Java FX: Creating a Form

Learning Goal: Understand the basics of screen layout, how to add controls to a layout pane, and how to create input events.

Part A – Create the Project

1) Create a New Java Project called JavaFXForm and include the packages as well as alter the main method as in our HelloWorld example.

   Add the following import statements:

```
 import javafx.scene.layout.*;
import javafx.geometry.*;
import javafx.scene.text.*;
import javafx.scene.control.*;
import javafx.scene.paint.*;
```

2) Type the following code after your main method()

```
@Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("JavaFX Welcome");

        primaryStage.show();
    }
```

3) Run your program – it will look weird/incomplete

Part B – Create a GridPane Layout

A GridPane layout enables you to create a flexible grid of rows and columns in which to lay out controls. You can place controls in any cell in the grid, and you can make controls span cells as needed.

4) Add this code below `primaryStage.show();`

```
GridPane grid = new GridPane();
grid.setAlignment(Pos.CENTER);  //change the default position from top left to center
grid.setHgap(10);    //spacing between rows
grid.setVgap(10);    //spacing between columns
grid.setPadding(new Insets(25, 25, 25, 25));  //space around edges

Scene scene = new Scene(grid, 300, 275);    //as the window is resized, nodes are also resized
primaryStage.setScene(scene);
```

Part C – Add Text, Labels, and Text Fields

5) Add this code after the previous line of code:

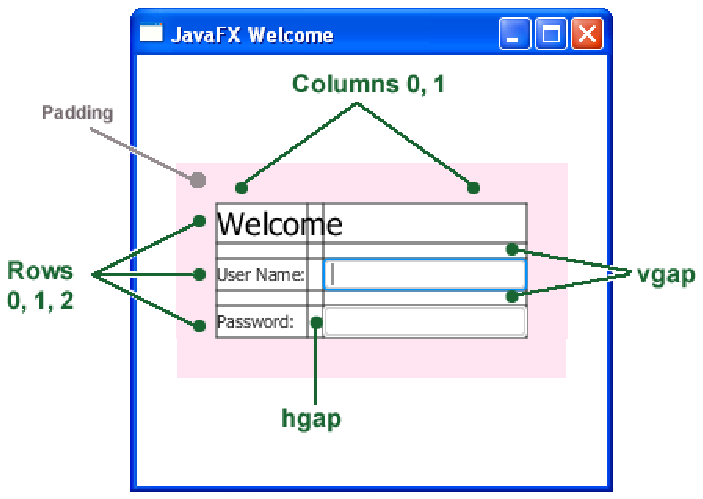
```
// create a Text object that cannot be edited
Text scenetitle = new Text("Welcome");
scenetitle.setFont(Font.font("Tahoma", FontWeight.NORMAL, 20));
grid.add(scenetitle, 0, 0, 2, 1);  // adds the scenetitle to the layout grid
// the title is in position 0,0 (top left)
// the title spans 2 columns and 1 row

// create a Label object with text userName at column 0, row 1
Label userName = new Label("User Name:");
grid.add(userName, 0, 1);

// create a TextField object that can be edited
TextField userTextField = new TextField();
grid.add(userTextField, 1, 1);

Label pw = new Label("Password:");
grid.add(pw, 0, 2);

PasswordField pwBox = new PasswordField();
grid.add(pwBox, 1, 2);
```

6) Run the application and see how it looks.

7) When working with a grid pane, you can display the grid lines, which is useful for debugging.

Try adding `grid.setGridLinesVisible(true)` after the line that adds the password field

Run your application to check it out with the gridlines!



Part D – Add a Button and Text

The final two controls required for the application are a Button control for submitting the data and a Text control for displaying a message when the user presses the button.

8) Add this code **before the code for the scene**

```
//Create the button and position it on bottom right
Button btn = new Button("Sign in");
HBox hbBtn = new HBox(10);
hbBtn.setAlignment(Pos.BOTTOM_RIGHT);
hbBtn.getChildren().add(btn);
grid.add(hbBtn, 1, 4);
```

9) Run your application!

10) Now, add a Text control for displaying a message. Add this code before the code for the scene. You will not see this part until you finish the next section!

```
//Add a Text control
final Text actiontarget = new Text();
grid.add(actiontarget, 1, 6);
```

Part E – Add Code to Handle an Event

Finally, we are going to make the button display the text message when the user presses it!

11) Add the following code

```
// registers an event handler that sets actiontarget to "Sign in button pressed" in red
btn.setOnAction(new EventHandler<ActionEvent>() {

    @Override
    public void handle(ActionEvent e) {
        actiontarget.setFill(Color.FIREBRICK);
        actiontarget.setText("Sign in button pressed");
    }
});
```

12) Run your application

13) Add this code to retrieve the values of the textboxes

```
//Try this code under the handle(ActionEvent e)
System.out.println("Hello");
System.out.println(userTextField.getText());

System.out.println(pwBox.getText());
```

14) Run your application and see what happens!

Programming Activity/ Mini Assignment

a) Edit your code to check the user's username and password against the provided list of acceptable combinations. Display a welcome message if access is granted, or a message of denial if not.

User Name:	Password:
MsMcDougall	ICS3U
John	snappy
CharlieM	123456
Katie	katie1

b) Organize your code to implement a method that is called from the event handler. For instance, you could create a method called checkPassword with two String parameters that returns a Boolean value (true or false).

c) Allow users 5 tries to gain access, and then quit the program with an appropriate message. This would be an acceptable situation for a global variable to keep track of the number of tries. This message may occur in the Console window.

d) Bonus: Have the appropriate messages appear as alerts rather than in the console window (an alert is similar to a message box in C# or Visual Basic)

Here is a snippet that implements an alert:

```
// add to your import statements
import javafx.scene.control.Alert.AlertType;

// place this code where you want the alert to appear
Alert myAlert = new Alert(AlertType.INFORMATION);
myAlert.setTitle("First Alert");
myAlert.setHeaderText("This is my first Alert!");
myAlert.setContentText("This is an important message for you, user");
myAlert.showAndWait();
```

Submit your code via the dropbox. This is a mini-assingment.