Assignment 2

- Final spec update was yesterday, please make sure you are using the latest spec.
- Check the changes in the spec, they are mostly clarifications, or covering edge cases that were previously undefined.
- The due date for the assignment has been extended to 9am on Tuesday, October 2nd (Note that October 1st is a public holiday).
- Make the most of this week (where all pracs are assignment help sessions), because it will be hard to get help during the holidays.

Testing in Assignment 2

- You have been asked to write tests for the Action class (ActionTest) and SparseTileArray class (SparseTileArrayTest).
- For Action, there are two cases that you have not done in the previous assignment:
  1. Testing that the correct output is written to System.out.
  2. Testing that an object can be loaded from a BufferedReader.
- Correct output can be tested by using the `System.setOut()` (see `OutputTest.java`)
- Input can be given to a BufferedReader using StringReader:

```
BufferedReader br = new BufferedReader(new
StringReader("my input string"));
```

Week 9.1 — GUIs

GUIs?

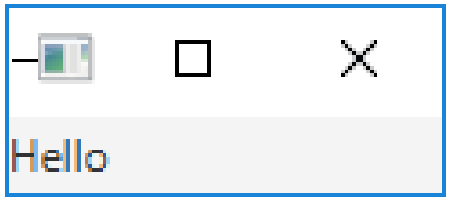
- Why GUIs?
  - Programs people *directly* interact with often have them.
  - They are a nice example of OO.
  - Useful from the perspective of event-driven programming.
- Extra considerations:
  - Getting GUI's "wrong" is very easy - there is a lot of complexity which goes into designing good GUIs (which we won't be covering in this course).

Java GUIs

Java first party GUI toolkits:

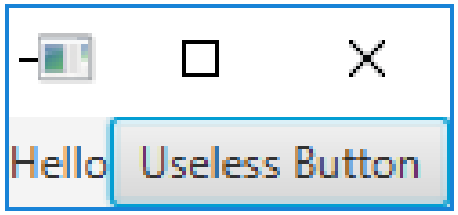
- A.W.T. Abstract Window Toolkit (`java.awt`)
  - In Java1.0
  - Designed to look like native applications
  - Relative positioning
- Swing (`javax.swing`)
  - Uses custom widgets
  - Looks the same everywhere - "Java" visual style, but can be themed
  - Could do fancier things - 2d, 3d, animation
  - Not actively developed anymore
- **JavaFX**
  - Uses newer Java features
  - We're using this one
    - We are not using all of it
  - Things it supports which we won't be using:
    - CSS
    - JFXML

HelloGUI.java




```
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.stage.Stage;

public class HelloGui extends javafx.application.Application {

  public static void main(String[] args) {
    launch(args); //Static method inherited from Application
    //Creates an object of this class and calls start()
  }

  // Called to start doing application stuff
  public void start(Stage stage) {
    stage.setTitle("Greetings");
    Label lab=new Label("Hello ");
    Scene scene=new Scene(lab); // can only hold one "control"
    stage.setScene(scene);
    stage.setHeight(60);
    stage.setWidth(60);
    stage.show();

  }
}
```

HelloGUI2



- Use layout panes (in `javafx.scene.layout`) to group `Nodes` together into a single "control".
  - This can be done recursively to form a graph.
- Note `col, row` ordering for placement within grid.
- Java will try to identify the minimum sensible size for the window.

`HelloGUI2.java`

Main points of layout panes

- Content forms a graph of nodes (scene graph)
  - Controls
  - Panes, which provide grids
  - Panes can be nested
- The graph is added to a scene (there could be multiple scenes)
- A window is a stage
- A stage can display one scene at a time
- Program doesn't automatically end when start() completes
  - it will respond to events (if it has the programming to support it)

Controls

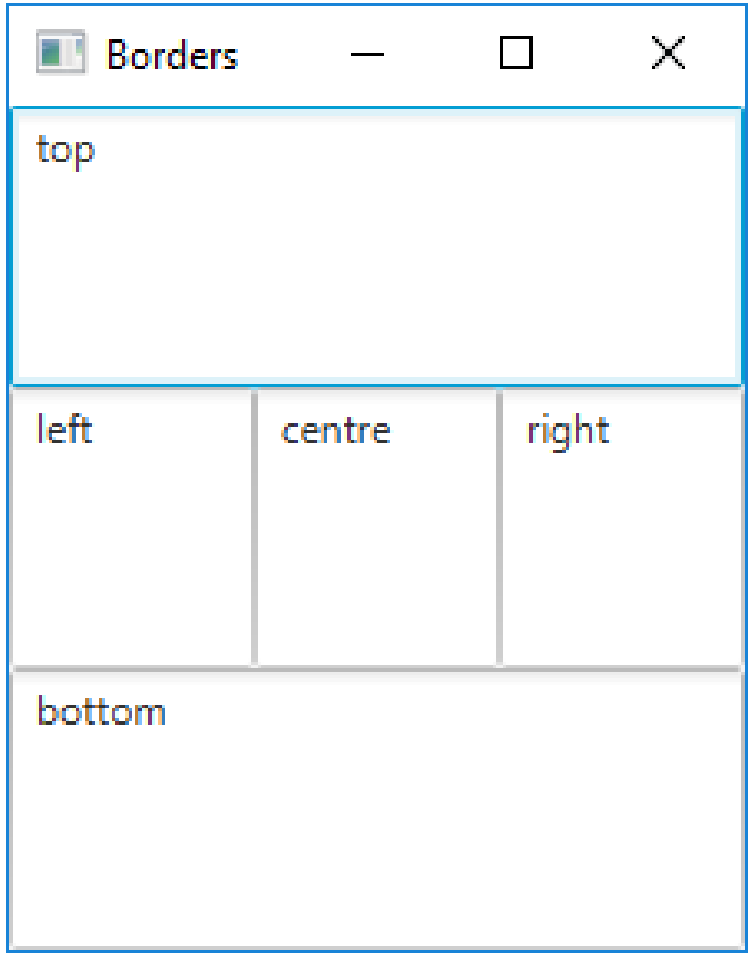
In the `javafx.scene.control`.

- Button
- Label
- TextArea
- TextField

Pane

javafx.scene.layout.Pane

`BorderGui.java`



This example uses the BorderPane class.

We can:

- control behaviour when resizing
- include or leave out any combination

of panes

For anyone familiar with other Java GUI approaches:
javafx doesn't have a layout manager class, the Pane class acts as the layout manager.

"Ugly" Panes

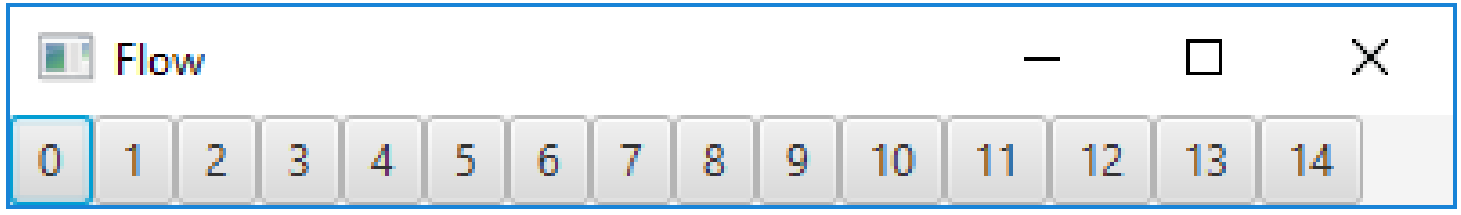
javafx.scene.layout has many options for laying out GUIs:

- FlowPane
- HBox: Organises elements horizontally
- VBox: Organises elements vertically
- ...

javafx.scene.layout

Classes

AnchorPane
Background
BackgroundFill
BackgroundImage
BackgroundPosition
BackgroundSize
Border
BorderImage
BorderPane
BorderStroke
BorderStrokeStyle
BorderWidths
ColumnConstraints
ConstraintsBase
CornerRadii
CornerRadiiConverter
FlowPane
GridPane
HBox
Pane
Region
RowConstraints
StackPane
TilePane
VBox



`FlowGui.java`

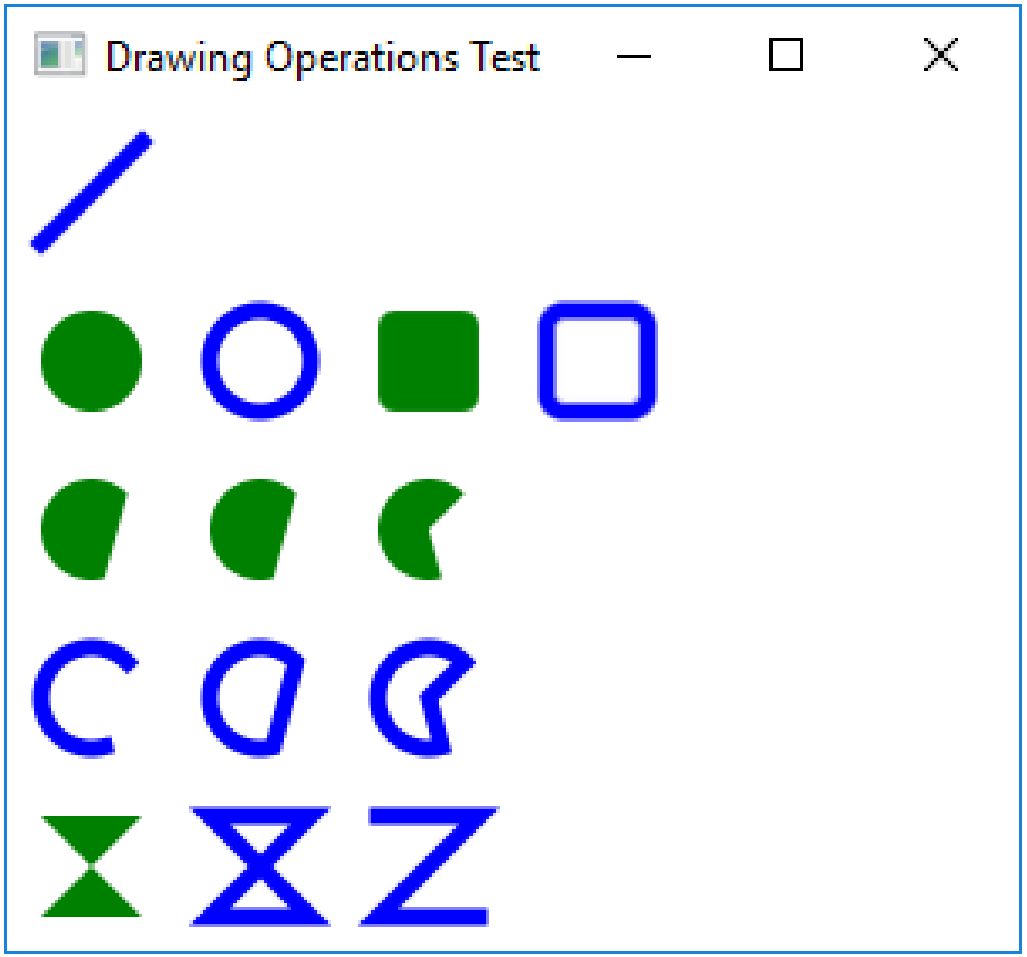
Flow GUI allows us to add child elements to a list. The GUI is then generated automatically by placing elements horizontally (or vertically) while there is room.

"Children" in a GUI are components within an element.

Canvas

A canvas is an interface element which allows us to draw programmatically.

`BasicOpsTest.java`



Source: https://docs.oracle.com/javase/8/javafx/graphics-tutorial/canvas.htm