Group 4 members: Isaac Supeene and Braeden Soetaert

- Our program was developed using Ruby version 1.9.3 and requires at least gtk+ version 2.16. (This is the same version used in the TA's lab8 demo)

- Score values and leaderboards have been removed due to time constraints and will be deferred to project 5.

- We decided to implement only 1 AI opponent for now. This opponent just plays in a random column on his turn so he is fairly easy to beat.

- For window size we went with a default of 900 pixels wide and 600 high. The game is resizable though so this is just the default size when the GUI is first opened. Also, the New menu option under File allows the user to end the current game and immediately begin a new single or 2 player game.

- We now have both a CLIClient and a GUIClient which interact with the user directly, and forward the user's inputs to the Controller or the GameManager.  The Views are notified of state changes by the Game, and update the GUI with the current gamestate. The clients are also responsible for deciding which type of view to create based on the number of players and the type of game that is to be played.

- Currently we do not use Namespaces in our solution. Namespaces are not really needed in our solution currently as it is all of one piece. Namespaces could be useful in the future if we put multiple programs together but they are not necessary at this point in time.

- Currently, our exception handling strategy attempts not to disturb the player's experience if the game can continue.  An error message is printed to the console, but the game is not terminated.  This is because in many cases, the exception may not actually affect gameplay (for example, there could be errors checking the very edge of the board for victory conditions, where a victory is unlikely).  For this reason, the error messages are obvious when using the CLI (as a developer or tester might when debugging), but not when using the GUI.

Our answer to the last question in the design doc did not mention the exception hierarchy so it will be talked about here. For this problem the exception hierarchy is not too important as the errors we experience will most likely be unrecoverable errors.Other exceptions can occur when loading or saving a game. Right now this functionality is rather primitive as it is more of a feature for assignment 5 so currently it catches all exceptions that occur with the load and save and report that an error has occurred to the user. Exceptions and error-handling will be much more important in assignment 5 where a more comprehensive save/load mechanism will be needed, as well as exceptions to do with networks and leaderboards.

Regarding the need for a CLI, a very important use for a CLI is to enable automated testing (and also make manual testing easier).  The CLI also enabled a better parallel workflow, since one of us was able to work on the backend and test it via the CLI, while the other worked on the GUI, and then integrated it with the backend.