckBoxDemo {
  @Override

  protected BorderPane getPane() {
    BorderPane pane = super.getPane();

    VBox paneForRadioButtons = new VBox(20);
    paneForRadioButtons.setPadding(new Insets(5, 5, 5, 5));
    paneForRadioButtons.setStyle
      ("-fx-border-width: 2px; -fx-border-color: green");

    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);
      }
    });

    return pane;
  }
```

class TextField



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.

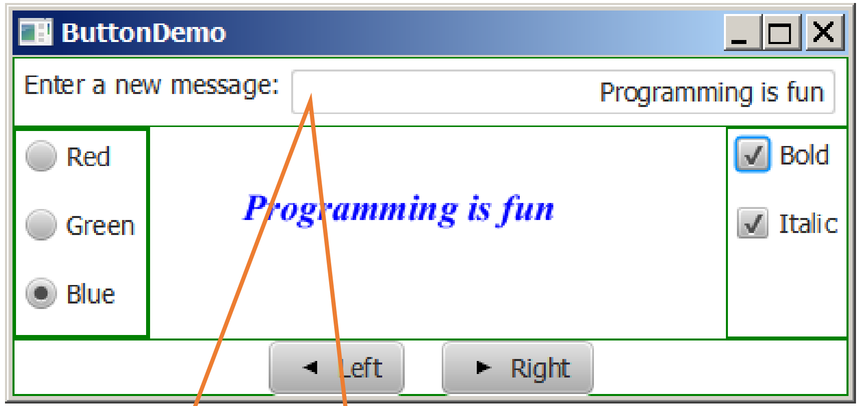
Specifies the handler for processing the action event on the text field.

Creates an empty text field.

Creates a text field with the specified text.

Som vi alt har sett i lånekalkulatoren: Brukes både til output og til input.

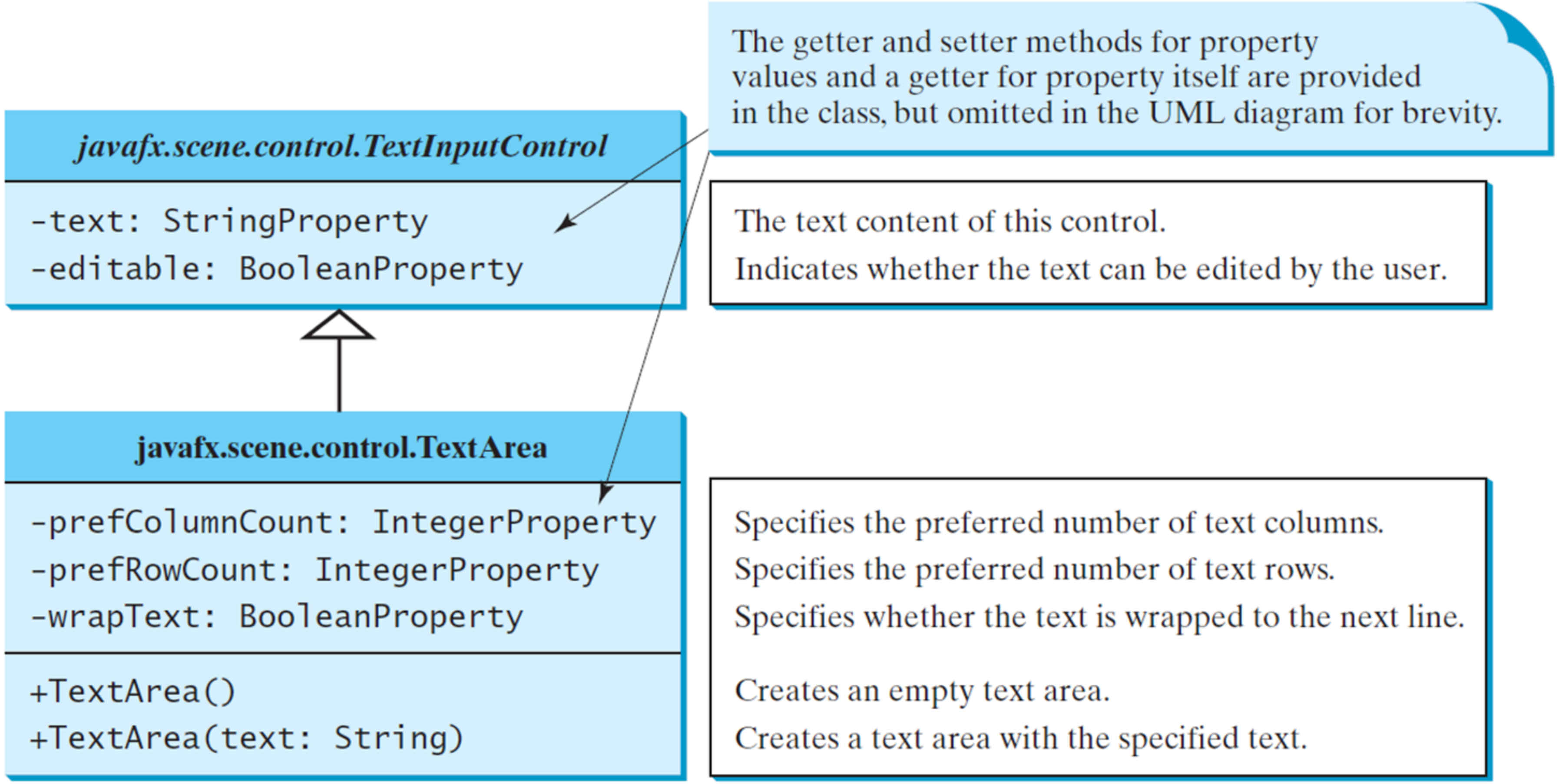
TextFieldDemo



Ny Label og TextField er lagt i en BorderPane. Kunne brukt HBox, GridPane og FlowPane, men BorderPane prioriterer senterelementet, TextField’en

```
public class TextFieldDemo extends RadioButtonDemo {
  @Override
  protected BorderPane getPane() {
    BorderPane pane = super.getPane();

    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);

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

    return pane;
  }
```

class 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.

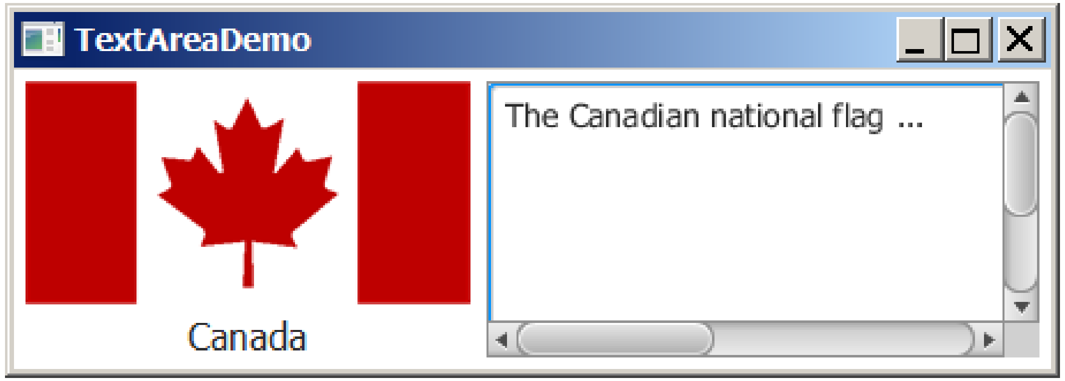
Specifies whether the text is wrapped to the next line.

Creates an empty text area.

Creates a text area with the specified text.

Tillater et helt område med tekst – menge linjer.

class DescriptionPane



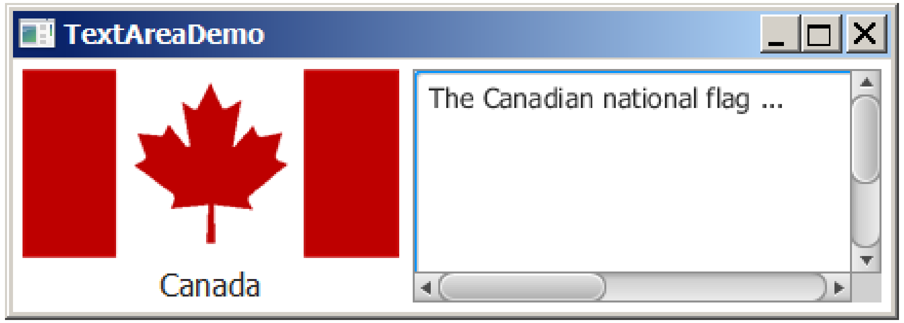
The Canadian national flag ...

Canada

```
public class DescriptionPane extends BorderPane {
  /** Label for displaying an image and a title */
  private Label lblImageTitle = new Label();

  /** Text area for displaying text */
  private TextArea taDescription = new TextArea();

  public DescriptionPane() {
    // Center the icon and text and place text under icon
    lblImageTitle.setContentDisplay(ContentDisplay.TOP);
    lblImageTitle.setPrefSize(200,  100);

    // Set the font in the label and the text field
    lblImageTitle.setFont(new Font("SansSerif", 16));
    taDescription.setFont(new Font("Serif", 14));

    taDescription.setWrapText(true);
    taDescription.setEditable(false);

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

    // Place label and scroll pane in the border pane
    setLeft(lblImageTitle);
    setCenter(scrollPane);
    setPadding(new Insets(5, 5, 5, 5));
  }
```

```
/** Set the title */
  public void setTitle(String title) {
    lblImageTitle.setText(title);
  }

  /** Set the image view */
  public void setImageView(ImageView icon) {
    lblImageTitle.setGraphic(icon);
  }

  /** Set the text description */
  public void setDescription(String text) {
    taDescription.setText(text);
  }
}
```

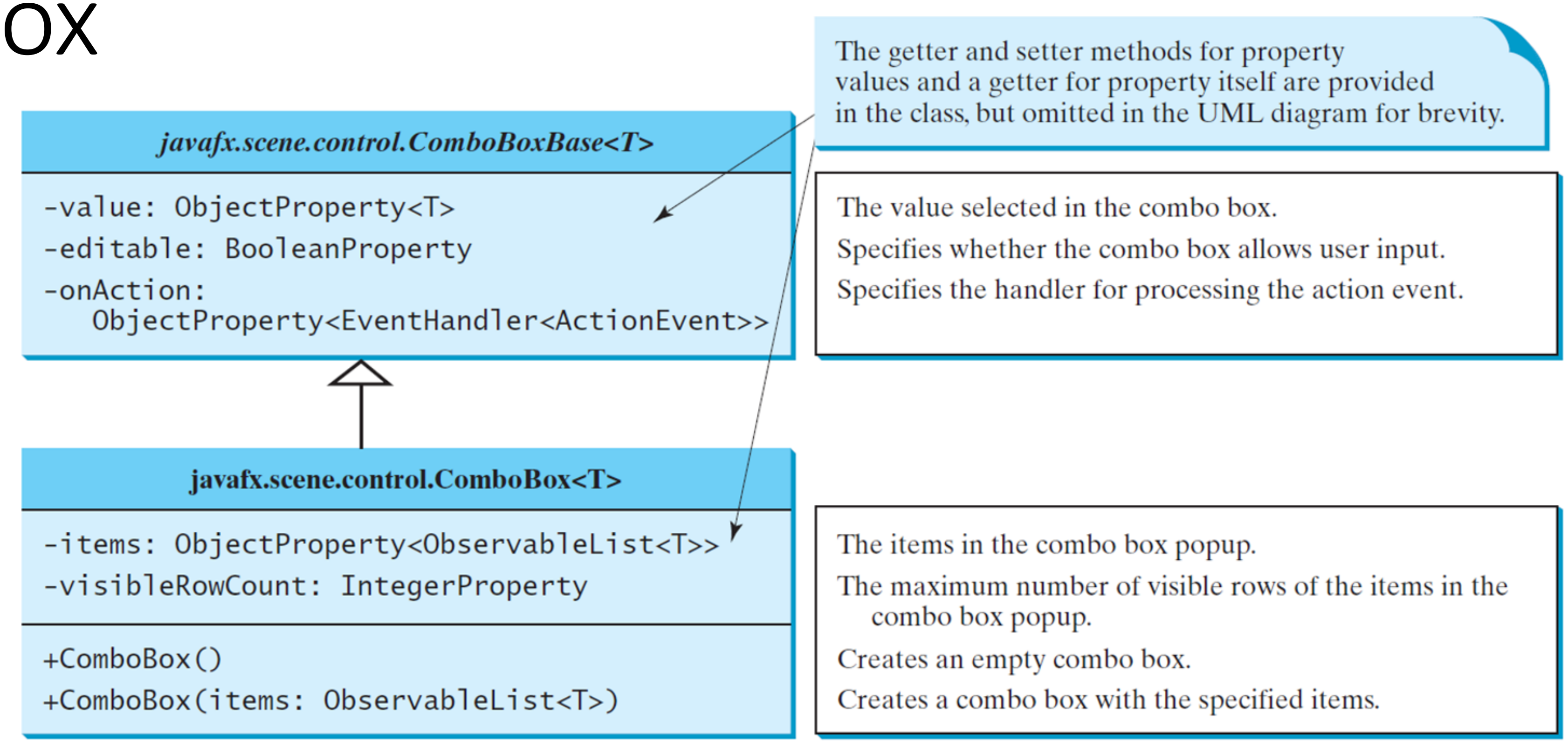
class TextAreaDemo



The Canadian national flag ...

```
public class TextAreaDemo extends Application {
  @Override
  public void start(Stage primaryStage) {
    // Declare and create a description pane
    DescriptionPane descriptionPane = new DescriptionPane();

    // Set title, text and image in the description pane
    descriptionPane.setTitle("Canada");
    String description = "The Canadian national flag ...";
    descriptionPane.setImageView(new ImageView(
      "https://home.usn.no/lonnesta/kurs/OBJ2000/image/ca.gif"));
    descriptionPane.setDescription(description);

    // Create a scene and place it in the stage
    Scene scene = new Scene(descriptionPane, 450, 200);
    primaryStage.setTitle("TextAreaDemo");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

class ComboBox



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>

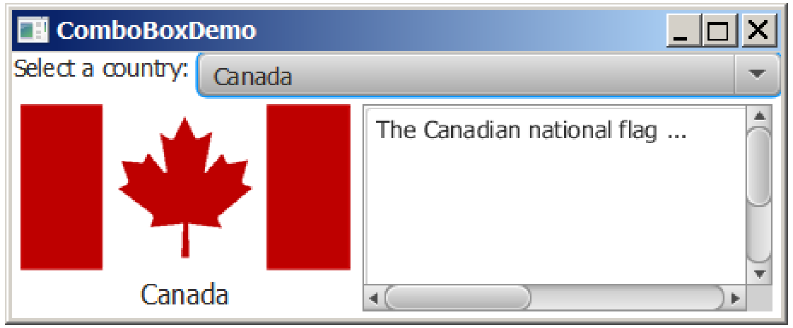
```
-value: ObjectProperty<T>
-editable: BooleanProperty
-onAction:
   ObjectProperty<EventHandler<ActionEvent>>
```

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.



Select a country: Canada

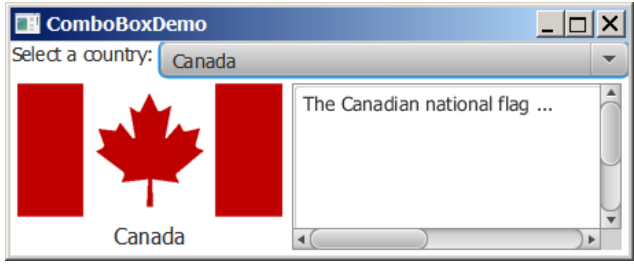
The Canadian national flag ...

Canada

ComboBox / valgliste / drop-down-liste :
Vis fram alternativer, la bruker velge

ComboBoxDemo

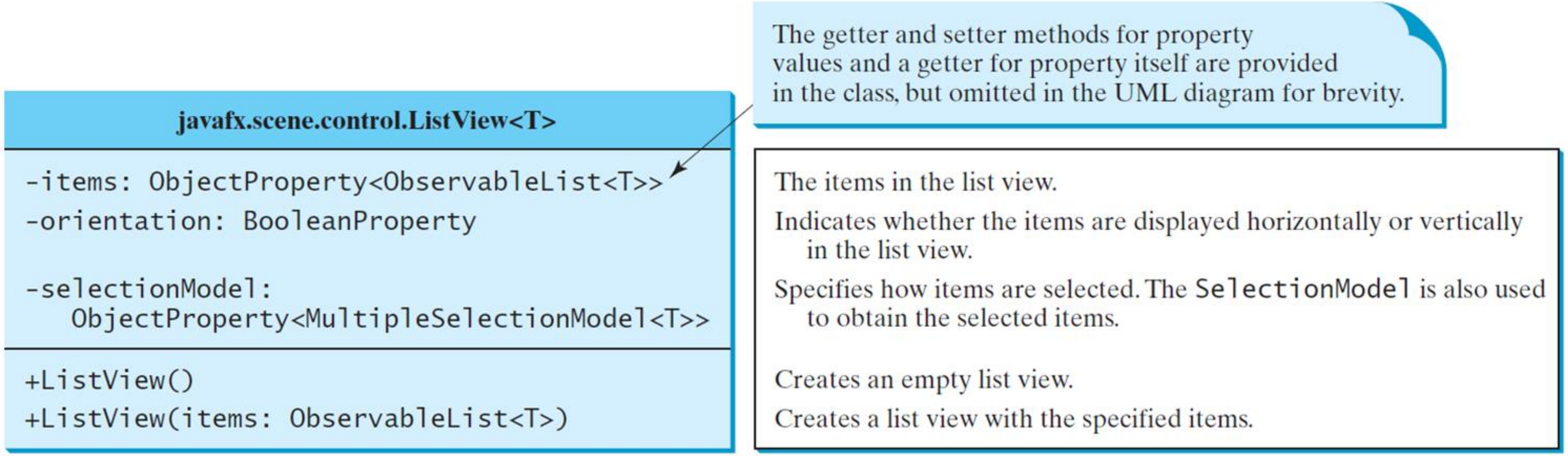
```
public class ComboBoxDemo extends Application {
  // Declare an array of Strings for flag titles
  private String[] flagTitles = {"Canada", "China", "Denmark",
    "France", "Germany", "India", "Norway", "United Kingdom",
    "United States of America"};

  // Declare an ImageView array for the national flags of 9 countries
  private ImageView[] flagImage = {new ImageView("image/ca.gif"),
    new ImageView("image/china.gif"), new ImageView("image/denmark.gif"),
    new ImageView("image/fr.gif"), new ImageView("image/germany.gif"),
    new ImageView("image/india.gif"), new ImageView("image/norway.gif"),
    new ImageView("image/uk.gif"), new ImageView("image/us.gif")};

  // Declare an array of strings for flag descriptions
  private String[] flagDescription = new String[9];

  // Declare and create a description pane
  private DescriptionPane descriptionPane = new DescriptionPane();

  // Create a combo box for selecting countries
  private ComboBox<String> cbo = new ComboBox<>(); // flagTitles

  @Override
  public void start(Stage primaryStage) {
    // Set text description
    flagDescription[0] = "The Canadian national flag ...";
    flagDescription[1] = "Description for China ... ";
    ...
    flagDescription[8] = "Description for US ... ";
```



The Canadian national flag ...

```
    // Set the first country (Canada) for display
    setDisplay(0);

    // Add combo box and description pane to the border pane
    BorderPane pane = new BorderPane();

    BorderPane paneForComboBox = new BorderPane();
    paneForComboBox.setLeft(new Label("Select a country: "));
    paneForComboBox.setCenter(cbo);
    pane.setTop(paneForComboBox);
    cbo.setPrefWidth(400);
    cbo.setValue("Canada");

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

    // Display the selected country
    cbo.setOnAction(e -> setDisplay(items.indexOf(cbo.getValue())));

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 450, 170);
    primaryStage.setTitle("ComboBoxDemo");
    primaryStage.setScene(scene);
    primaryStage.show();
  }

  /** Set display information on the description pane */
  public void setDisplay(int index) {
    descriptionPane.setTitle(flagTitles[index]);
    descriptionPane.setImageView(flagImage[index]);
    descriptionPane.setDescription(flagDescription[index]);
  }
```

class 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.

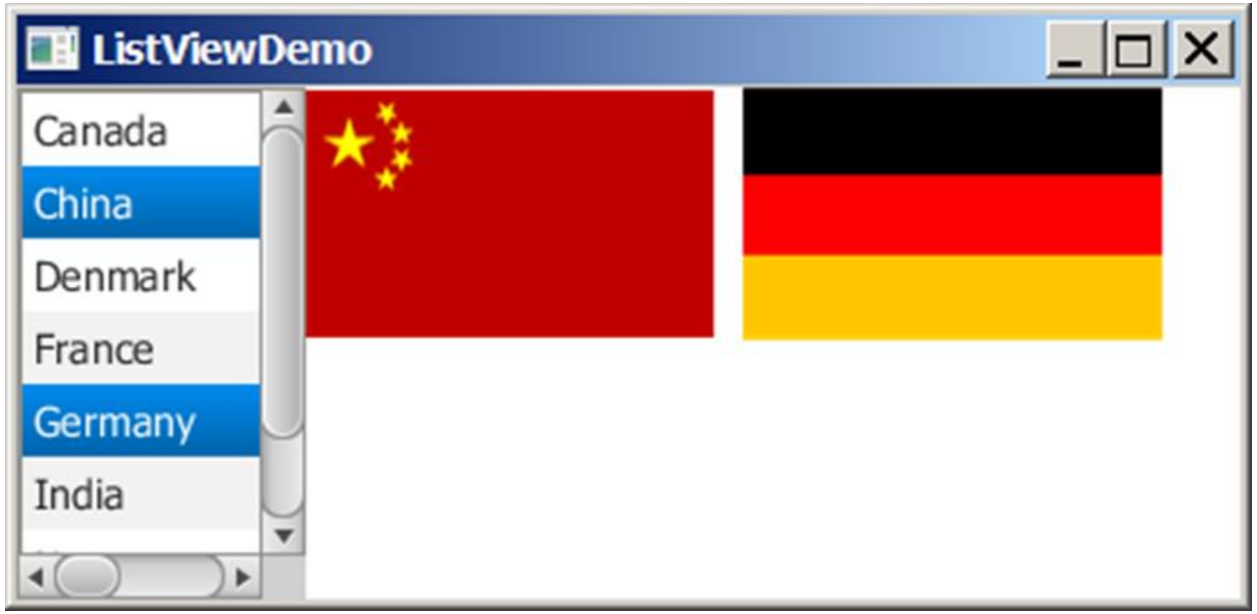
Creates an empty list view.

Creates a list view with the specified items.

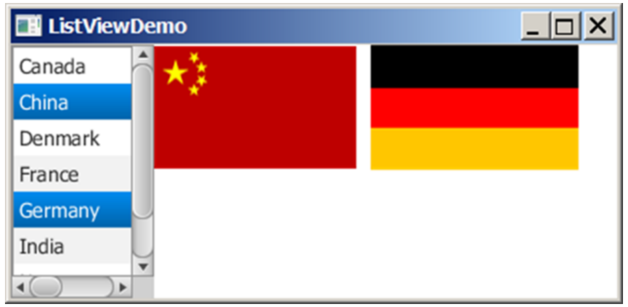
ListView ligner mye på ComboBox men tillater at bruker velger flere elementer

ListViewDemo

Her skal bruker kunne velge flere land fra lista og få vist fram flaggene.



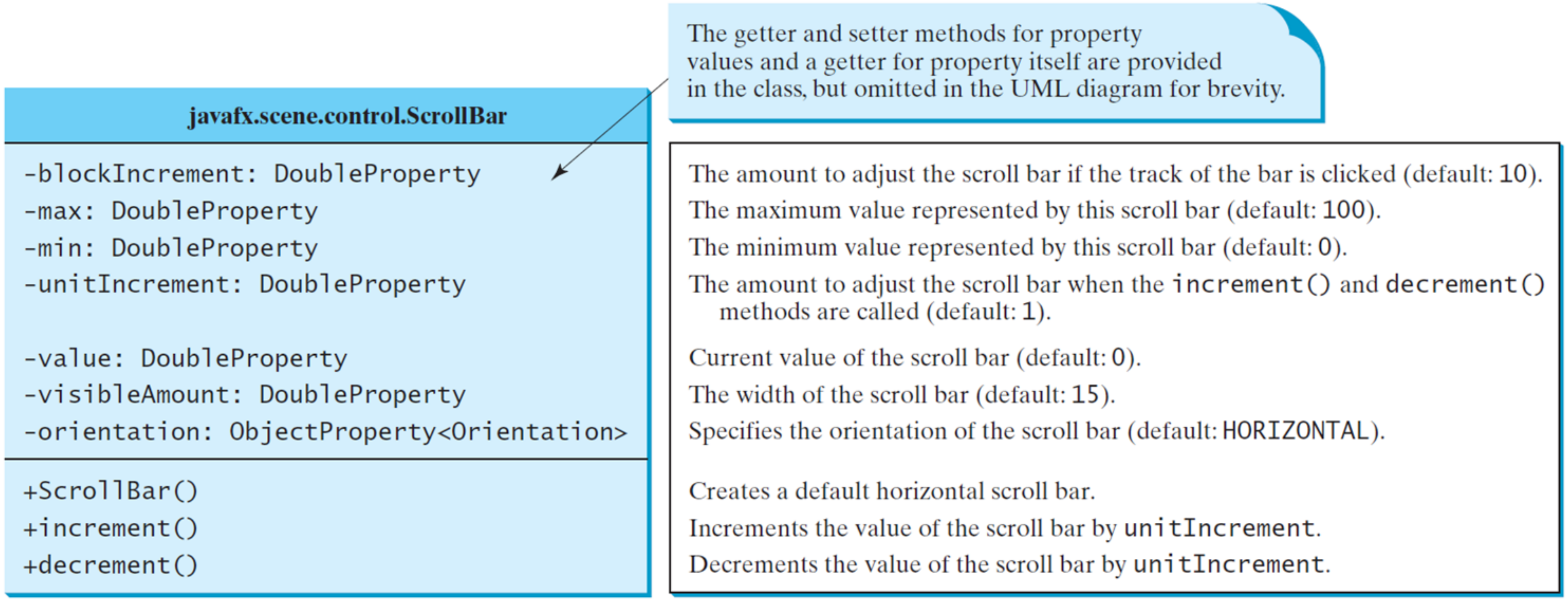
ListViewDemo



```
public class ListViewDemo extends Application {
  // Declare an array of Strings for flag titles
  private String[] flagTitles = {"Canada", "China",
    "Denmark", "France", "Germany", "India", "Norway",
    "United Kingdom", "United States of America"};

  // Declare an ImageView array for the national flags
  private ImageView[] ImageViews = {
    new ImageView("image/ca.gif"),
    new ImageView("image/china.gif"),
    new ImageView("image/denmark.gif"),
    new ImageView("image/fr.gif"),
    new ImageView("image/germany.gif"),
    new ImageView("image/india.gif"),
    new ImageView("image/norway.gif"),
    new ImageView("image/uk.gif"),
    new ImageView("image/us.gif")
  };
```

```
@Override
public void start(Stage primaryStage) {
  ListView<String> lv = new ListView<>
    (FXCollections.observableArrayList(flagTitles));
  lv.setPrefSize(140, 400);
  lv.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);

  // Create a pane to hold image views
  FlowPane imagePane = new FlowPane(10, 10);
  BorderPane pane = new BorderPane();
  pane.setLeft(new ScrollPane(lv));
  pane.setCenter(imagePane);

  lv.getSelectionModel().selectedItemProperty().addListener(
    ov -> {
      imagePane.getChildren().clear();
      for (Integer i: lv.getSelectionModel().getSelectedIndices()) {
        imagePane.getChildren().add(ImageViews[i]);
      }
  });

  // Create a scene and place it in the stage
  Scene scene = new Scene(pane, 450, 170);
  primaryStage.setTitle("ListViewDemo");
  primaryStage.setScene(scene);
  primaryStage.show();
}
```

class ScrollBar



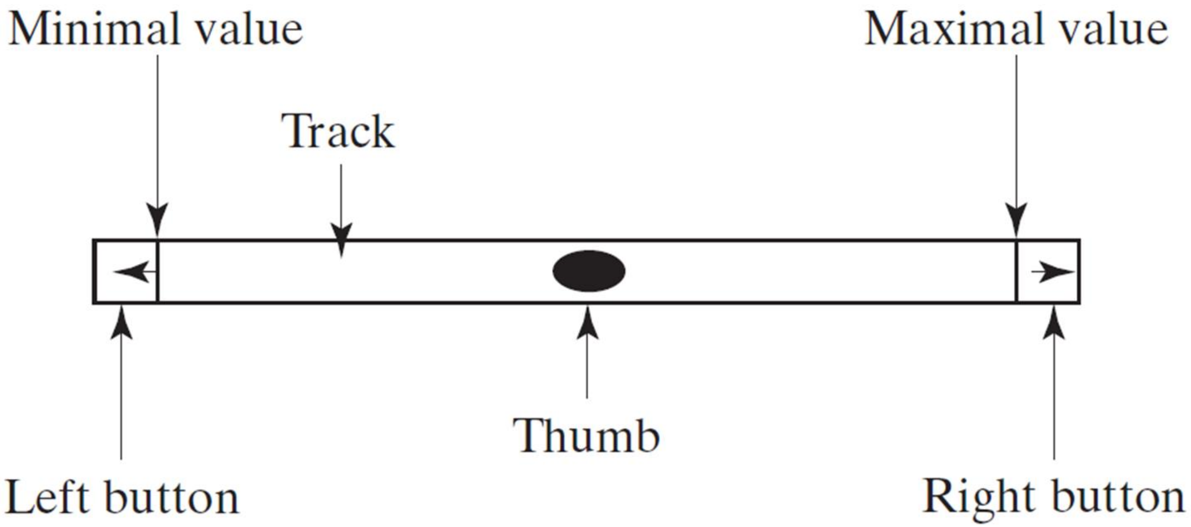
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

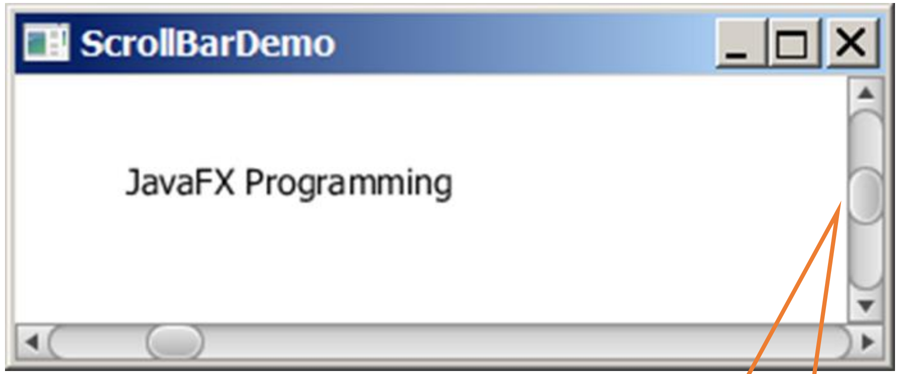
javafx.scene.control.ScrollBar

`-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`.

Velg verdi ved å skyve på en knapp – horisontalt eller vertikalt

class ScrollBar

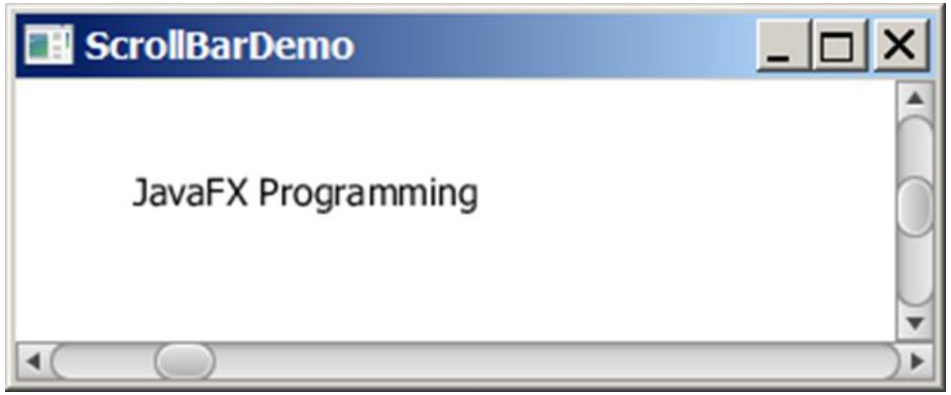




JavaFX Programming

Kan eksempelvis bruke ScrollBar til å «scrolle» teksten rundt omkring. Dette kan også løses med ScrollPane

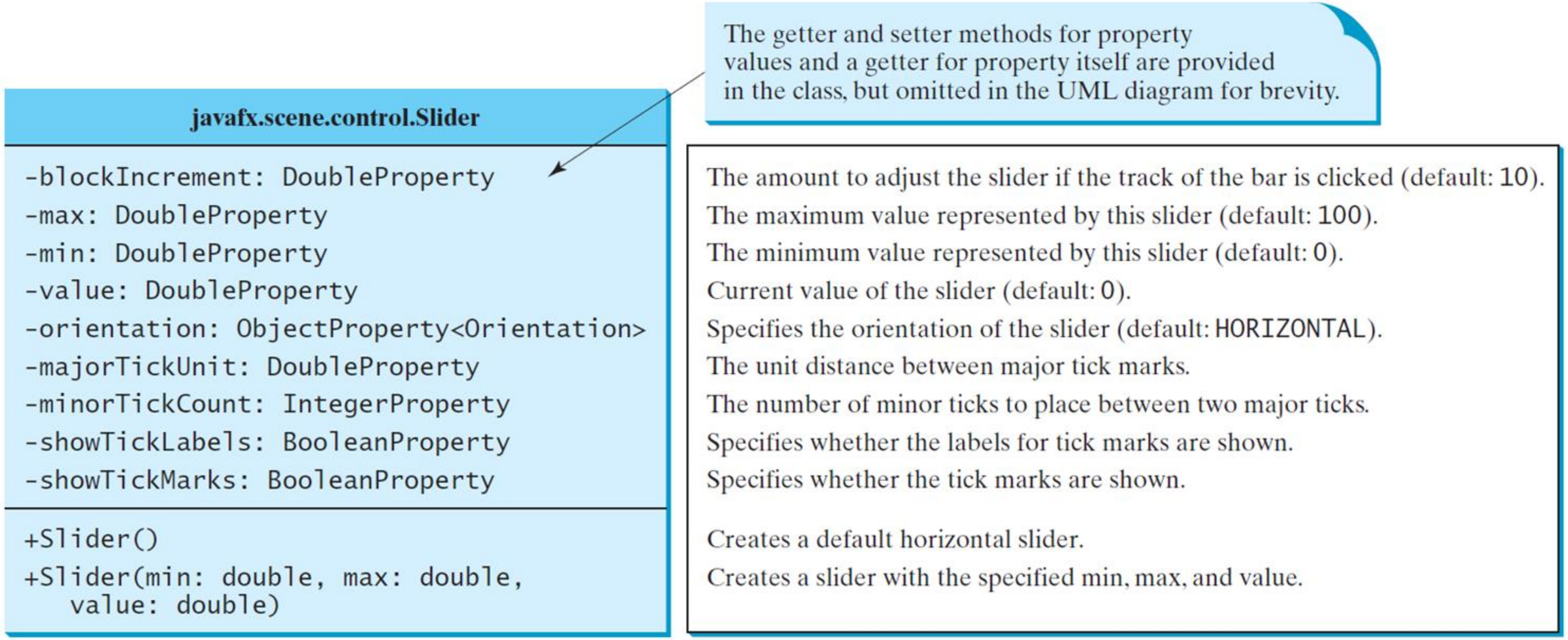
ScrollBarDemo



```
public class ScrollBarDemo extends Application {
  @Override
  public void start(Stage primaryStage) {
    Text text = new Text(20, 20,
      "JavaFX Programming");

    ScrollBar sbHorizontal = new ScrollBar();
    ScrollBar sbVertical = new ScrollBar();
    sbVertical.setOrientation(Orientation.VERTICAL);

    // Create a text in a pane
    Pane paneForText = new Pane();
    paneForText.getChildren().add(text);

    // Create pane to hold text and scroll bars
    BorderPane pane = new BorderPane();
    pane.setCenter(paneForText);
    pane.setBottom(sbHorizontal);
    pane.setRight(sbVertical);
```

```
// Listener for horizontal scroll bar value change
    sbHorizontal.valueProperty().addListener(ov ->
      text.setX(sbHorizontal.getValue() *
        paneForText.getWidth() / sbHorizontal.getMax()));

    // Listener for vertical scroll bar value change
    sbVertical.valueProperty().addListener(ov ->
      text.setY(sbVertical.getValue() *
        paneForText.getHeight() / sbVertical.getMax()));

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 450, 170);
    primaryStage.setTitle("ScrollBarDemo");
    primaryStage.setScene(scene);
    primaryStage.show();
  }
```

class Slider



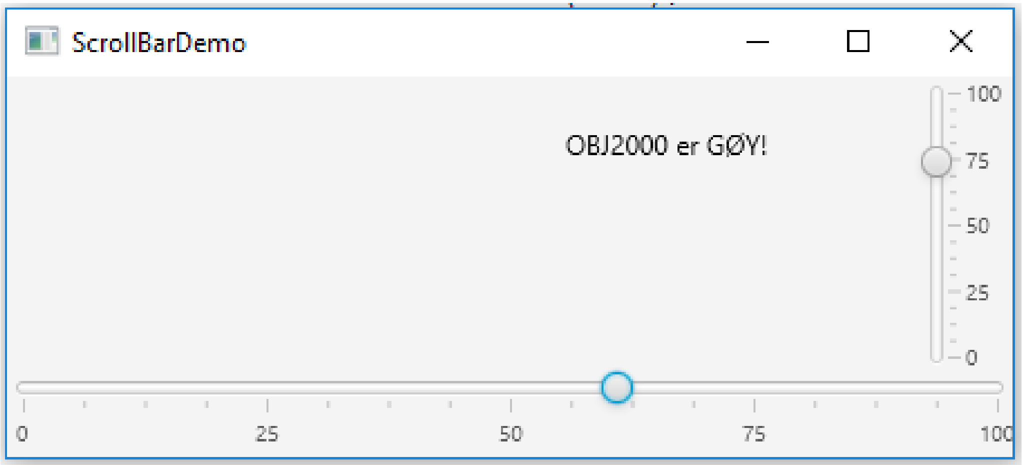
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the UML diagram for brevity.

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.

Ligner mye på ScrollBar men har flere egenskaper.
Brukes gjerne når man skal velge en helt bestemt verdi.

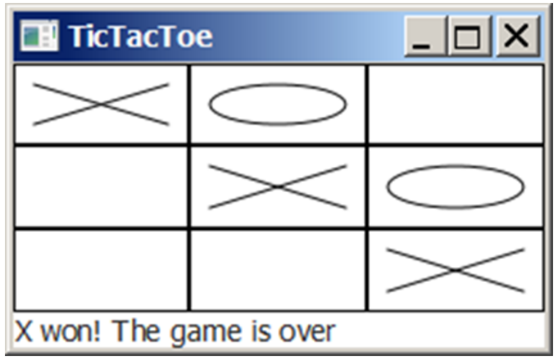
SliderDemo

Det burde være greit å skrive om ScrollBarDemo til å bruke Slider ist. for ScrollBar.

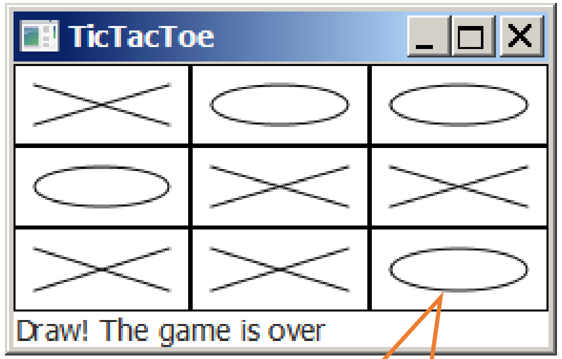


```
@Override
public void start(Stage primaryStage) {
  Text text = new Text(20, 20, "OBJ2000 er GØY!");

  Slider slHorizontal = new Slider();
  slHorizontal.setShowTickLabels(true);
  slHorizontal.setShowTickMarks(true);

  Slider slVertical = new Slider();
  slVertical.setOrientation(Orientation.VERTICAL);
  slVertical.setShowTickLabels(true);
  slVertical.setShowTickMarks(true);
  slVertical.setValue(100);

  // Videre er det likt ScrollBarDemo!
```

Case Study: TicTacToe - TrippTrappTresko

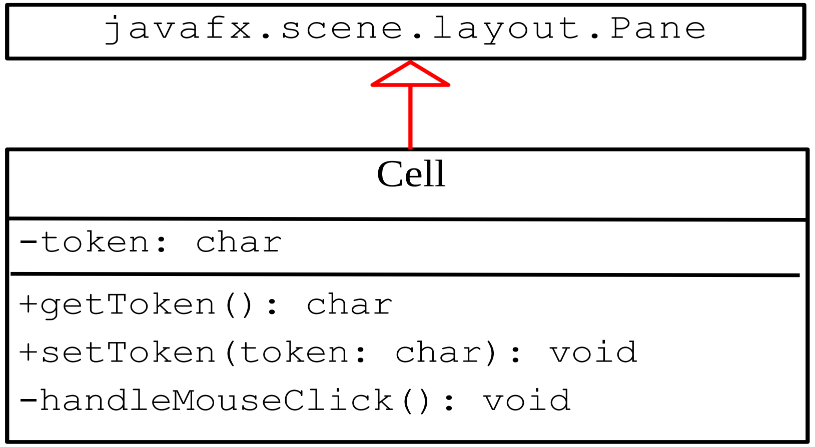


X won! The game is over



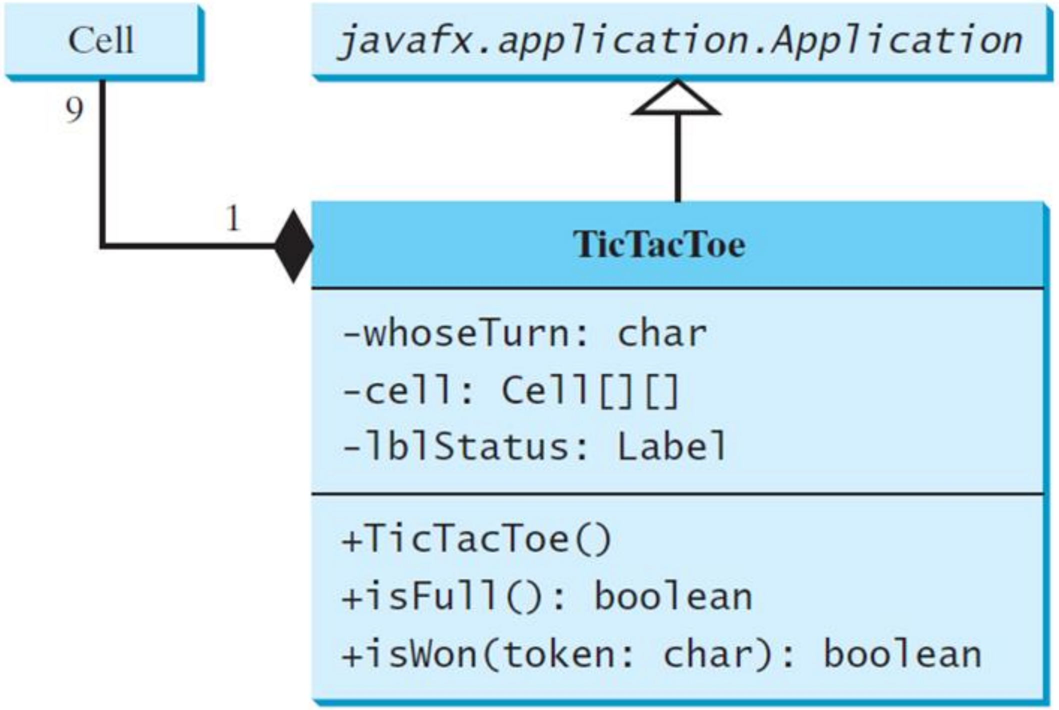
Draw! The game is over

Hver rute på brettet er en Celle



```
javafx.scene.layout.Pane

Cell
-token: char
+getToken(): char
+setToken(token: char): void
-handleMouseClick(): void
```

Case Study: TicTacToe - TrippTrappTresko



```
-whoseTurn: char
-cell: Cell[][]
-lblStatus: Label

+TicTacToe()
+isFull(): boolean
+isWon(token: char): boolean
```

Indicates which player has the turn, initially X.
A 3 × 3, two-dimensional array for cells.
A label to display game status.

Constructs the TicTacToe user interface.
Returns true if all cells are filled.
Returns true if a player with the specified token has won.

Case Study: TicTacToe - TrippTrappTresko

```java
public class TicTacToe extends Application {
  // Indicate which player has a turn, initially it is the X player
  private char whoseTurn = 'X';

  // Create and initialize cell
  private Cell[][] cell =  new Cell[3][3];

  // Create and initialize a status label
  private Label lblStatus = new Label("X's turn to play");

  @Override
  public void start(Stage primaryStage) {
    // Pane to hold cell
    GridPane pane = new GridPane();
    for (int i = 0; i < 3; i++)
      for (int j = 0; j < 3; j++)
        pane.add(cell[i][j] = new Cell(), j, i);

    BorderPane borderPane = new BorderPane();
    borderPane.setCenter(pane);
    borderPane.setBottom(lblStatus);

    // Create a scene and place it in the stage
    Scene scene = new Scene(borderPane, 450, 170);
    primaryStage.setTitle("TicTacToe"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
```

Case Study: TicTacToe - TrippTrappTresko

```
/** Determine if the cell are all occupied */
 public boolean isFull() {
   for (int i = 0; i < 3; i++)
     for (int j = 0; j < 3; j++)
       if (cell[i][j].getToken() == ' ')
         return false;

   return true;
 }

 /** Determine if the player with the specified token wins */
 public boolean isWon(char token) {
   for (int i = 0; i < 3; i++)
     if (cell[i][0].getToken() == token
         && cell[i][1].getToken() == token
         && cell[i][2].getToken() == token) {
       return true;
     }

   for (int j = 0; j < 3; j++)
     if (cell[0][j].getToken() ==  token
         && cell[1][j].getToken() == token
         && cell[2][j].getToken() == token) {
       return true;
     }

   if (cell[0][0].getToken() == token
       && cell[1][1].getToken() == token
       && cell[2][2].getToken() == token) {
     return true;
   }

   if (cell[0][2].getToken() == token
       && cell[1][1].getToken() == token
       && cell[2][0].getToken() == token) {
     return true;
   }

   return false;
 }  // end isWon
```

Case Study: TicTacToe - TrippTrappTresko

```
// An inner class for a cell
 public class Cell extends Pane {
   // Token used for this cell
   private char token = ' ';

   public Cell() {
     setStyle("-fx-border-color: black");
     this.setPrefSize(800, 800);
     this.setOnMouseClicked(e -> handleMouseClick());
   }

   /** Return token */
   public char getToken() {
     return token;
   }
```

```
/** Set a new token */
  public void setToken(char c) {
    token = c;

    if (token == 'X') {
      Line line1 = new Line(10, 10,
        this.getWidth() - 10, this.getHeight() - 10);
      line1.endXProperty().bind(this.widthProperty().subtract(10));
      line1.endYProperty().bind(this.heightProperty().subtract(10));
      Line line2 = new Line(10, this.getHeight() - 10,
        this.getWidth() - 10, 10);
      line2.startYProperty().bind(
        this.heightProperty().subtract(10));
      line2.endXProperty().bind(this.widthProperty().subtract(10));

      // Add the lines to the pane
      this.getChildren().addAll(line1, line2);
    }
    else if (token == 'O') {
      Ellipse ellipse = new Ellipse(this.getWidth() / 2,
        this.getHeight() / 2, this.getWidth() / 2 - 10,
        this.getHeight() / 2 - 10);
      ellipse.centerXProperty().bind(
        this.widthProperty().divide(2));
      ellipse.centerYProperty().bind(
          this.heightProperty().divide(2));
      ellipse.radiusXProperty().bind(
          this.widthProperty().divide(2).subtract(10));
      ellipse.radiusYProperty().bind(
          this.heightProperty().divide(2).subtract(10));
      ellipse.setStroke(Color.BLACK);
      ellipse.setFill(Color.WHITE);

      getChildren().add(ellipse); // Add the ellipse to the pane
    }
  }
```

Case Study: TicTacToe - TrippTrappTresko

```
/* Handle a mouse click event */
   private void handleMouseClick() {
     // If cell is empty and game is not over
     if (token == ' ' && whoseTurn != ' ') {
       setToken(whoseTurn); // Set token in the cell

       // Check game status
       if (isWon(whoseTurn)) {
         lblStatus.setText(whoseTurn + " won! The game is over");
         whoseTurn = ' '; // Game is over
       }
       else if (isFull()) {
         lblStatus.setText("Draw! The game is over");
         whoseTurn = ' '; // Game is over
       }
       else {
         // Change the turn
         whoseTurn = (whoseTurn == 'X') ? 'O' : 'X';
         // Display whose turn
         lblStatus.setText(whoseTurn + "'s turn");
       }
     }
   }
 }
```

```
 /**
   * main method
   */
  public static void main(String[] args) {
    launch(args);
  }
}
```