Jess Monnier
CSD-402 Module 11
11 May 2025

JavaFX Accordion and ScrollPane Layout Controls

JavaFX offers numerous options for controlling the layout of an application. To a less experienced programmer like myself, the wide array of controls to choose from can feel a bit overwhelming, but it helps to know what each option was made to accomplish and how they can be used together to achieve multiple layout goals. The ScrollPane and Accordion controls are perfect for illustrating this point because they synergize very well when used together in an application.

The Accordion control is designed very specifically to take TitledPane controls as children, and so it is helpful to first explain a little about the functionality of TitledPanes. They are layout elements with a title bar that can be clicked to collapse or expand the actual contents of the pane. This feature can be useful for layouts with vertical space restrictions or for implementing progressive disclosure in an application.

Alone, TitledPanes can be expanded and collapsed completely independently of one another. A user could have multiple open at once. An Accordion's role is to limit its child TitledPanes such that only one may be expanded at a time. Clicking to expand one of the collapsed children will automatically collapse the TitledPane that is currently expanded. This sort of functionality is not uncommon on the checkout page of online stores, allowing the user to focus on one section at a time: Shipping Information, then Billing Information, then Payment Information, et cetera.

Where Accordions are specifically limited to TitledPane children, a ScrollPane can have pretty much any other layout control as a child, including the Accordion itself. I felt that the scrolling functionality would pair well with an Accordion, allowing the user to scroll to view the contents of the expanded pane, and so I combined these layout controls in a sample application to try my hand at JavaFX.

I leapt to using the ScrollPane as the top-most parent layout control within the application's Stage because I knew I wanted to test vertical scrolling functionality, but this control can be useful in many other ways within a layout. It could be used within a TitledPane if a developer wanted to ensure that all TitledPanes expanded to a specific height to introduce an element of consistency into the layout, for example; that would allow the TitledPane to include more content than the height restriction would otherwise allow. Many of the examples I stumbled across for ScrollPanes online demonstrated

allowing a user to scroll horizontally and vertically to view more an image placed into a limited space.

One of the nice things about JavaFX is that these controls and many others share a lot of properties and methods due to being inherited from common parent classes. This makes it a bit easier to tweak the layout of an application until it's just right. For example, when I was building my sample application for this assignment, I wanted to adjust the padding of some of the sections. I learned that the method to use was `.setPadding()` paired with the geometry component `Insets()`. I first adjusted the padding of the ScrollPane, and then I set out to adding padding within the TitledPanes. I quickly learned that the same method worked there, but did not produce quite the result I expected; instead, it added padding *between* the TitledPane elements. Fortunately, the same method *also* worked on the Labels that contained my text, which was where I actually wanted the padding.

JavaFX is a very powerful tool for designing applications, and the great flexibility that it offers naturally means that there are a lot of methods and properties associated with each of its numerous classes and tools. The learning curve for maximizing the use of the various capabilities of JavaFX can be quite steep, but fortunately there are a lot of resources out there to help. For this assignment, I found a number of resources particularly helpful, including those listed in the References section but also the Azul Zulu build of OpenJDK, which helped save my sanity when getting JavaFX installed and working with Visual Studio Code was driving me bonkers. As I learn more about programming, I am grateful to have grown up in an era that taught me to research and tackle problems in a hands-on way in order to better learn from them, but I also appreciate the modern availability of AI resources like ChatGPT that can help find answers to oddly specific questions that don't easily turn up helpful results with a Google search (such as how to get an Accordion to auto-resize its width as the application window is resized). As we continue this degree program and beyond, I am looking forward to gaining more experience and comfort with JavaFX and with other approaches to creating GUI applications.

References

ChatGPT. (2025, May 11). *Response to a question about APA citation and JavaFX ScrollPane/Accordion*. OpenAI. https://openai.com/chatgpt

Oracle. (n.d.). *Accordion (JavaFX 8)*. Oracle. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Accordion.html

Oracle. (n.d.). *ScrollPane (JavaFX 2 UI Controls)*. Oracle. https://docs.oracle.com/javafx/2/ui_controls/scrollpane.htm

Oracle. (n.d.). *TitledPane (JavaFX 8)*. Oracle. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TitledPane.html

TutorialsPoint. (n.d.). *JavaFX Accordion*. TutorialsPoint. https://www.tutorialspoint.com/javafx/javafx_accordion.htm

TutorialsPoint. (n.d.). *JavaFX ScrollPane*. TutorialsPoint. https://www.tutorialspoint.com/javafx/javafx_scrollpane.htm