Java Application Programming - Lab 1

Neil Kingdom

Algonquin College

1. Layout

1.1. General

In general, I would like to keep my picross application to be as modular as possible. From past experience with Swing, I've learned a few lessons about layout that I'd like to implement here. Firstly, I will start with a main JPanel that will overlay the JFrame since we don't really want to mess with the JFrame directly. The main JPanel will then contain more JPanels for the left, top, center, and right sections. The reasoning for this is to allow flexibility for layout manangers. Each JPanel will have its own layout manager so that I will have more control over the layout of each section. This allows for more dynamic scaling as well when we call pack().

1.2. Details

The details of this application will likely have certain modifications that need to be made throughout the course of development, but I'll try my best to describe what I envision. Firstly, the main JPanel will have a 5px inset from the JFrame so that there is a 5px spacing around the entire border of the window. This will simply make things look a bit tidier and neater. Each JPanel for the left, top, center, and right JPanel will potentially also have a 5px gap in between them, however, I haven't decided whether or not this should be 5px or 1px (remains to be determined). The hint tiles (with the numbers), as well as the actual picross tiles will have 1px gaps between them. This will hopefully give a sort of sudoku look to the game. A practical reasoning for this choice is to avoid jarring oddities when scaling the window.

The tile width and height of picross playing tiles will likely be 60px x 60px since that will fit on most screens, though they should be scalable so this is only the default size. The hint tiles will be created individually to maintain the sudoku grid-like look that I am aiming for. This raises many complications with scaling that will just need to be dealt with somehow. Ideally the hint tiles will sort of expand based on the number of hints that there are. So for instance, if there are 3 groups of tiles that ought to be marked eg. [(1), (1), (1)] then that hint tile will portrude outwards more than the other tiles which might only have 1 or two groups of tiles that need to be marked. This is easily achieved since the hint tiles will be in their own JPanel. This means, setting the width or height of that JPanel to the tallest/widest (depending on the top or left set) hint tile. The other details about the checkbox, and right dialogue section are not as important in terms of spacing so I feel that it's unnecesary to cover that.

2. JComponents

Finally, I'd like to cover some of the JComponents that I will use and my reasoning for each. Firstly, the obvious one: JCheckBox for the "mark" checkbox. The hint panels will use JTextArea components. This is because they accept plain text. JTextFields do not make sense here since the user should not be inputting text. JEditorPanes use document text, which is not desirable either. JTextArea can disable user input, change the font and size of text, and is simple to use, hence my decision to use JTextArea for the hint tiles. For the picross tiles, I think the best choice here is JLabels. This is simply because they are sort of a blank slate component and can be easily sized, colored, etc. I could technically go with buttons here, however, I feel that that would look extremely tacky and would just overall be more resource intensive. Finally, for the right panel... The icon at the top will likely just be a JLabel once again since they handle icons quite nicely. I think for the points and time sections, I will just use JTextField and disable the editing. This is simply because the size of JTextField by default would look good I feel, and I wont have to resize anything. The reset button will obviously be a JButton. And finally, the dialogue panel will likely also be a JTextArea since only plain text is required here. This **might** potentially change to a JEditorPane if I feel like adding some aditional pizzaz to the box, but I doubt that I will need that sort of thing unless perhaps I needed a moving background or something. This will also be added to a JScrollPane since you can add JTextAreas to JScrollPanes. This will allow scrolling for text if it spills off of the screen.

3. Conclusion

In conclusion, I feel confident that my design makes sense from an aesthetic and practical point of view and I hope that you agree. I feel quite strong about the details of my design as well as the components that I've decided to use, although there are always complications that may lead me to change certain design elements. I hope that you are able to agree on most of my points!