COMP3021 Java Programming

Supplementary note set: UI Control and Multimedia

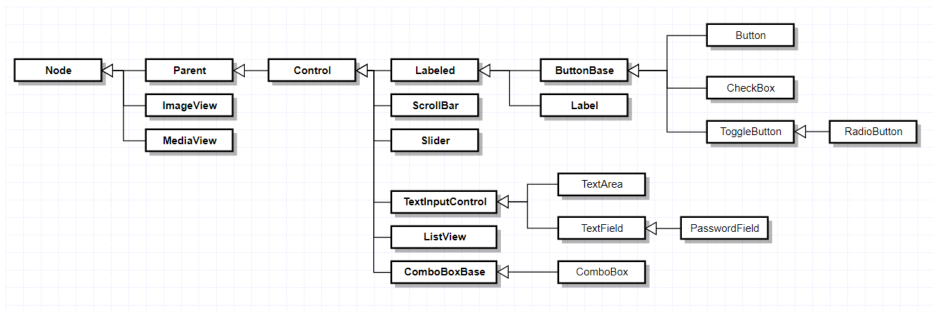
Dr. Alex Lam

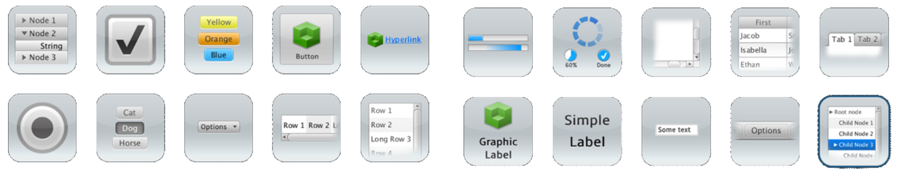
Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China

Motivations

- A Graphical User Interface (GUI) makes a system user-friendly and easy to use
- Creating a GUI requires creativity and knowledge of how GUI components work
- GUI components are designed to be flexible and versatile so that we can create a wide assortment of useful user interfaces
- We will visit some popular JavaFX GUI components, which support event-driven programming

Frequently Used UI Controls





Labeled Abstract Class (javafx.scene.control.Labeled)

- A label is a display area for a short text, a node, or both. If 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

Method	Description
ObjectProperty<Pos> alignment	Specifies the alignment of the text and node in the labeled
ObjectProperty<ContentDisplay> contentDisplay	Specifies the position of the node relative to the text using the constants TOP, BOTTOM, LEFT and RIGHT defined in ContentDisplay
ObjectProperty<Node> graphic	A graphic for the labeled
DoubleProperty graphicsTextGap	The gap between the graphics and text
ObjectProperty<Paint> textFill	The paint used to fill the text
StringProperty text	A text for the labeled
BooleanProperty underline	Whether text should be underlined
BooleanProperty wrapText	Whether text should be wrapped if the text exceeds the width

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

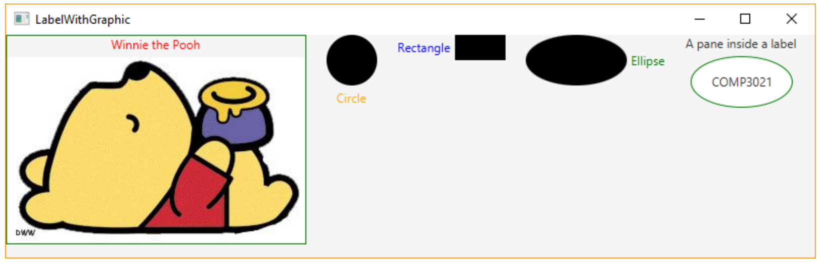
Label (javafxc.scene.control.Label)

Method	Description
Label()	Creates an empty label
Label(String text)	Creates a label with the specified text
Label(String text, Node graphics)	Creates a label with the specified text and graphic

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.HBox;
import javafx.scene.control.ContentDisplay;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.Ellipse;

public class LabelWithGraphic extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    ImageView pooh = new ImageView(new Image("pooh.jpg"));
    Label lb1 = new Label("Winnie the Pooh", pooh);
    lb1.setStyle("-fx-border-color: green; -fx-border-width: 1");
    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);
    lb3.setTextFill(Color.BLUE);

    Label lb4 = new Label("Ellipse", new Ellipse(50, 50, 50, 25));
    lb4.setContentDisplay(ContentDisplay.LEFT);
    lb4.setTextFill(Color.GREEN);

    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("COMP3021"));
    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);

    Scene scene = new Scene(pane, 800, 220); // Create a scene
    primaryStage.setTitle("LabelWithGraphic"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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

Output of the Last Example



ButtonBase and Button

(javafx.scene.control.ButtonBase & javafx.scene.control.Button)

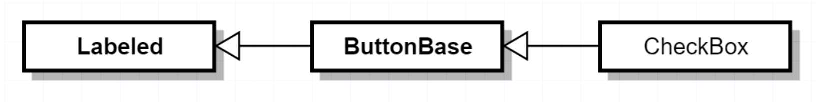
- ButtonBase class

Method	Description
ObjectProperty <EventHandler <ActionEvent>> onAction	Defines a handler for handling a button's action

(Defines a handler for handling a button's action)
The getter and setter methods for property values and a getter for property itself are provided in the class, but omitted in the table for brevity

- Button class

Method	Description
Button()	Creates an empty button
Button(String text)	Creates a button with the specified text
Button(String text, Node graphic)	Creates a button with the specified text and graphic



ButtonBase and Button

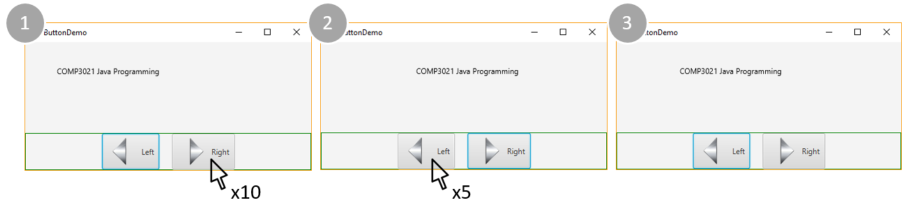
- A Button triggers an action event when clicked
- JavaFX provides regular buttons, toggle buttons, check box buttons, and radio buttons
- The common features of these buttons are defined in ButtonBase and Labeled classes

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.layout.HBox;
import javafx.scene.control.Button;
import javafx.scene.image.ImageView;
import javafx.scene.text.Text;
import javafx.geometry.Pos;

public class ButtonDemo extends Application {
  protected Text text = new Text(50, 50, "COMP3021 Java Programming");

  protected BorderPane getPane() {
    HBox paneForButtons = new HBox(20);
    Button btLeft = new Button("Left", new ImageView("left.png"));
    Button btRight = new Button("Right", new ImageView("right.png"));
```

Example

```java
    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 // Override the start method in the Application class
  public void start(Stage primaryStage) {
    Scene scene = new Scene(getPane(), 450, 200); // Create a scene
    primaryStage.setTitle("ButtonDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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

Output of the Last Example

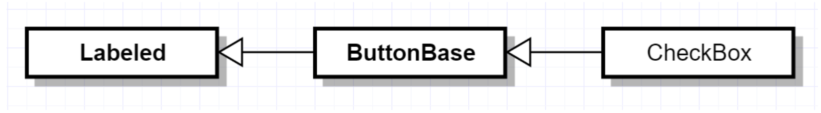

1
ButtonDemo
COMP3021 Java Programming
Left
Right
x10
2
ttonDemo
COMP3021 Java Programming
Left
Right
x5
3
tonDemo
COMP3021 Java Programming
Left
Right

CheckBox

(javafx.scene.control.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

Method	Description
BooleanProperty selected	Indicates whether this check box is checked
CheckBox()	Creates an empty check box
CheckBox(String text)	Creates a check box with the specified text

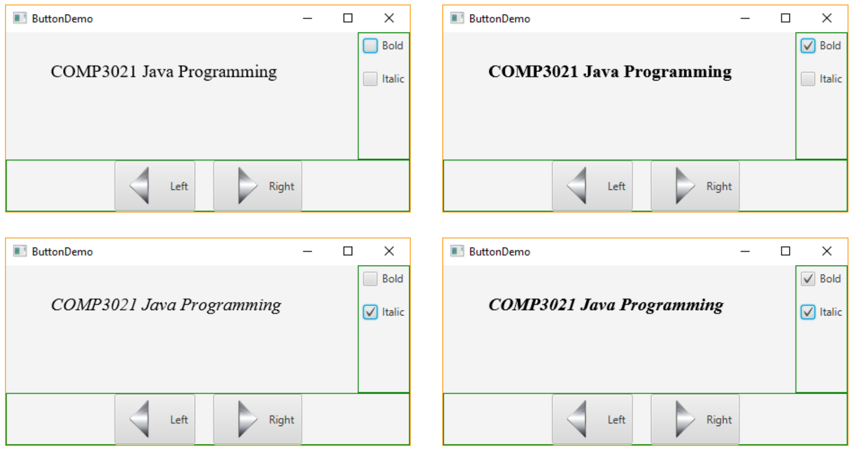


Example

```java
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.geometry.Insets;
import javafx.scene.control.CheckBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;

public class CheckBoxDemo extends ButtonDemo {
  @Override // Override the getPane() method in the super class
  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);
```

```java
    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);

    return pane; // Return a new pane
  }

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

Output of the Last Example



RadioButton (javafx.scene.control.RadioButton)

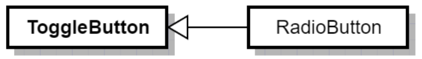
- Radio buttons enable us to choose an item from a group of choices

ToggleButton (javafx.scene.control.ToggleButton)

Method	Description
BooleanProperty selected	Indicates whether the button is selected
ObjectProperty<ToggleGroup> toggleGroup	Specifies the button group to which the button belongs
ToggleButton()	Creates an empty toggle button
ToggleButton(String text)	Creates a toggle button with the specified text
ToggleButton(String text, Node graphic)	Creates a toggle button with the specified text and graphic

RadioButton

Method	Description
RadioButton()	Creates an empty radio button
RadioButton(String text)	Creates a radio button with the specified text

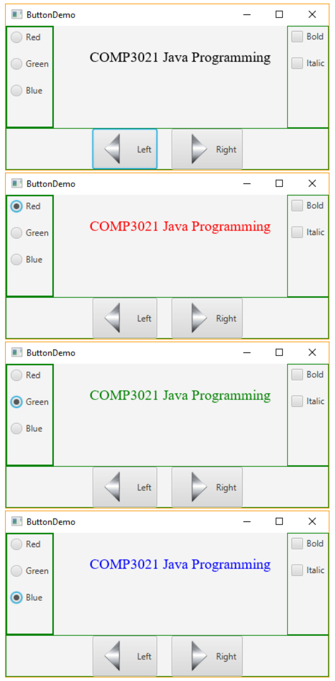


Example

```
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.geometry.Insets;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.paint.Color;

public class RadioButtonDemo extends CheckBoxDemo {
  @Override // Override the getPane() method in the super class
  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;
  }

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

content...



TextField (javafx.scene.control.TextField)

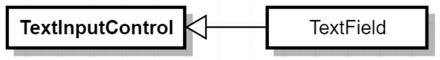
- A TextField can be used to enter or display a string

TextInputControl (javafx.scene.control.TextInputControl)

Method	Description
StringProperty text	The text content of this control
BooleanProperty editable	Indicates whether the text can be edited by the user

TextField

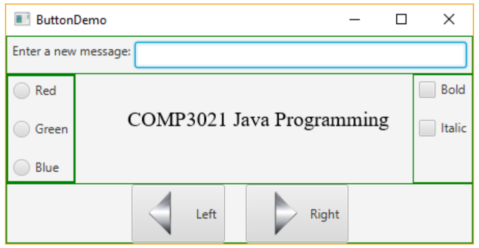
Instance Variable / Method	Description
StringProperty text	The text content of this control
BooleanProperty editable	Indicates whether the text can be edited by the user
ObjectProperty<Pos> alignment	Specifies how the text should be aligned in the text field
IntegerProperty prefColumnCount	Specifies the preferred number of columns in the text field
ObjectProperty <EventHandler < *ActionEvent* >> onAction	Specifies the handler for processing the action event on the text field
TextField()	Creates an empty text field
TextField(String text)	Creates a text field with the specified text

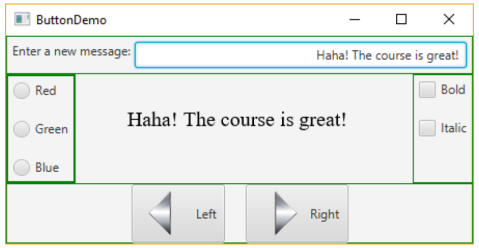


Example

```
import javafx.scene.layout.BorderPane;
import javafx.geometry.Insets;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.geometry.Pos;

public class TextFieldDemo extends RadioButtonDemo {
  @Override // Override the getPane() method in the super class
  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;
  }

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

Output of the Last Example





TextArea (javafx.scene.control.TextArea)

- A TextArea enables the user to enter multiple lines of text

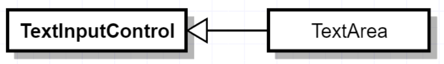
TextInputControl (javafx.scene.controlTextInputControl)

Instance Variable	Description
StringProperty text	The text content of this control
BooleanProperty editable	Indicates whether the text can be edited by the user

TextArea

Instance Variable / Method	Description
IntegerProperty prefColumnCount	Specifies the preferred number of text columns
IntegerProperty prefRowCount	Specifies the preferred number of text rows
BooleanProperty wrapText	Specifies whether the text is wrapped to the next line
TextArea()	Creates an empty text area
TextArea(String text)	Creates a text area with the specified text

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



Example

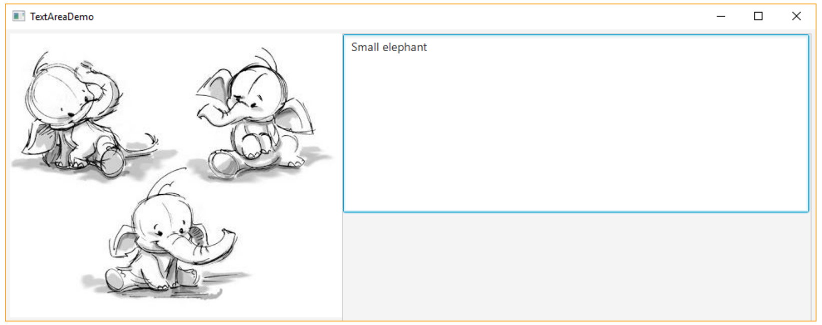
```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.image.ImageView;

public class TextAreaDemo extends Application {
  @Override // Override the start method in the Application class
  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("Elephant");
    String description = "Small elephant";
    descriptionPane.setImageView(new ImageView("elephant.jpg"));
    descriptionPane.setDescription(description);

    Scene scene = new Scene(descriptionPane, 975, 350); // Create a scene
    primaryStage.setTitle("TextAreaDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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

Example

```java
import javafx.scene.layout.BorderPane;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.ContentDisplay;
import javafx.geometry.Insets;
import javafx.scene.control.Label;
import javafx.scene.control.TextArea;
import javafx.scene.image.ImageView;
import javafx.scene.text.Font;

public class DescriptionPane extends BorderPane {
  private Label lblImageTitle = new Label();
  private TextArea taDescription = new TextArea();

  public DescriptionPane() {
    // Center the icon and text and place the text under the 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));
  }
```

Example

```
  /** 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); }
}
```


TextAreaDemo

Small elephant

ComboBox (javafx.scene.control.ComboBox<T>)

- A ComboBox, also known as a choice list or drop-down list, contains a list of items for which the user can choose

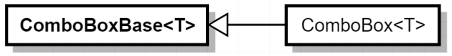
ComboBoxBase<T> (javafx.scene.control.ComboBoxBase<T>)

Instance Variable	Description
ObjectProperty<T> value	The value selected in the combo box
BooleanProperty editable	Specifies whether the combo box allows user input
ObjectProperty<ObjectProperty <ActionEvent>>	Specifies the handler for processing the action event

ComboBox<T>

Instance Variable / Method	Description
ObjectProperty <ObservableList<T>> items	The items in the combo box popup
IntegerProperty visibleRowCount	The maximum number of visible rows of the items in the combo box popup
ComboBox	Creates an empty combo box
ComboBox(ObservableList<T> items)	Creates a combo box with the specified items

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



Example

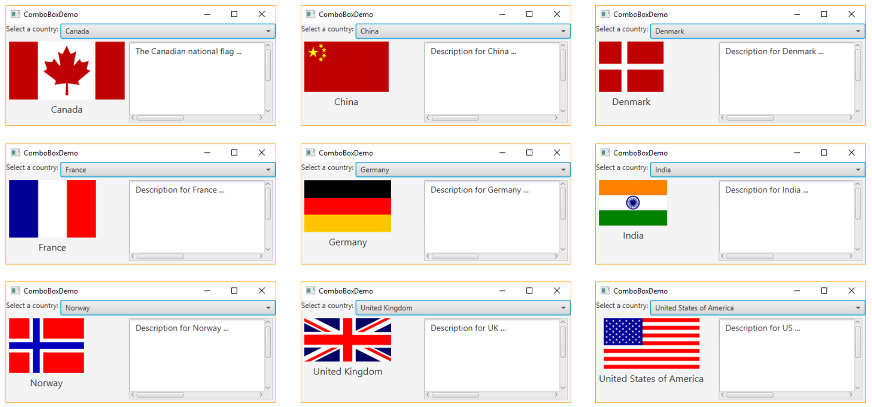
- This example lets users view an image and a description of a country's flag by selecting the country from a combo box

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.image.ImageView;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;

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 // Override the start method in the Application class
public void start(Stage primaryStage) {
  // Set text description
  flagDescription[0] = "The Canadian national flag ...";
  flagDescription[1] = "Description for China ... ";
  flagDescription[2] = "Description for Denmark ... ";
  flagDescription[3] = "Description for France ... ";
  flagDescription[4] = "Description for Germany ... ";
  flagDescription[5] = "Description for India ... ";
  flagDescription[6] = "Description for Norway ... ";
  flagDescription[7] = "Description for UK ... ";
  flagDescription[8] = "Description for US ... ";

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

  // 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())));

      Scene scene = new Scene(pane, 450, 170); // Create a scene
      primaryStage.setTitle("ComboBoxDemo"); // Set the stage title
      primaryStage.setScene(scene); // Place the scene in the stage
      primaryStage.show(); // Display the stage
    }

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

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

Output of the Last Example


ComboBoxDemo
Select a country: Canada
The Canadian national flag ...
Canada
Select a country: China
Description for China ...
China
Select a country: Denmark
Description for Denmark ...
Denmark
Select a country: France
Description for France ...
France
Select a country: Germany
Description for Germany ...
Germany
Select a country: India
Description for India ...
India
Select a country: Norway
Description for Norway ...
Norway
Select a country: United Kingdom
Description for UK ...
United Kingdom
Select a country: United States of America
Description for US ...
United States of America

ListView (javafx.scene.control.ListView<T>)

- A ListView 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

Instance Variable / Method	Description
ObjectProperty <ObservableList<T>> items	The items in the list view
BooleanProperty orientation	Indicates whether the items are displayed horizontally or vertically in the list view
ObjectProperty <MultipleSelectionMode<T>> selectionModel	Specifies how items are selected. The SelectionModel is also used to obtain the selected items
ListView()	Creates an empty list view
ListView(Observable<T>) items	Creates a list view with specified items

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

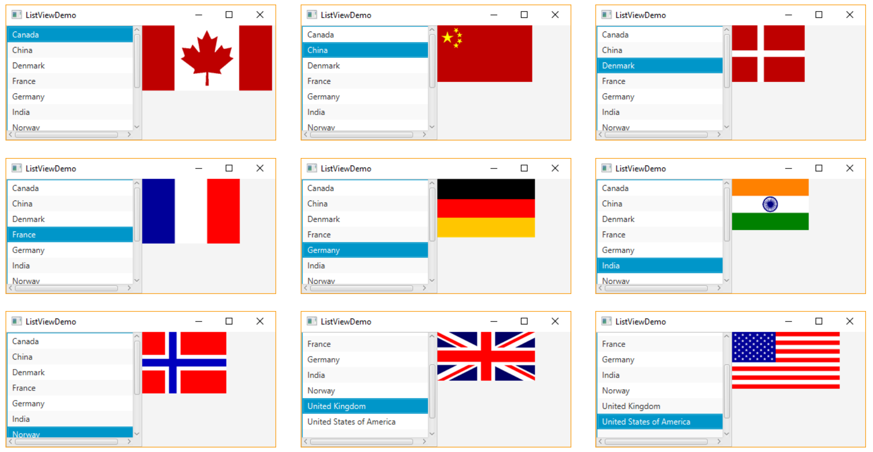
Example: Using ListView

- This example gives a program that lets users select countries in a list and display the flags of the selected countries in the labels

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.ListView;
import javafx.scene.image.ImageView;
import javafx.collections.FXCollections;
import javafx.scene.control.SelectionMode;

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 of 9 countries
  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 // Override the start method in the Application class
    public void start(Stage primaryStage) {
      ListView<String> lv = new ListView<>
        (FXCollections.observableArrayList(flagTitles));
      lv.setPrefSize(400, 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]);
          }
      });

      Scene scene = new Scene(pane, 450, 170); // Create a scene
      primaryStage.setTitle("ListViewDemo"); // Set the stage title
      primaryStage.setScene(scene); // Place the scene in the stage
      primaryStage.show(); // Display the stage
    }

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

Output of the Last Example



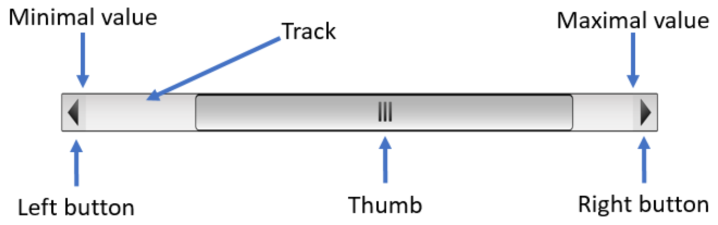
ScrollBar (javafx.scene.control.ScrollBar)

- A ScrollBar enables the user to select from a range of values
- The scrollbar appears in two styles: horizontal and vertical

Instance Variable / Method	Description
DoubleProperty blockIncrement	The amount to adjust the scroll bar if the track of the bar is clicked (default: 10)
DoubleProperty max	The maximum value represented by this scroll bar (default 100)
DoubleProperty min	The minimum value represented by this scroll bar (default 0)
DoubleProperty unitIncrement	The amount to adjust the scroll bar when the increment() and decrement() methods are called (default: 1)
DoubleProperty value	Current value of the scroll bar (default: 0)
DoubleProperty visibleAmount	The width of the scroll bar (default: 15)
ObjectProperty<Orientation> orientation	Specifies the orientation of the scroll bar (default: HORIZONAL)
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

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

Scroll Bar Properties



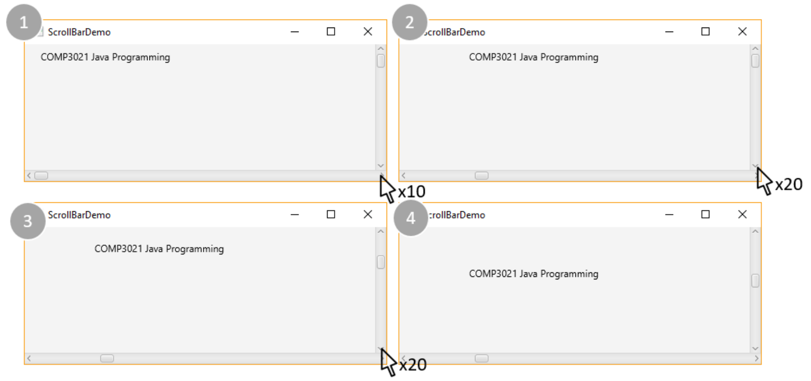
Example

- 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

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.control.ScrollBar;
import javafx.scene.text.Text;
import javafx.geometry.Orientation;

public class ScrollBarDemo extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    Text text = new Text(20, 20, "COMP3021 Java 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);
```

```java
    // Create a border 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()));

    Scene scene = new Scene(pane, 450, 170); // Create a scene
    primaryStage.setTitle("ScrollBarDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

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

Output of the Last Example


1
ScrollBarDemo
COMP3021 Java Programming
2
ScrollBarDemo
COMP3021 Java Programming
x10
x20
3
ScrollBarDemo
COMP3021 Java Programming
4
ScrollBarDemo
COMP3021 Java Programming
x20

Slider (javafx.scene.control.Slider)

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

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

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

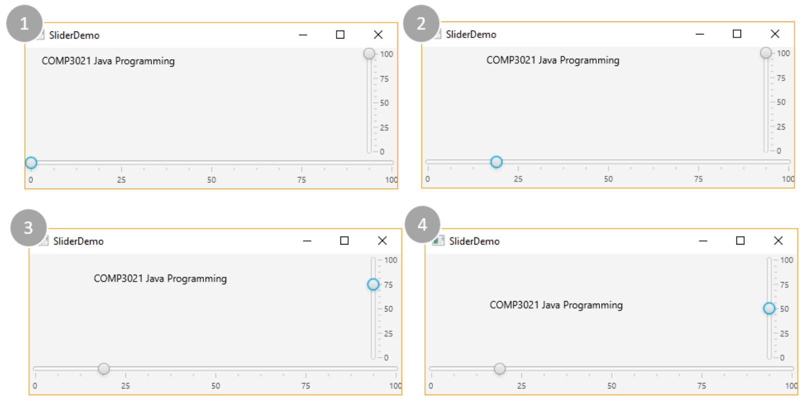
Example

```java
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.control.Slider;
import javafx.scene.text.Text;
import javafx.geometry.Orientation;

public class SliderDemo extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    Text text = new Text(20, 20, "COMP3021 Java Programming");

    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);

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

Example

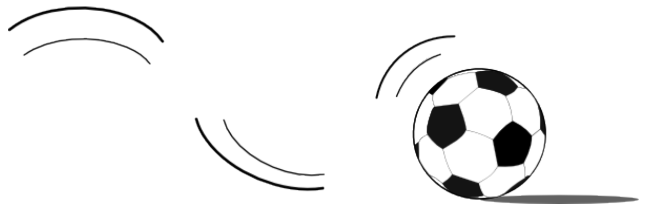
```java
    // Create a border pane to hold text and scroll bars
    BorderPane pane = new BorderPane();
    pane.setCenter(paneForText);
    pane.setBottom(slHorizontal);
    pane.setRight(slVertical);

    slHorizontal.valueProperty().addListener(ov ->
      text.setX(slHorizontal.getValue() * paneForText.getWidth() /
        slHorizontal.getMax()));

    slVertical.valueProperty().addListener(ov ->
      text.setY((slVertical.getMax() - slVertical.getValue())
        * paneForText.getHeight() / slVertical.getMax()));

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

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

Output of the Last Example



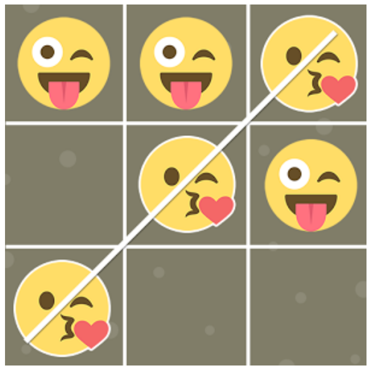
Case Study: Bouncing Ball

- BallPane.java
  `http://www.cs.armstrong.edu/liang/intro10e/html/BallPane.html`
- BounceBallSlider.java
  `http://www.cs.armstrong.edu/liang/intro10e/html/BounceBallSlider.html`



Case Study: Tic Tac Toe

- TicTacToe.java
  `http://www.cs.armstrong.edu/liang/intro10e/html/TicTacToe.html`



Media (javafx.scene.media.Media)

- We 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

Instance Variable / Method	Description
ReadOnlyProperty<Duration> duration	The duration in seconds of the source media
ReadOnlyIntegerProperty width	The width in pixels of the source video
ReadOnlyIntegerProperty height	The height in pixels of the source video
Media(String source)	Creates a Media from a URL source

MediaPlayer (javafx.scene.media.MediaPlayer)

- The MediaPlayer class plays and controls the media with properties such as autoPlay, currentCount, cycleCount, mute, volume, and totalDuration

Instance Variable / Method	Description
BooleanProperty autoPlay	Specifies whether the playing should start automatically
ReadOnlyIntegerProperty currentCount	The number of completed playback cycles
IntegerProperty cycleCount	Specifies the number of time the media will be played
BooleanProperty mute	Specifies whether the audio is muted
DoubleProperty volume	The volume for the audio
ReadOnlyObjectProperty<Duration> totalDuration	The amount of time to play the media from start to finish
MediaPlayer(Media media)	Creates a player for a specified media
void play()	Plays the media
void pause()	Pauses the media
void seek()	Seeks the player to a new playback time

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

MediaView (javafx.scene.media.MediaView)

- 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

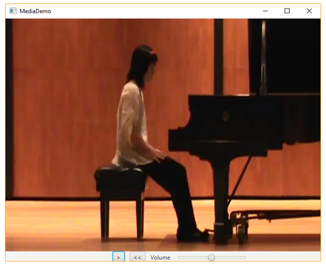
Instance Variable / Method	Description
DoubleProperty x	Specifies the current x-coordinate of the media view
DoubleProperty y	Specifies the current y-coordinate of the media view
ObjectProperty <MediaPlayer> mediaPlayer	Specifies a media player for the media view
DoubleProperty fitWidth	Specifies the width of the view for the media to fit
DoubleProperty fitHeight	Specifies the height of the view for the media to fit
MediaView()	Creates an empty media view
MediaView(MediaPlayer mediaPlayer)	Creates a media view with the specified media player

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

Example

- This example displays a video in a view
- We can use the play / pause button to play or pause the video and use the rewind button to restart the video, and use the slider to control the volume of the audio

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Region;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.geometry.Pos;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.util.Duration;
```



```
public class MediaDemo extends Application {
  private static final String MEDIA_URL =
    "https://liveexample.pearsoncmg.com/common/sample.mp4";
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    Media media = new Media(MEDIA_URL);
    MediaPlayer mediaPlayer = new MediaPlayer(media);
    MediaView mediaView = new MediaView(mediaPlayer);
```

```
    Button playButton = new Button(">");
    playButton.setOnAction(e -> {
      if (playButton.getText().equals(">")) {
        mediaPlayer.play();
        playButton.setText("||");
      } else {
        mediaPlayer.pause();
        playButton.setText(">");
      }
    });

    Button rewindButton = new Button("<<");
    rewindButton.setOnAction(e -> mediaPlayer.seek(Duration.ZERO));
    Slider slVolume = new Slider();
    slVolume.setPrefWidth(150);
    slVolume.setMaxWidth(Region.USE_PREF_SIZE);
    slVolume.setMinWidth(30);
    slVolume.setValue(50);
    mediaPlayer.volumeProperty().bind(slVolume.valueProperty().divide(100));
    HBox hBox = new HBox(10);
    hBox.setAlignment(Pos.CENTER);
    hBox.getChildren().addAll(playButton, rewindButton,
      new Label("Volume"), slVolume);
    BorderPane pane = new BorderPane();
    pane.setCenter(mediaView);
    pane.setBottom(hBox);
    Scene scene = new Scene(pane, 650, 500); // Create a scene
    primaryStage.setTitle("MediaDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }
  public static void main(String[] args) { launch(args); }
}
```

WebView (javafx.scene.web.WebView)

- The WebView class is also a subclass of Node that manages a WebEngine and displays its content

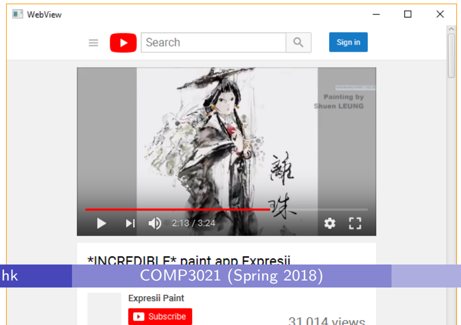
Instance Variable / Method	Description
ReadOnlyDoubleProperty width	Width of this WebView
ReadOnlyDoubleProperty height	Height of this WebView
DoubleProperty prefWidth	Specifies the preferred width of the web view
DoubleProperty prefHeight	Specifies the preferred height of the web view
WebView()	Creates a WebView object
WebEngine getEngine()	Returns the WebEngine object
void setPrefWidth(double value)	Sets preferred width of the web view
void setPrefHeight(double value)	Sets preferred height of the web view

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

Example

- This example displays a Youtube video in a web view

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.web.WebView;

public class WebViewDemo extends Application {
  @Override public void start(Stage primaryStage) {
    WebView webview = new WebView();
    webview.getEngine().load("https://youtu.be/Tu305qvVVHo");
    webview.setPrefSize(640, 480);
    primaryStage.setTitle("WebView");
    primaryStage.setScene(new Scene(webview));
    primaryStage.show();
  }
  public static void main(String[] args) { launch(args); }
}
```



That's all!

Any questions?