Ryan Monnier

CSD 402

Module-11 - JavaFX Paper

11-May-2025

JavaFX is a powerful framework for GUIs in Java that are cross-platform capable. It was introduced by Sun Microsystems as a modern alternative to Swing. JavaFX offers enhanced capabilities for creating desktop applications UIs with improvements over AWT and Swing like better sterling with CSS and the use of FXML.

JavaFX supports a wide range of UI components, including things like buttons, labels, text fields, and layout containers. It also has advanced features that I haven’t quite tapped into yet like 2D and 3D graphics, media playback, and integrating web content. The layouts in JavaFX are managed using various layout panes such as HBox, VBox, BorderPane, FlowPane, Accordion and GridPane, each of which serves a different purpose for arranging UI elements. Some of these layouts are a bit intuitive like HBox and VBox, which are used to arrange elements in a horizontal or vertical manner. But the functionality of other layouts like Accordion and GridPane seem to go a bit beyond what their names suggest.

The Accordion class in JavaFX is a container that allows for display of multiple panes in a manner which presents them inside of collapsible sections. The panes are each represented by a TitledPane, which can be expanded or collapsed by the user. When the user is presented with a large amount of information, this functionality gives the user the ability to filter in or out the information that they want to view, making the experience more manageable. This component is ideal for settings panels, menus, or any interface requiring grouped information which you want to provide the user with some control over which portions they’re viewing.

As mentioned, the key feature of Accordion is that each TitledPane can be expanded or collapsed, allowing users to focus on one section at a time. However, the Accordion can be set to allow multiple panes to be expanded simultaneously or it can be set to restrict it to one expanded pane at a time if the developer would like to have more granular control over the user experience. The title of each TitledPane can of course also be customized to reflect the content it holds.

Another really powerful class in JavaFX is the GridPane layout container. I was originally thinking that this class would just be used to present data sets or similar information, but after finding a couple of examples online, it’s clear that this is used for that as well as so much more. One example that stood out was using GridPane for an entire login screen which consisted of Email and Password fields as well as the buttons to submit the data or clear the fields. This arranges the interface in a nice predictable and uniform manner. The GridPane class arranges its children in a flexible grid of rows and columns, and can be particularly useful for forms, tables, and any layout requiring precise control over the positioning of elements.

The primary feature of GridPane is that developers can define the number of rows and columns, as well as their sizes, to control the layout precisely. Each of the cells in the grid can have its own alignment and padding, which gives the developer fine-grained control over the UI appearance. Nodes that are added to the layout container are referred to as children, and these children can span multiple rows or columns, which ends up allowing for complex layouts within the grid. Another complexity that this container offers is that the cells can be made resizable to adapt to different screen sizes or content changes, allowing for a smooth display on a variety of platforms.

Having explored both Accordion and GridPane, it’s easy to see how powerful these containers can be, and they’re only two of the containers contained in JavaFX. Utilization of these classes enable developers to create sophisticated and user-friendly desktop applications. The functionality of these features really begins to skyrocket when a developer starts combining them. An Accordion can be placed inside of a GridPane, or vice versa, depending on the needs of the developer. Since every container is a Node, it can simply be added as the child to another layout. This russian nesting doll approach really means that the limitations of these containers are really only bound by the creativity of the developer!

References

TutorialsPoint. (n.d.). *JavaFX - HBox Layout*. TutorialsPoint.<https://www.tutorialspoint.com/javafx/javafx_hbox_layout.htm>

TutorialsPoint. (n.d.). *JavaFX - VBox Layout*. TutorialsPoint.<https://www.tutorialspoint.com/javafx/javafx_vbox_layout.htm>

Gluon. (n.d.). *JavaFX*. Gluon. <https://gluonhq.com/products/javafx/>

Oracle. (n.d.). *Using JavaFX UI Controls*. Oracle Help Center.<https://docs.oracle.com/javafx/2/layout/jfxpub-layout.htm>

Visual Studio Code Documentation. (n.d.). *Java in Visual Studio Code*. Microsoft.<https://code.visualstudio.com/docs/languages/java>