When Java was introduced back in 1995 developers need a way to create applications that users could interact with. The developers needed to gain input from there users such as providing a location for them to insert their username and password or checking boxes for specific questions to gain insight. According to our textbook “Introduction to java programming and Data Structures Comprehensive Version” by Y. Daniel Liang when Java was introduced, the graphical user interfaces (GUI) classes were bundled in a library known as the Abstract Windows Toolkit (AWT). AWT soon updated and expanded to Swing which led to an all-new GUI platform JavaFX. JavaFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java, according to the website openjfx.io. In this paper we will be looking at two different layout panes, FlowPane and GridPane, what they are used for and how they differentiate from each other.

According to our textbook FlowPane arranges the visual components (nodes) in the pane horizontally from left to right, or vertically from top to bottom, in the order in which they are added (Liang, 2020). The website java2s.com adds that the nodes with wrap to the next line when horizontal space is less than the total of all the nodes’ widths (java2s.com). By default, a FlowPane layout flows nodes from left to right according to the website java2s.com. If one wanted to change the orientation or alignment, they could us the respective properties like in the example below taken from the openjfx.io documentation page and java2s.com site:

FlowPane flow = new FlowPane(Orientation.VERTICAL);

flow.setAlignment(Pos.TOP\_RIGHT);

Orientation.VERTICAL makes the FlowPane vertical and the flow.setAlignment(Pos.TOP\_RIGHT) starts the nodes on the top right of the pane.

FlowPane also has two properties that one can use to set the horizontal and vertical gap between the nodes by using the hgap or vgap properties. Our textbook states that the default for hgap and vgap is 0 pixels (Liang, 2020) so they would but up right next to each other. Using the FlowPane of flow from above, one would set a the hgap and vgap by writing flow.setHgap(5) and/or flow.setVgap(5). This would set the gap in between each node to 5 pixels horizontally and vertically. To add or remove a node you would use the add(node), addAll(node1, node2,…), remove(node) or removeAll() methods respectively.

Graphical user interface, text, application, email

Description automatically generatedUsing listing 14.10 ShowFlowPane.java from our textbook (Liang, 2020) as inspiration I created a FlowPane example that asks for a user’s first name, middle initial, and last name. The code creates a 300 by 200 scene that has six nodes in it, three labels and three text fields. The FlowPan’s orientation is the default of horizontal. In the first picture (left), because the stage is only 300 pixels wide there isn’t enough width for all six nodes to fit so the last two nodes (Last Name and its text field) gets moved down to the next line. There is a black space in between the nodes due to a Hgap of 5 pixels and Graphical user interface, text, application

Description automatically generateda Vgap of 10 pixels. The second picture (right) shows what the FlowPane looks like when I increased the width of the screen. Because the width of the stage is larger, all six nodes fit on the horizontal line. The picture still shows the Hgap of 5 pixels, but the Vgap of 10 pixels isn’t needed because there isn’t a second line of nodes.

The GridPane is like the FlowPane except it arranges the nodes in a grid (matrix) formation according to your textbook (Liang, 2020). Java2s.com states that the GridPane is usually used to layout entry form with read-only labels on the first column and input fields on the second column. GridPane has the same properties of alignment, hgap and vgap like FlowPane, but because GridPane deals with a grid pattern, there is also gridLinesVisible property that has a default value of false, but if turned to true one would see the grid lines. Adding a node to a GridPane is similar FlowPane except you have to add the location of the node in the grid. An example for our textbook is below:

GridPane pane = new GridPane();

Pane.add(new Label(“First Name:”), 0, 0);

Pane.add(new TextField(), 1 , 0);

The first node, which is the Label of “First Name:”, is located on the top left grid, column 1, row 1, but because Java starts at 0 it reads like column 0, row 0. The second node, which is the TextField, is located on the top right of the grid, which reads like column 1, row 0. According to our textbook one can add multiple noes to a specified column or row (Liang, 2020).

Graphical user interface, text, application

Description automatically generatedGraphical user interface, website

Description automatically generatedUsing Listing 14.11 ShowGridPane.java as inspiration from our textbook, I again created a scene that has labels and input fields for the users first and last name, and middle initial. Using the code I centered the grid in scene so no matter the size the grid will always be in the middle. In the first picture (left) you can see that all the labels are in the first column and all the input fields are in the second column. By default, each node is aligned to the left of each grid section. The button on the bottom right of the pictures has it's alignment changed to be on the right of the column so it matches the end of the input fields. The second picture (right) shows what happens when the scene is enlarged, you’ll see that the grid stays center and just the scene got bigger. According to java2s.com one can code the second column to resize as the window is being resized. This is achieved by the code of column2.setHgrow(Priority.ALWAYS).

In conclusion, FlowPane is used more when one would want all their nodes to be either horizontally or vertically in one line. The only way the nodes would go to another line is if the width of the window isn’t big enough. Using GridPane is more conducive to the label, text field form as it creates a visually pleasing window that is easy to read and navigate.

Reference

Liang, Y. D. (2020). Chapter 14: JavaFX Basics. In *Introduction to java programming and Data Structures Comprehensive Version* (pp. 541–592). Pearson.

N.A, N. A. (n.d.). *Class FlowPane*. FlowPane (JavaFX 19). Retrieved February 2, 2023, from https://openjfx.io/javadoc/19/javafx.graphics/javafx/scene/layout/FlowPane.html

N.A, N. A. (n.d.). *JavaFX Tutorial - JavaFX FlowPane*. JavaFX tutorial - javafx introduction. Retrieved February 1, 2023, from http://www.java2s.com/Tutorials/Java/JavaFX/0320\_\_JavaFX\_FlowPane.htm

N.A, N. A. (n.d.). *JavaFX Tutorial - JavaFX GridPane*. JavaFX tutorial - javafx introduction. Retrieved February 1, 2023, from http://www.java2s.com/Tutorials/Java/JavaFX/0340\_\_JavaFX\_GridPane.htm

N.A, N. A. (n.d.). JavaFX. Retrieved February 1, 2023, from https://openjfx.io/