Graphical User Interface

Han-fen Hu

Tony Gaddis (2019) Starting Out with Java: From Control Structures through Data Structures, 4th Edition

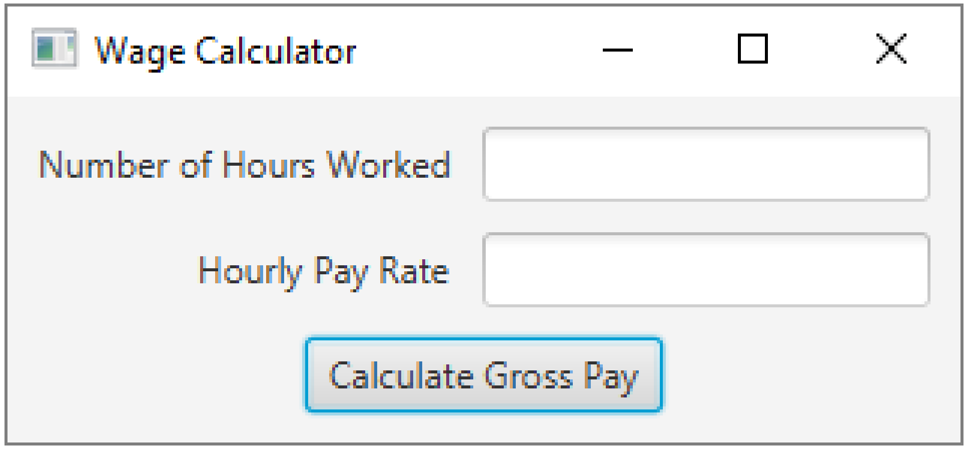
UNLV

Outline

- Graphical User Interfaces
- Introduction to JavaFX
- Process of Creating a JavaFX Program
- Event Handling
- Layout Management

Graphical User Interfaces (1)

- A GUI is a graphical window or windows that provide interaction with the user
- A window in a GUI commonly consists of several controls that present data to the user and/or allow interaction with the application.
  - Some of the common GUI controls are buttons, labels, text fields, check boxes, and radio buttons.



Graphical User Interfaces (2)

- Programs that operate in a GUI environment must be event-driven
  - An event is an action that takes place within a program, such as the clicking of a button
- Part of writing a GUI application is creating event listeners
  - An event listener is a method that automatically executes when a specific event occurs

Introduction to JavaFX (1)

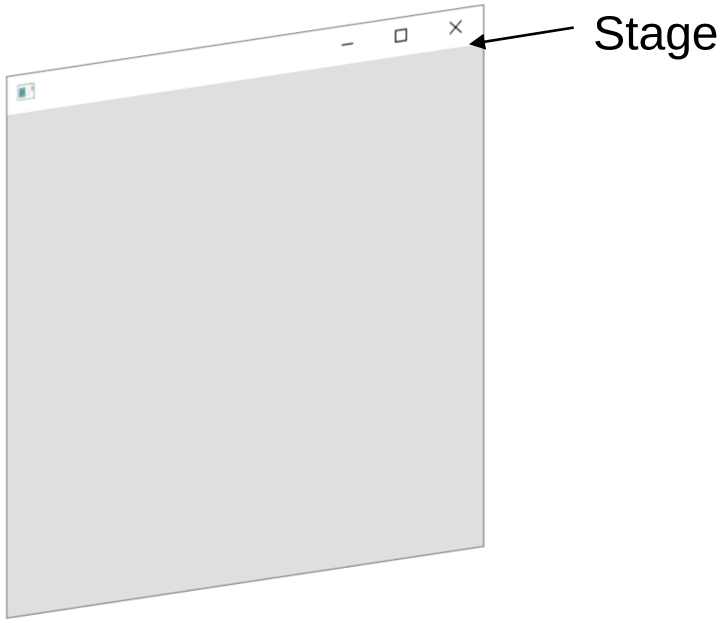
- JavaFX is a Java library for developing rich applications that employ graphics.
  - GUI applications, as well as applications that display 2D and 3D graphics
  - Standalone graphics applications that run on your local computer
  - Applications that run from a remote server
  - Applications that are embedded in a Web page

Introduction to JavaFX (2)

- JavaFX uses a theater metaphor to describe the structure of a GUI.
  - A theater has a stage
  - On the stage, a scene is performed by actors

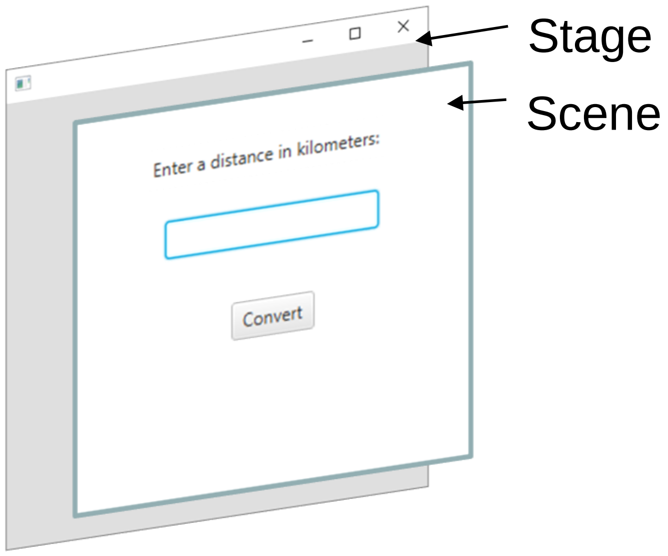
Introduction to JavaFX (3)

- In JavaFX, a stage is an empty window



Introduction to JavaFX (4)

- The scene is a collection of GUI objects (controls) that are contained within the window.
- You can think of the GUI objects as the actors that make up the scene.



Introduction to JavaFX (5)

- The Application Class
  - An abstract class that is the foundation of all JavaFX applications
  - JavaFX applications must extend the Application class
  - The Application class has an abstract method named start, which is the entry point for the application
  - Because the start method is abstract, you must override it

General Layout of a JavaFX Program

- Various import statements
- A class that extends the `Application` class
- A `start` method
- A `main` method

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
Other import statements...

public class ClassName extends Application {
   public static void main(String[] args){
      // Launch the application.
      launch(args);
   }

   @Override
   public void start(Stage primaryStage) {
      // Insert startup code here.
   }
}
```

UNLV

Layout of a JavaFX Program

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
```
Other import statements…

```
public class ClassName extends Application {
   public static void main(String[] args){
      // Launch the application.
      launch(args);
   }

   @Override
   public void start(Stage primaryStage) {
      // Insert startup code here.
   }
}
```

Necessary `import` statements

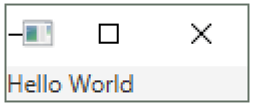
A static `main` method that calls the inherited `launch` method

A class that extends the `Application` class

A `start` method that accepts a Stage argument. This method is called by the inherited `launch` method.

Example of a JavaFX Program

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;

public class HelloWorld extends Application{
  public static void main(String[] args){
      launch(args);
   }

   @Override
   public void start(Stage primaryStage){
      Label messageLabel = new Label("Hello World");   //Make a Label control
      VBox vbox = new VBox(messageLabel);              //Put the Label in a VBox
      Scene scene = new Scene(vbox);     //Make the VBox the root node in the scene
      primaryStage.setScene(scene);      //Set the scene to the stage
      primaryStage.show();               //Show the stage (display it)
   }
}
```


Hello World

Creating Controls

- Process for creating a control
  - Import the class for the control from the necessary javafx package.

```
import javafx.scene.control.Label;
import javafx.scene.control.Button;
```

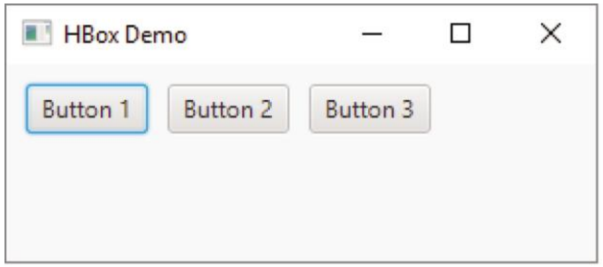
  - Instantiate the class, calling the desired constructor.

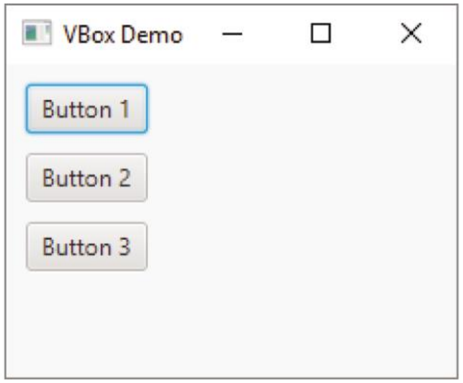
```
Label messageLabel = new Label("Hello World");
Button mybutton = new Button("Click Me");
```

UNLV

Layout Containers

- Layout containers are used to arrange the positions of controls on the screen.
- JavaFX provides many layout containers to arrange the controls (More about the layout containers later)
  - HBox: a single horizontal row
  - VBox: a single vertical row

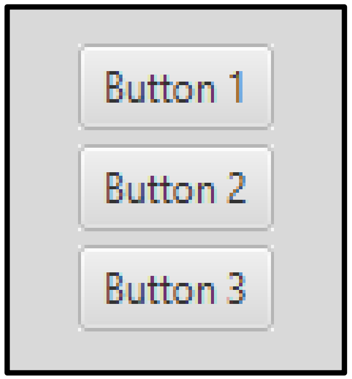




UNLV

Adding Controls to a Layout Container

VBox



```
Button b1 = new Button("Button 1");
Button b2 = new Button("Button 2");
Button b3 = new Button("Button 3");

VBox vbox = new VBox(b1, b2, b3);
```

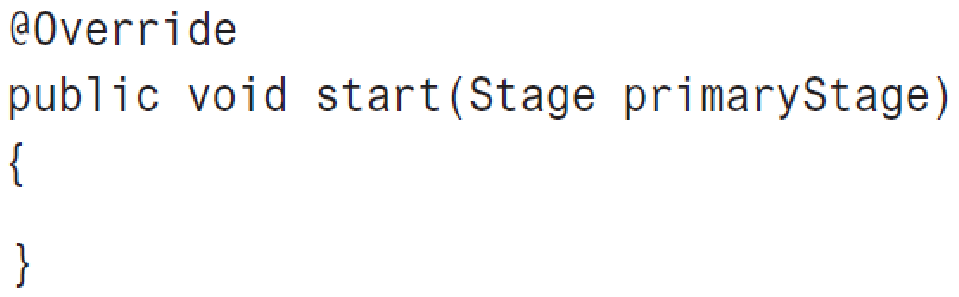
Creating a Scene

- To create a scene, you instantiate the `Scene` class (in `the javafx.scene` package)
- Then, you add root nodes to the scene

```
// Create a Label control.
Label messageLabel = new Label("Hello World");

// Create an HBox container and add the Label.
HBox hbox = new HBox(messageLabel);

// Create a Scene and add the HBox as the root node.
Scene scene = new Scene(hbox);
```

Adding the Scene to the Stage

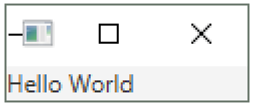
- Once a `Scene` object is created, add it to the application's stage.
  - The stage is an instance of the `Stage` class (from the `javafx.stage` package)
  - You do not have to instantiate the Stage class. It is created automatically, and passed as an argument to the start method.



```
@Override
public void start(Stage primaryStage)
{

}
```

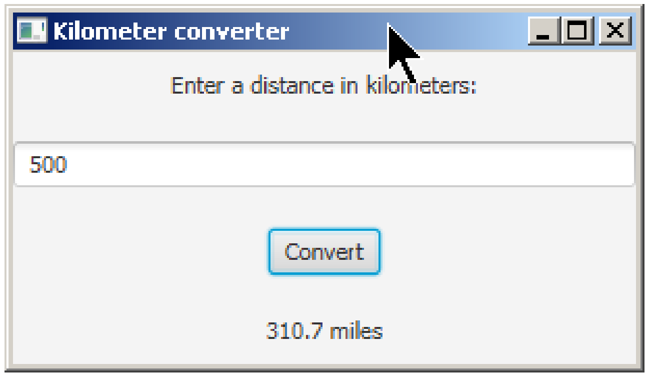
Example of a JavaFX Program

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;

public class HelloWorld extends Application{
  public static void main(String[] args){
      launch(args);
   }

   @Override
   public void start(Stage primaryStage){
      Label messageLabel = new Label("Hello World");     //Make a Label control
      VBox vbox = new VBox(messageLabel);                //Put the Label in a VBox
      Scene scene = new Scene(vbox);     //Make the VBox the root node in the scene
      primaryStage.setScene(scene);      //Set the scene to the stage
      primaryStage.show();               //Show the stage (display it)
   }
}
```


Hello World

Lab (1)

- DistanceConverter.java
  - A simple program for distance conversion



Enter a distance in kilometers:

500

310.7 miles

More About HBox and Vbox (1)

- To add spacing between the items in an HBox or VBox

```
HBox hbox = new HBox(10, label1,
label2, label3);

VBox vbox = new VBox(20, button1,
button2, button3);
```

UNLV

More About HBox and Vbox (2)

- Padding is space that appears around the inside edge of a container.
  - The HBox and VBox containers have a setPadding method.
  - You pass an `Insets` object as an argument to the setPadding method.
  - The Insets object specifies the number of pixels of padding.

```
hbox.setPadding(new Insets(10));
```

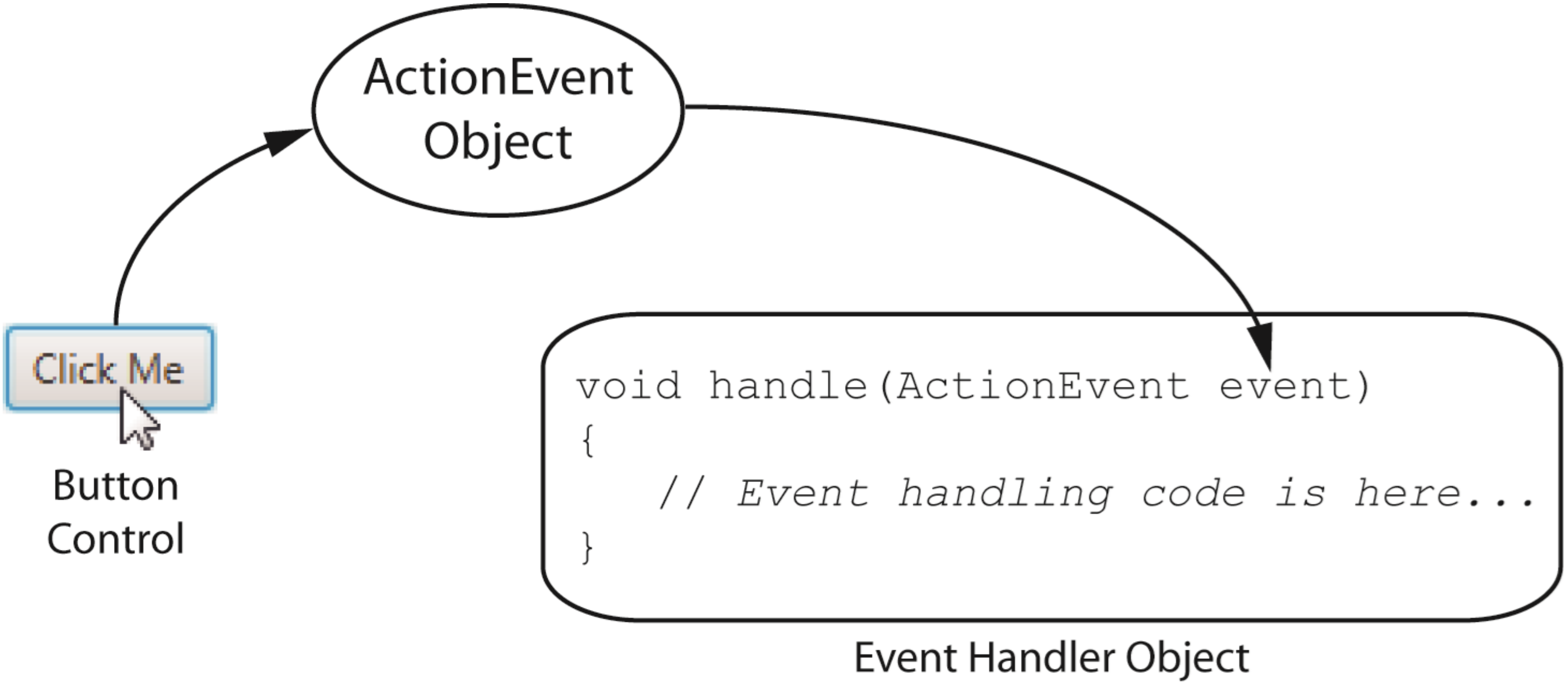
- The `Insets` class is in the `javafx.geometry` package.

UNLV

Handling Events (1)

- An event is an action that takes place within a program, such as the clicking of a button
  - When an event takes place, the control that is responsible for the event creates an event object in memory
  - The event object contains information about the event
- The control that generated the event object is know as the event source
- It is possible that the event source is connected to one or more event listeners

Handling Events (2)



```
void handle(ActionEvent event)
{
    // Event handling code is here...
}
```

Event Objects

- Event objects are instances of the Event class (from the `javafx.event` package), or one of its subclasses.
  - For example, a Button click generates an `ActionEvent` object. `ActionEvent` is a subclass of the `Event` class.

Event Handlers

- Event handlers are objects
- We write event handler classes that implement the `EventHandler` interface (from the `javafx.event` package)
- The `EventHandler` interface specifies a void method named `handle` that has a parameter of the Event class (or one of its subclasses)

```
class ButtonClickHandler implements EventHandler<ActionEvent>{
    @Override
    void handle(ActionEvent event){
        // Write event handling code here.
    }
}
```

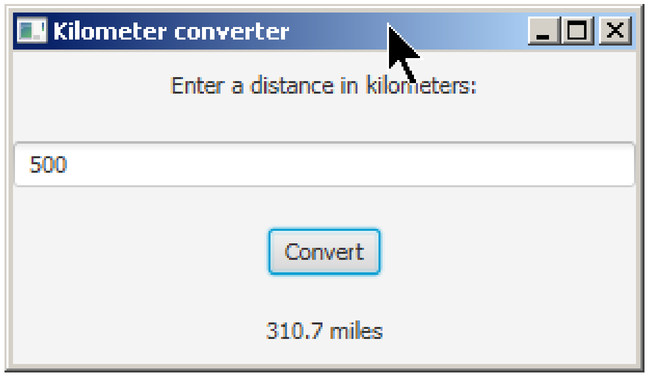
Registering an Event Handler

- The process of connecting an event handler object to a control is called registering the event handler
- When the user clicks the button, the event handler object’s handle method will be executed
- Button controls have a method named setOnAction that registers an event handler:

```
mybutton.setOnAction(new ButtonClickHandler());
```

Lab (2)

- DistanceConverter.java
  - Now we can implement the event



TextField (1)

- At runtime, the user can type text into a TextField control.
- In the program, you can retrieve the text that the user entered.
- The `TextField` class is in the `javafx.scene.control` package.
- To create an empty TextField:

```
TextField myTextField = new TextField();
```

TextField (2)

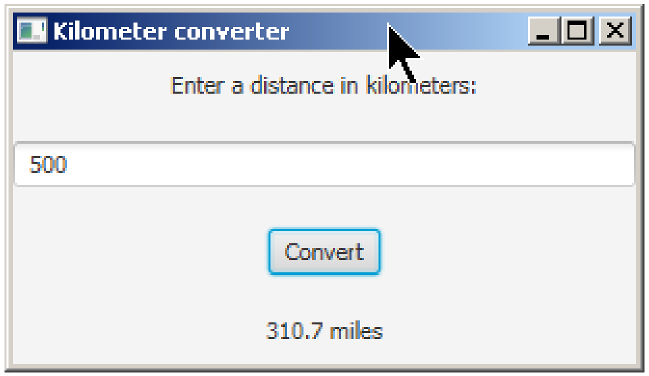
- To retrieve the text that the user has typed into a `TextField` control, call the control’s `getText()` method.
- The method returns the value that has been entered, as a `String`.
- Example

```
String input;
input = myTextField.getText();
```

UNLV

Lab (3)

- DistanceConverter.java
  - Let’s finish up the application



Enter a distance in kilometers:

500

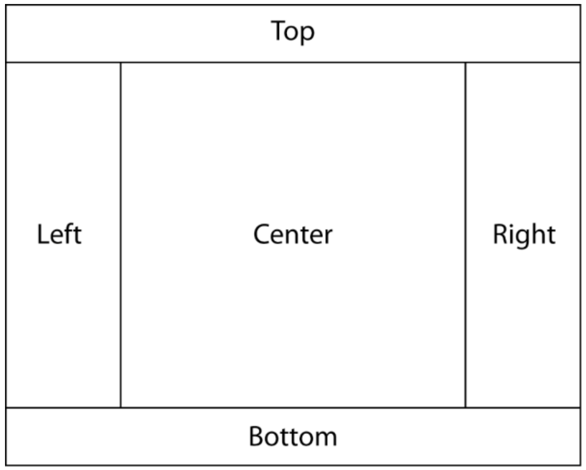
310.7 miles

Layout Panes

- Benefits of using Layout Panes
  - Adjust the user interface automatically to the device/screen size.
- Built-in Layout Panes
  - VBox: Arranges controls vertically in a single column
  - HBox: Arranges controls horizontally in a single row
  - GridPane: Places controls in a grid of rows and columns
  - FlowPane: Lay out the controls children in a flow that wraps at the boundary
  - AnchorPane: Anchor copntrols to the top, bottom, left side, or center of the pane
  - BorderPane: Lays out controls in the top, bottom, right, left, or center region
  - StackPane – Places controls in a back-to-front single stack.
  - TilePane: Places controls in uniformly sized layout cells or tiles
- A layout pane can be added to another layout pane to achieve a more complex user interface design.

BorderPane (1)

- `BorderPane` layout container manages controls in five regions



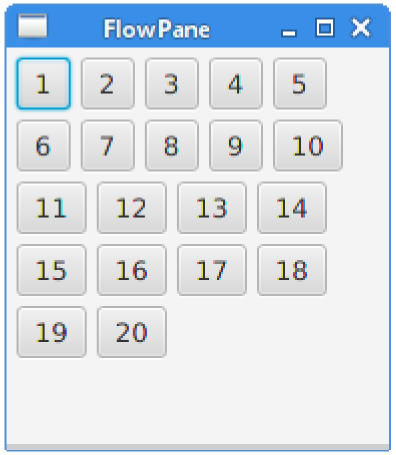
- Only one object at a time may be placed into a `BorderPane` region.
- Typically, you put controls into another type of layout container, then you put that container into one of the `BorderPane` regions.

BorderPane (2)

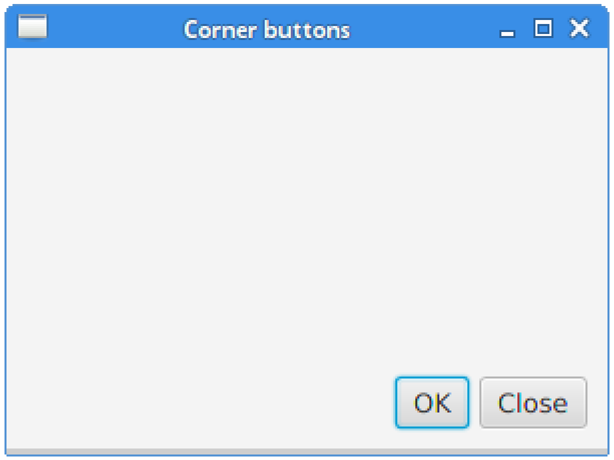
- The BorderPane class provides the following methods to add controls to specific regions:
  - setCenter
  - setTop
  - setBottom
  - setLeft
  - setRight
- Lab
  - BorderPaneExample.java

Other Layout Containers (1)

- Example of FlowPane



- AnchorPane: anchors the edges of child nodes to an offset from the anchor pane's edges

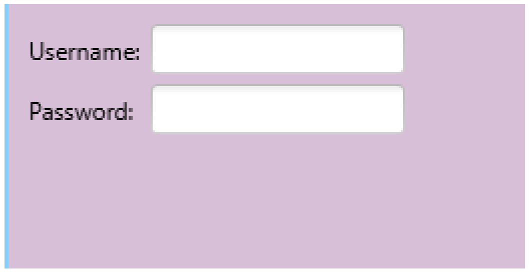


Source: http://zetcode.com/gui/javafx/layoutpanes/

Other Layout Containers (2)

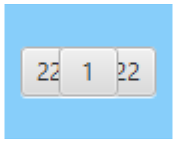
- GridPane

```
GridPane grid = new GridPane();
grid.setPadding(new Insets(10, 10, 10, 10));
Text username = new Text("Username:");
grid.add(username, 0, 0);
TextField text = new TextField();
text.setPrefColumnCount(10); grid.add(text, 1, 0);
Text password = new Text("Password:");
grid.add(password, 0, 1);
TextField text2 = new TextField();
```



- StackPane

```
StackPane root = new StackPane();
Button btn1 = new Button(" 1 ");
Button btn2 = new Button("22222222");
root.getChildren().addAll(btn2, btn1);
root.setStyle("-fx-background-color: #87CEFA;");
```



Nesting Components in a Layout

- Adding components to panels and then nesting the panels inside the regions can overcome the single component limitation of layout regions.
- By adding panes to a region and then adding the objects to the panes, sophisticated layouts can be achieved.