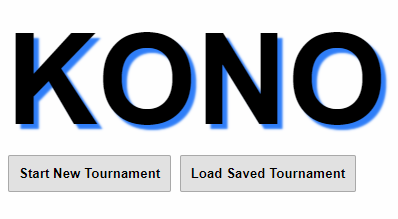
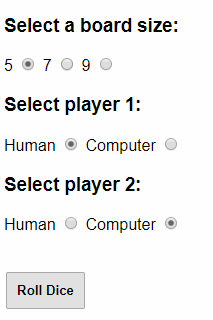
***Todd Weisse - 04.28.18 - Kono JavaScript User Manual***

*Using the Application:*

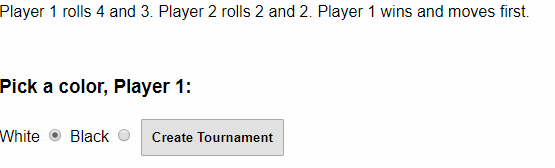
* When starting the application, a welcome screen appears with 2 options for the user:



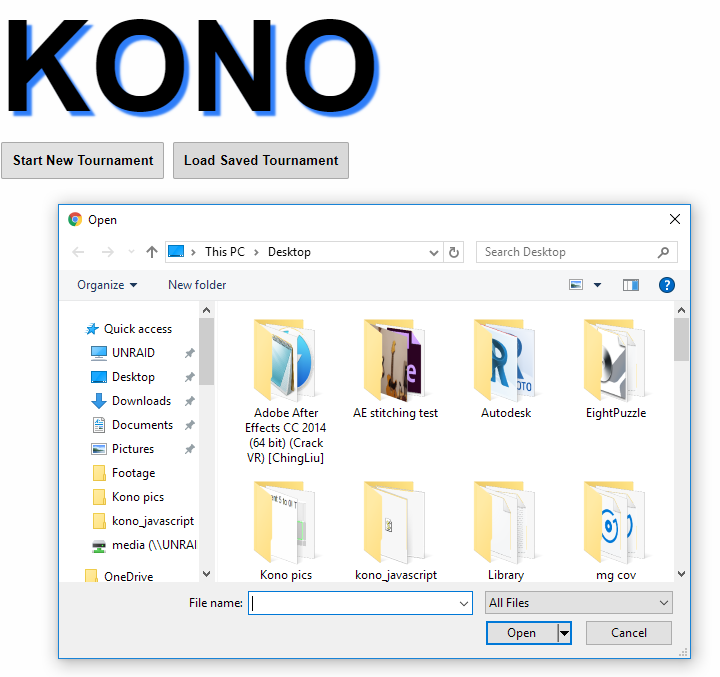
* + Clicking on “Start New Tournament” will bring the user to a new screen where they can configure a new tournament:



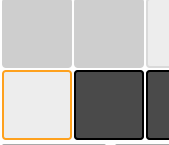
* + - Board size and opponent type must be selected. The user can then press the “roll” button, which will roll dice to see which player will move first. The winner also selects the color they would like to play as. When all fields are selected, the user can press the “Start Tournament” button to start the first game in the tournament.



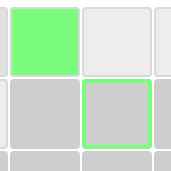
* + The user can also load a tournament from a previously saved file by pressing the “Load Tournament” button. This will take them to a seperate screen where a filename can be entered to load data from. Files must be placed in the root of the SD card directory, which varies from device to device.



* Either by loading from an existing tournament file, or by creating a new one, the user will end up on the Play screen. On this screen, each user takes turns making moves until one of them wins or quits the game. Each players’ pieces are represented by either a white or black box on the board, with super-pieces represented by an orange border. Information about the current game is displayed to the left under the board, and a log of all of the moves which have been played by both players will be displayed below that.

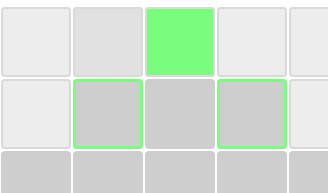
 

* + For the human’s turn, 4 buttons are available: “Ask For Help”, “Quit Game”, “Save And Quit”, and “Skip Move”
    - The “Help” button will use the computer AI system to pick a good move for the human to play. The suggested move will be displayed on the screen, with the suggested piece highlighted in green and the direction flashing.



* + - The “Quit” button will forfeit the current game. The game will exit, the player will lose 5 points, and the final scores will be displayed.

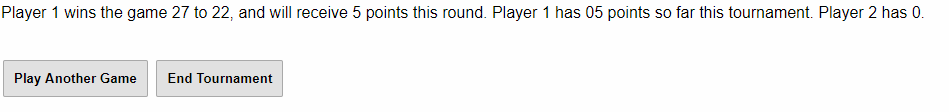
* + - The player can select their own move to play by clicking on one of their pieces, which will turn it green. All of the possible moves that the selected piece has will flash green, any of which can be clicked to execute the move. If the user decides that they do not want to play the piece which is currently selected, they can click again on the same piece to deselect it and select another, or simply click another of their pieces which will select it instead. Each player gets up to two moves per turn. If the player does not want to play his second move, he can click the “Skip” button, which will forfeit that turn.



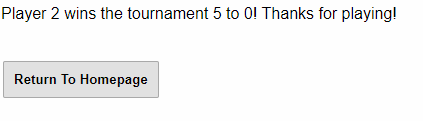
* + For the computer, the “Skip” button is replaced with one which reads “Move”. The user must press this button in order for the computer to execute a move. Note that the “Help” and “Quit” buttons will be disabled for the computer player, since the computer cannot ask itself for help, nor should the user be able to force the computer to quit. The computer will quit on its own if it decides that there are no moves worth playing.

* + The user can press the “Save” button at any point during the tournament to save the current progress into a file to resume later. The user will be prompted with the system dialog for saving a file, where they can name the file and pick a location to save it to. After saving is complete, the current game will exit and the page will return to the welcome page.

* + The game ends when a player gets all of his remaining pieces into the opponent’s home locations, loses all of his pieces, or quits the game. When this occurs, the user will be navigated to a page where the final scores are displayed. He will also be prompted if he would like to play another game.



* + - If the user presses “Play Another Game”, signifying that he would like to continue the tournament, he will be brought to another screen where a new game can be configured in much the same way a new tournament was configured. By pressing “Start Next Game” from this screen, a new game will launch and the tournament will continue.
  + If the user presses “End Tournament” instead, the tournament will end, and both players’ final tournament score will be shown along with the overall winner.



*Bugs:*

* Final tournament scores displaying very high value for one of the players. I have a feeling it is simply a string printing error, where a number is being printed where it shouldn’t be next to the actual score of the player.
* AI system will occasionally get “stuck” and move the same piece back and forth. This can be corrected in-game by moving an opponent’s piece to the spot that the computer keeps moving its piece, which forces it to move a different piece.

*Missing Features:*

* Board coordinates shown in the GUI.
* Dice loading from a file. Specific dice roll values can still be configured for testing, but they are placed in an array in the code rather than read from a file at runtime.

*Class Descriptions:*

* *Board:* Represents a game board for a given game. Pieces can be moved around on this board when playing.
  + The Bboard class contains a subclass (acting as a struct) to hold information about each Cell, such as occupant color and point value.
* *BoardView:* Prints a Board object to the screen to let the user track the progress and moves in a game.
* *Game:* Holds a Board and two Players and allows each to alternate making moves. The Game class handles a complete game from start to finish.
* *Tournament:* Allows 2 Players to play several games, and tracks each player’s points throughout. When the players no longer wish to keep playing, the tournament decides who has won based upon who has accumulated more points.
  + The Tournament class contains a subclass (acting as a struct) to hold information about each player in the tournament, like color and points scored overall.
* *Serializer:* Handles the saving and loading of games from files.
* *Player:* A Base class for a player of a game. Human and Computer are both derived from this.
  + *Human:* Represents a human player in a game. Allows the user to enter their own move and ask the computer AI for help.
  + *Computer:* Represents a computer player in a game. The AI system will choose a good move for itself and execute it, allowing the computer to play against a human.
* *Move:* Represents a move on the board. Holds both the move information as well as the reasoning for the move in the case of the computer AI.

*Why I choose JavaScript:*

* Already knew how to build a working UI in HTML, and style it with CSS. I thought that would speed up the process.
* All types are nullable, which I wanted in Java.

*Language Experience:*

* Liked:
  + All types nullable.
  + Functions can be nested in other functions. I used this to organize helper functions that otherwise would not not be called except by a particular function.
  + Functions can accept any parameters, the code can handle what to do in each case.
* Didn’t Like:
  + No type checking at all, so errors only crop up when they occur.
  + Function closures dont allow access to variables that you would think you could access. This drove me crazy until I learned how it worked.

*Project Log:*

* 4.24.18:
  + Created the Board class. (1.5 hrs)
  + Created the game’s homepage, and game page to display the board. (1 hr)
  + Created a BoardView class to get the board displayed on the screen. (.75 hrs)
  + Worked on an intuitive system for making moves, which displays all possible moves to the user by flashing them green. (3 hrs)
* 4.25.18:
  + Created a Player class, as well as Human and Computer to test moving on the board. The computer class does not yet have AI (2 hrs)
  + Extended the Board class with more helper functions that I will need later, such as determining who has won the game. (1.5 hrs)
  + Tested moving on the board. (.5 hrs)
  + Added to the game page the display of player’s scores. (.25 hrs)
* 4.26.18 into 4.27.18:
  + Created a game class, tested it with the current game page. (2 hrs)
  + Created a page to be displayed after a game is over, displaying the final scores of each player. (1.5 hr)
  + Implemented a second move for each player, and modified the board display to only allow moves from the first piece to be available for the second move. (2 hrs)
  + Created all necessary html pages, such as tournament creation and a home page, etc. Most do not do anything yet. (1.5 hr)
  + Created a tournament class, tested a bit but need to implement UI integration for complete testing. (2.5 hrs)
  + Scores of tournament shown after game is won (.33 hrs)
  + Tournament creation page linked to the model, tournaments can now be created by the user. (2.25 hrs.)
  + Created and completed serialization class (2.75 hrs.)
  + Created and completed computer AI. It is still buggy, needs tweaking. (2.75 hrs)
  + Serialization tweaked due to having to save two players of the same type. (1.5 hrs)
  + Tournament continue page integrated with model, allows for user to play another game in a tournament. Issues occurred with tournament scoring and changing player color. Needed to tweak Player class. (1.75 hr)
  + Tested with provided cases to make sure loading an AI working (1 hr)
  + Tweaked AI due to strange issues, improved quite a bit. (2 hrs)
  + Got saving files from the browser working. (.75 hrs)
  + AI tweaked to move a specific piece for the computer’s 2nd move. (1 hr.)