Alisa Steensen 2/25/2025

Module M11 Programming Assignment

JavaFx provides many different tools that can be used for generating applications. This allows users to create text boxes, like buttons and layouts while creating the user interface. When you are deciding on how to layout the user interface, JavaFX will help to organize and provide different layout managers to help with this.

In JavaFX different layouts are helpful for different situations. HBox and BorderPane are two examples of the features you can use. HBox is good for when needing a simple row of items. And BorderPane is helpful when you need a structured layout that has different sections. You can use both of these features together to create different designs. By understanding the differences in these layouts, you can create a well designed user interface.

HBox is one of the commonly used tools. This layout is an extension of the Pane Class. The JavaFX HBox component is a layout component which positions all its child nodes (components) in a horizontal row. The Java HBox component is represented by the class javafx.scene.layout.HBox (Jenkov, n.d.)

The HBox layout will arrange things in a straight horizontal line. When designing with multiple buttons, labels, or images, they will line up from left to right. Similar to a bookshelf, each book is placed next to the other in a single line. That is how HBox works, putting everything side by side in a row.

When using the HBox you will use what is called child nodes to arrange items. Child nodes are a data structure, like a tree, that is connected to another node (Parent Node). The child nodes are placed next to each other in the order they were added. You will need to use a default spacing between them which is 0 pixels, however, you can customize this. When controlling the alignment of the boxes, the alignment is controlled by the alignment property which defaults to Pos.TOP\_LEFT. Additionally, during resizing of the HBox, if resized too large than the preferred layout width, the children will resize to default and leave the additional spacing unused.

The constructors for the Hbox are Hbox(), which creates an HBox object with no nodes. And HBox(double s), which creates an Hbox with spacing in between nodes. These are both important to consider when designing your layout.

To create the Hbox you will use HBox hbox = new HBox(). You will then need to have the constructor that will take in the variable length list of components that will need to be laid out. For example, to do this you will use:

Button button1 = new Button(“Button Number 1”);

Button button2 = new Button(“Button Number 2”);

In this example, the layout will have two Button instances next to each other in a horizontal row. (Jenkov, n.d.)

Commonly Used Methods:

* getAligment() - Provides the current alignment property value.
* getSpacing() - Provides the current spacing between the children nodes.
* setAlignment(Pos Value) - When setting the alignment of the Hbox.
* getChildren() - Provides the nodes that are in the Hbox.

BorderPane is another layout feature of the JavaFX used for laying out children (child nodes in the tree). Instead of things being a single row like HBox, the items are divided on the screen in five areas, the top, left, right bottom, and center positions. Each of these sections have the ability to hold different things. This keeps everything organized and easier to manage. For example, you might want to put a title at the top, a menu to the left, and the main content in the center. Additionally, BorderPane may be styled with backgrounds and borders using CSS styling. (Oracle, n.d.)

The constructors for the BorderPane class are: BorderPane(): which will create a new BorderPane. BorderPane(Node c): which ccreates a new BorderPane with the specific node in the center. BorderPane(Node center, Node top, Node right, Node bottom, Node left): which creates a BorderPane layout with the given nodes to use on each of the main layout areas.

Key Methods:

* getAlignment(Node c) - Provides the alignment of the node.
* getBottom() - Provides the bottom node.
* getCenter() - Provides the center node.
* getLeft() - Provides the left side of the node.
* getRight() - Provides the right side of the node.
* getTop() - Provides the top of the node.
* setAlignment(Node c, Pos v) - Sets the alignment of node c to position v.
* setBottom(Node v) - Sets the bottom node.
* setCenter(Node v) - Sets the center node.
* setLeft(Node v) - Sets the left node.
* setRight(Node v) - Sets the right node.
* setTop(Node v) - sets the top node.

When resizing the children top and bottom, it will be resized to their preferred heights and extend the width of the border pane. The same for the left and right sides to their preferred heights. “BorderPane lays out each child set in the five positions regardless of the child's visible property value; unmanaged children are ignored.” (Oracle, n.d.)

In conclusion, both BorderPane and HBox are important features of JavaFX that can be used to create well desgined layouts. HBox provides the horizontal layout and BorderPane divides the screen in different sections. Understanding these layouts will better assist developers in a well designed layout.

References:

Jenkov, J. (n.d.). *JavaFX HBox*. Jenkov.com. <https://jenkov.com/tutorials/javafx/hbox.html>

GeeksforGeeks. (n.d.). *JavaFX HBox class*. GeeksforGeeks. <https://www.geeksforgeeks.org/javafx-hbox-class/>

Oracle. (n.d.). *HBox (JavaFX 8)*. Oracle Documentation. <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/HBox.html>

Fiveable. (n.d.). *Child node*. Fiveable. <https://library.fiveable.me/key-terms/data-structures/child-node>

Oracle. (n.d.). *BorderPane (JavaFX 8)*. Oracle Documentation. <https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/BorderPane.html>

GeeksforGeeks. (n.d.). *JavaFX BorderPane class*. GeeksforGeeks. <https://www.geeksforgeeks.org/javafx-borderpane-class/>

Oracle. (n.d.). *BorderPane (JavaFX 2)*. Oracle Documentation. <https://docs.oracle.com/javafx/2/api/javafx/scene/layout/BorderPane.html>