Rachel Shaw
CSD-402
2/28/2024

JavaFX BorderPane and GridPane Classes

JavaFX offers a vast range of features for creating creative and innovative user interfaces, including several classes dedicated to the layout of applications. Some of these classes include BorderPane, FlowPane, GridPane, and several others. With so many layout options, it may be difficult for programmers to decide which is best for their projects, especially when a few produce similar results. BorderPane and GridPane, for example, are both layout structures that take on a grid-like format. However, there are a few key differences between them that may assist in deciding which one to use.

The BorderPane class is meant to assist in creating a simple layout of a scene in JavaFX. The design of a BorderPane object is divided into five different areas in which nodes can be placed: top, bottom, left, right, and center. However, any of these areas may be removed from the layout if they are not defined during the object's construction. The class has three constructors. The first is a no-argument constructor, which will create the basic five-area layout. The second accepts a node argument and sets it as the center of the BorderPane. The third and final constructor accepts five node arguments that allow programmers to use a preconstructed node as each of the areas. However, suppose the no-argument constructor or the single-argument constructor is used. In that case, one can also assign nodes to each layout area using the setter methods for each area. Alternatively, several elements can be arranged within the same area using a method that will set the alignment of a given element within an area in the Pane.

The size of a BorderPane object depends on the width of the top and bottom areas and the height of the left and right areas, with the center area filling the space left in the center. That said, the width and height of a BorderPane may be customized by setting desired width and height values for each of its size-defining areas. Additionally, the BorderPane class implements the "Styleable" interface, which allows objects to be customized with CSS. For an example of the BorderPane class in use, please refer to example one.

The GridPane class, as can be assumed by its namesake, aligns its children neatly within a grid with rows and columns. Unlike Borderpane, GridPane is quite flexible as rows and columns adjust to fit the content within them. Alternatively, the size of each column or row can be adjusted using constraints with values or percentages. However, placing items inside a GridPane's cells is similar to placing items inside BorderPane areas as it is done using row and column indexes that determine where each item belongs within the Pane. Additionally, like BorderPane, items can be aligned within a cell using similar constraints. Elements can also span multiple columns and rows of a GridPane object if needed, which is done by defining which column and row the element begins on and which column and row it ends on. Although, unfortunately, a GridPane object cannot be manipulated by CSS as it does not implement the "Styleable" interface. To see an example of a GridPane layout, please refer to example two.

Both BorderPane and GridPane are relatively simple and beginner-friendly as BorderPane provides a simple layout with five sections, and GridPane follows a simple row/column layout. However, choosing between them depends on how flexible the layout must be. The layout of a BorderPane is fixed, and while programmers can change the size of areas to a certain degree, they cannot alter the page's overall layout beyond removing areas by leaving them undefined. On the other hand, the GridPane offers more flexibility as each individual cell is resizable, and placement of elements is less restricted because they can span multiple cells

(which they cannot do in a BorderPane.) However, if a more basic, uniform layout will suffice, BorderPane is an excellent option as it is arguably the simplest of the two. This is primarily because the class automatically takes care of most of the placement, and programmers only need to worry about node placement within the area. Additionally, BorderPane has an advantage regarding customizability as it supports modifications with CSS.

Conclusively, while these two classes may appear similar when implemented, there are several key differences between the BorderPane and GridPane classes that give each an advantage depending on a programmer's priorities. If flexibility is required, GridPane is the way to go. In contrast, if simplicity and customization are priorities, BorderPane is a great choice.

References:

Oracle. (2013a). *BorderPane (JavaFX 8)*. Docs.oracle.com. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/BorderPane.html

Oracle. (2013b, December 9). *GridPane (JavaFX 2.2)*. Oracle.com. https://docs.oracle.com/javafx/2/api/javafx/scene/layout/GridPane.html