Ryan King Senior Project proposal, 2/5/2019
CMPS 450
Dr. Miller

My project will be a chess engine.
Chess is a 2-player game where the objective is to threaten the opponent with inescapable capture, come to a draw, or force the opponent to resign.

There are six types of pieces in chess:

The pawn, which is the weakest piece in the game. It can only move 1 square forward if the square in front of it is unoccupied, but it has the privilege of moving 2 squares on its first move assuming that square and the one before it is unoccupied. It captures by moving forward a single space diagonally. It can be promoted into a knight, bishop, rook, or queen if it reaches the opposite side of the board in a process known as *promotion*. It can also capture another pawn *en passant*, meaning if it is on its fourth rank, it may capture a pawn that moved two squares on the move that immediately preceded the current one. The capture finishes like an ordinary capture. It is depicted as a short piece with bumps that taper with height. If this opportunity is not taken immediately, the pawn will lose this privilege on the next turn. It is depicted as a series of tapering balls.

All other pieces capture the same way they move. If a piece can capture another, only that square the captured piece is resting on can be traveled to on the current turn. If a friendly piece is on a square the piece can move to, in the case of a knight, it cannot move. Otherwise, the piece can land up to the square immediately preceding said piece.

The king, which is the most important piece of the game. He can move one square in any direction so long as that square does not result in him being threatened (in check). He also has a special move called castling, which he can perform with either of his rooks if neither he nor the rook has moved, the spaces between them are empty, and the king is not in check that turn and will not pass through check while he attempts to castle. This rule about avoiding check does not apply to the rook. Such a move can only be initiated through clicking the king. He is depicted as a crown with two visible “humps” and a small cross in the center. Placing the king into inescapable capture (checkmate) is the primary objective of the game.

The knight, which moves uniquely. It can move either two squares in one direction and one square in a perpendicular direction, or vice versa. Its movement is not restrained by

any pieces in its way, allowing it to “jump” over them. It is depicted as the head and neck of a horse.

The bishop, which moves diagonally. It is depicted as a miter.

The rook (or castle), which moves horizontally. It can castle with a king if the king hasn’t moved yet. It is depicted as a tower of a castle.

The queen, which has the combined movements of a bishop and a rook. She is depicted as a crown with five spikes that have balls on their ends.

My program will contain an AI which will use Minimax as its move strategy.

Challenges I anticipate to run into include making every piece move differently, taking move history into account, making a user-friendly UI, synchronizing the model with the view, implementing the AI, multithreading, pawn promotion, serialization with FEN file formats, resuming games from serialization strings, accounting for differences between humans and computers, and efficiency.

To represent game pieces and the playing field, the program contains an abstract Piece class which is derived into any of the above six pieces, a Tile class which contains its own color, row, column, and a Piece pointer, a Board class which contains sixty-four tiles alternating between black and white. An internal BoardBuilder class exists to help instantiate boards.

A Player class will also be included and will be derived to Human and Computer.

For the GUI, the program will have a GUIBoard which contains sixty-four GUITiles as explained above. Each GUITile will contain a Tile object and will have access to its piece (if any) as well as most of its fields.

The main engine class (DarkBlue) will have a GUIBoard, two Players, a source tile, a destination tile, a moved piece object, a GameWatcher class, and many symbolic constants.
The GUI will be written using the Java Swing API and will make ample use of the MouseListener interface, specifically `mouseClicked()`. Messages will be displayed to the human player via JOptionPane. Strings can be given to the program in Forsyth-Edwards notation (FEN) to resume a game from a certain point or to create your own game.

An abstract Move class exists to modify the board. It will extend to RegularMove (a piece moves to a random spot without capturing another), AttackingMove (a piece captures another, except for pawns capturing *en passant*), CastlingMove (the king and rook switching places on his first turn), and an EnPassantMove (For capturing *en passant* as described above).

A Utilities interface contains most numeric constants and a few utility methods.
A BoardUtilities interface contains methods that can determine if coordinates are valid, colors for the GUITiles, etc.

A GameState enumeration will provide an easy way to keep track of the game and make the code more readable.

A Factory interface will provide factory methods to construct pieces, tiles, etc.