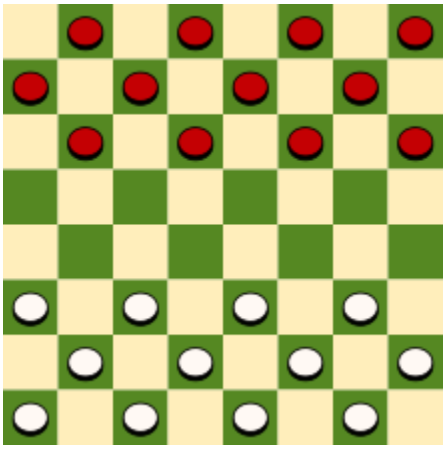
AP(IT) Assessed Coursework

Design Document

The chosen game for this coursework is English draughts. The aim of the game is to win, which happens as described in the Rules section below.

Rules:

“Starting position



The starting setup; Red moves first.

Each player starts with 12 men on the dark squares of the three rows closest to that player's side (see diagram). The row closest to each player is called the kings row or crownhead. The player with the darker-coloured pieces moves first. Then turns alternate.

Move rules

There are two different ways to move in English draughts:

1. Simple move: A simple move consists of moving a piece one square diagonally to an adjacent unoccupied dark square. Uncrowned pieces can move diagonally forward only; kings can move in any diagonal direction.
2. Jump: A jump consists of moving a piece that is diagonally adjacent an opponent's piece, to an empty square immediately beyond it in the same direction. (Thus "jumping over" the opponent's piece.) Men can jump diagonally forward only; kings can jump in any diagonal direction. A jumped piece is considered "captured" and removed from the game. Any piece, king or man, can jump a king.

Multiple jumps are possible, if after one jump, another piece is immediately eligible to be jumped—even if that jump is in a different diagonal direction. If more than one multi-jump is available, the player can choose which piece to jump with, and which sequence of jumps to make. The sequence chosen is not required to be the one that maximizes the number of jumps in the turn; however, a player must make all available jumps in the sequence chosen.

Jumping is always mandatory: if a player has the option to jump, he must take it, even if doing so results in disadvantage for the jumping player. For example, a mandated single jump might set up the player such that the opponent has a multi-jump in reply. Any piece can move in any direction if it is going to double jump. Which means that if you jump your opponents piece you can go backward in order to jump another piece.

Kings

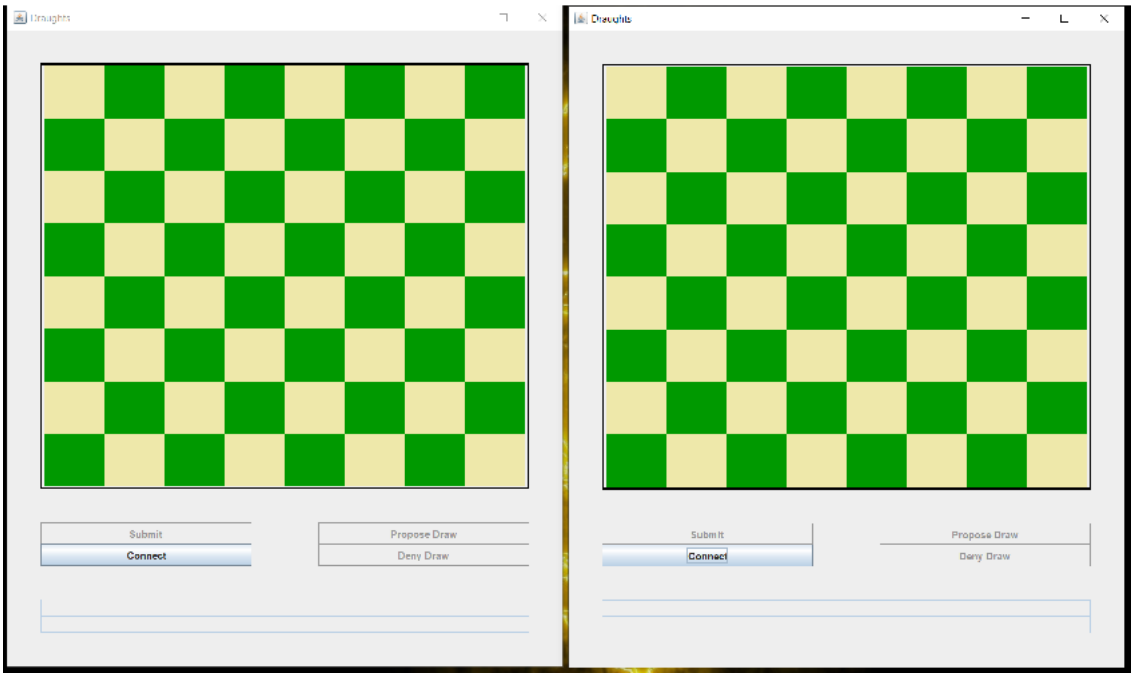
If a man moves into the kings row on the opponent's side of the board, it is crowned as a king and gains the ability to move both forward and backward. If a man moves into the kings row or if it jumps into the kings row, the current move terminates; the piece is crowned as a king but cannot jump back out as in a multi-jump, until another move.

End of game

A player wins by capturing all of the opponent's pieces or by leaving the opponent with no legal move. The game ends in a draw if neither side can force a win, or by agreement (one side offering a draw, the other accepting).**”** (text and picture taken from https://en.wikipedia.org/wiki/English_draughts)

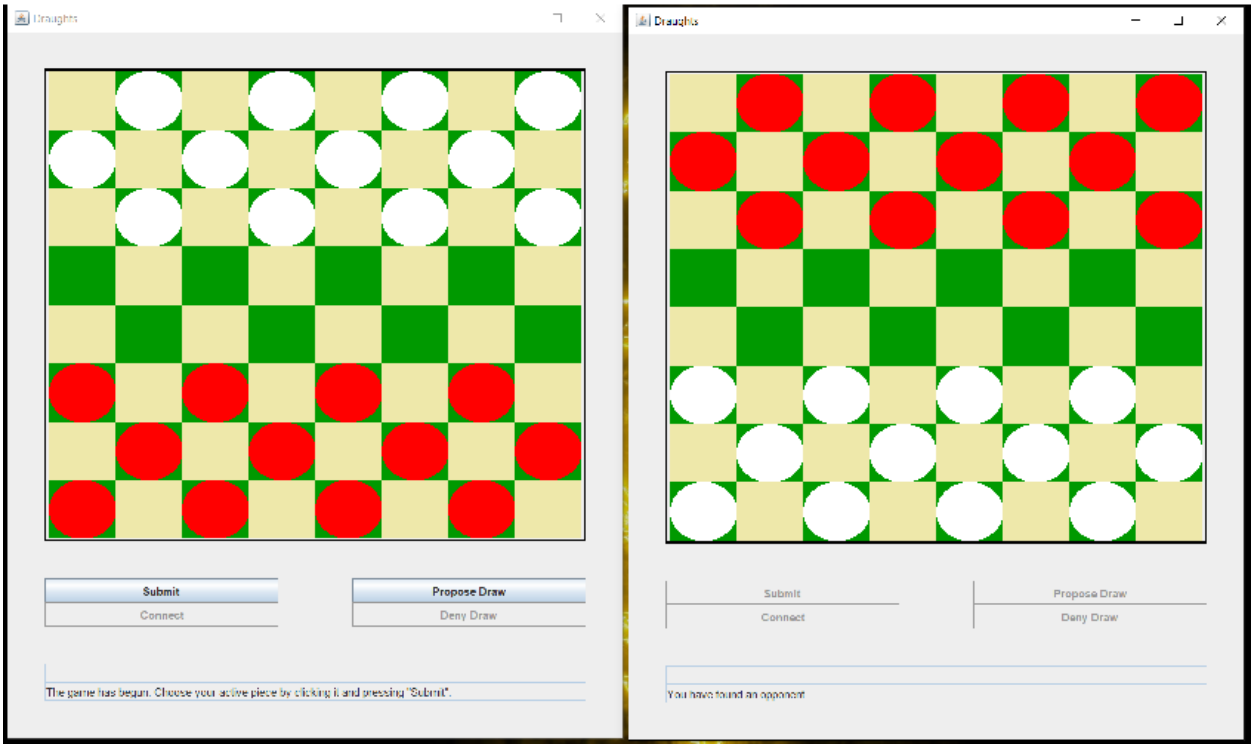
Design:

In the current implementation of the game the Client class’ main method starts up two Client windows(initially on top of one another) on the host computer which try to connect to localhost when the “Connect” button is clicked.



The Server class’ main method needs to be started up before the Clients attempt to connect in order for the connection to be possible.

After both Clients connect successfully, their colour is randomly selected and the game begins with the red player being first. A player begins with his pieces at the board’s bottom.



There are two JTextFields on each Client, which are used to present messages to the player regarding what they currently need to do and if they made any errors in previous selections.

At the beginning of their turn, a player must first click on a piece and press submit. If the player clicked on a valid piece(one that exists, belongs to them, can move and can jump if there is piece that can jump in the current turn) then the player must click on the new position for the piece and press submit, otherwise an error message will be displayed and the player will need to select a piece again. If the new position is a valid one(depending on who the player is, whether his piece is a king and what type of move it's making) the piece is moved and the player is potentially given another move if the piece has jumped, hasn't become a king and can still jump, otherwise an error message will be displayed and the player will need to select a move again. After all moves a player can make for a turn have been made, control passes to the next player.

During their turn a player may propose a draw via the "Propose Draw" button. Their opponent will then be able to either accept it or deny it on their next turn. Acceptance is done via the "Propose Draw" button and denial via the "Deny Draw" button, which becomes active after a draw proposal. A player is informed about whether their opponent proposed or denied a draw after their turn starts.

At the beginning of each turn the server checks if any player has legal moves left. If none of them have legal moves left, the game ends in a draw. If just one of them hasn't got legal moves left, the game ends with them losing.

If any player loses all their pieces in game, the game ends with them losing.

After the game is over, each player is given 3 seconds to read whether they won/lost or the game ended in a draw, and then the game restarts from the beginning automatically.

At any point, any Client can press X on their window, which will stop the Server and both Clients, and close their windows.

Application level protocol:

The Application level protocol consists of two custom Serializable Java classes: RequestPacket and ResponsePacket. They basically exist as wrappers of the following fields for each:

RequestPacket:

```
private boolean hasProposedDraw;
private boolean hasDeniedDraw;
private boolean hasProposedPiece;
private boolean hasProposedMove;
private int proposedRow;
private int proposedColumn;
```

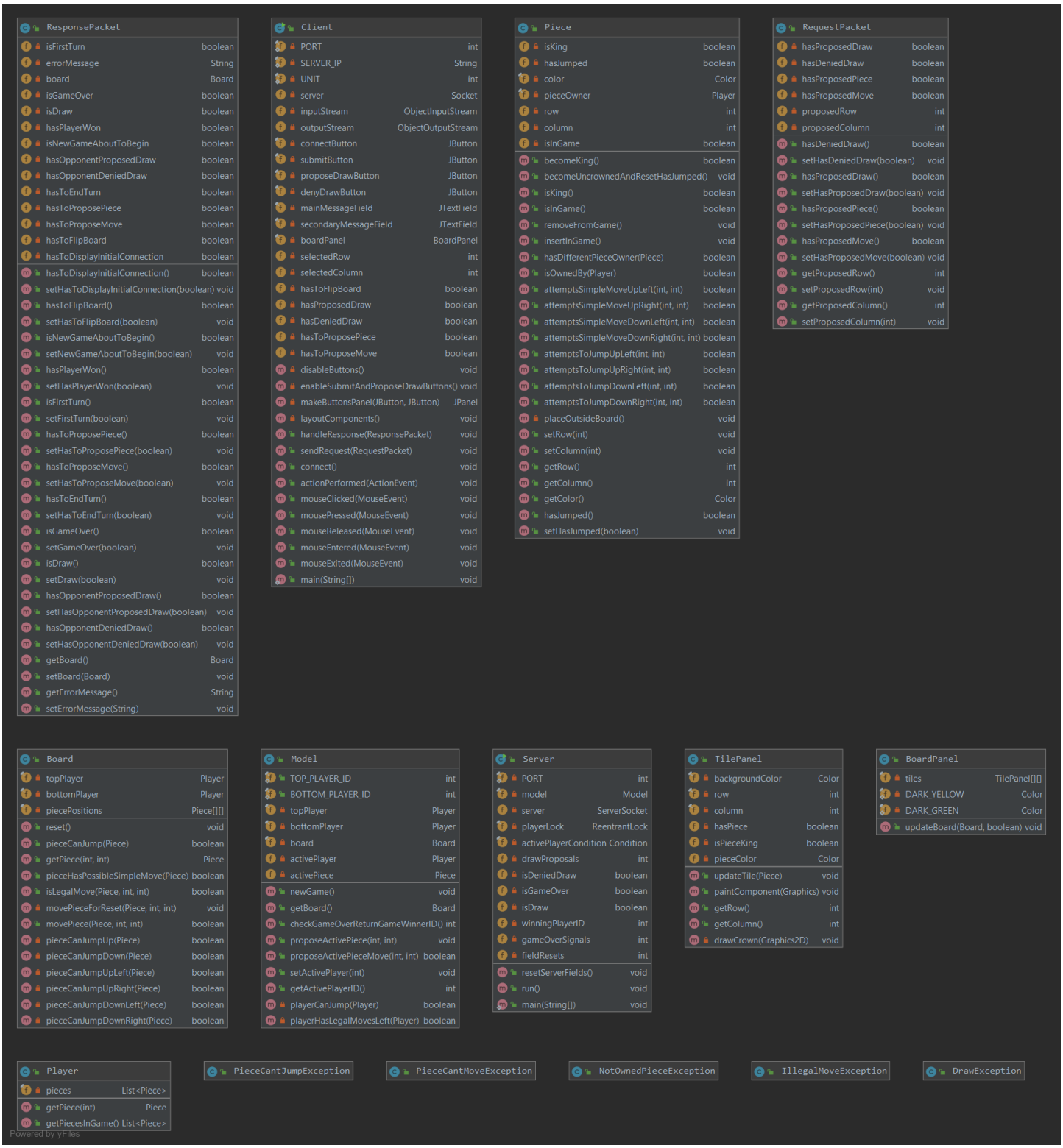
ResponsePacket:

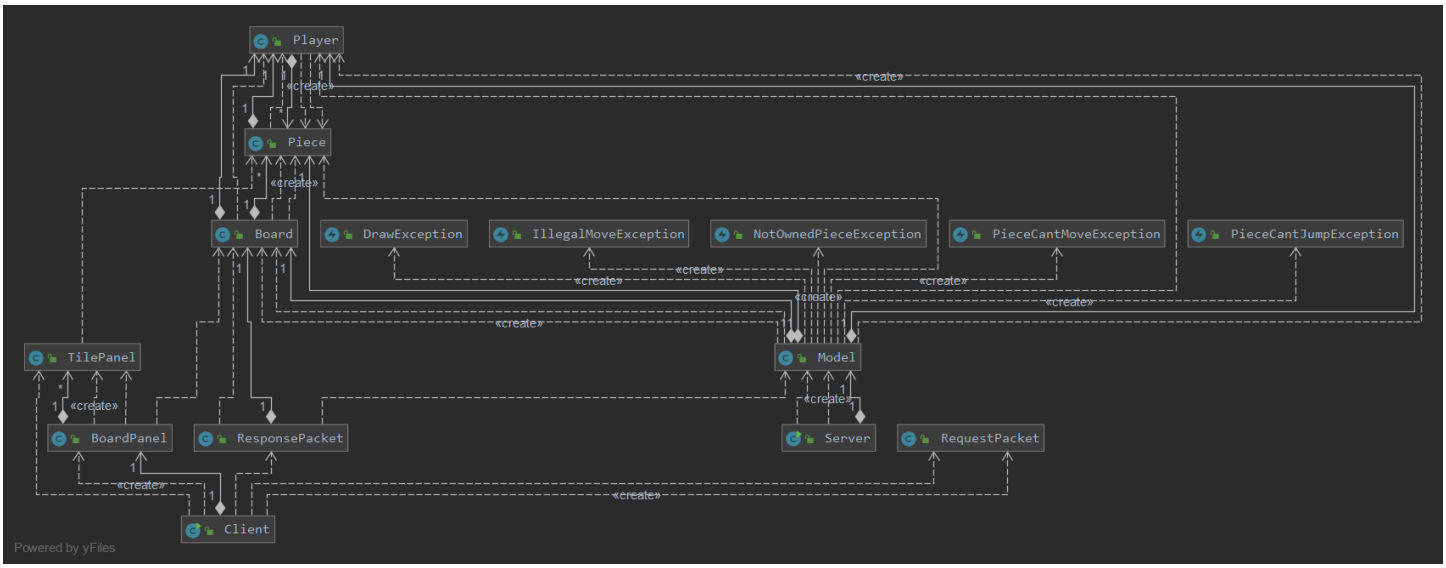
```
private boolean isFirstTurn;
private String errorMessage;
private Board board;
private boolean isGameOver;
private boolean isDraw;
private boolean hasPlayerWon;
private boolean isNewGameAboutToBegin;
private boolean hasOpponentProposedDraw;
private boolean hasOpponentDeniedDraw;
private boolean hasToEndTurn;
private boolean hasToProposePiece;
private boolean hasToProposeMove;
private boolean hasToFlipBoard;
private boolean hasToDisplayInitialConnection;
```

The RequestPacket is sent from a Client to the Server and the ResponsePacket is sent from the Server to a Client. The Client-Server communication for each turn starts with the Server “waking up”(turning on the player’s buttons) the active player’s Client with an initial Response and is then followed by Request-Response pairs until the player ends their turn and goes back to “sleep”(their buttons are disabled until they get their next turn). Then the other player goes through the same mechanism.

The RequestPacket’s fields are mostly self-explanatory based on their names. The ResponsePacket’s fields are mostly self-explanatory as well except for a few standouts. The board field sends a board object(the irrelevant fields for the Client have been made transient) which holds information about the newest piece distribution on the board. Every Piece object(the irrelevant fields for the Client have been made transient) on the board holds information about its colour and whether or not it is a king to ensure proper rendering. The hasToFlipBoard field reflects whether the player’s Client needs to flip the board it received from the Server to ensure that the player’s pieces start out on the bottom of their screen. The hasToDisplayInitialConnection field serves to display the fact that an opponent has been found to both players before the first player starts their turn.

UML Diagrams(continues on next page):




Player
Piece
Board
DrawException
IllegalMoveException
NotOwnedPieceException
PieceCantMoveException
PieceCantJumpException
TilePanel
Model
BoardPanel
ResponsePacket
Server
RequestPacket
Client
«create»
1
*
Powered by yFiles