**Futoshiki**

**Displaying the data:**

For the display of my futoshiki puzzle, I made the GUI so that it would use a border pane that I would then use to separate my window into different compartments. As a result of this, I had a top, left, right, centre and bottom sections to use. For the top pane, I turned this into a HBox that would simply hold the games title. I did this so that I could have the alignment that I desired. Next, for the left pane I used a Vbox which contained an introduction to the game, how to play and the ‘Hints’ that the player requested. I provided these hints as after doing solve, the completely filled grid would be copied into solved before clearing roughly ⅘ of the values again which would be used for the start of the game. Therefore, when the player would press ‘Hint’, the value would be taken from the solved grid as a long as the tile is editable, it will display the value with the location it needs to go. For the text, I wanted the two headings and hints to be bold to improve its appearance; I did this by making them into text field, which I then edited. However, for the other normal text, I did not wish for the to have anything edited so I simply made them using labels. For the right pane, I used a VBox again that was used to list all the error that the user was having with locations and that would be removed when the problem was solved or set back to pencil. Next, for the bottom pane I used a HBox for the button ‘quit’, and combo boxes ‘colour’, ‘size’, ‘difficulty’ (and others mentioned in optional extras). When a ‘difficulty’ is selected, 1 of 3 random fills would be used that differ from each other by the amount of random constraints that will be used. The ‘size’ will simply use that as the puzzle creation parameters. Finally, for the center, I used a grid pane, which would be positioned in the middle of the area and would use both buttons and labels for the numbers and constraints. Depending on the ‘size’ that the user makes the puzzle, this will affect how large the buttons and labels are in proportion to the space it must fill. To make the grid look correct, I made it so that say, for example the user makes the grid 5x5, the grid will actually be 10x10 but it will alternate between filling it with numbers of constraints using the mod function. I would get this information by doing array location value calls divided by 2 for the iteration (so that the it doesn't try and grab a number in index 8 to put in location 8 as this is out-of-bounds), then applying that to the corresponding grid location. If the ‘quit’ button is clicked, a commiserations message will appear. Whenever anything is changed in the game, the display will be called again to ensure that everything is always up-to-date with what the player has done at that point in time.

**Editing the data within the grid:**

If the player were to left click on the tile, as long as it is editable, it will increment by 1 each left click until it reaches its limit then will return back to clear then will start incrementing from 1 again. However, if it was not editable then nothing will happen to the grid. A relevant message will also appear for any illegal moves the player makes. In addition to this, if the user were to right click an editable tile, the tile will be set to true. The set true/pencil will make it become a true value that will be reverted back to a pencil mark if the button has been pressed again. If all of the tiles have been set to true and are legal, the player can press ‘Check’ and a congratulations message will appear as the game has been complete and the player will have the option to either start again of close the window.

**Optional extras:**

Label ‘Name’, buttons ‘Hint’, ‘new’, check’ and combo boxes ‘colour’. The ‘Name’ would be used to give personalisation to the messages that may appear from doing certain things. ‘New’ would create a new puzzle of the same size and difficulty. ‘Check’ would search the whole grid to make sure that its legal and has values. ‘Colour’ would change the top and bottom pane colours.