ADSV 2420, Advanced Programming I

REMEMBER TO CLEAN AND BUILD before you run any class. JavaFX uses the last created JAR. Any changes to your code will not be reflected on the screen unless you clean and build.

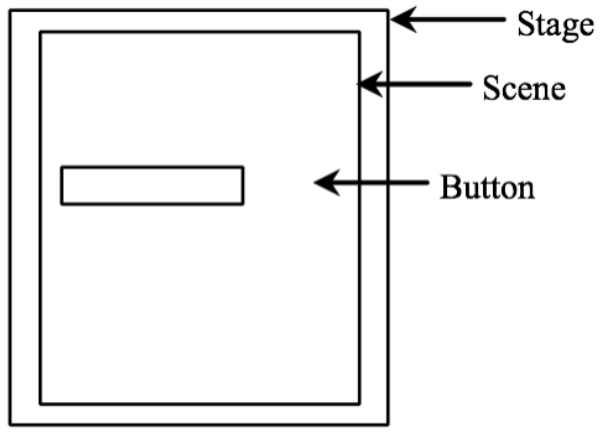
1. Use Netbeans and Create a new project named **FX1:**
   File > New Project > JavaFX > JavaFX Application.
   You will get the code given below.

Explanation of code: The class *FX1* is derived from class *Application* and we override method *start* . The method *start* takes as a parameter *primaryStage* which is of type *Stage.*

1. We create a *Button* (line 15) and we add to the button a listener (lines 17-25) using an anonymous class (new *EventHandler).* We will see anonymous classes a little later.
2. We create a *StackPane* and we add the *Button* to the children of the *StackPane.* (lines28-28)
3. We create a *Scene* and we add the *StackPane to the Scene.* (line 30)
4. We add the *Scene to the Stage* (line 33)
5. We show the *Stage* (line 34)

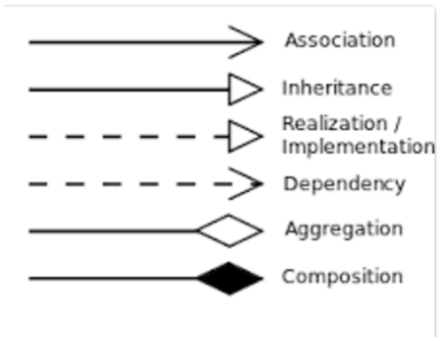
Go through every line of code, and understand it. Run the program.

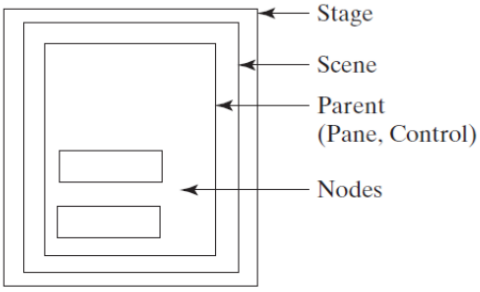
```
1   package fx1;
2   import javafx.application.Application;
3   import javafx.event.ActionEvent;
4   import javafx.event.EventHandler;
5   import javafx.scene.Scene;
6   import javafx.scene.control.Button;
7   import javafx.scene.layout.StackPane;
8   import javafx.stage.Stage;
9
10  public class FX1 extends Application
11  {
12      @Override
13      public void start(Stage primaryStage)
14      {
15          Button btn = new Button();
16          btn.setText("Say 'Hello World'");
17          btn.setOnAction(new EventHandler<ActionEvent>()
18          {
19
20              @Override
21              public void handle(ActionEvent event)
22              {
23                  System.out.println("Hello World!");
24              }
25          });
26
27          StackPane root = new StackPane();
28          root.getChildren().add(btn);
29
30          Scene scene = new Scene(root, 300, 250);
31
32          primaryStage.setTitle("Hello World!");
33          primaryStage.setScene(scene);
34          primaryStage.show();
35      }
36
37      /**...3 lines */
40      public static void main(String[] args)
41      {
42          launch(args);
43      }
44
45  }
```

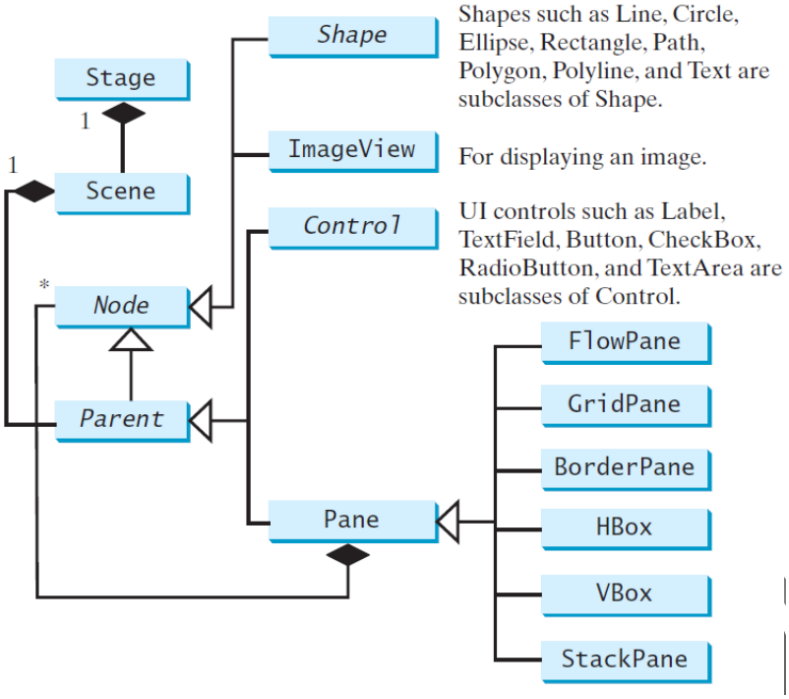


Understanding of JavaFX Architecture

1. The Stage HAS-A a Scene
2. The Scene HAS-A Parent that IS-A Node.
3. Pane could have (HAS-A) many Node(s) such as Shape(s) ImageView(s)
4. Control (Button, TextField) IS-A Parent. Parent IS-A Node
5. FlowPane, GridPane etc, IS-A Pane. Pane IS-A Parent. Parent IS-A Node.



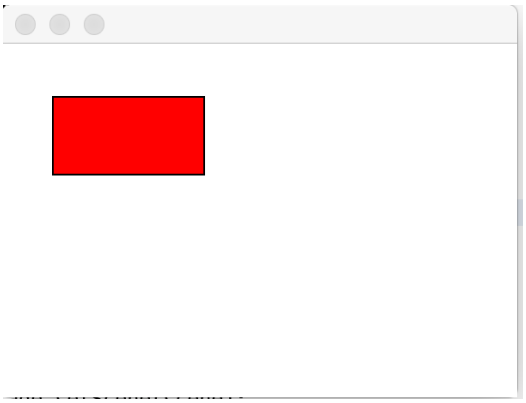




2. Add a new class to your package and name it *ShowCirlce* .
Run it to display a red circle with blue perimeter of radius 20, centered at x=150 and y = 80.

```
1   package fx1;
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.scene.shape.Circle;
7   import javafx.stage.Stage;
8
9   public class ShowCircle
10          extends Application
11  {
12
13      @Override
14      public void start(Stage primaryStage)
15      {
16          Pane pane = new Pane();
17          Circle c = new Circle( 20 ) ;
18          c.setCenterX(150);
19          c.setCenterY( 80 );
20          pane.getChildren().add( c );
21          Scene  scene = new Scene( pane, 300, 200);
22          c.setStroke( Color.BLUE);
23          c.setFill(Color.RED);
24          primaryStage.setScene(scene);
25          primaryStage.show();
26      }
27       public static void main(String[] args)
28      {
29          launch(args);
30      }
31  }
```

3. Add a new class to your package and name it *ShowRectangle* as you did with ShowCircle.
Use **Rectangle r = new Rectangle(30,30, 88, 44);** to create a rectangle at x=30, y=30, width =88, height = 44.Make its perimeter (stroke) black, and its body (fill) red.



4. The following code uses the *binding property* that enables a *target object* **Circle displayed on screen** to be bound to a *source object* ***Circle in memory***. If the value in the object ***Circle in memory*** changes, then the target property of **Circle displayed on screen** is also changed automatically. The target object is simply called a *binding object* or a *binding property*. In lines 21 and 22 we **bind** the center of the circle (x, y) to half of the width and hight of the containing Pane, so the circle is always centered. Size(increase/decrease) the Window (Pane) and you will see that circle increases/ decreases and remains centered.

```
1   package fx1;
2
3   import javafx.application.Application;
4   import javafx.scene.Scene;
5   import javafx.scene.layout.Pane;
6   import javafx.scene.paint.Color;
7   import javafx.scene.shape.Circle;
8   import javafx.stage.Stage;
9
10  public class ShowCircleCentered extends Application
11  {
12
13      @Override // Override the start method in the Application class
14      public void start(Stage primaryStage)
15      {
16          // Create a pane to hold the circle
17          Pane pane = new Pane();
18
19          // Create a circle and set its properties
20          Circle circle = new Circle();
21          circle.centerXProperty().bind( pane.widthProperty().divide(2) );
22          circle.centerYProperty().bind( pane.heightProperty().divide(2) );
23          circle.setRadius(50);
24          circle.setStroke(Color.BLACK);
25          circle.setFill(Color.WHITE);
26          pane.getChildren().add(circle); // Add circle to the pane
27
28          // Create a scene and place it in the stage
29          Scene scene = new Scene(pane, 200, 200);
30          primaryStage.setTitle("ShowCircleCentered"); // Set the stage title
31          primaryStage.setScene(scene); // Place the scene in the stage
32          primaryStage.show(); // Display the stage
33      }
34
35      /**
36       * The main method is only needed for the IDE with limited JavaFX support.
37       * Not needed for running from the command line.
38       */
39      public static void main(String[] args)
40      {
41          launch(args);
42      }
43  }
```

5. Add a new class ShowRectangleProportional. Bind its width and height properties to the width and height of the pane respectively as show below:

```
r.widthProperty().bind( pane.widthProperty().divide(2) );
r.heightProperty().bind( pane.heightProperty().divide(2) );
```

The default (x,y) coordinates are (0,0) for its left top corner. Make the rectangle display at (50, 50). The rectangle is green with blue perimeter. Use setX, setY to display the rectangle at (50, 50).

6. One direction Binding. The code below **binds** d1 to d2. If d2 changes, then d1 changes. Create the class and run the program.

```
1  package fx1;
2
3  import javafx.beans.property.DoubleProperty;
4  import javafx.beans.property.SimpleDoubleProperty;
5
6  public class BindingDemo
7  {
8
9      public static void main(String[] args)
10     {
11         DoubleProperty d1 = new SimpleDoubleProperty(1);
12
13         DoubleProperty d2 = new SimpleDoubleProperty(2);
14
15         d1.bind(d2);
16         System.out.println("d1 is " + d1.getValue()
17                 + " and d2 is " + d2.getValue());
18
19         d2.setValue(70.2);
20         System.out.println("d1 is " + d1.getValue()
21                 + " and d2 is " + d2.getValue());
22         //d1.set(33); //remove the comment to crash the program.
23         //When you bind you cannot change. Is like getting married.
24     }
25 }
26
```

7. Bidirectional binding. The code below **binds** d1 to d2 and d2 to d1. If d1 changes, then d2 changes and if d2 changes then d1 changes. Create the class and run the program.

```
1   package fx1;
2
3   import javafx.beans.property.DoubleProperty;
4   import javafx.beans.property.SimpleDoubleProperty;
5
6   public class BidirectionalBindingDemo
7   {
8
9       public static void main(String[] args)
10      {
11          DoubleProperty d1 = new SimpleDoubleProperty(1);
12          DoubleProperty d2 = new SimpleDoubleProperty(2);
13
14          System.out.println("d1 is " + d1.getValue()
15                  + " and d2 is " + d2.getValue());
16          d1.bindBidirectional(d2);
17          System.out.println("d1 is " + d1.getValue()
18                  + " and d2 is " + d2.getValue());
19          d1.setValue(50.1);
20          System.out.println("d1 is " + d1.getValue()
21                  + " and d2 is " + d2.getValue());
22          d2.setValue(70.2);
23          System.out.println("d1 is " + d1.getValue()
24                  + " and d2 is " + d2.getValue());
25      }
26  }
27
```

8. (10 points) Create a class Checkerboard that displays a checkerboard in which each square is of type Rectangle with a fill color yellow or blue. Your checkerboard should start at (0,0). The size of the screen should be initially 800x800. Bind the width and height of each square to 1/10 of the screen width and height (800/10 = 80), which makes the size of the square width=80, height=80. Hint: Use an 8x8 2D array of type Rectangle and set the properties (binding-width, binding-height, x, y) of each rectangle inside your nested for loops.