SEG2105 Assign 5 – Group 53 – Mark Kasun (3806554), Patrick Shortt (6036229)

Problem Statement

The system will be responsible for handling a chess game played between two players.

Requirements

The system will include a chess board and all the pieces on said board. Each piece type will have certain valid move combinations. The player will have the ability to move pieces on the board when it is their turn. The system will be responsible for generating a graphical representation of the board for each player to see. In early versions, the program will generate a textual display on a command line interface, with options to expand to html or android interfaces in the future. The system will programmed in such a way as to make it easily modifiable to other variants of chess and even non-chess board games.

User Stories

Start New Game: Either player must be able to start a new game and let another player join it. Upon starting the game, the user must decide who goes first or to randomly decide who goes first. The system will then generate the starting board and all necessary components.

Join New Game: A user must be able to join a game set up by another user. The system will check that no one else has already joined that game and connect the two users.

Taking a Turn: Assuming it is the specific player's turn, the player must be able to select a piece and select a destination square. The system then determines whether this is a valid move. If it is not a valid move, a message is sent to the user. Otherwise, the move and consequences are resolved and the next player's turn begins.

Ending the Game: The user may end the game prematurely by resigning or offering a draw to the other player. In the event of a draw offer, the other player will be prompted to accept or deny the draw. The game will also naturally end when one player's King is placed in checkmate.

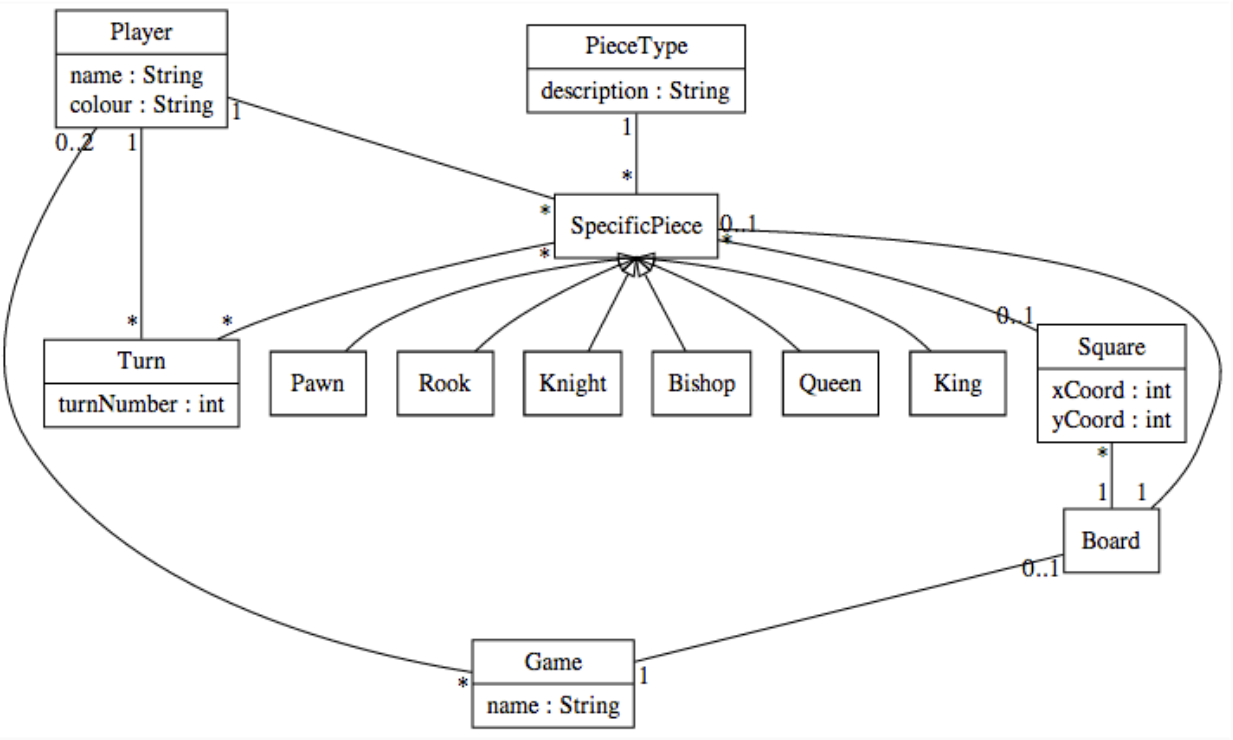
Architecture

For this chess application we will be designing it using Java. In early versions the game will be rendered using command line and text based representation. By using Java, we have the option to add an Android UI in future releases assuming time is available.

The server will be responsible for the bulk of the system. This is a thin-client system. The server will keep data on the current game state, determine validity of moves, and ensure users are unable to cheat while playing the game.

The client will be responsible for receiving commands from the user and transmitting them to the server. Such commands will be starting a new game, taking a turn, resigning, offering a draw, etc. The client will also be responsible for displaying the board state to the users.

UML Diagram



Messages between Server and Client

Client1: Start new game <arguments>
Server: “New game started successfully; Waiting for opponent.”
Client2: Join new game <arguments>.
Server: Generate board and send to clients.
Server: “Game started; Awaiting move from white.”
Client1: <square1 to square2>
Server: Send updated board state to clients.
Client2: <square1 to square2>
Server: Send updated board state to clients.
…
Client1: End game.
Server: <Client1> has ended the game.