The biggest challenge which in and of itself was not too difficult, only required a lot of testing, was the algorithm for checking whether a particular place on the board could be used to place a chip there. What I did was I wrote 8 recursive methods that check availability in each direction (whether there exists at least one other player piece in that direction followed by one this player piece) and a method that calls all these methods and checks whether the particular place is not already occupied. Writing 8 methods was a great idea (as opposed to writing everything in one method, in loops rather than making recursive methods) because they later came in handy when writing a method that flips the pieces in all these directions – this method checks each direction separately and flips that direction only if that’s required.

The other biggest challenge was making some sort of computer intelligence. I wrote a method that copies an existing board and tries out all available moves to see which one gives the highest score. In my findings, checking only 1 step ahead is enough in most gameplays as it is still hard to win against a computer.

Some of the things I didn’t know how to do before starting this project was adding graphical interface for file saving and loading, as well as GUI in general. But these things were not difficult to implement once the text-based version was finished. I even improved the given GUI java file by changing how the board is displayed – instead of showing symbols on top of buttons, I colored the buttons instead.

I’m particularly proud of my GUI integration with the text-based version, particularly how the computer knows when to place a piece for itself – usually the game waits for the player to make turn and then immediately makes a computer’s turn. If computer stars first, then its method that makes a turn for it is called immediately as soon as the game is started.