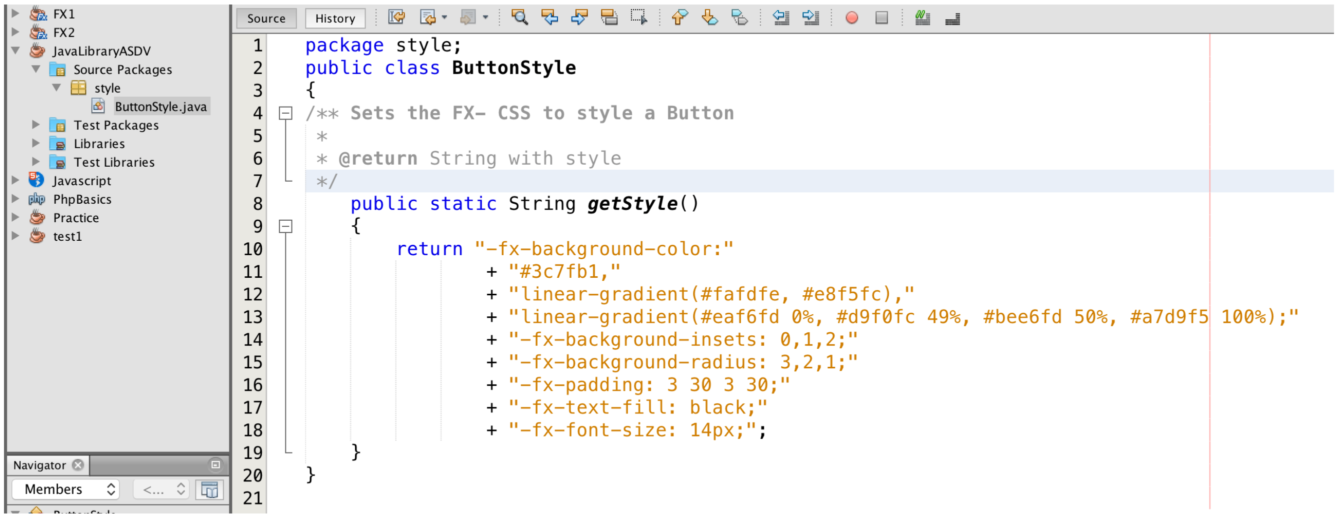
JavaFX II

REMEMBER TO CLEAN AND BUILD before you run any class.

1. How to create a Library of classes . We will create a **Library** for a standard-style a JAVA-FX Button.
   File > New Project > Java > **Java Class Library (NOT** Java Application). Name the project JavaLibraryASDV.

2. Create a new Package name style, create class **ButtonStyle** under the package **style** and either type the code below or download the file ButtonStyle.java from canvas to avoid typing errors. Make sure you **include** the **comments** in lines 4 to 7 for Javadoc.



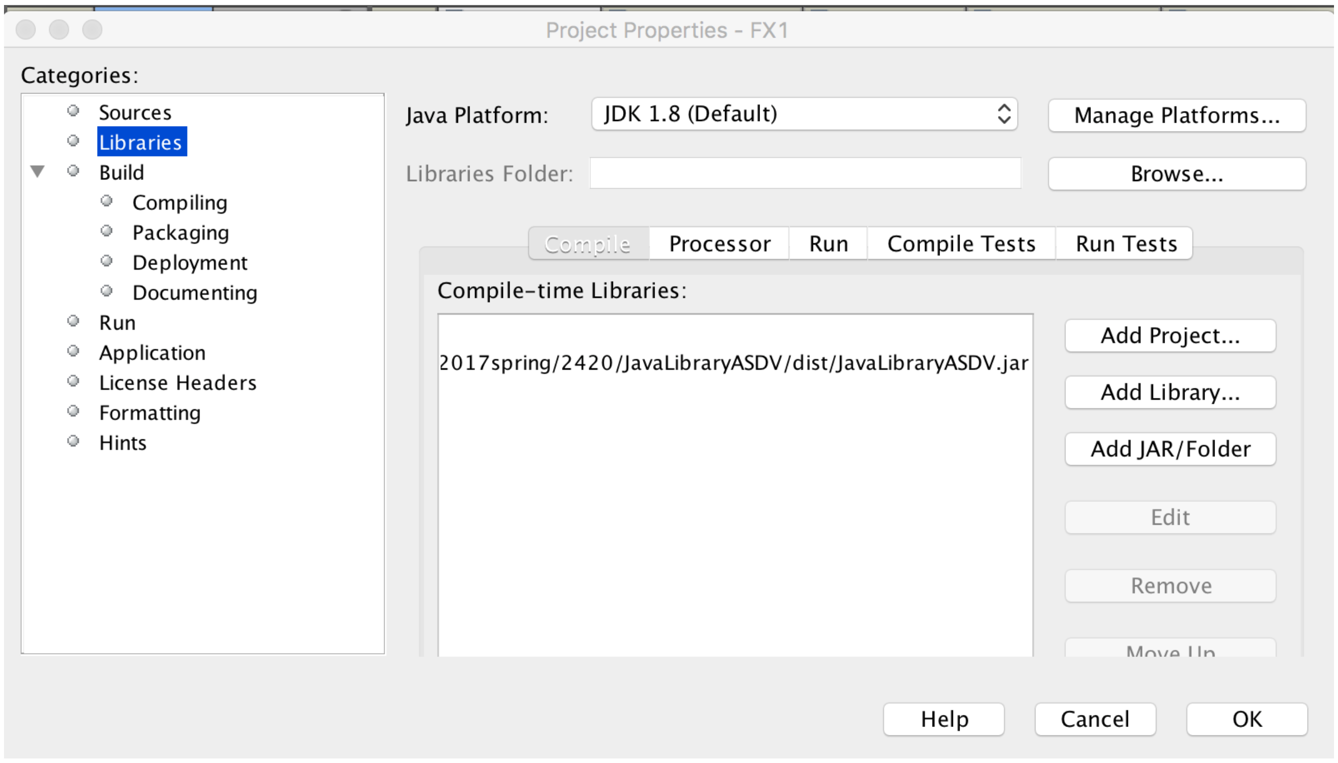
```
1  package style;
2  public class ButtonStyle
3  {
4  /** Sets the FX- CSS to style a Button
5   *
6   * @return String with style
7   */
8      public static String getStyle()
9      {
10         return "-fx-background-color:"
11                 + "#3c7fb1,"
12                 + "linear-gradient(#fafdfe, #e8f5fc),"
13                 + "linear-gradient(#eaf6fd 0%, #d9f0fc 49%, #bee6fd 50%, #a7d9f5 100%);"
14                 + "-fx-background-insets: 0,1,2;"
15                 + "-fx-background-radius: 3,2,1;"
16                 + "-fx-padding: 3 30 3 30;"
17                 + "-fx-text-fill: black;"
18                 + "-fx-font-size: 14px;";
19     }
20 }
21
```

3. Run> Clean and Build Project
4. Run > Generate Javadoc
5. Run> Clean and Build Project (again)

You are done with the creation of the **Library** , JavaLibraryASDV.jar . In the future you may

add more classes similarly to this project and rebuild.
To add this library to your existing JavaFX project named FX1:. **Right click** on the coffee cup of FX1 and click Properties.

6. Select **Libraries**, **Compile**, **Add Jar/Folder** AND NAGIVATE to where the JavaLibraryASDV.jar is in your computer. Click OK.
......JavaLibraryASDV/dist/ JavaLibraryASDV.jar.



Java Platform: JDK 1.8 (Default)

Libraries Folder:

Compile-time Libraries:

2017spring/2420/JavaLibraryASDV/dist/JavaLibraryASDV.jar

7. To test your new Button of the class library we will introduce officially EVENT DRIVEN PROGRAMMING.

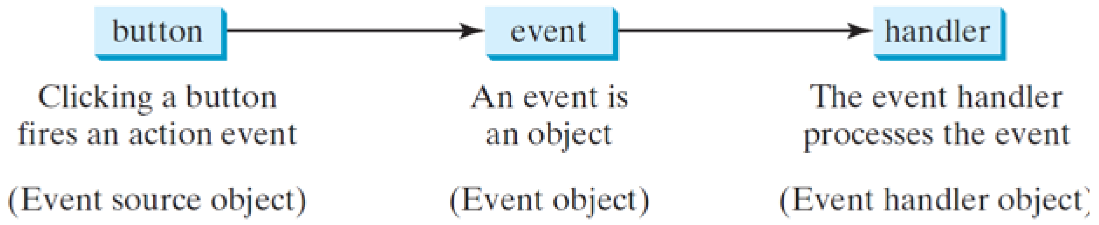


FIGURE 1

As you see in the **figure 1** above, a button-click **CREATES** an **OBJECT** (event). We take this OBJECT (event) and we handle it in a HANDLER method.

In **figure 2** below, the a partial hierarchy of Inheritance of Events is shown:

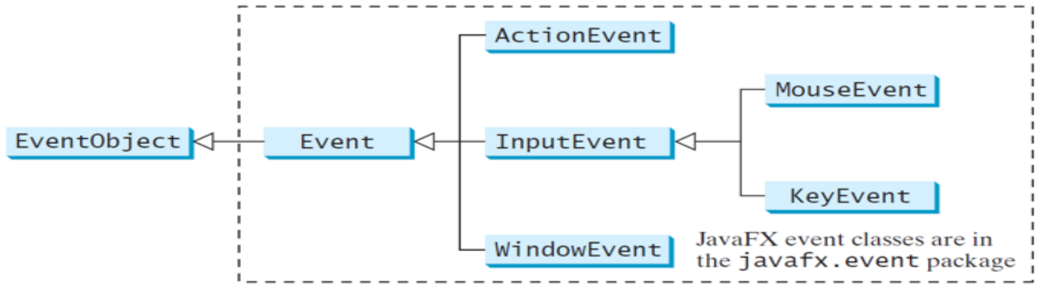
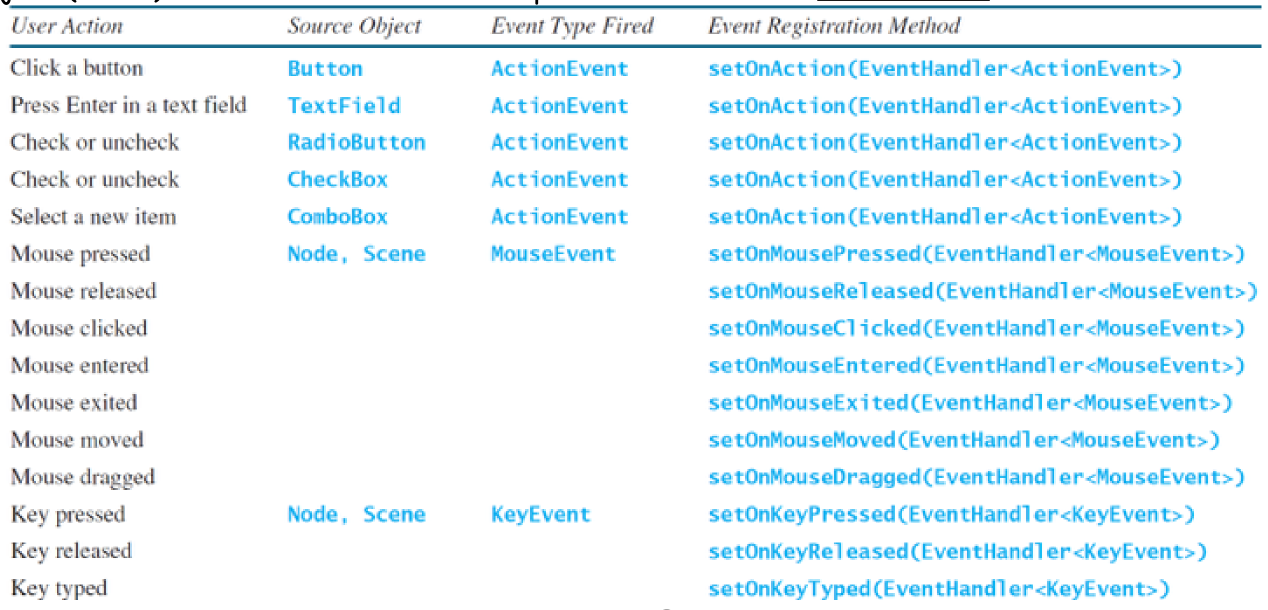


FIGURE 2

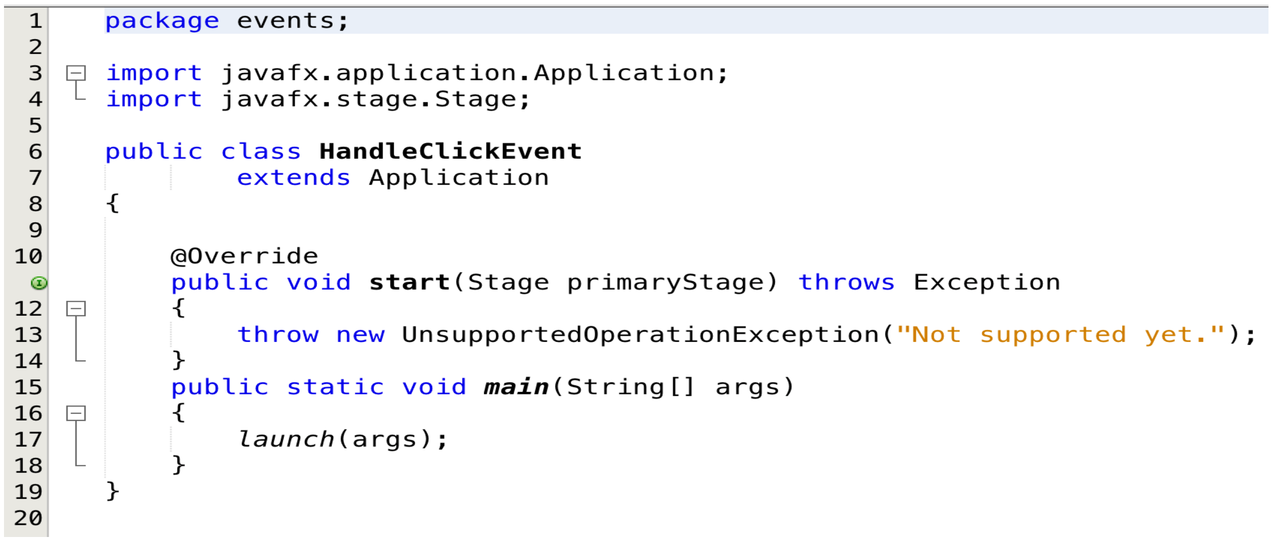
In **figure 3** below, the clicking of a Button generates an <u>ActionEvent</u> . Wee will register an object (event) with Java-Machine and implement the method <u>**setOnAction**</u> .



User Action	*Source Object*	*Event Type Fired*	*Event Registration Method*
Click a button	`Button`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Press Enter in a text field	`TextField`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Check or uncheck	`RadioButton`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Check or uncheck	`CheckBox`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Select a new item	`ComboBox`	`ActionEvent`	`setOnAction(EventHandler<ActionEvent>)`
Mouse pressed	`Node, Scene`	`MouseEvent`	`setOnMousePressed(EventHandler<MouseEvent>)`
Mouse released			`setOnMouseReleased(EventHandler<MouseEvent>)`
Mouse clicked			`setOnMouseClicked(EventHandler<MouseEvent>)`
Mouse entered			`setOnMouseEntered(EventHandler<MouseEvent>)`
Mouse exited			`setOnMouseExited(EventHandler<MouseEvent>)`
Mouse moved			`setOnMouseMoved(EventHandler<MouseEvent>)`
Mouse dragged			`setOnMouseDragged(EventHandler<MouseEvent>)`
Key pressed	`Node, Scene`	`KeyEvent`	`setOnKeyPressed(EventHandler<KeyEvent>)`
Key released			`setOnKeyReleased(EventHandler<KeyEvent>)`
Key typed			`setOnKeyTyped(EventHandler<KeyEvent>)`

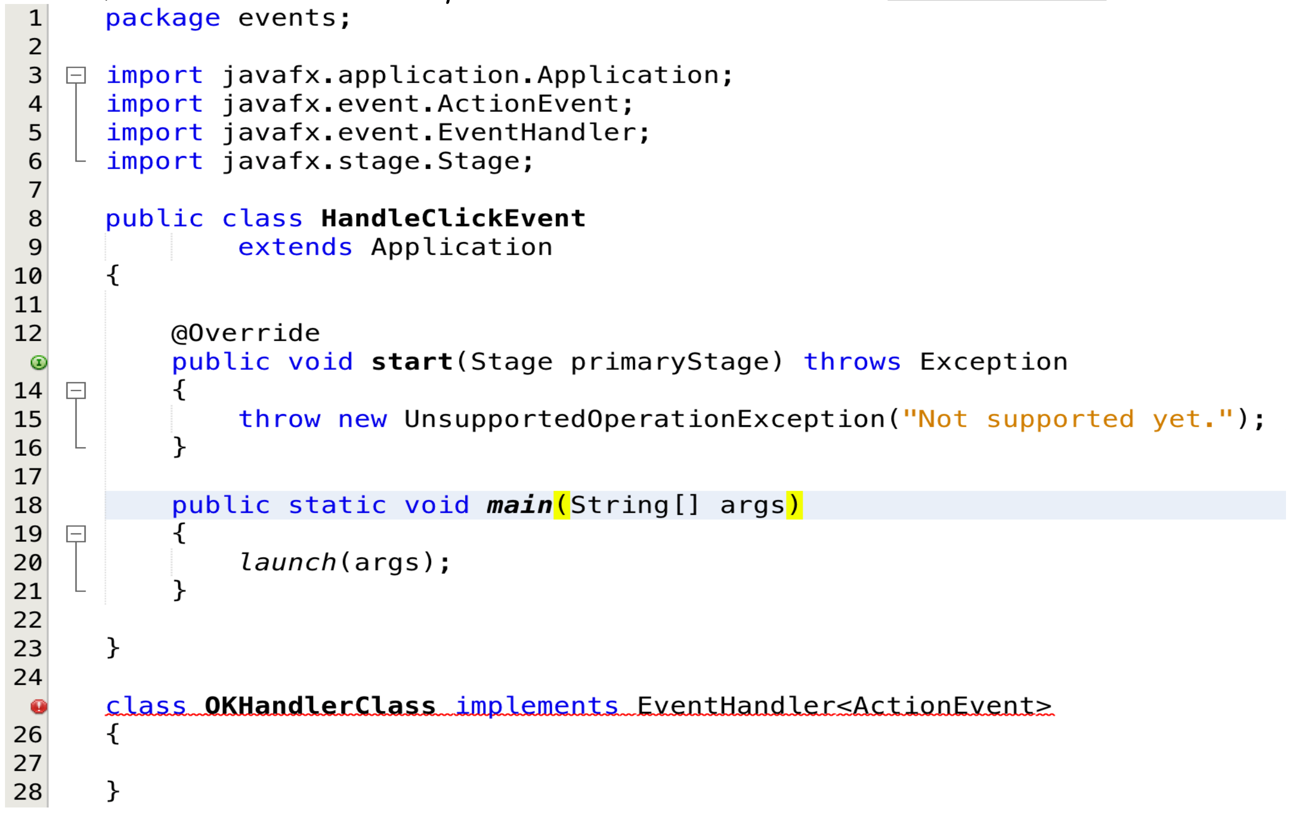
FIGURE 3

8. Let as put Event Driven Programing to works. Create a new package in your FX1 application and name it events. Create a new class HandleClickEvent and derive it from Application. The class handles the clicking of button(s).



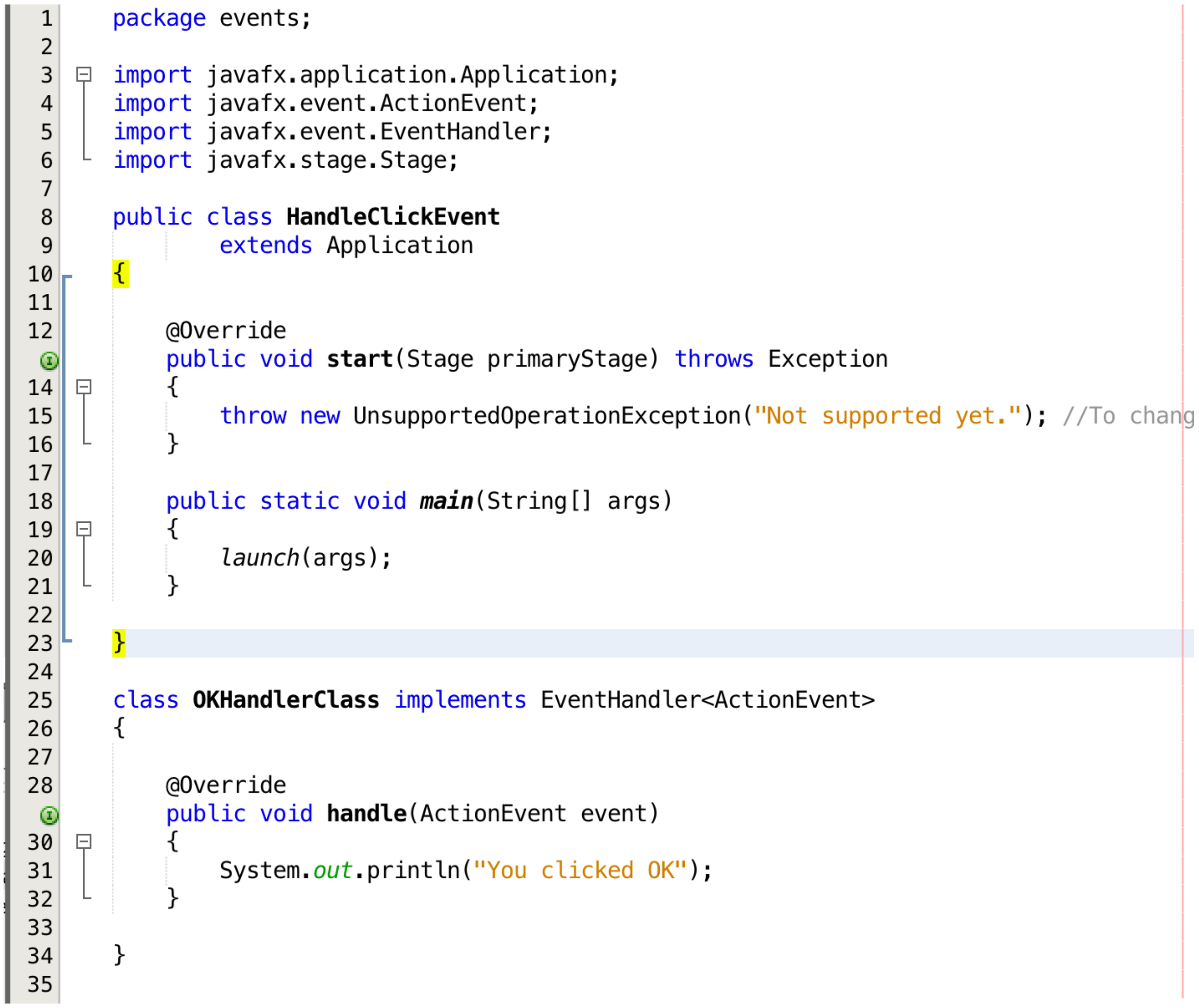
```
package events;

import javafx.application.Application;
import javafx.stage.Stage;

public class HandleClickEvent
        extends Application
{

    @Override
    public void start(Stage primaryStage) throws Exception
    {
        throw new UnsupportedOperationException("Not supported yet.");
    }
    public static void main(String[] args)
    {
        launch(args);
    }
}

```

9. Add the class OKHandlerClass which implements EventHandler to handle clicking. Lines 28 to 28 below, and click the bulb to *Implement all abstract methods* in OKHandlerClass



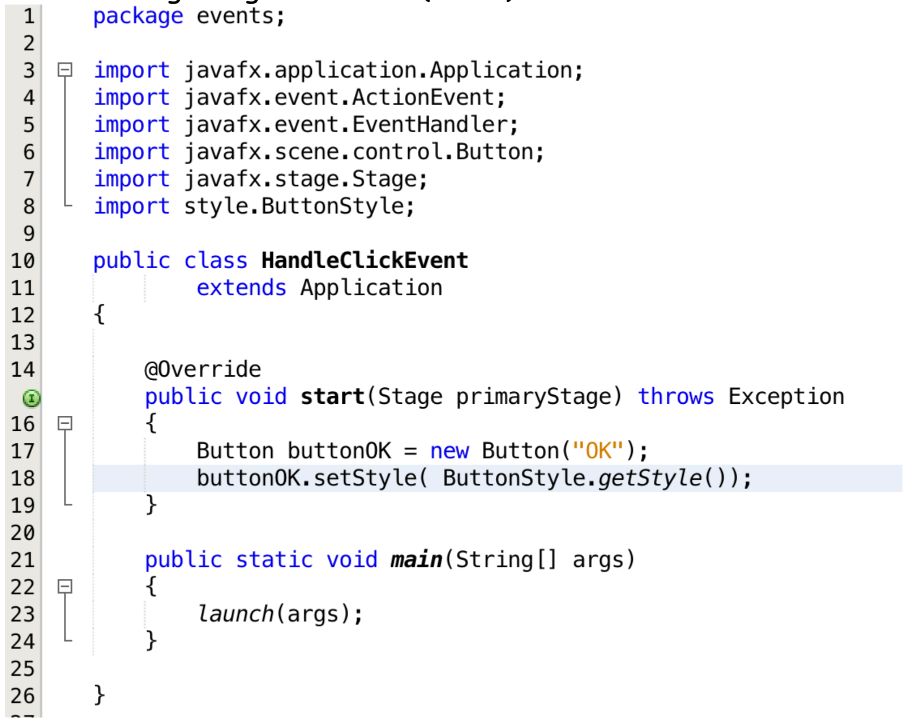
```
 1  package events;
 2
 3  import javafx.application.Application;
 4  import javafx.event.ActionEvent;
 5  import javafx.event.EventHandler;
 6  import javafx.stage.Stage;
 7
 8  public class HandleClickEvent
 9          extends Application
10  {
11
12      @Override
        public void start(Stage primaryStage) throws Exception
14      {
15          throw new UnsupportedOperationException("Not supported yet.");
16      }
17
18      public static void main(String[] args)
19      {
20          launch(args);
21      }
22
23  }
24
    class OKHandlerClass implements EventHandler<ActionEvent>
26  {
27
28  }
```

10. Type line 31. Your program now contains 2 classes and it should look as shown below:



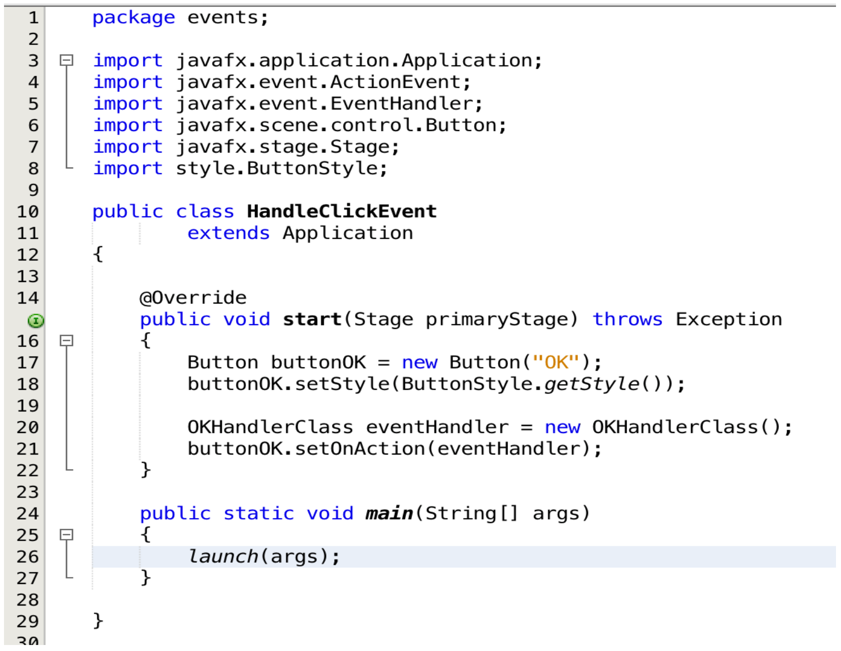
```
 1  package events;
 2
 3  import javafx.application.Application;
 4  import javafx.event.ActionEvent;
 5  import javafx.event.EventHandler;
 6  import javafx.stage.Stage;
 7
 8  public class HandleClickEvent
 9          extends Application
10  {
11
12      @Override
    public void start(Stage primaryStage) throws Exception
14      {
15          throw new UnsupportedOperationException("Not supported yet."); //To chang
16      }
17
18      public static void main(String[] args)
19      {
20          launch(args);
21      }
22
23  }
24
25  class OKHandlerClass implements EventHandler<ActionEvent>
26  {
27
28      @Override
    public void handle(ActionEvent event)
30      {
31          System.out.println("You clicked OK");
32      }
33
34  }
35
```

11. Inside your start() method, create a button in line 17 and set its style in line 18 as shown below. After you type line 19, Netbeans should import the library class style.ButtonStlye we created in the beginning of this lab (line 8).



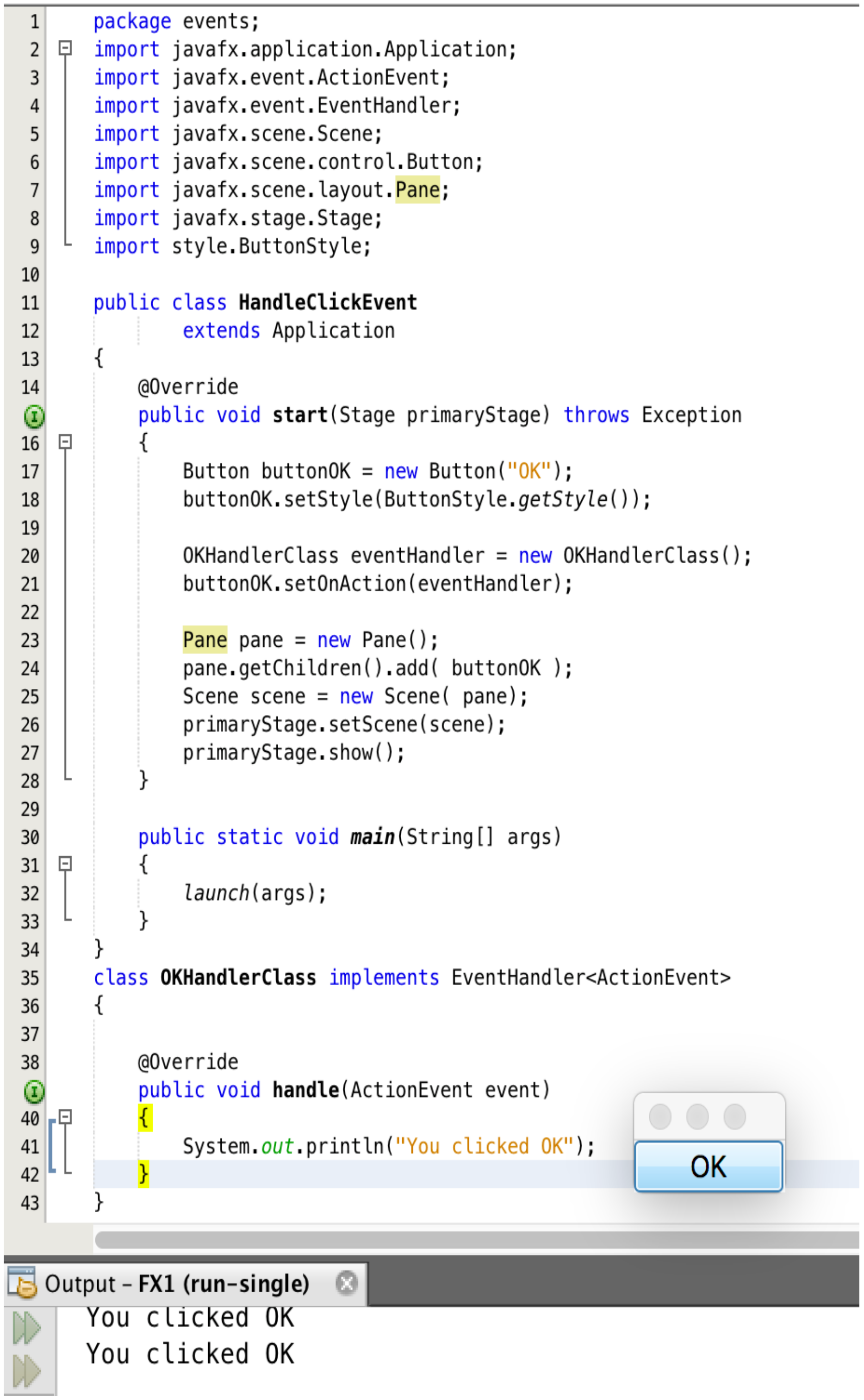
```
 1    package events;
 2
 3    import javafx.application.Application;
 4    import javafx.event.ActionEvent;
 5    import javafx.event.EventHandler;
 6    import javafx.scene.control.Button;
 7    import javafx.stage.Stage;
 8    import style.ButtonStyle;
 9
10    public class HandleClickEvent
11            extends Application
12    {
13
14        @Override
          public void start(Stage primaryStage) throws Exception
16        {
17            Button buttonOK = new Button("OK");
18            buttonOK.setStyle( ButtonStyle.getStyle());
19        }
20
21        public static void main(String[] args)
22        {
23            launch(args);
24        }
25
26    }
```

12. Create the Event Handler, eventHandler variable, line 20, and REGISTER the handler with the method setOnAction, line 21. Now the button object buttonOK has registred the object eventHandler to handle the clicking of the button.



```
 1    package events;
 2
 3    import javafx.application.Application;
 4    import javafx.event.ActionEvent;
 5    import javafx.event.EventHandler;
 6    import javafx.scene.control.Button;
 7    import javafx.stage.Stage;
 8    import style.ButtonStyle;
 9
10    public class HandleClickEvent
11            extends Application
12    {
13
14        @Override
          public void start(Stage primaryStage) throws Exception
16        {
17            Button buttonOK = new Button("OK");
18            buttonOK.setStyle(ButtonStyle.getStyle());
19
20            OKHandlerClass eventHandler = new OKHandlerClass();
21            buttonOK.setOnAction(eventHandler);
22        }
23
24        public static void main(String[] args)
25        {
26            launch(args);
27        }
28
29    }
```

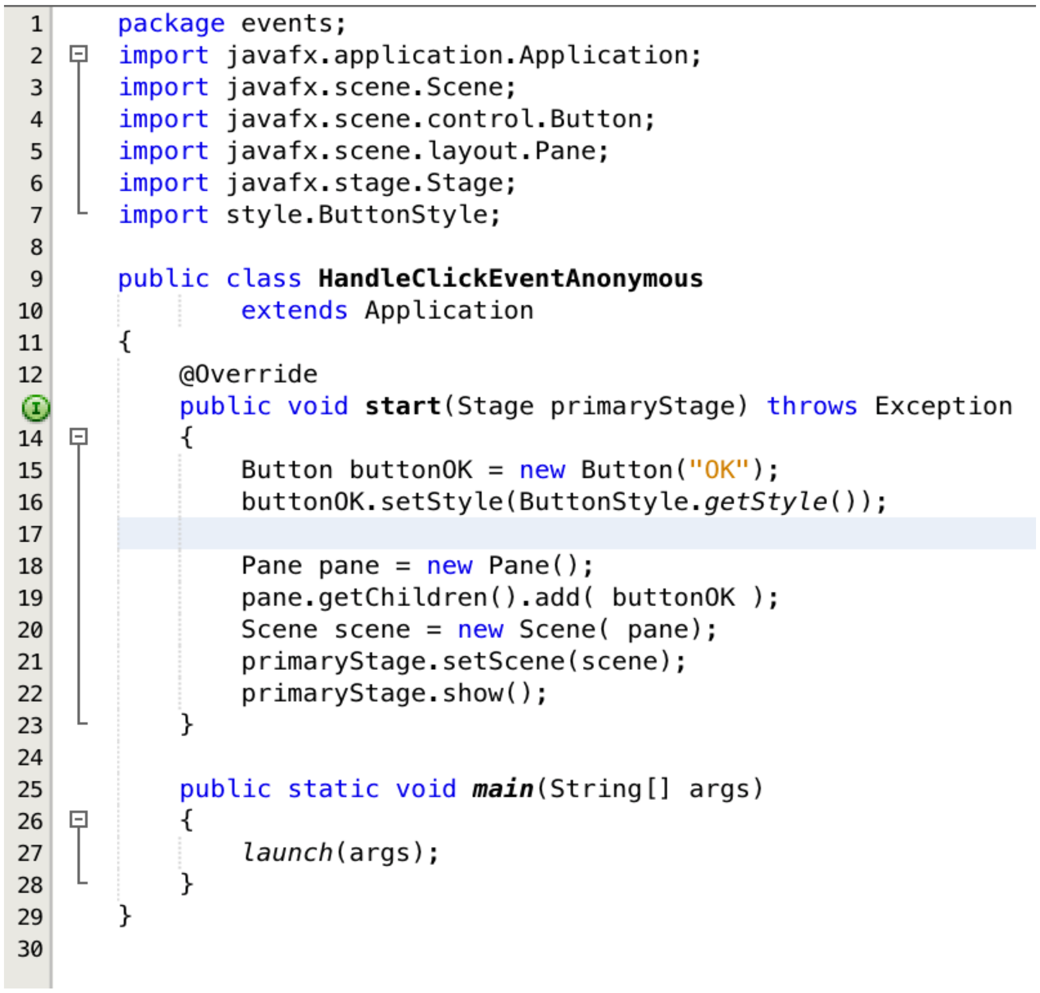
13. Follow standard procedure to test your Java-FX program: Create a Pane, add the Button to the Pane. Create a Scene, add the Pane to the Scene. Add the Scene to the Stage. You are done. Clean and build. Then run it.



```
package events;
import javafx.application.Application;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import style.ButtonStyle;

public class HandleClickEvent
        extends Application
{
    @Override
    public void start(Stage primaryStage) throws Exception
    {
        Button buttonOK = new Button("OK");
        buttonOK.setStyle(ButtonStyle.getStyle());

        OKHandlerClass eventHandler = new OKHandlerClass();
        buttonOK.setOnAction(eventHandler);

        Pane pane = new Pane();
        pane.getChildren().add( buttonOK );
        Scene scene = new Scene( pane);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
class OKHandlerClass implements EventHandler<ActionEvent>
{

    @Override
    public void handle(ActionEvent event)
    {
        System.out.println("You clicked OK");
    }
}
```

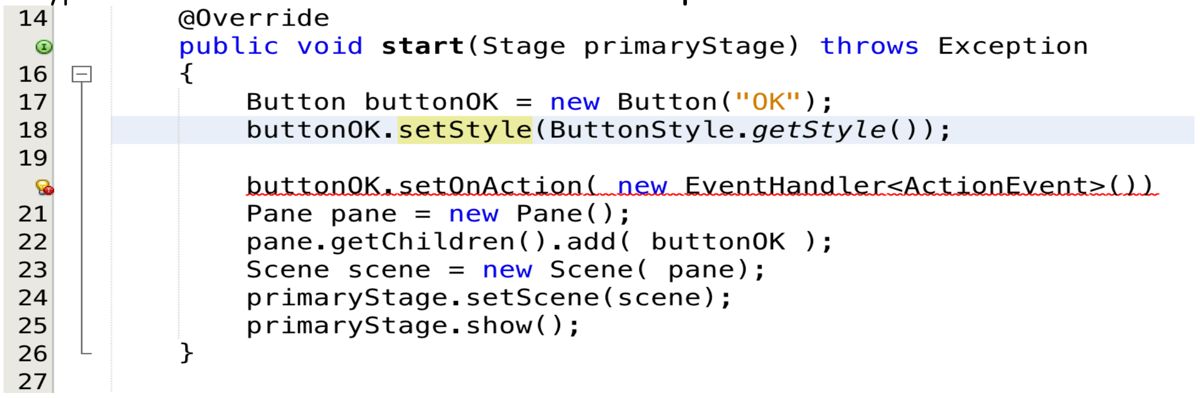
```
You clicked OK
You clicked OK
```

14. Now we will introduce officially **Anonymous classes**. In package events , right click the class HandleClickEvent. **Copy**. Then, **paste** inside the same package but **refactor** the class with the name HandleClickAnonymous. Delete the class OKHandlerClass, and delete the 2 lines of the registration of the button-handler and the attachment to the handler. The resulting code after the deletions is shown below.



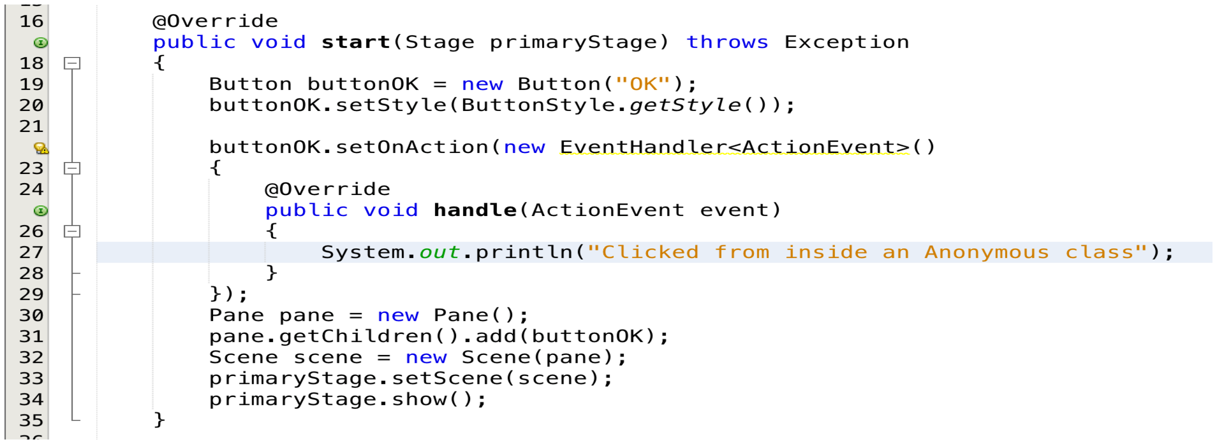
```java
package events;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.Pane;
import javafx.stage.Stage;
import style.ButtonStyle;

public class HandleClickEventAnonymous
        extends Application
{
    @Override
    public void start(Stage primaryStage) throws Exception
    {
        Button buttonOK = new Button("OK");
        buttonOK.setStyle(ButtonStyle.getStyle());

        Pane pane = new Pane();
        pane.getChildren().add( buttonOK );
        Scene scene = new Scene( pane);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

15. Type line 20 shown below. Then click the bulb to **Implement all abstract methods.**



```java
14      @Override
        public void start(Stage primaryStage) throws Exception
16      {
17          Button buttonOK = new Button("OK");
18          buttonOK.setStyle(ButtonStyle.getStyle());
19
            buttonOK.setOnAction( new EventHandler<ActionEvent>())
21          Pane pane = new Pane();
22          pane.getChildren().add( buttonOK );
23          Scene scene = new Scene( pane);
24          primaryStage.setScene(scene);
25          primaryStage.show();
26      }
27
```

21. After you clicked *Implement all abstract methods*, replace the thrown exception of method handle with line 27 shown below.
    Anonymous class: The **parameter** of the method setOnAction is an **ANONYMOUS class** of type EventHandler<ActionEvent> , with its implementation following after the brace at line 23.
    Clean and build, then run, to see the clicking output. As you see we PASSED IN THE WHOLE CLASS as an argument to the parameter of method setOnAction instead of just a variable .

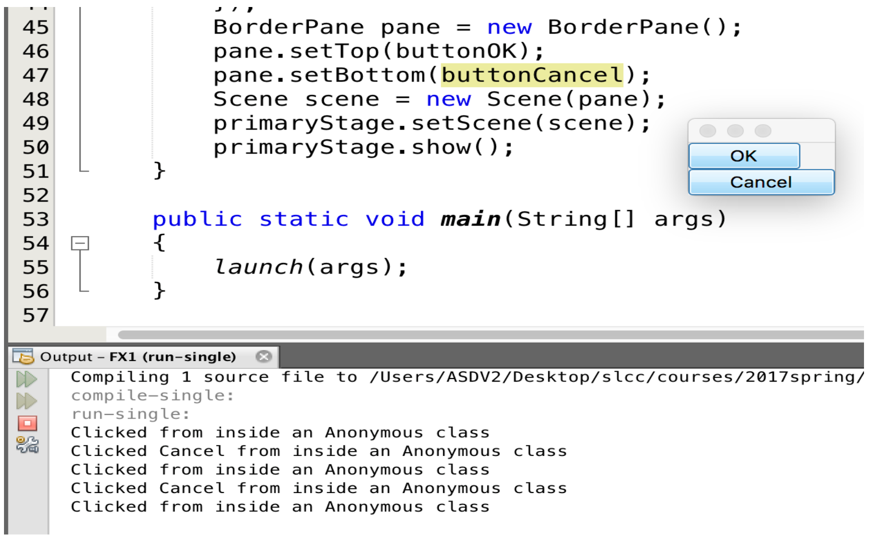


```
16      @Override
17      public void start(Stage primaryStage) throws Exception
18      {
19          Button buttonOK = new Button("OK");
20          buttonOK.setStyle(ButtonStyle.getStyle());
21
22          buttonOK.setOnAction(new EventHandler<ActionEvent>()
23          {
24              @Override
25              public void handle(ActionEvent event)
26              {
27                  System.out.println("Clicked from inside an Anonymous class");
28              }
29          });
30          Pane pane = new Pane();
31          pane.getChildren().add(buttonOK);
32          Scene scene = new Scene(pane);
33          primaryStage.setScene(scene);
34          primaryStage.show();
35      }
```

22. Modify **HandleClickEvent.java** and add class **Cancel**HandlerClass implements EventHandler<ActionEvent> **below** the class class **OK**HandlerClass implements EventHandler<ActionEvent>

    The handle method of class CancelHandlerClass prints " You clicked cancel"
    Inside the start() create another button and register its clicking and we did for the OK button.
    Instead of Pane, use BorderPane (look it up to see how it behaves,, as it IS-A a Pane).

23. Modify the HandleClickEventAnonymous class to and add a Cancel button after the OK button. Have an anonymous class handle the clicking event as we did for the OK button. Use a BorderPane as shown below. We will see more Pane types later.

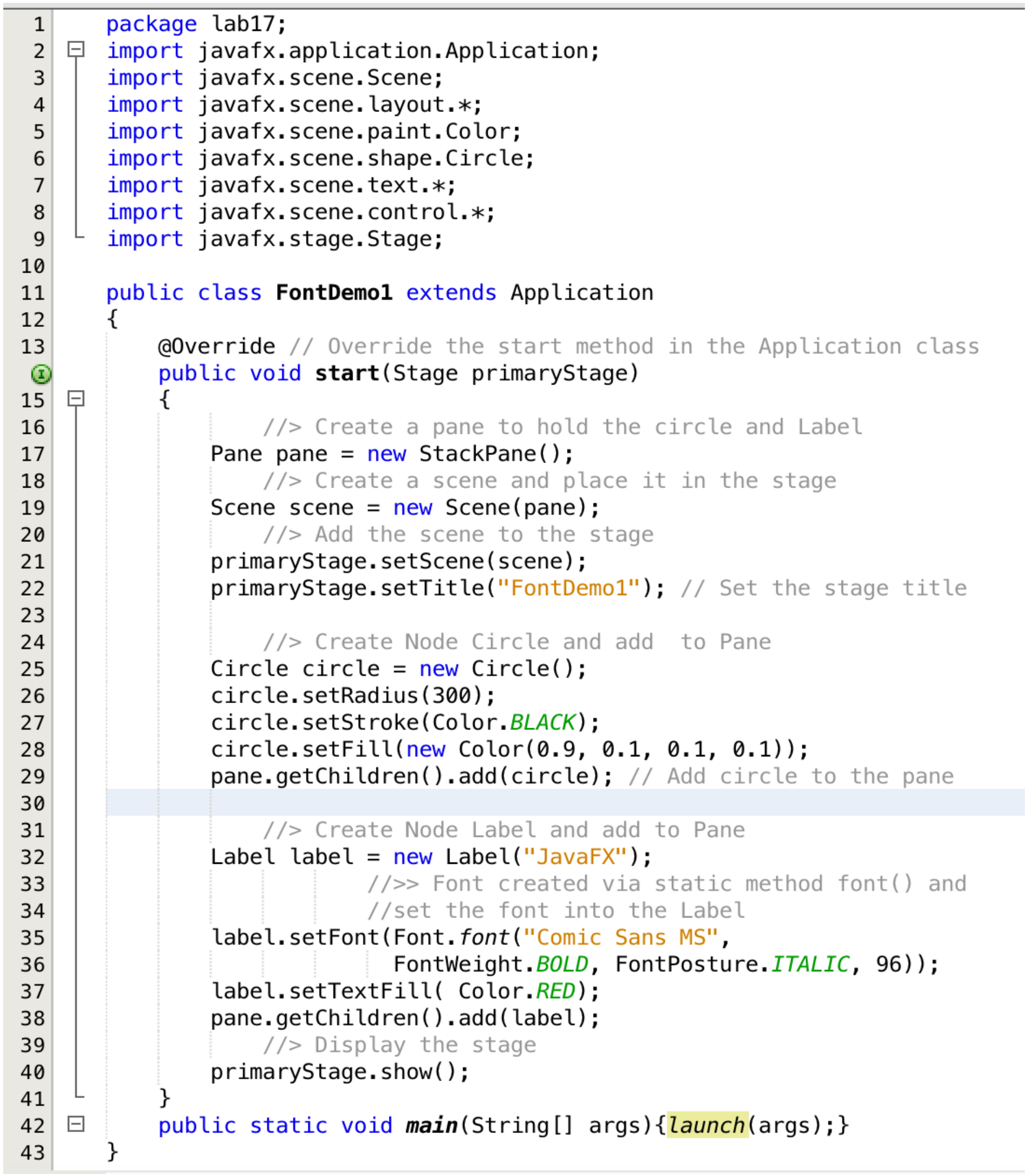


```
        BorderPane pane = new BorderPane();
        pane.setTop(buttonOK);
        pane.setBottom(buttonCancel);
        Scene scene = new Scene(pane);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

```
Compiling 1 source file to /Users/ASDV2/Desktop/slcc/courses/2017spring/
compile-single:
run-single:
Clicked from inside an Anonymous class
Clicked Cancel from inside an Anonymous class
Clicked from inside an Anonymous class
Clicked Cancel from inside an Anonymous class
Clicked from inside an Anonymous class
```

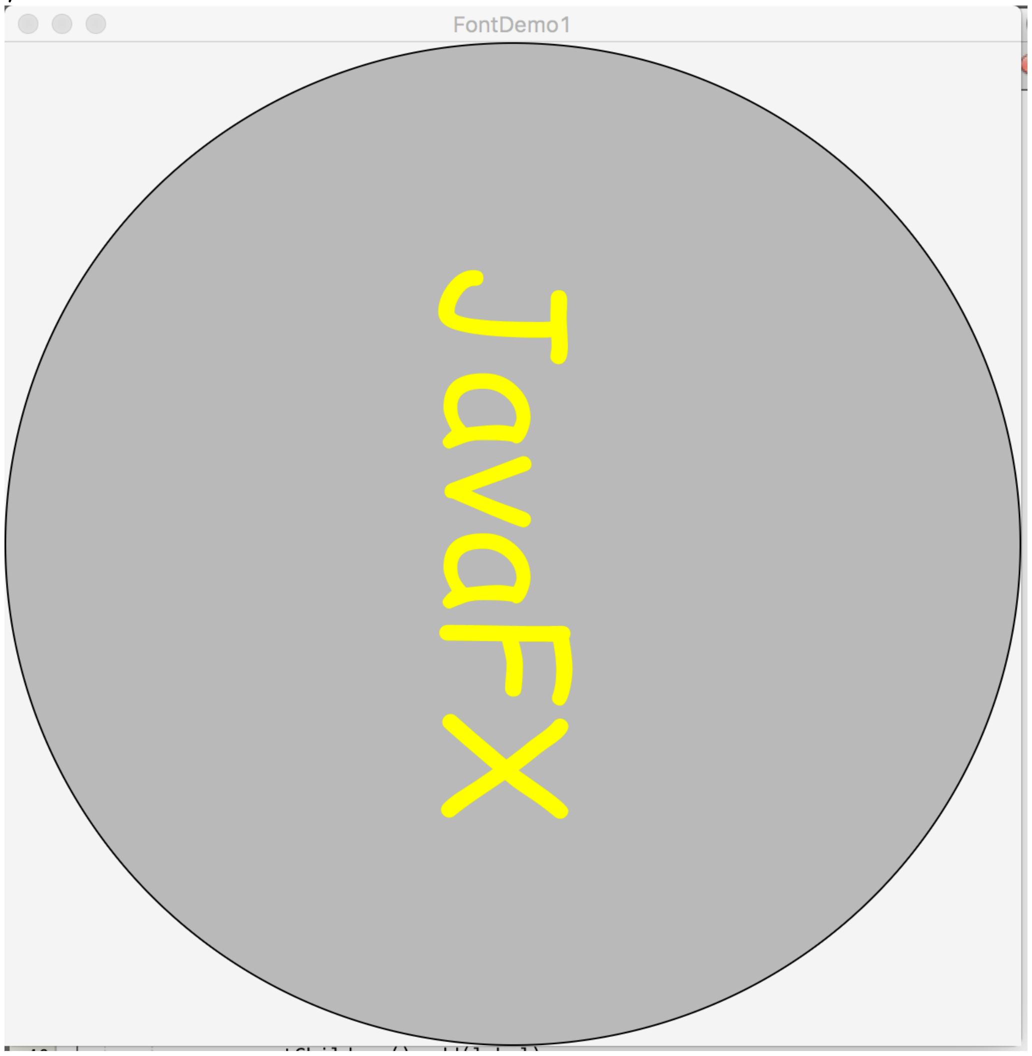
Fonts

24. Create the class FontDemo1 below. The class add to pane a Circle, and a label to the circle. It prints the label using font Comic MS of size 96.



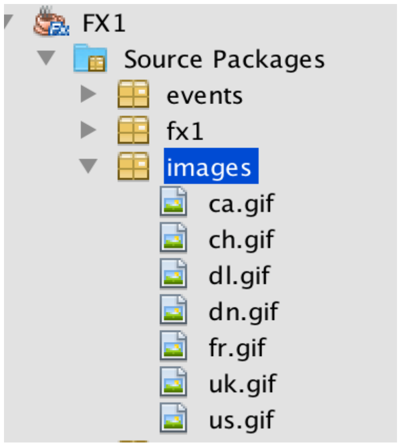
```java
package lab17;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.text.*;
import javafx.scene.control.*;
import javafx.stage.Stage;

public class FontDemo1 extends Application
{
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage)
    {
            //> Create a pane to hold the circle and Label
        Pane pane = new StackPane();
            //> Create a scene and place it in the stage
        Scene scene = new Scene(pane);
            //> Add the scene to the stage
        primaryStage.setScene(scene);
        primaryStage.setTitle("FontDemo1"); // Set the stage title

            //> Create Node Circle and add  to Pane
        Circle circle = new Circle();
        circle.setRadius(300);
        circle.setStroke(Color.BLACK);
        circle.setFill(new Color(0.9, 0.1, 0.1, 0.1));
        pane.getChildren().add(circle); // Add circle to the pane

            //> Create Node Label and add to Pane
        Label label = new Label("JavaFX");
                    //>> Font created via static method font() and
                    //set the font into the Label
        label.setFont(Font.font("Comic Sans MS",
                      FontWeight.BOLD, FontPosture.ITALIC, 96));
        label.setTextFill( Color.RED);
        pane.getChildren().add(label);
            //> Display the stage
        primaryStage.show();
    }
    public static void main(String[] args){launch(args);}
}
```

25. Create the class FontDemo2. Create a gray background for the circle, Yellow text for the label, and rotated as shown below.



Images

26. Instead of a Pane we will use an Hbox to add the American flag. **Images must** to be .gif.
    In your project, create a folder and name it images. Add all the flags posted in canvas to that folder.



FX1
Source Packages
events
fx1
images
ca.gif
ch.gif
dl.gif
dn.gif
fr.gif
uk.gif
us.gif

We will use Insets for padding within a Node:

Insets(double top, double right, double bottom, double left)
Constructs a new Insets instance with four different offsets.

We will use Hbox instead of Pane

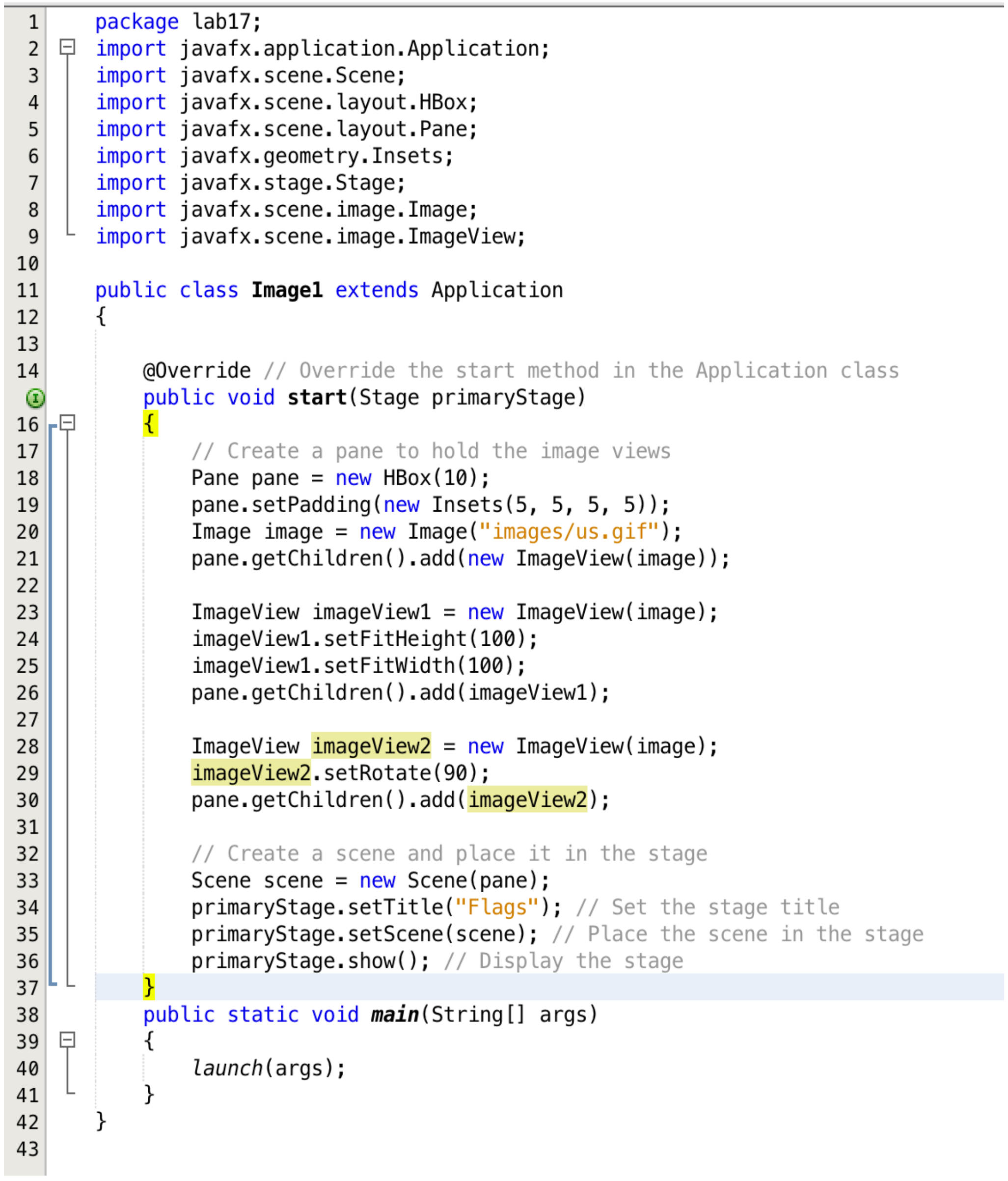
```
public class HBox
extends Pane
```

HBox lays out its children in a single horizontal row. If the hbox has a border and/or padding set, then the contents will be layed out within those insets.

HBox example:

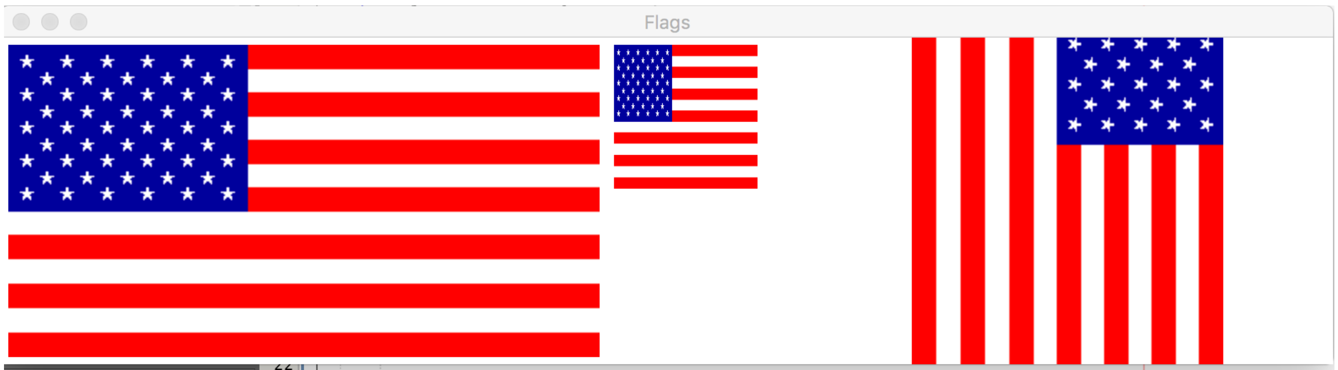
```
HBox hbox = new HBox(8); // spacing = 8
hbox.getChildren().addAll(new Label("Name:), new TextBox());
```

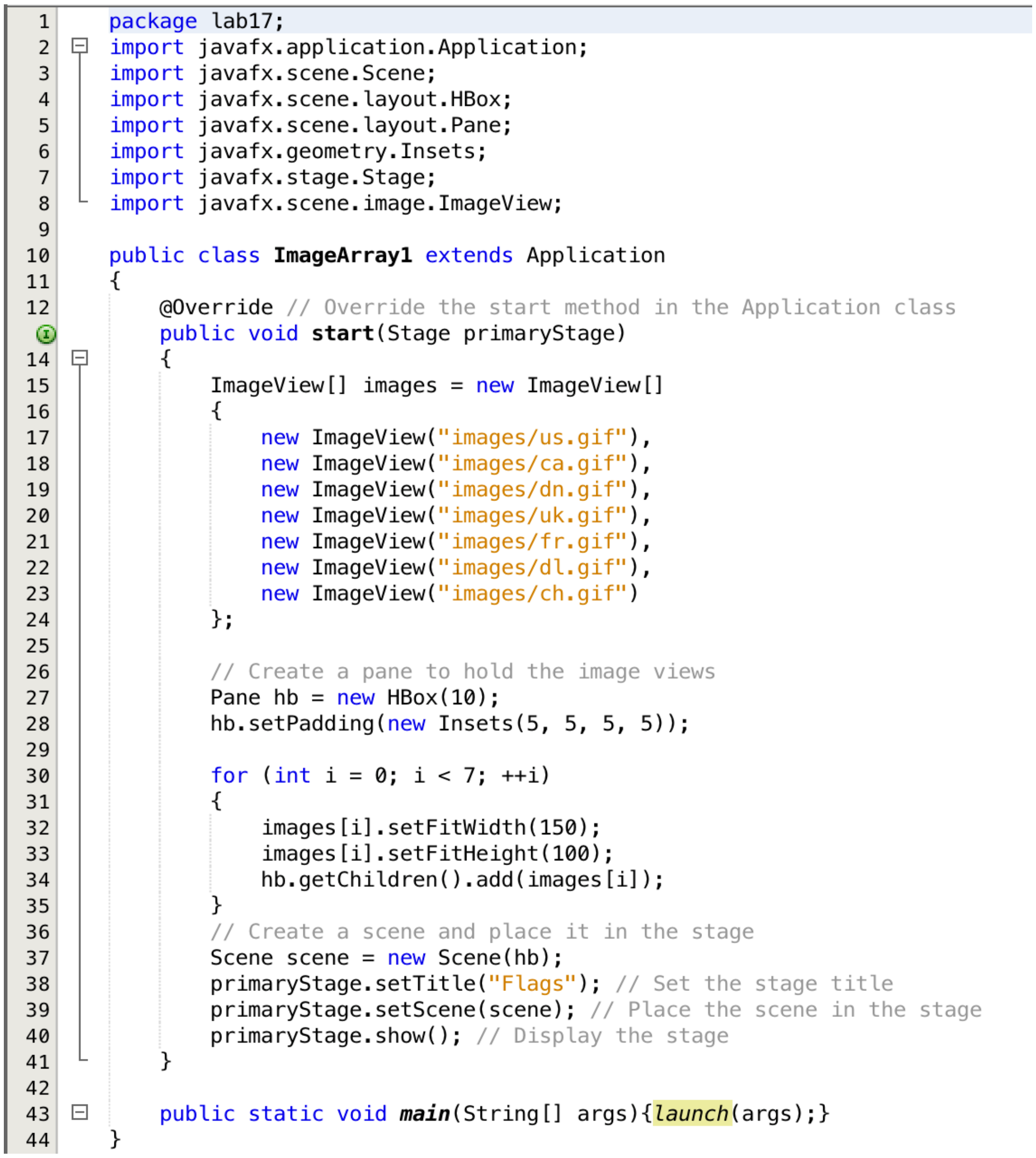
27. Create class Image1. Display the American flag 3 times inside an Hbox. Insets is the padding within the cells of the Hbox.



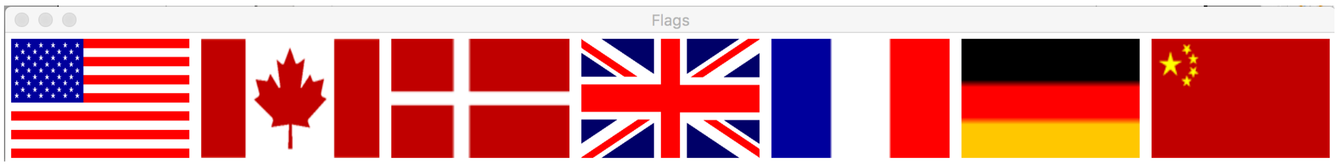
```java
package lab17;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;

public class Image1 extends Application
{

    @Override // Override the start method in the Application class
    public void start(Stage primaryStage)
    {
        // Create a pane to hold the image views
        Pane pane = new HBox(10);
        pane.setPadding(new Insets(5, 5, 5, 5));
        Image image = new Image("images/us.gif");
        pane.getChildren().add(new ImageView(image));

        ImageView imageView1 = new ImageView(image);
        imageView1.setFitHeight(100);
        imageView1.setFitWidth(100);
        pane.getChildren().add(imageView1);

        ImageView imageView2 = new ImageView(image);
        imageView2.setRotate(90);
        pane.getChildren().add(imageView2);

        // Create a scene and place it in the stage
        Scene scene = new Scene(pane);
        primaryStage.setTitle("Flags"); // 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);
    }
}

```

28. Create class ImageArray1.The class display an array of flags of the same size inside an Hbox.

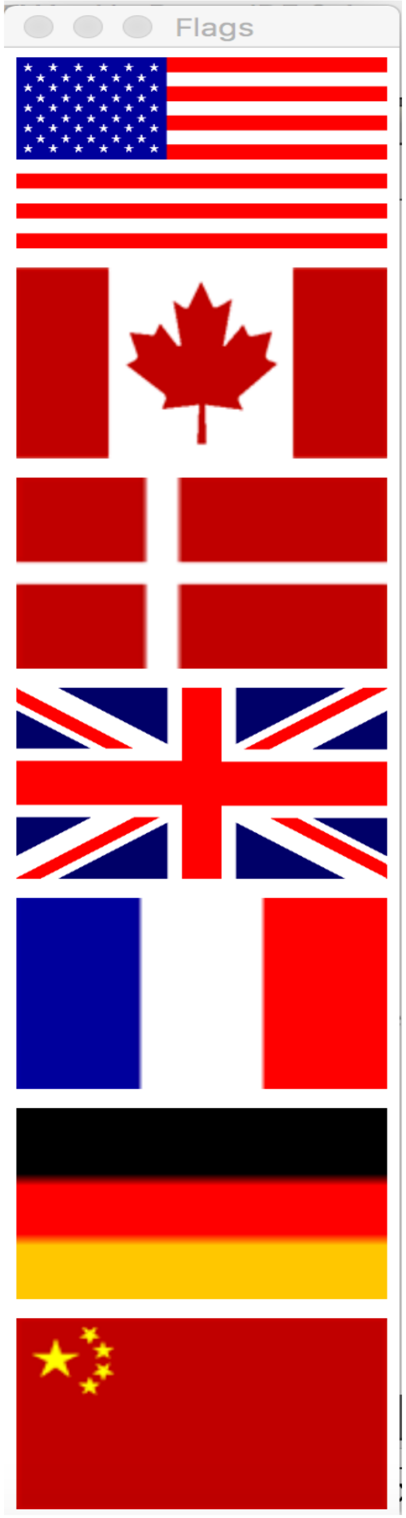




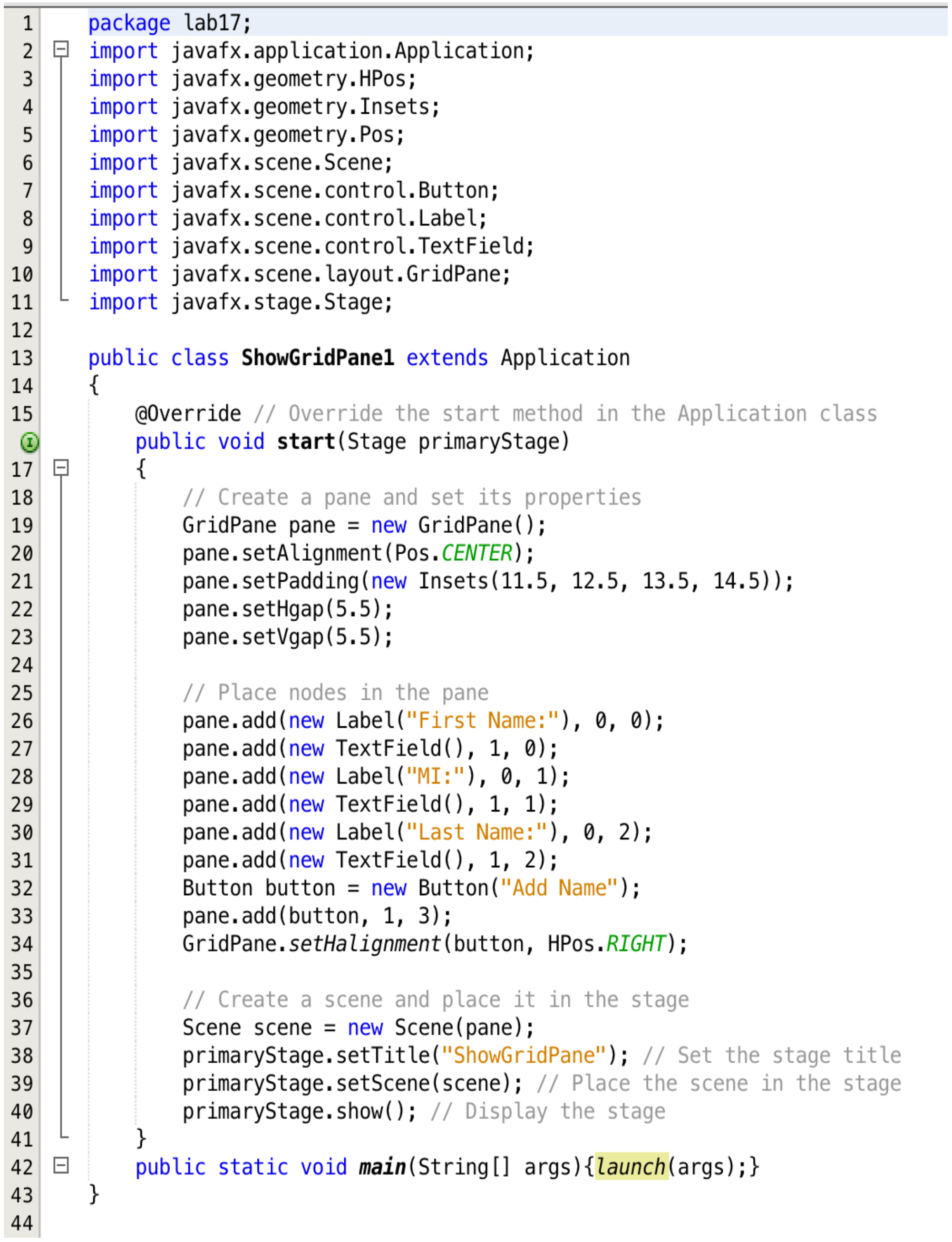
```
package lab17;
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.image.ImageView;

public class ImageArray1 extends Application
{
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage)
    {
        ImageView[] images = new ImageView[]
        {
            new ImageView("images/us.gif"),
            new ImageView("images/ca.gif"),
            new ImageView("images/dn.gif"),
            new ImageView("images/uk.gif"),
            new ImageView("images/fr.gif"),
            new ImageView("images/dl.gif"),
            new ImageView("images/ch.gif")
        };

        // Create a pane to hold the image views
        Pane hb = new HBox(10);
        hb.setPadding(new Insets(5, 5, 5, 5));

        for (int i = 0; i < 7; ++i)
        {
            images[i].setFitWidth(150);
            images[i].setFitHeight(100);
            hb.getChildren().add(images[i]);
        }
        // Create a scene and place it in the stage
        Scene scene = new Scene(hb);
        primaryStage.setTitle("Flags"); // 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);}
}
```



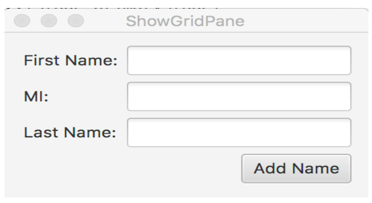
29. Create class ImageArray2.The class displays an array of flags of the same size inside a VBox. The VBox only differs form the HBox in that it add its nodes vertically.



30. Create a class ShowGridePane1. GridPane lays out its children within a flexible grid of rows and columns. If a border and/or padding is set, then its content will be layed out within those insets.

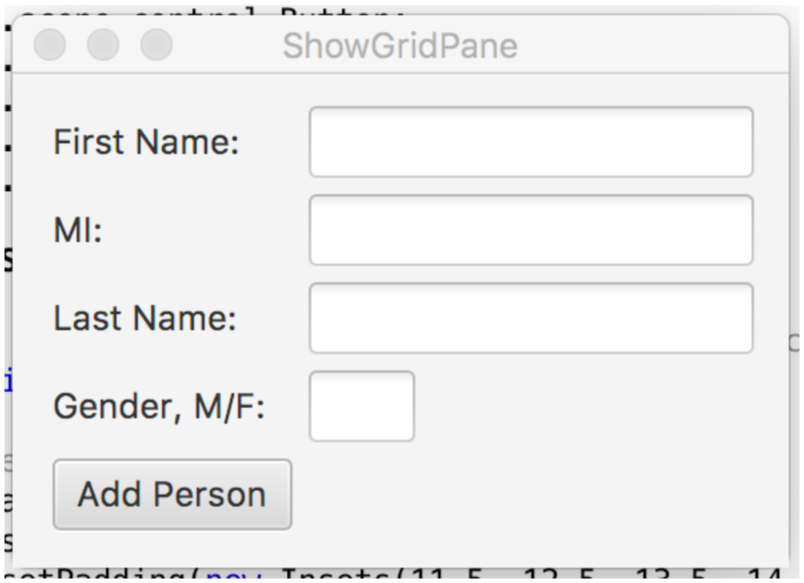


```java
package lab17;
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class ShowGridPane1 extends Application
{
    @Override // Override the start method in the Application class
    public void start(Stage primaryStage)
    {
        // Create a pane and set its properties
        GridPane pane = new GridPane();
        pane.setAlignment(Pos.CENTER);
        pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
        pane.setHgap(5.5);
        pane.setVgap(5.5);

        // Place nodes in the pane
        pane.add(new Label("First Name:"), 0, 0);
        pane.add(new TextField(), 1, 0);
        pane.add(new Label("MI:"), 0, 1);
        pane.add(new TextField(), 1, 1);
        pane.add(new Label("Last Name:"), 0, 2);
        pane.add(new TextField(), 1, 2);
        Button button = new Button("Add Name");
        pane.add(button, 1, 3);
        GridPane.setHalignment(button, HPos.RIGHT);

        // Create a scene and place it in the stage
        Scene scene = new Scene(pane);
        primaryStage.setTitle("ShowGridPane"); // 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);}
}
```

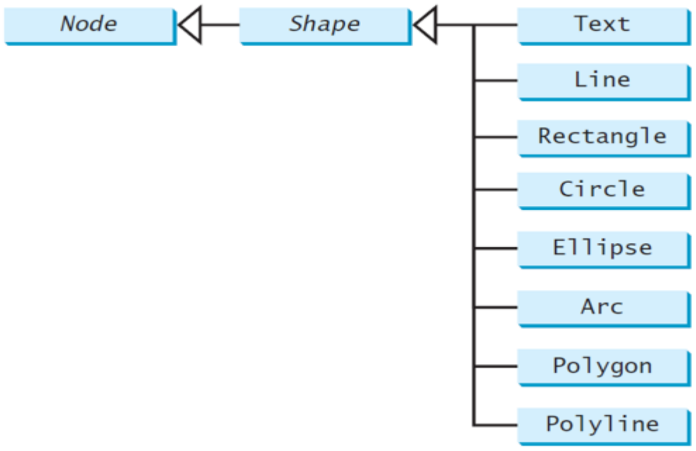


31. Create a class ShowGridePane2. Modify your GridPane1 to add a new textfield for the gender and place the button to the left as shown below.

For the size of the gender textfield	`TextField tf = new TextField();`
	`tf.setMaxSize(40, 20);`
	`pane.add(tf, 1, 3);`

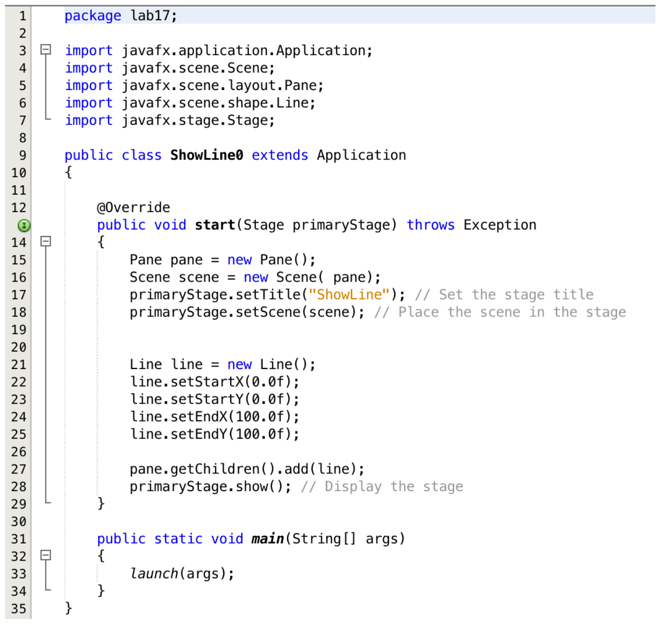


SHAPES and Binding

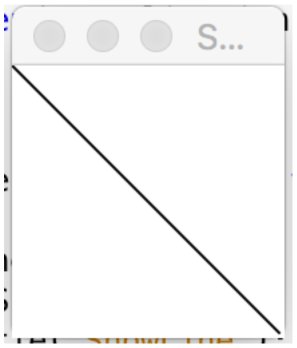


We already experimented with shape Rectangle. Now will experiment with Line Text and Arc

32. Create a class Line0. Implement and understand the starting end ending points of the line in the code shown below. SIZE the WINDOW of your output. You will size that the line REMAINS the same size.



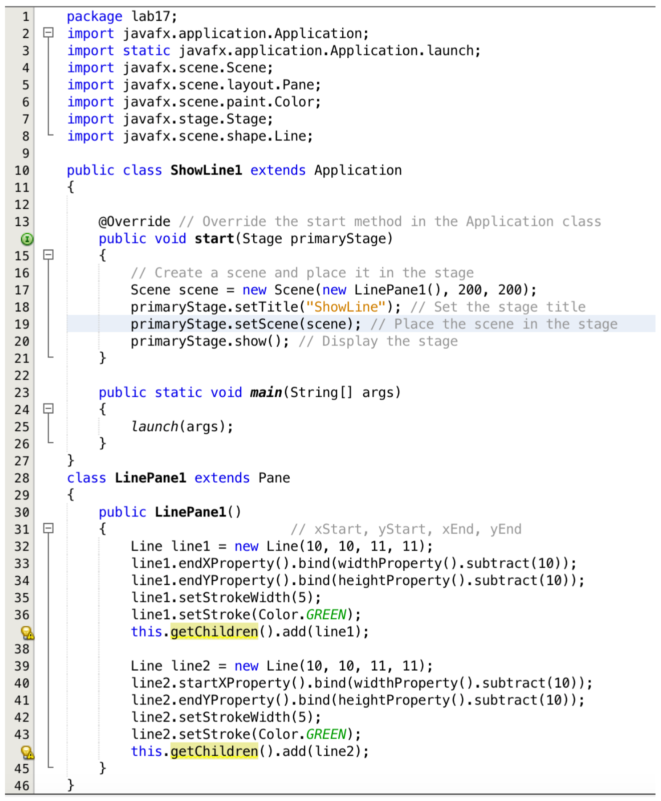
```java
package lab17;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.shape.Line;
import javafx.stage.Stage;

public class ShowLine0 extends Application
{

    @Override
    public void start(Stage primaryStage) throws Exception
    {
        Pane pane = new Pane();
        Scene scene = new Scene( pane);
        primaryStage.setTitle("ShowLine"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage


        Line line = new Line();
        line.setStartX(0.0f);
        line.setStartY(0.0f);
        line.setEndX(100.0f);
        line.setEndY(100.0f);

        pane.getChildren().add(line);
        primaryStage.show(); // Display the stage
    }

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



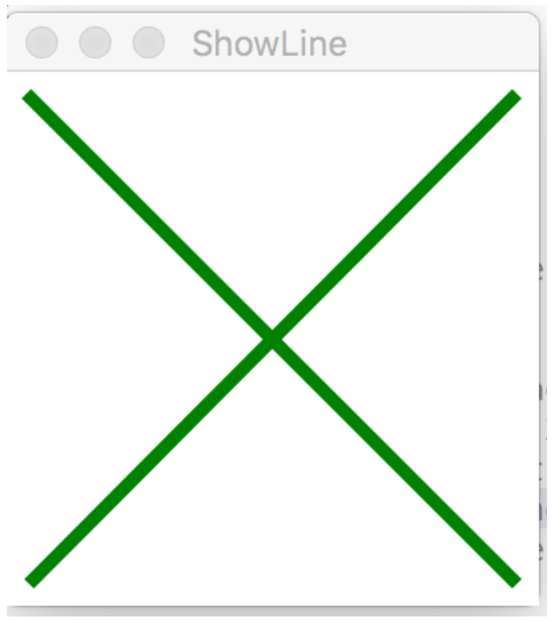
33. Create class Line1.We will remedy the problem with sizing the window of Line0. When the window increases then we could increase the size of the line if we please. It is done by BINDING the start and the end of the line to the sizes of the window which is a Pane. Type in the code given. The line always starts at point (10, 11)

Line 33, binds the endX property of line to the **width** of the Pane. When the Pane increases or decreases then the line increases or decreases.

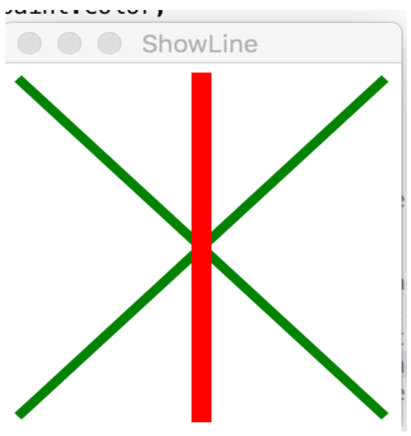
Line 33, binds the endY property of line to the **height** of the Pane. When the Pane increases or decreases then the line increases or decreases.



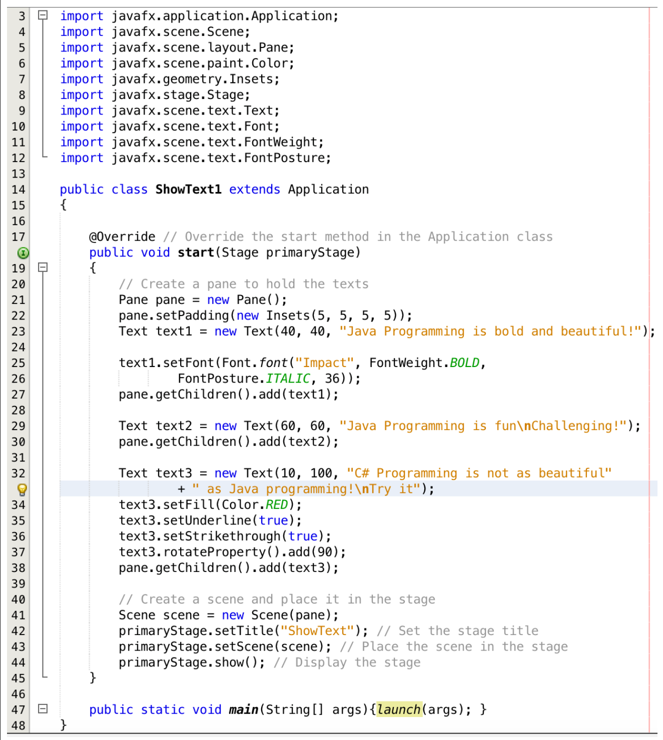
```
 1  package lab17;
 2  import javafx.application.Application;
 3  import static javafx.application.Application.launch;
 4  import javafx.scene.Scene;
 5  import javafx.scene.layout.Pane;
 6  import javafx.scene.paint.Color;
 7  import javafx.stage.Stage;
 8  import javafx.scene.shape.Line;
 9
10  public class ShowLine1 extends Application
11  {
12
13      @Override // Override the start method in the Application class
        public void start(Stage primaryStage)
15      {
16          // Create a scene and place it in the stage
17          Scene scene = new Scene(new LinePane1(), 200, 200);
18          primaryStage.setTitle("ShowLine"); // Set the stage title
19          primaryStage.setScene(scene); // Place the scene in the stage
20          primaryStage.show(); // Display the stage
21      }
22
23      public static void main(String[] args)
24      {
25          launch(args);
26      }
27  }
28  class LinePane1 extends Pane
29  {
30      public LinePane1()
31      {                       // xStart, yStart, xEnd, yEnd
32          Line line1 = new Line(10, 10, 11, 11);
33          line1.endXProperty().bind(widthProperty().subtract(10));
34          line1.endYProperty().bind(heightProperty().subtract(10));
35          line1.setStrokeWidth(5);
36          line1.setStroke(Color.GREEN);
            this.getChildren().add(line1);
38
39          Line line2 = new Line(10, 10, 11, 11);
40          line2.startXProperty().bind(widthProperty().subtract(10));
41          line2.endYProperty().bind(heightProperty().subtract(10));
42          line2.setStrokeWidth(5);
43          line2.setStroke(Color.GREEN);
            this.getChildren().add(line2);
45      }
46  }
```



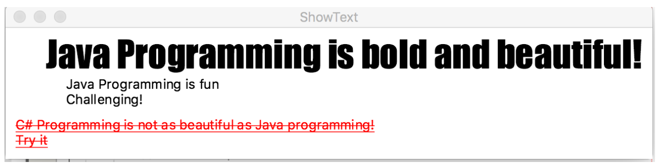
34. Create class Line2. Modify class Line1 and add a vertical red line of stroke 10 which as shown below which increases and decrease as the window is sized.



35. Create a class ShowText1. Test the code.



```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.FontPosture;

public class ShowText1 extends Application
{

    @Override // Override the start method in the Application class
    public void start(Stage primaryStage)
    {
        // Create a pane to hold the texts
        Pane pane = new Pane();
        pane.setPadding(new Insets(5, 5, 5, 5));
        Text text1 = new Text(40, 40, "Java Programming is bold and beautiful!");

        text1.setFont(Font.font("Impact", FontWeight.BOLD,
                FontPosture.ITALIC, 36));
        pane.getChildren().add(text1);

        Text text2 = new Text(60, 60, "Java Programming is fun\nChallenging!");
        pane.getChildren().add(text2);

        Text text3 = new Text(10, 100, "C# Programming is not as beautiful"
                + " as Java programming!\nTry it");
        text3.setFill(Color.RED);
        text3.setUnderline(true);
        text3.setStrikethrough(true);
        text3.rotateProperty().add(90);
        pane.getChildren().add(text3);

        // Create a scene and place it in the stage
        Scene scene = new Scene(pane);
        primaryStage.setTitle("ShowText"); // 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); }
}
```



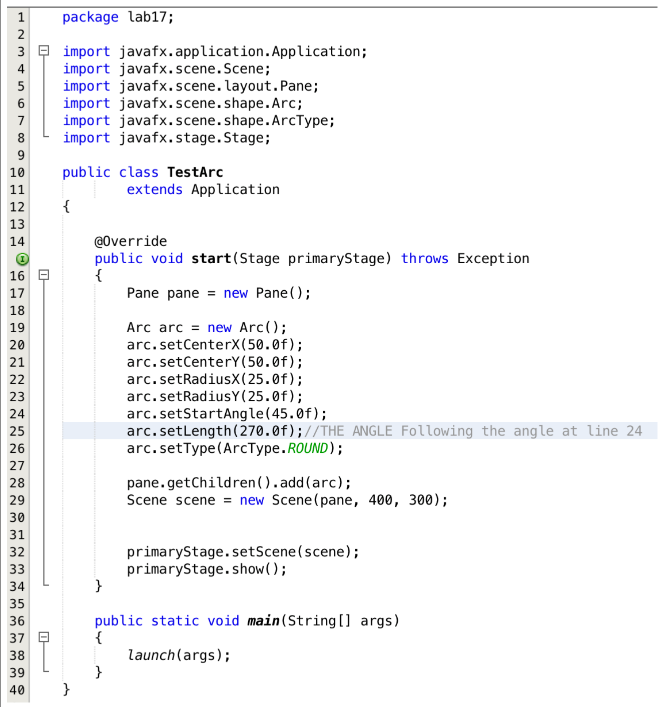
ShowText

Java Programming is bold and beautiful!

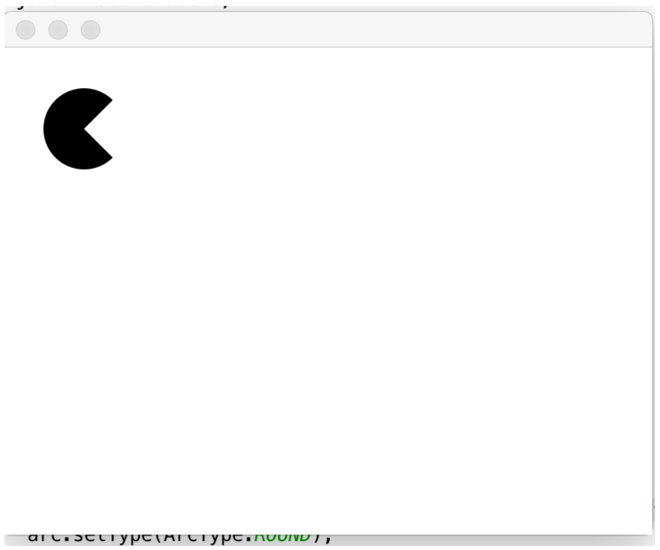
Java Programming is fun
Challenging!

~~C# Programming is not as beautiful as Java programming!~~
~~Try it~~

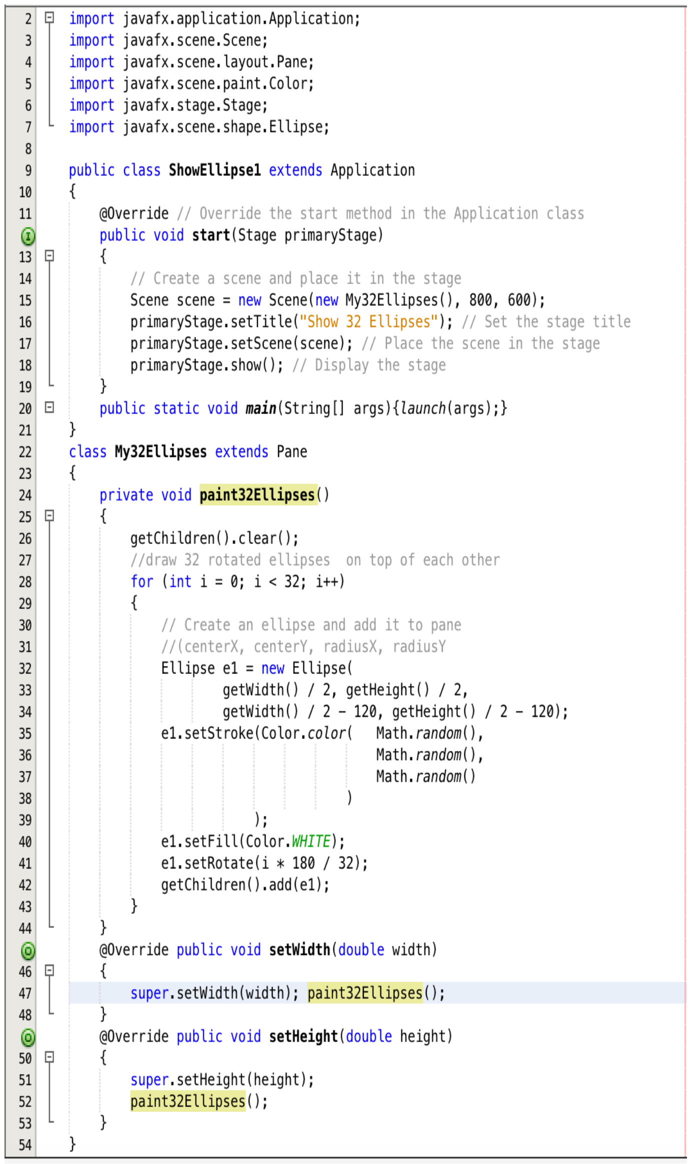
36. Create a class TestArc. Arc is like an ellipse but shows only parts of the perimeter, either solid arc or just the arc. We will draw a solid arc. Study lines 20 to 27 for its creation which are self-explanatory. We start at 45 degrees(line 24) and we add an angle of 270 degrees(line 25). And we have a Pacman.



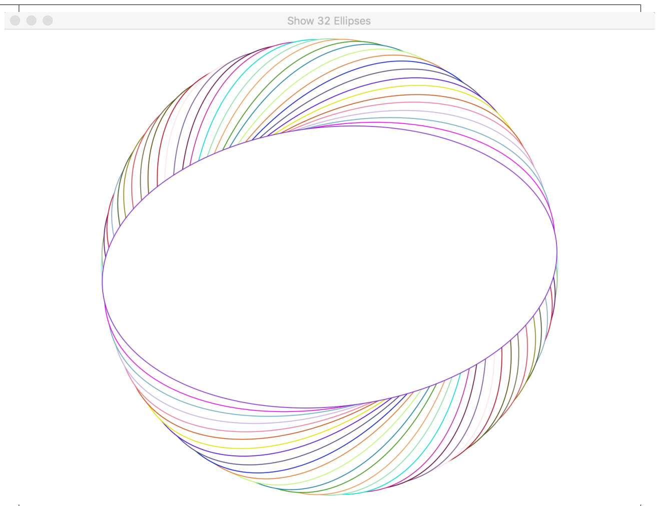
```java
1   package lab17;
2
3   import javafx.application.Application;
4   import javafx.scene.Scene;
5   import javafx.scene.layout.Pane;
6   import javafx.scene.shape.Arc;
7   import javafx.scene.shape.ArcType;
8   import javafx.stage.Stage;
9
10  public class TestArc
11          extends Application
12  {
13
14      @Override
15      public void start(Stage primaryStage) throws Exception
16      {
17          Pane pane = new Pane();
18
19          Arc arc = new Arc();
20          arc.setCenterX(50.0f);
21          arc.setCenterY(50.0f);
22          arc.setRadiusX(25.0f);
23          arc.setRadiusY(25.0f);
24          arc.setStartAngle(45.0f);
25          arc.setLength(270.0f);//THE ANGLE Following the angle at line 24
26          arc.setType(ArcType.ROUND);
27
28          pane.getChildren().add(arc);
29          Scene scene = new Scene(pane, 400, 300);
30
31
32          primaryStage.setScene(scene);
33          primaryStage.show();
34      }
35
36      public static void main(String[] args)
37      {
38          launch(args);
39      }
40  }
```

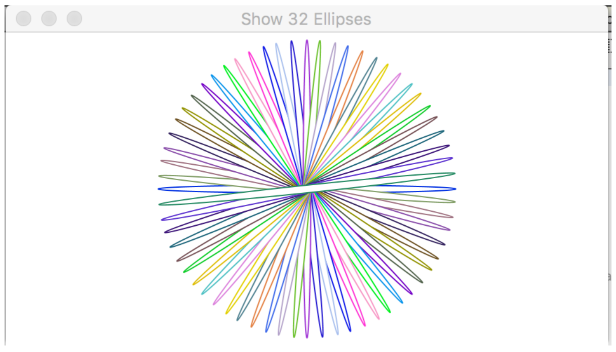


37. Create class <u>ShowEllipse1</u>. The class <u>My32Ellipses</u> that starts at line 22 is derived from <u>Pane</u>. However, we do not bind the coordinates of the ellipses to the Pane. What we do is we overload the methods <u>setWidth</u> and <u>setSize</u> of the the parent(super) of Pane (Node). When the window is sized these methods are called and the size of the ellipses increase or decrees depending on winch way we size the window. The creation of each ellipse is explained in comment lines 31, 32. Run the program and size the ellipse SMALL so the ellipses look like a star. It is a beautiful star.



```
 2  import javafx.application.Application;
 3  import javafx.scene.Scene;
 4  import javafx.scene.layout.Pane;
 5  import javafx.scene.paint.Color;
 6  import javafx.stage.Stage;
 7  import javafx.scene.shape.Ellipse;
 8
 9  public class ShowEllipse1 extends Application
10  {
11      @Override // Override the start method in the Application class
        public void start(Stage primaryStage)
13      {
14          // Create a scene and place it in the stage
15          Scene scene = new Scene(new My32Ellipses(), 800, 600);
16          primaryStage.setTitle("Show 32 Ellipses"); // Set the stage title
17          primaryStage.setScene(scene); // Place the scene in the stage
18          primaryStage.show(); // Display the stage
19      }
20      public static void main(String[] args){launch(args);}
21  }
22  class My32Ellipses extends Pane
23  {
24      private void paint32Ellipses()
25      {
26          getChildren().clear();
27          //draw 32 rotated ellipses  on top of each other
28          for (int i = 0; i < 32; i++)
29          {
30              // Create an ellipse and add it to pane
31              //(centerX, centerY, radiusX, radiusY
32              Ellipse e1 = new Ellipse(
33                      getWidth() / 2, getHeight() / 2,
34                      getWidth() / 2 - 120, getHeight() / 2 - 120);
35              e1.setStroke(Color.color(   Math.random(),
36                                          Math.random(),
37                                          Math.random()
38                                      )
39                          );
40              e1.setFill(Color.WHITE);
41              e1.setRotate(i * 180 / 32);
42              getChildren().add(e1);
43          }
44      }
        @Override public void setWidth(double width)
46      {
47          super.setWidth(width); paint32Ellipses();
48      }
        @Override public void setHeight(double height)
50      {
51          super.setHeight(height);
52          paint32Ellipses();
53      }
54  }
```


Show 32 Ellipses


Show 32 Ellipses