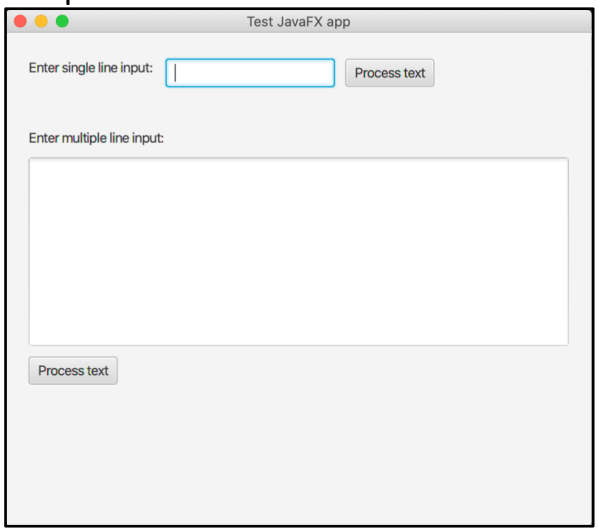
Setting up layout skins with CSS

- Many options are available for setting up visual styles of your chosen layout
  - *setStyle()* method is used for setting up style parameters
  - Cascading style sheets (CSS) language is used for setting up these parameters
- A pretty good tutorial, as a starting point:
  - https://docs.oracle.com/javafx/2/css_tutorial/jfxpub-css_tutorial.htm

```
myLayout.setStyle("-fx-padding: 10;" +
        "-fx-border-style: solid inside;" +
        "-fx-border-width: 2;" +
        "-fx-border-insets: 5;" +
        "-fx-border-radius: 5;" +
        "-fx-border-color: blue;" +
        "-fx-background-color: #F2F5A9;");
```

Example: using text fields with nested layouts in JavaFX app

Output:



Setup nodes for single line input

Setup nodes for multiple line input

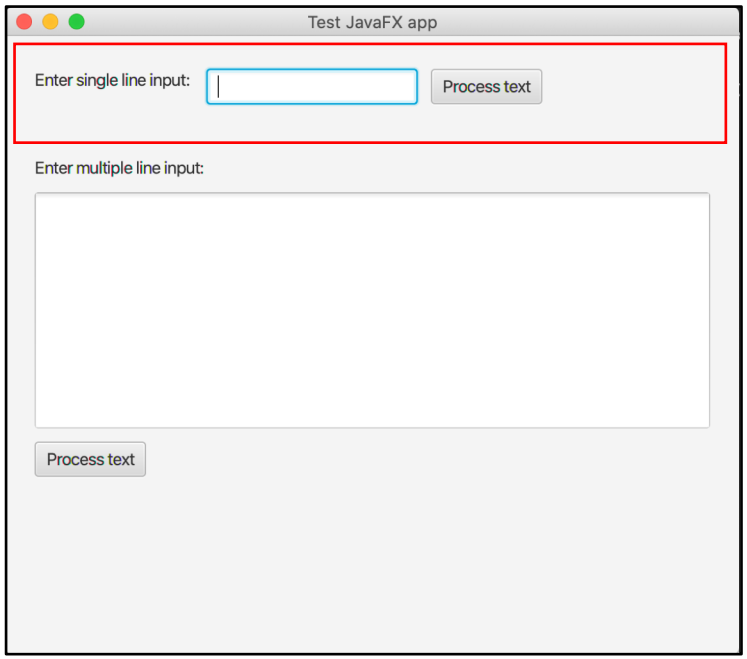
Setup the screen GUI (scene and stage)

Setup event handling

Ensure all needed imports

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        // setup the panel with single line input:
        Label lblInputSingle = new Label("Enter single line input: ");
        TextField txtSingleLine = new TextField();
        Button btnProcessSingle = new Button("Process text");

        HBox singleLineLayout = new HBox();
        singleLineLayout.setPadding(new Insets(15, 12, 15, 12));
        singleLineLayout.setSpacing(10);
        singleLineLayout.getChildren().addAll(lblInputSingle, txtSingleLine, btnProcessSingle);

        // setup the panel with multiple line input:
        Label lblInputMultiple = new Label("Enter multiple line input: ");
        TextArea txtMultipleLine = new TextArea();
        Button btnProcessMultiple = new Button("Process text");

        VBox multipleLineLayout = new VBox();
        multipleLineLayout.setPadding(new Insets(15, 12, 15, 12));
        multipleLineLayout.setSpacing(10);
        multipleLineLayout.getChildren().addAll(lblInputMultiple, txtMultipleLine, btnProcessMultiple);

        // setup the gui screen:
        Label lblOutput = new Label();
        lblOutput.setMinWidth(150);
        lblOutput.setMinHeight(100);

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(singleLineLayout, multipleLineLayout, lblOutput);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.sizeToScene();
        stage.setTitle("Test JavaFX app");
        stage.show();

        // setup event handling
        EventHandler<MouseEvent> processTextEventHandler = new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                String displayText;
                Object source = (Object) event.getSource();

                if((Button)source == btnProcessSingle){
                    displayText = txtSingleLine.getText();
                } else{
                    displayText = txtMultipleLine.getText();
                }

                lblOutput.setText(displayText);
            }
        };
        btnProcessSingle.addEventHandler(MouseEvent.MOUSE_CLICKED, processTextEventHandler);
        btnProcessMultiple.addEventHandler(MouseEvent.MOUSE_CLICKED, processTextEventHandler);
    }

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

Example: using text fields with nested layouts in JavaFX app (cont'd)



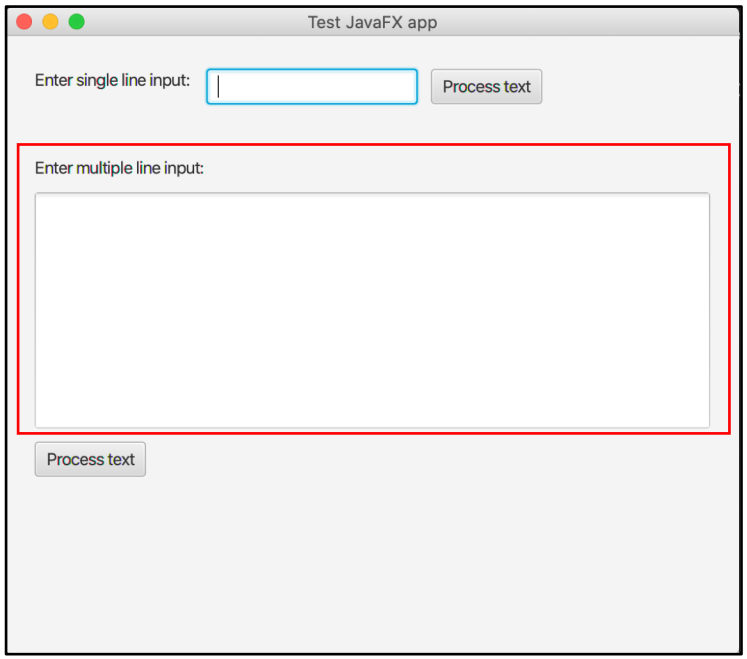
...

```
// setup the panel with single line input:
Label lblInputSingle = new Label("Enter single line input: ");
TextField txtSingleLine = new TextField();
Button btnProcessSingle = new Button("Process text");

HBox singleLineLayout = new HBox();
singleLineLayout.setPadding(new Insets(15, 12, 15, 12));
singleLineLayout.setSpacing(10);
singleLineLayout.getChildren().addAll(lblInputSingle, txtSingleLine, btnProcessSingle);
```

...

Setup single line input panel as HBox layout

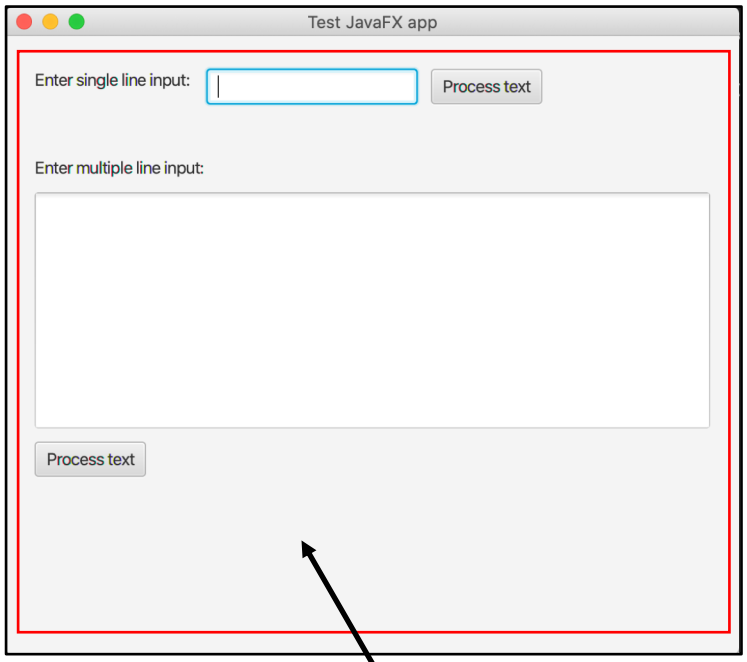
Example: using text fields with nested layouts in JavaFX app (cont'd)




```
...
// setup the panel with multiple line input:
Label lblInputMultiple = new Label("Enter multiple line input: ");
TextArea txtMultipleLine = new TextArea();
Button btnProcessMultiple = new Button("Process text");

VBox multipleLineLayout = new VBox();
multipleLineLayout.setPadding(new Insets(15, 12, 15, 12));
multipleLineLayout.setSpacing(10);
multipleLineLayout.getChildren().addAll(lblInputMultiple, txtMultipleLine, btnProcessMultiple);
...
```

Setup multiple line input panel as VBox layout

Example: using text fields with nested layouts in JavaFX app (cont'd)



Setup an empty label for output

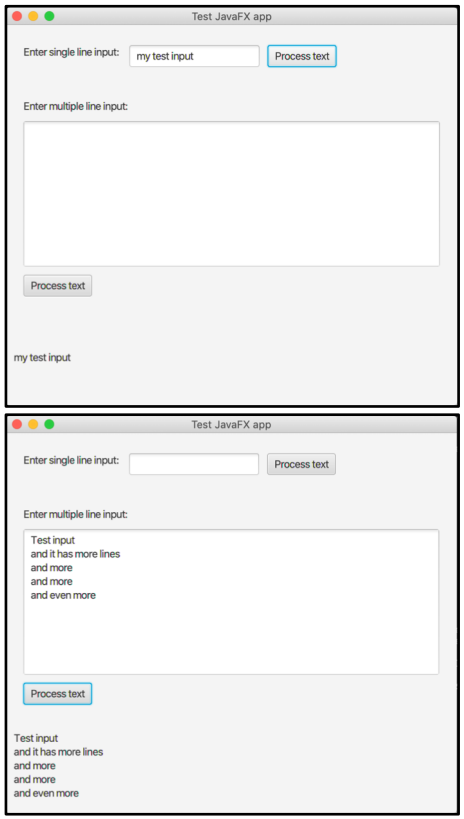
Nested layouts

```
...
//setup the gui screen:
Label lblOutput = new Label();
lblOutput.setMinWidth(150);
lblOutput.setMinHeight(100);

VBox root = new VBox();
root.setPadding(new Insets(10, 10, 10, 10));
root.setSpacing(10);
root.getChildren().addAll(singleLineLayout, multipleLineLayout, lblOutput);

Scene scene = new Scene(root);
stage.setScene(scene);

stage.setX(200);
stage.setY(300);
stage.sizeToScene();
stage.setTitle("Test JavaFX app");
stage.show();
...
```

automatically size the window to fit the scene

Setup the scene as VBox layout

Example: using text fields with nested layouts in JavaFX app (cont'd)



We could setup different event handlers for each button

Test which button is the source of the event

Add the mouse click handler to both buttons

```
...
// setup event handling
EventHandler<MouseEvent> processTextEventHandler = new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event) {
        String displayText;
        Object source = (Object) event.getSource();

        if((Button)source == btnProcessSingle){
            displayText = txtSingleLine.getText();
        } else{
            displayText = txtMultipleLine.getText();
        }

        lblOutput.setText(displayText);
    }
};
btnProcessSingle.addEventHandler(MouseEvent.MOUSE_CLICKED, processTextEventHandler);
btnProcessMultiple.addEventHandler(MouseEvent.MOUSE_CLICKED, processTextEventHandler);
...
```

Example: button shadow effects

Mouse over the button:

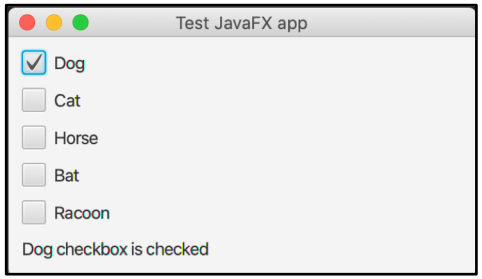
Mouse is not over the button:

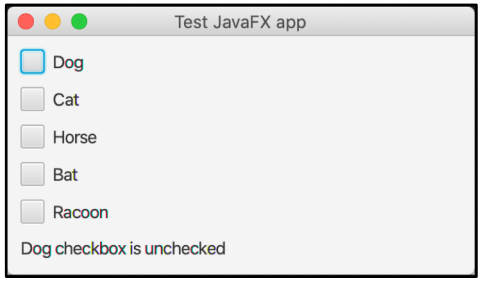
New shadow

Anonymous event handlers

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.effect.DropShadow;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Button btnButton1 = new Button("Test button");
        DropShadow shadow = new DropShadow();

        btnButton1.addEventHandler(MouseEvent.MOUSE_ENTERED,
        new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                btnButton1.setEffect(shadow);
            }
        });

        btnButton1.addEventHandler(MouseEvent.MOUSE_ENTERED,
        new EventHandler<MouseEvent>() {
            @Override
            public void handle(MouseEvent event) {
                btnButton1.setEffect(null);
            }
        });

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(btnButton1);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

Example: using checkboxes






```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.beans.value.*;
```

Add checkboxes, note that chbxBat allows intermediate state

Setup checkbox event handler

Simply update the label based on the state of the last clicked checkbox

Add the handler to all checkboxes

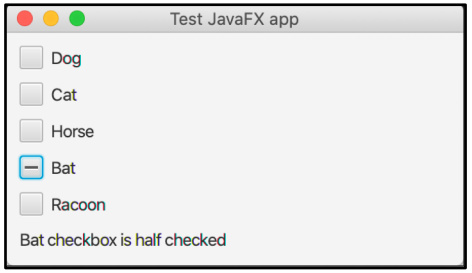
```
public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        CheckBox chbxDog = new CheckBox("Dog");
        CheckBox chbxCat = new CheckBox("Cat");
        CheckBox chbxHorse = new CheckBox("Horse");
        CheckBox chbxBat = new CheckBox("Bat");
        chbxBat.setAllowIndeterminate(true);
        CheckBox chbxRacoon = new CheckBox("Racoon");
        Label lblLatestAction = new Label();

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(chbxDog,chbxCat,chbxHorse,chbxBat,chbxRacoon,lblLatestAction);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();

        EventHandler<ActionEvent> checkBoxEventHandler = new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event){
                String state;
                Object source = (Object) event.getSource();

                if(((CheckBox)source).isSelected()){
                    state = "checked";
                } else{
                    state = "unchecked";
                }

                if((CheckBox)source == chbxDog){
                    lblLatestAction.setText("Dog checkbox is "+state);
                } else if((CheckBox)source == chbxCat){
                    lblLatestAction.setText("Cat checkbox is "+state);
                } else if((CheckBox)source == chbxHorse){
                    lblLatestAction.setText("Horse checkbox is "+state);
                } else if((CheckBox)source == chbxBat){
                    if(chbxBat.isIndeterminate()){
                        lblLatestAction.setText("Bat checkbox is half checked");
                    }else if(chbxBat.isSelected()){
                        lblLatestAction.setText("Bat checkbox is checked");
                    }else{
                        lblLatestAction.setText("Bat checkbox is unchecked");
                    }
                } else if((CheckBox)source == chbxRacoon){
                    lblLatestAction.setText("Racoon checkbox is "+state);
                }
            }
        };

        chbxDog.setOnAction(checkBoxEventHandler);
        chbxCat.setOnAction(checkBoxEventHandler);
        chbxHorse.setOnAction(checkBoxEventHandler);
        chbxBat.setOnAction(checkBoxEventHandler);
        chbxRacoon.setOnAction(checkBoxEventHandler);
    }

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

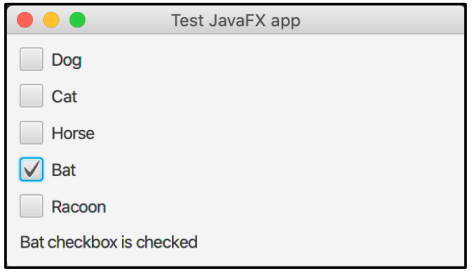
Example: using checkboxes (cont'd)

...

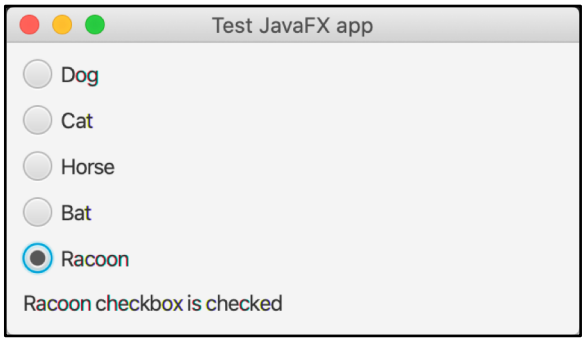
```
EventHandler<ActionEvent> checkBoxEventHandler = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event){
        String state;
        Object source = (Object) event.getSource();

        if(((CheckBox)source).isSelected()){
            state = "checked";
        } else{
            state = "unchecked";
        }

        if((CheckBox)source == chbxDog){
            lblLatestAction.setText("Dog checkbox is "+state);
        } else if((CheckBox)source == chbxCat){
            lblLatestAction.setText("Cat checkbox is "+state);
        } else if((CheckBox)source == chbxHorse){
            lblLatestAction.setText("Horse checkbox is "+state);
        } else if((CheckBox)source == chbxBat){
            if(chbxBat.isIndeterminate()){
                lblLatestAction.setText("Bat checkbox is half checked");
            }else if(chbxBat.isSelected()){
                lblLatestAction.setText("Bat checkbox is checked");
            }else{
                lblLatestAction.setText("Bat checkbox is unchecked");
            }
        } else if((CheckBox)source == chbxRacoon){
            lblLatestAction.setText("Racoon checkbox is "+state);
        }
    }
};
```

...





Example: using radio buttons



Have to create a toggle group to which add all grouped radio buttons

```java
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.beans.value.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        RadioButton rdDog = new RadioButton("Dog");
        RadioButton rdCat = new RadioButton("Cat");
        RadioButton rdHorse = new RadioButton("Horse");
        RadioButton rdBat = new RadioButton("Bat");
        RadioButton rdRacoon = new RadioButton("Racoon");
        ToggleGroup rbGroup = new ToggleGroup();
        rbGroup.getToggles().addAll(rdDog,rdCat,rdHorse,rdBat,rdRacoon);

        Label lblLatestAction = new Label();

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(rdDog,rdCat,rdHorse,rdBat,rdRacoon,lblLatestAction);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();

        EventHandler<ActionEvent> rdEventHandler = new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event){
                String state;
                Object source = (Object) event.getSource();

                if(((RadioButton)source).isSelected()){
                    state = "checked";
                } else{
                    state = "unchecked";
                }

                if((RadioButton)source == rdDog){
                    lblLatestAction.setText("Dog checkbox is "+state);
                } else if((RadioButton)source == rdCat){
                    lblLatestAction.setText("Cat checkbox is "+state);
                } else if((RadioButton)source == rdHorse){
                    lblLatestAction.setText("Horse checkbox is "+state);
                } else if((RadioButton)source == rdBat){
                    lblLatestAction.setText("Bat checkbox is "+state);
                } else if((RadioButton)source == rdRacoon){
                    lblLatestAction.setText("Racoon checkbox is "+state);
                }
            }
        };

        rdDog.setOnAction(rdEventHandler);
        rdCat.setOnAction(rdEventHandler);
        rdHorse.setOnAction(rdEventHandler);
        rdBat.setOnAction(rdEventHandler);
        rdRacoon.setOnAction(rdEventHandler);
    }

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

Example: using choice box

```java
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Label lblAnimals = new Label("Animals:");
        ChoiceBox<String> animals = new ChoiceBox<>();
        animals.getItems().addAll("Dog", "Cat", "Horse", "Bat", "Racoon");
        Label lblSelection = new Label();

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(lblAnimals,animals,lblSelection);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();

        EventHandler<ActionEvent> lstEventHandler = new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent event){
                lblSelection.setText(animals.getValue());
            }
        };

        animals.setOnAction(lstEventHandler);
    }

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

Example: using menus

This label will change the text as a part of event handling for menu items

Create menu objects

Create menu bar and add menus to it

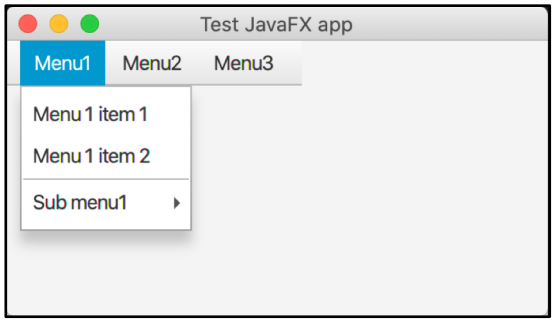
AnchorPane will allow us to anchor lblSelection at the bottom of the window

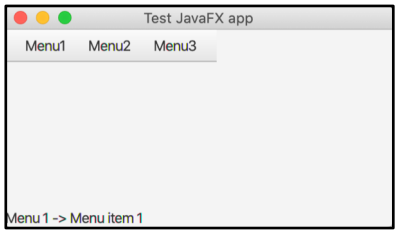
Setup a scene and stage

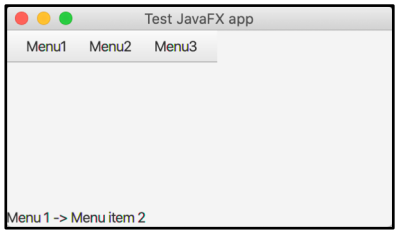
Setup stage options and show it

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.image.ImageView;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Label lblSelection = new Label();

        // menu 1 elements:
        Menu menu1 = new Menu("Menu1");
        ...

        // menu 2 elements:
        Menu menu2 = new Menu("Menu2");
        ...

        // menu 3 elements:
        Menu menu3 = new Menu("Menu3");
        ...

        MenuBar menuBar = new MenuBar();
        menuBar.getMenus().add(menu1);
        menuBar.getMenus().add(menu2);
        menuBar.getMenus().add(menu3);

        AnchorPane root = new AnchorPane();
        root.getChildren().addAll(menuBar, lblSelection);
        root.setBottomAnchor(lblSelection, 0.0);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

Example: using menus (cont'd)







Event handlers for the menu items

Don't forget to add menu items and separators to the appropriate menu object

```
// menu 1 elements:
Menu menu1 = new Menu("Menu1");

MenuItem menuItem1 = new MenuItem("Menu 1 item 1");
menu1.getItems().add(menuItem1);
menuItem1.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Menu item 1");
    }
});

MenuItem menuItem2 = new MenuItem("Menu 1 item 2");
menu1.getItems().add(menuItem2);
menuItem2.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Menu item 2");
    }
});

SeparatorMenuItem menuSeparator = new SeparatorMenuItem();
menu1.getItems().add(menuSeparator);

Menu submenu1 = new Menu("Sub menu1");
menu1.getItems().add(submenu1);

MenuItem menuItem3 = new MenuItem("Menu 1 submenu 1 item 1");
submenu1.getItems().add(menuItem3);
menuItem3.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Sub menu 1 -> Menu item 1");
    }
});

Button btnMenuItem4 = new Button("Button menu item");
CustomMenuItem menuItem4 = new CustomMenuItem();
menuItem4.setContent(btnMenuItem4);
menuItem4.setHideOnClick(false);
submenu1.getItems().add(menuItem4);
btnMenuItem4.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Sub menu 1 -> Menu item 2 (button)");
    }
});

TextField txtMenuItem5 = new TextField();
CustomMenuItem menuItem5 = new CustomMenuItem();
menuItem5.setContent(txtMenuItem5);
menuItem5.setHideOnClick(false);
submenu1.getItems().add(menuItem5);
menuItem5.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Sub menu 1 -> Menu item 3 ("+txtMenuItem5.getText()+")");
    }
});
```

Create a menu

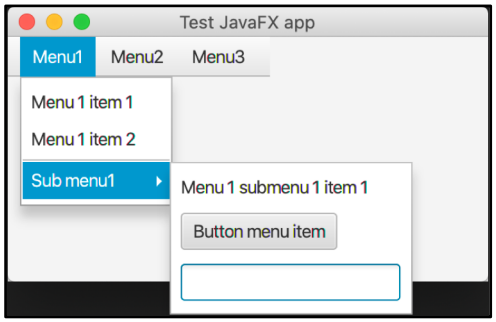
Create menu item 1

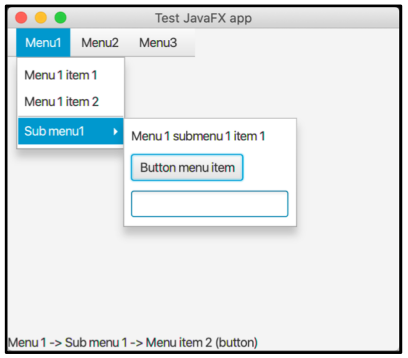
Create menu item 2

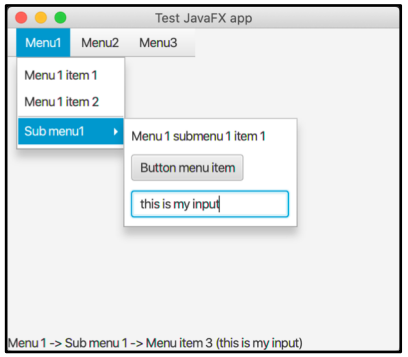
Create separator

Create submenu object

Example: using menus (cont'd)







...

```
// menu 1 elements:
Menu menu1 = new Menu("Menu1");

MenuItem menuItem1 = new MenuItem("Menu 1 item 1");
menu1.getItems().add(menuItem1);
menuItem1.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Menu item 1");
    }
});

MenuItem menuItem2 = new MenuItem("Menu 1 item 2");
menu1.getItems().add(menuItem2);
menuItem2.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Menu item 2");
    }
});

SeparatorMenuItem menuSeparator = new SeparatorMenuItem();
menu1.getItems().add(menuSeparator);

Menu submenu1 = new Menu("Sub menu1");
menu1.getItems().add(submenu1);

MenuItem menuItem3 = new MenuItem("Menu 1 submenu 1 item 1");
submenu1.getItems().add(menuItem3);
menuItem3.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Sub menu 1 -> Menu item 1");
    }
});

Button btnMenuItem4 = new Button("Button menu item");
CustomMenuItem menuItem4 = new CustomMenuItem();
menuItem4.setContent(btnMenuItem4);
menuItem4.setHideOnClick(false);
submenu1.getItems().add(menuItem4);
btnMenuItem4.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Sub menu 1 -> Menu item 2 (button)");
    }
});

TextField txtMenuItem5 = new TextField();
CustomMenuItem menuItem5 = new CustomMenuItem();
menuItem5.setContent(txtMenuItem5);
menuItem5.setHideOnClick(false);
submenu1.getItems().add(menuItem5);
menuItem5.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 1 -> Sub menu 1 -> Menu item 3 ("+txtMenuItem5.getText()+")");
    }
});
```

Regular menu item within submenu

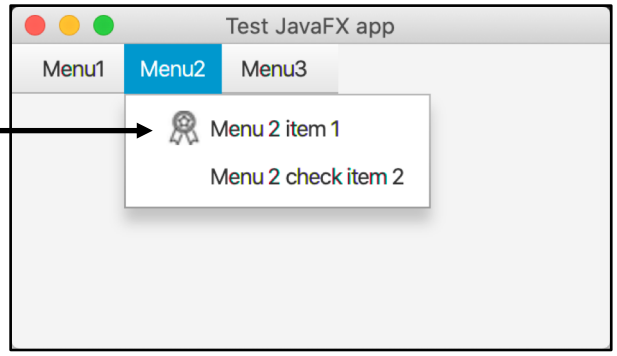
Create a menu item that is a button

Create a menu item that is a text field

...

Example: using menus (cont'd)

Menu item with an image

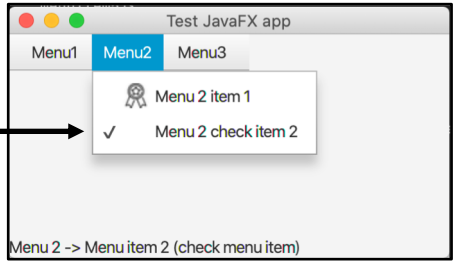


...

```
// menu 2 elements:
Menu menu2 = new Menu("Menu2");

MenuItem menuItem6 = new MenuItem("Menu 2 item 1");
menuItem6.setGraphic(new ImageView("./my_app_icon7.png"));
menu2.getItems().add(menuItem6);
menuItem6.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 2 -> Menu item 1");
    }
});

CheckMenuItem menuItem7 = new CheckMenuItem("Menu 2 check item 2");
menu2.getItems().add(menuItem7);
menuItem7.setOnAction(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        lblSelection.setText("Menu 2 -> Menu item 2 (check menu item)");
    }
});
```

...

Check menu item



Example: using menus (cont'd)

...

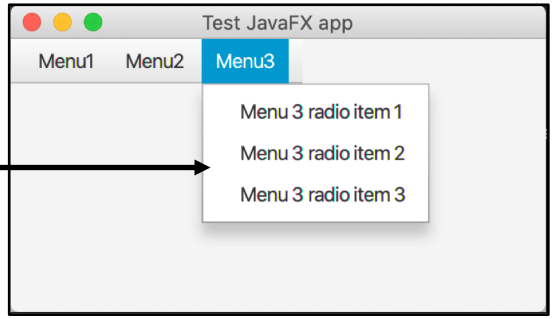
Don't forget to use toggle group

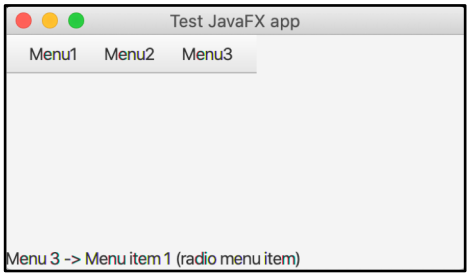
```
// menu 3 elements:
Menu menu3 = new Menu("Menu3");

RadioMenuItem menuItem8 = new RadioMenuItem("Menu 3 radio item 1");
RadioMenuItem menuItem9 = new RadioMenuItem("Menu 3 radio item 2");
RadioMenuItem menuItem10 = new RadioMenuItem("Menu 3 radio item 3");

ToggleGroup toggleGroup = new ToggleGroup();
toggleGroup.getToggles().add(menuItem8);
toggleGroup.getToggles().add(menuItem9);
toggleGroup.getToggles().add(menuItem10);

menu3.getItems().add(menuItem8);
menu3.getItems().add(menuItem9);
menu3.getItems().add(menuItem10);

EventHandler<ActionEvent> rdmenuEventHandler = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event){
        RadioMenuItem source = (RadioMenuItem)event.getSource();
        if(source == menuItem8){
            lblSelection.setText("Menu 3 -> Menu item 1 (radio menu item)");
        } else if(source == menuItem9){
            lblSelection.setText("Menu 3 -> Menu item 2 (radio menu item)");
        } else if(source == menuItem10){
            lblSelection.setText("Menu 3 -> Menu item 3 (radio menu item)");
        }
    }
};
menuItem8.setOnAction(rdmenuEventHandler);
menuItem9.setOnAction(rdmenuEventHandler);
menuItem10.setOnAction(rdmenuEventHandler);
```

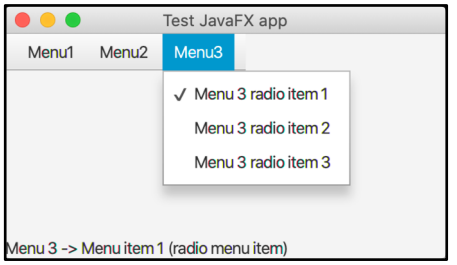
One event handler for all radio menu items

...

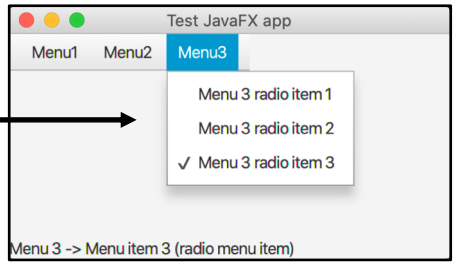
These are radio menu items







Only one menu item can have a checkmark next to it



Example: using menus (cont'd)

Additional options for event handling:

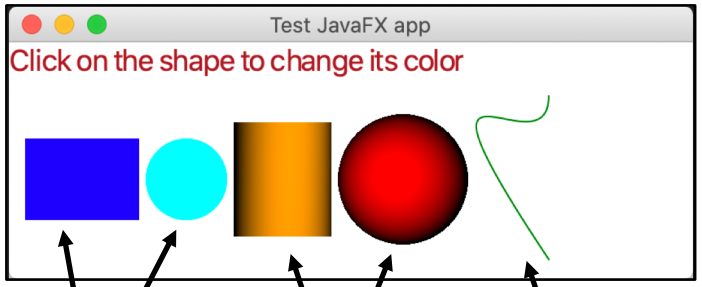
```
menu.setOnShowing(new EventHandler<ActionEvent>()
{
    @Override public void handle(ActionEvent event)
    {
        ...
    }
});
menu.setOnShown(...);
menu.setOnHiding(...);
menu.setOnHidden(...);
```

Example: working with shapes in JavaFX

Make sure all the imports are there

https://docs.oracle.com/javase/8/javafx/api/javafx/scene/paint/Color.html

Output:



2-D shapes 3-D shapes Lines and curves

Create shapes, set parameters, implement functionality

Use FlowPane to arrange the shapes

Use BorderPane to arrange text and FlowPane

Setup the scene, stage, and display

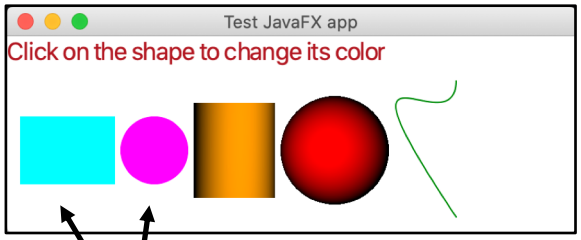
```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Rectangle;
import javafx.scene.shape.Cylinder;
import javafx.scene.shape.Sphere;
import javafx.scene.shape.CubicCurveTo;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;
import javafx.scene.paint.Color;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.text.*;
import java.util.*;

public class Test extends Application {
    private ArrayList<Color> palette = new ArrayList<Color>(Arrays.asList(Color.PINK,
            Color.MAGENTA, Color.CORAL, Color.BLACK, Color.CYAN, Color.BROWN,
            Color.BLUE, Color.ORANGE, Color.GREEN, Color.YELLOW, Color.LAVENDER));

    @Override
    public void start(Stage stage)
    {
        // working with 2-D shapes:
        ...

        // working with 3-D shapes:
        ...

        // working with lines and curves
        ...

        // adding text to the scene:
        ...

        FlowPane center = new FlowPane();
        center.setPadding(new Insets(10, 10, 10,10));
        center.setVgap(4);
        center.setHgap(4);
        center.getChildren().addAll(rectangle, circle, cylinder, sphere, path);

        BorderPane root = new BorderPane();
        root.setTop(message);
        root.setCenter(center);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(150);
        stage.setMinWidth(200);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

Example: working with shapes in JavaFX (cont'd)

Add circle shape and setup visual options and event handling

```
...
// working with 2-D shapes:
Circle circle = new Circle();
circle.setRadius(25.0f);
circle.setFill(Color.CYAN);
circle.addEventHandler(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event){
        circle.setFill(palette.get(new Random().nextInt(palette.size())));
    }
});

Rectangle rectangle = new Rectangle();
rectangle.setHeight(50.0f);
rectangle.setWidth(70.0f);
rectangle.setFill(Color.BLUE);
rectangle.addEventHandler(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event){
        rectangle.setFill(palette.get(new Random().nextInt(palette.size())));
    }
});
...
```

Output:



Shapes change color when the mouse is clicked on the shape

Add rectangle shape and setup visual options and event handling

Generate a random integer position in the palette list and grab color at that position

Example: working with shapes in JavaFX (cont'd)

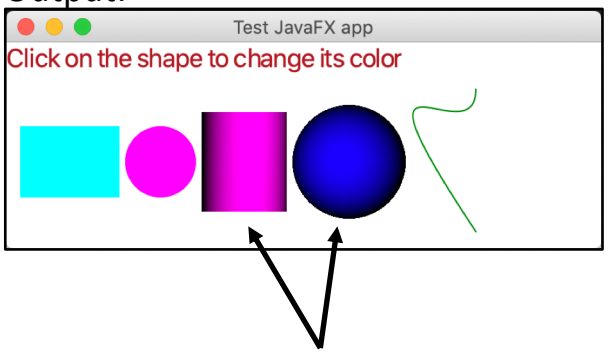
Class that adds texture and color distribution to a 3-D shape

…

```
// working with 3-D shapes:
Cylinder cylinder = new Cylinder();
cylinder.setRadius(30.0f);
cylinder.setHeight(70.0f);
PhongMaterial phMaterial1 = new PhongMaterial();
phMaterial1.setDiffuseColor(Color.ORANGE);
cylinder.setMaterial(phMaterial1);
cylinder.addEventHandler(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event){
        PhongMaterial phMaterial = new PhongMaterial();
        phMaterial.setDiffuseColor(palette.get(new Random().nextInt(palette.size())));
        cylinder.setMaterial(phMaterial);
    }
});

Sphere sphere = new Sphere();
sphere.setRadius(40.0f);
PhongMaterial phMaterial2 = new PhongMaterial();
phMaterial2.setDiffuseColor(Color.RED);
sphere.setMaterial(phMaterial2);
sphere.addEventHandler(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event){
        PhongMaterial phMaterial = new PhongMaterial();
        phMaterial.setDiffuseColor(palette.get(new Random().nextInt(palette.size())));
        sphere.setMaterial(phMaterial);
    }
});
```

…

Output:



Shapes change color when the mouse is clicked on the shape

PhongMaterial class is in *javafx.scene.paint* package. Read Java API for information about other classes in this package

Example: working with shapes in JavaFX (cont'd)

Generic path from point A to point B

…

Create movement from point A to point B

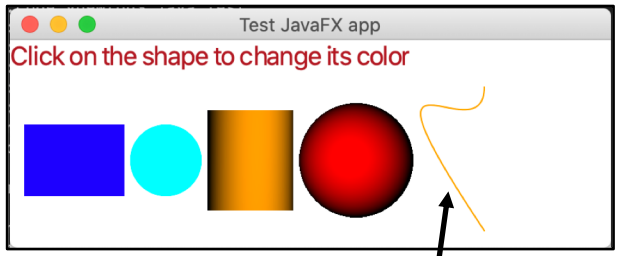
```
// working with lines and curves
Path path = new Path();
MoveTo moveTo = new MoveTo();
moveTo.setX(100.0f);
moveTo.setY(150.0f);
CubicCurveTo cubicTo = new CubicCurveTo();
cubicTo.setControlX1(0.0f);
cubicTo.setControlY1(0.0f);
cubicTo.setControlX2(100.0f);
cubicTo.setControlY2(100.0f);
cubicTo.setX(100.0f);
cubicTo.setY(50.0f);
path.getElements().add(moveTo);
path.getElements().add(cubicTo);
path.setStroke(Color.GREEN);
path.addEventHandler(MouseEvent.MOUSE_CLICKED, new EventHandler<MouseEvent>() {
    @Override
    public void handle(MouseEvent event){
        path.setStroke(palette.get(new Random().nextInt(palette.size())));
    }
});
```

Create a cubic curve

set color

…

Output:



Line change color when the mouse is clicked on it

Properties of *PhongMaterial* class (3-D shape materials)

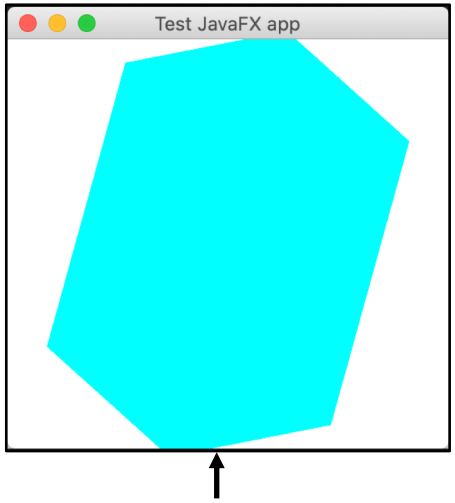
- **diffuseColor** − represents a diffuse color
- **bumpMap** − represents a map stored as a RGB Image
- **diffuseMap** − represents a diffuse map.
- **selfIlluminationMap** − represents a self-illumination map
- **specularMap** − represents a specular map
- **specularColor** − represents a specular color
- **specularPower** − represents a specular power

JavaFX shapes:

Class	Description
Arc	The `Arc` class represents a 2D arc object, defined by a center point, start angle (in degrees), angular extent (length of the arc in degrees), and an
ArcTo	A path element that forms an arc from the previous coordinates to the specified x and y coordinates using the specified radius.
Box	The `Box` class defines a 3 dimensional box with the specified size.
Circle	The `Circle` class creates a new circle with the specified radius and center location measured in pixels Example usage.
ClosePath	A path element which closes the current path.
CubicCurve	The `CubiCurve` class defines a cubic Bézier parametric curve segment in (x,y) coordinate space.
CubicCurveTo	Creates a curved path element, defined by three new points, by drawing a Cubic Bézier curve that intersects both the current coordinates and th `(controlX1,controlY1)` and `(controlX2,controlY2)` as Bézier control points.
Cylinder	The `Cylinder` class defines a 3 dimensional cylinder with the specified size.
Ellipse	The `Ellipse` class creates a new ellipse with the specified size and location in pixels
HLineTo	Creates a horizontal line path element from the current point to x.
Line	This Line represents a line segment in `(x,y)` coordinate space.
LineTo	Creates a line path element by drawing a straight line from the current coordinate to the new coordinates.
Mesh	Base class for representing a 3D geometric surface.
MeshView	The `MeshView` class defines a surface with the specified 3D mesh data.
MoveTo	Creates an addition to the path by moving to the specified coordinates.
Path	The `Path` class represents a simple shape and provides facilities required for basic construction and management of a geometric path.
PathElement	The `PathElement` class represents an abstract element of the **`Path`** that can represent any geometric objects like straight lines, arcs, quadratic cu
Polygon	Creates a polygon, defined by an array of x,y coordinates.
Polyline	Creates a polyline, defined by the array of the segment points.
QuadCurve	The `Quadcurve` class defines a quadratic Bézier parametric curve segment in (x,y) coordinate space.
QuadCurveTo	Creates a curved path element, defined by two new points, by drawing a Quadratic Bézier curve that intersects both the current coordinates and `controlY)` as a Bézier control point.
Rectangle	The `Rectangle` class defines a rectangle with the specified size and location.
Shape	The `Shape` class provides definitions of common properties for objects that represent some form of geometric shape.
Shape3D	The `Shape3D` base class provides definitions of common properties for objects that represent some form of 3D geometric shape.
Sphere	The `Sphere` class defines a 3 dimensional sphere with the specified size.
SVGPath	The `SVGPath` class represents a simple shape that is constructed by parsing SVG path data from a String.
TriangleMesh	Defines a 3D triangle mesh that consists of its associated `VertexFormat` and a set of separate arrays of vertex components such as points, norma triangles of the mesh.
VertexFormat	Defines the format of the vertices in a mesh.
VLineTo	Creates a vertical line path element from the current point to y.

https://docs.oracle.com/

Example: using animation (2-D shape)



This hexagon rotates around its center point

Use *javafx.animation* package to add animation to your application

Read Java API for information about other types of transitions and ways to create animation effects

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.shape.Polygon;
import javafx.scene.paint.Color;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.text.*;
import java.util.*;
import javafx.util.Duration;
import javafx.animation.RotateTransition;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Polygon hexagon = new Polygon();
        hexagon.getPoints().addAll(new Double[]{
         200.0, 50.0,
         400.0, 50.0,
         450.0, 150.0,
         400.0, 250.0,
         200.0, 250.0,
         150.0, 150.0,
        });
        hexagon.setFill(Color.CYAN);

        RotateTransition rotateTransition = new RotateTransition();
        rotateTransition.setDuration(Duration.millis(1000));
        rotateTransition.setNode(hexagon);
        rotateTransition.setByAngle(360);
        rotateTransition.setCycleCount(50);
        rotateTransition.setAutoReverse(false);
        rotateTransition.play();

        BorderPane root = new BorderPane();
        root.setCenter(hexagon);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(300);
        stage.setMinWidth(300);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

Create an instance of a hexagon with given apex points

Introduce rotation animation

How many cycles to do animation

Bigger number means slower rotation

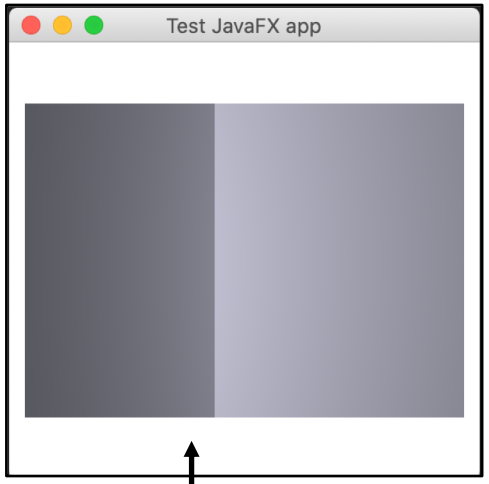
Make the animation movement go in reverse or not

Start animation

Example borrowed from www.tutorialspoint.com

Example: using animation (3-D shape)

Output:



This box rotates around y-axis

Box is a 3-D shape
Specify dimensions

As seen before, 3-D shapes need material from which they are made specified

Setup rotation for the box

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.shape.Box;
import javafx.scene.paint.Color;
import javafx.scene.paint.PhongMaterial;
import javafx.scene.transform.Rotate;
import javafx.scene.text.*;
import java.util.*;
import javafx.util.Duration;
import javafx.animation.RotateTransition;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Box box = new Box();
        box.setWidth(200.0);
        box.setHeight(200.0);
        box.setDepth(200.0);

        PhongMaterial boxMaterial = new PhongMaterial();
        boxMaterial.setDiffuseColor(Color.LAVENDER);
        box.setMaterial(boxMaterial);

        RotateTransition rotateTransition = new RotateTransition();
        rotateTransition.setDuration(Duration.millis(10000));
        rotateTransition.setNode(box);
        rotateTransition.setAxis(Rotate.Y_AXIS);
        rotateTransition.setByAngle(360);
        rotateTransition.setCycleCount(50);
        rotateTransition.setAutoReverse(false);
        rotateTransition.play();

        BorderPane root = new BorderPane();
        root.setCenter(box);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(300);
        stage.setMinWidth(300);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

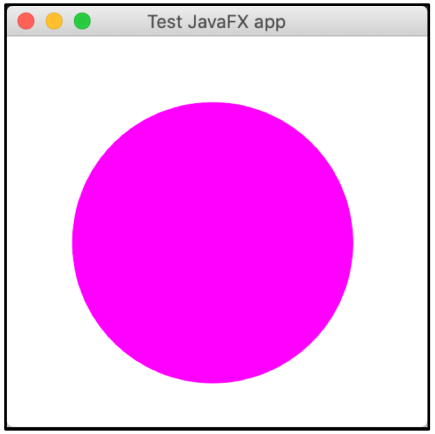
Other ways to animate your app

Class	Description
Animation	The class `Animation` provides the core functionality of all animations used in the JavaFX runtime.
AnimationTimer	The class `AnimationTimer` allows to create a timer, that is called in each frame while it is active.
FadeTransition	This `Transition` creates a fade effect animation that spans its `duration`.
FillTransition	This `Transition` creates an animation, that changes the filling of a shape over a `duration`.
Interpolator	The abstract class defines several `interpolate` methods, which are used to calculate interpolated values.
KeyFrame	Defines target values at a specified point in time for a set of variables that are interpolated along a **`Timeline`**.
KeyValue	Defines a key value to be interpolated for a particular interval along the animation.
ParallelTransition	This **`Transition`** plays a list of **`Animations`** in parallel.
PathTransition	This `Transition` creates a path animation that spans its **`PathTransition.duration`**.
PauseTransition	This `Transition` executes an **`Animation.onFinished`** at the end of its **`PauseTransition.duration`**.
RotateTransition	This `Transition` creates a rotation animation that spans its `duration`.
ScaleTransition	This `Transition` creates a scale animation that spans its **`ScaleTransition.duration`**.
SequentialTransition	This **`Transition`** plays a list of **`Animations`** in sequential order.
StrokeTransition	This `Transition` creates an animation, that changes the stroke color of a shape over a `duration`.
Timeline	A `Timeline` can be used to define a free form animation of any **`WritableValue`**, e.g.
Transition	An abstract class that contains the basic functionalities required by all `Transition` based animations, such as **`PathTransition`** and **`RotateTransition`**.
TranslateTransition	This `Transition` creates a move/translate animation that spans its **`TranslateTransition.duration`**.

https://docs.oracle.com/

Example: using *TranslateTransition* animation

Output:



The circle moves back and forth along y=-x line

Create an instance of circle

Create TranslateTransition and setup its options

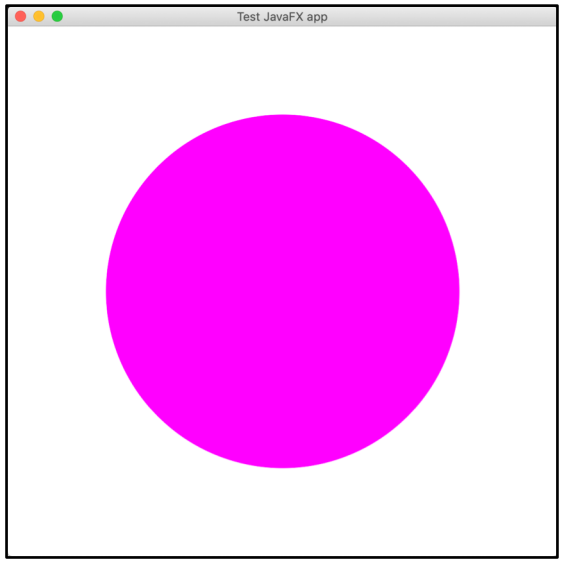
```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.shape.Circle;
import javafx.scene.paint.Color;
import javafx.scene.transform.Rotate;
import javafx.scene.text.*;
import java.util.*;
import javafx.util.Duration;
import javafx.animation.TranslateTransition;
import javafx.animation.Timeline;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Circle circle = new Circle();
        circle.setRadius(100.0f);
        circle.setFill(Color.MAGENTA);

        TranslateTransition translateTransition = new TranslateTransition();
        translateTransition.setDuration(Duration.millis(1000));
        translateTransition.setNode(circle);
        translateTransition.setByX(100);
        translateTransition.setByY(100);
        translateTransition.setCycleCount(Timeline.INDEFINITE);
        translateTransition.setAutoReverse(true);
        translateTransition.play();

        BorderPane root = new BorderPane();
        root.setLeft(circle);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(300);
        stage.setMinWidth(300);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

Infinite animation

Reverse animation (come back through the same path)

Example: using *ScaleTransition* animation

Output:



The circle inflates (increases in size) and deflates (decreases in size)

Create ScaleTransition and setup options

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.shape.Circle;
import javafx.scene.paint.Color;
import javafx.scene.transform.Rotate;
import javafx.scene.text.*;
import java.util.*;
import javafx.util.Duration;
import javafx.animation.ScaleTransition;
import javafx.animation.Timeline;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Circle circle = new Circle();
        circle.setRadius(100.0f);
        circle.setFill(Color.MAGENTA);

        ScaleTransition scaleTransition = new ScaleTransition();
        scaleTransition.setDuration(Duration.millis(1000));
        scaleTransition.setNode(circle);
        scaleTransition.setByY(1.5);
        scaleTransition.setByX(1.5);
        scaleTransition.setCycleCount(Timeline.INDEFINITE);
        scaleTransition.setAutoReverse(true);
        scaleTransition.play();

        BorderPane root = new BorderPane();
        root.setCenter(circle);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(600);
        stage.setMinWidth(600);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

Example: custom shapes with JavaFX

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.paint.Color;
import javafx.scene.canvas.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Canvas canvas = new Canvas(250, 250);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        gc.beginPath();
        gc.moveTo(50, 50);
        gc.bezierCurveTo(150, 20, 150, 150, 75, 150);
        gc.closePath();
        gc.setFill(Color.BLUE);
        gc.fill();
        gc.setStroke(Color.GREEN);
        gc.setLineWidth(10);
        gc.stroke();

        BorderPane root = new BorderPane();
        root.setCenter(canvas);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(300);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

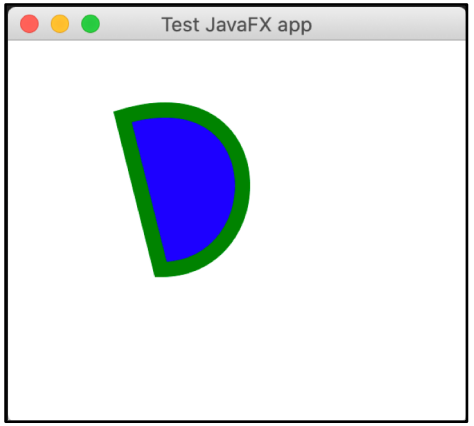
Create a 2-D canvas instance

Create and draw a path

Fill in the shape and create border

Adds a segment to make a cubic Bezier curve at the current path

Output:



Example borrowed from https://stackoverflow.com/, a post by jewelsea

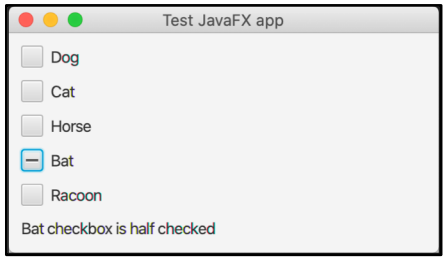
javafx.scene.canvas.GraphicsContext

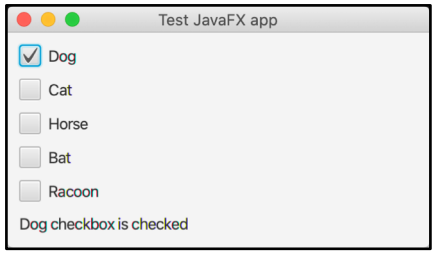
- Refer to the API for more methods and functionality
- Available functionality
  - Drawing
    - Images
    - Arcs
    - Paths
    - Polygon
    - Oval
    - Text
  - Filling in and drawing custom borders
  - Applying effects
  - Applying custom transformations

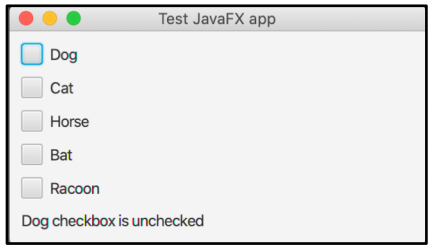
Using listeners with JavaFX

- Change listeners – a way to handle change events
  - Notified when the value of an *ObservableValue* changes
    - *ObservabalValue* interface (see Java API)
- *ObservableValue* objects represent content of nodes in JavaFX
  - E.g. *TextField*, *ListView*, *ChoiceBox*, *ComboBox*
  - *CheckBox*, *RadioButton*, *ToggleButton* are also *ObservableValue* objects
- We can add change and invalidation listeners to *ObservableValue* objects

Example: revisiting checkbox example to add change listeners








```java
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.beans.value.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        CheckBox chbxDog = new CheckBox("Dog");
        CheckBox chbxCat = new CheckBox("Cat");
        CheckBox chbxHorse = new CheckBox("Horse");
        CheckBox chbxBat = new CheckBox("Bat");
        chbxBat.setAllowIndeterminate(true);
        CheckBox chbxRacoon = new CheckBox("Racoon");
        Label lblLatestAction = new Label();

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(chbxDog,chbxCat,chbxHorse,chbxBat,chbxRacoon,lblLatestAction);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();

        chbxDog.selectedProperty().addListener(...);

        chbxCat.selectedProperty().addListener(...);

        chbxHorse.selectedProperty().addListener(...);

        chbxRacoon.selectedProperty().addListener(...);

        chbxBat.selectedProperty().addListener(...);

        chbxBat.indeterminateProperty().addListener(...);

    }


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

Example: revisiting checkbox example to add change listeners (cont'd)

Add a change listener to each checkbox

```
chbxDog.selectedProperty().addListener(new ChangeListener<Boolean>(){
    public void changed(ObservableValue<? extends Boolean> ov,
            final Boolean value, final Boolean newValue){
        if(newValue != null && newValue){
            lblLatestAction.setText("Dog checkbox is checked");
        }else{
            lblLatestAction.setText("Dog checkbox is unchecked");
        }
    }
});

chbxCat.selectedProperty().addListener(new ChangeListener<Boolean>(){
    public void changed(ObservableValue<? extends Boolean> ov,
            final Boolean value, final Boolean newValue){
        if(newValue != null && newValue){
            lblLatestAction.setText("Cat checkbox is checked");
        }else{
            lblLatestAction.setText("Cat checkbox is unchecked");
        }
    }
});

chbxHorse.selectedProperty().addListener(new ChangeListener<Boolean>(){
    public void changed(ObservableValue<? extends Boolean> ov,
            final Boolean value, final Boolean newValue){
        if(newValue != null && newValue){
            lblLatestAction.setText("Horse checkbox is checked");
        }else{
            lblLatestAction.setText("Horse checkbox is unchecked");
        }
    }
});

chbxRacoon.selectedProperty().addListener(new ChangeListener<Boolean>(){
    public void changed(ObservableValue<? extends Boolean> ov,
            final Boolean value, final Boolean newValue){
        if(newValue != null && newValue){
            lblLatestAction.setText("Racoon checkbox is checked");
        }else{
            lblLatestAction.setText("Racoon checkbox is unchecked");
        }
    }
});

chbxBat.selectedProperty().addListener(new ChangeListener<Boolean>(){
    public void changed(ObservableValue<? extends Boolean> ov,
            final Boolean value, final Boolean newValue){
        if(newValue != null && newValue){
            lblLatestAction.setText("Bat checkbox is checked");
        }else{
            lblLatestAction.setText("Bat checkbox is unchecked");
        }
    }
});

chbxBat.indeterminateProperty().addListener(new ChangeListener<Boolean>(){
    public void changed(ObservableValue<? extends Boolean> ov,
            final Boolean value, final Boolean newValue){
        if(newValue != null && newValue){
            lblLatestAction.setText("Bat checkbox is half checked");
        }
    }
});
```

For comparison, previous implementation:

```
EventHandler<ActionEvent> checkBoxEventHandler = new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event){
        String state;
        Object source = (Object) event.getSource();

        if(((CheckBox)source).isSelected()){
            state = "checked";
        } else{
            state = "unchecked";
        }

        if((CheckBox)source == chbxDog){
            lblLatestAction.setText("Dog checkbox is "+state);
        } else if((CheckBox)source == chbxCat){
            lblLatestAction.setText("Cat checkbox is "+state);
        } else if((CheckBox)source == chbxHorse){
            lblLatestAction.setText("Horse checkbox is "+state);
        } else if((CheckBox)source == chbxBat){
            if(chbxBat.isIndeterminate()){
                lblLatestAction.setText("Bat checkbox is half checked");
            }else if(chbxBat.isSelected()){
                lblLatestAction.setText("Bat checkbox is checked");
            }else{
                lblLatestAction.setText("Bat checkbox is unchecked");
            }
        } else if((CheckBox)source == chbxRacoon){
            lblLatestAction.setText("Racoon checkbox is "+state);
        }
    }
};
```

Note that *ChangeListener* here uses *Boolean* type because the checkbox has a Boolean state (*ObservableValue extends Boolean*)

Add the change listener to both selected and intermediate property for the Bat checkbox

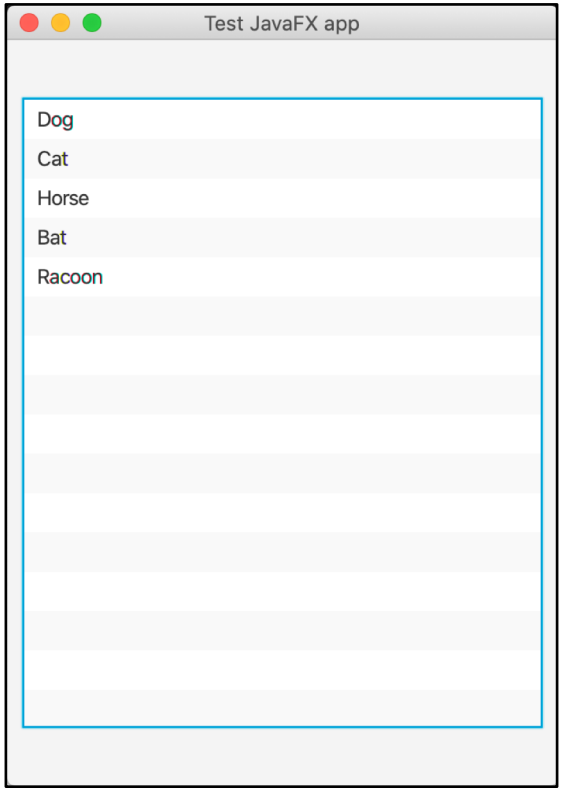
Example: using list view

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.scene.paint.Color;
import javafx.scene.canvas.*;
import javafx.beans.value.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Label lblSelection = new Label();
        Label lblAnimals = new Label();

        ListView<String> animals = new ListView<String>();
        animals.getItems().addAll("Dog", "Cat", "Horse", "Bat", "Racoon");
        //animals.getSelectionModel().selectFirst();

        animals.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<String>(){
            public void changed(ObservableValue<? extends String> ov, final String oldValue, final String newValue)
            {
                lblSelection.setText("Current selection: " + newValue);
            }
        });

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(lblAnimals, animals, lblSelection);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();
    }

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

can set selection in the list

Change listener for observable values in the list view; uses *String* type because values in the list view are a *String* type

Output:



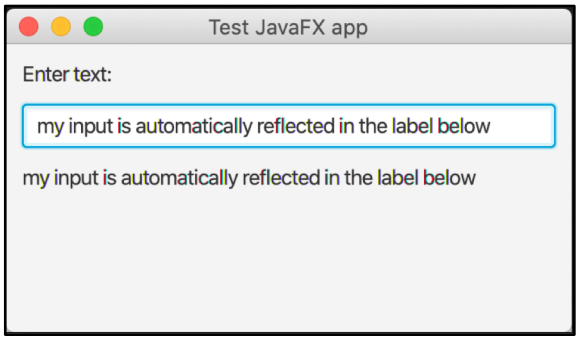
Example: using *ChangeListener* with *TextBox*

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.beans.value.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Label lblInput = new Label("Enter text:");
        TextField txtInput = new TextField();
        Label lblOutput = new Label();

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(lblInput, txtInput, lblOutput);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();

        txtInput.textProperty().addListener(new ChangeListener<String>(){
            public void changed(ObservableValue<? extends String> ov, final String value, final String newValue){
                lblOutput.setText(txtInput.getText());
            }
        });
    }

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

Add change listener to text property, which returns an ObservableValue; using String type again

Update the label text as the text in the text box changes

Output:

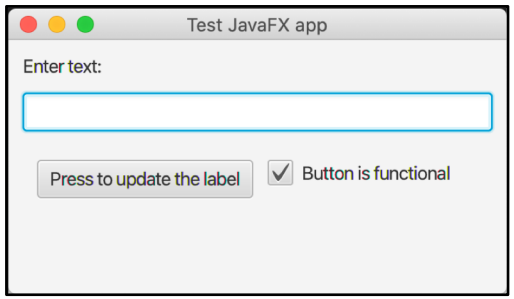


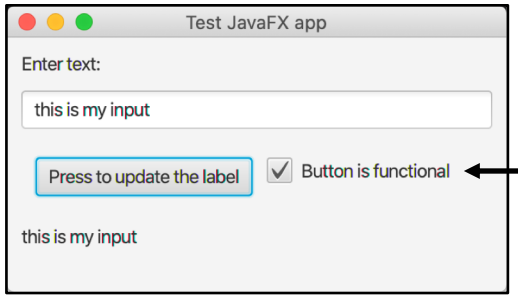
Adding and removing event filters

- When an event is triggered, the event capture phase occurs
  - Every node in the chain is polled for the appropriate event type filter
- Filters allow suppressing execution of event handlers under some condition
  - E.g. a mouse movement moves a character in the game around the screen, but you do not want that to happen when the menu is opened
- A filter is just an event
  - Events have a source, a target, and an event type (e.g. *MouseEvent.MOUSE_CLICKED*, *MouseEvent.ANY*, *KeyEvent.KEY_PRESSED*, *KeyEvent.KEY_RELEASED*, etc.)
- Use *addEventFilter()* method to add a filter
- Use *removeEventFilter()* method to remove a filter

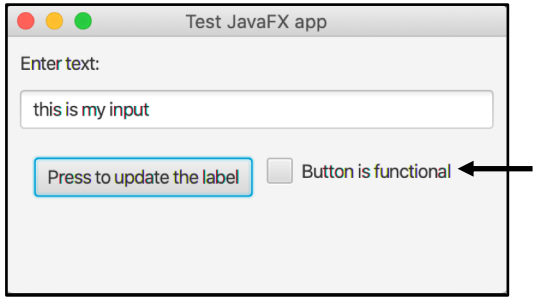
Example: using event filters

Output:





When the checkbox is checked the button responds to mouse click events



When the checkbox is unchecked the button does not respond to mouse click events

State change listener for the checkbox; when the checkbox is unchecked we add event filter, when it is checked we remove it

Create all the nodes, setup the layouts, the scene, and the stage

Event handler for the button

Event filter for the button; we use consume() method to suppress further handling of this event type

```
import javafx.application.Application;
import javafx.scene.*;
import javafx.stage.Stage;
import javafx.scene.control.*;
import javafx.scene.input.*;
import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.event.*;
import javafx.beans.value.*;

public class Test extends Application {
    @Override
    public void start(Stage stage)
    {
        Label lblInput = new Label("Enter text:");
        TextField txtInput = new TextField();
        Button btnInput = new Button("Press to update the label");
        CheckBox chkEnable = new CheckBox("Button is functional");
        chkEnable.setSelected(true);
        Label lblOutput = new Label();

        HBox ctrlPanel = new HBox();
        ctrlPanel.setPadding(new Insets(10, 10, 10, 10));
        ctrlPanel.setSpacing(10);
        ctrlPanel.getChildren().addAll(btnInput, chkEnable);

        VBox root = new VBox();
        root.setPadding(new Insets(10, 10, 10, 10));
        root.setSpacing(10);
        root.getChildren().addAll(lblInput, txtInput, ctrlPanel, lblOutput);

        Scene scene = new Scene(root);
        stage.setScene(scene);

        stage.setX(200);
        stage.setY(300);
        stage.setMinHeight(200);
        stage.setMinWidth(350);
        stage.setTitle("Test JavaFX app");
        stage.show();

        EventHandler<MouseEvent> btnEvent = new EventHandler<MouseEvent>(){
            @Override
            public void handle(MouseEvent event){
                lblOutput.setText(txtInput.getText());
            }
        };
        btnInput.addEventHandler(MouseEvent.MOUSE_CLICKED, btnEvent);

        EventHandler<MouseEvent> btnFilter = new EventHandler<MouseEvent>(){
            @Override
            public void handle(MouseEvent event){
                event.consume();
            }
        };

        chkEnable.selectedProperty().addListener(new ChangeListener<Boolean>(){
            public void changed(ObservableValue<? extends Boolean> ov, final Boolean value, final Boolean newValue){
                if(newValue != null && newValue){
                    scene.removeEventFilter(MouseEvent.MOUSE_CLICKED, btnFilter);
                }else{
                    lblOutput.setText("");
                    scene.addEventFilter(MouseEvent.MOUSE_CLICKED, btnFilter);
                }
            }
        });
    }

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

Closing remarks

- Java has a powerful GUI programming toolkit with many stylistic options
  - Top level container is a type of frame
  - Sub-containers are placed into the top container
  - Components are placed into the top container or sub-containers
- Refer to API for more information
- Use your google skills to look for examples
- Experiment, experiment, experiment!