Program Architecture

This is a Tic Tac Toe game that is written in Java 1.7.0_10. It implements two different ai's. One is a thoughtful ai that makes intelligent moves and the other is a naïve ai that will occasionally make bad moves. The program is divided up into 9 classes:

1. TicTacToe: This is the main class that handles command line arguments and starts the game.
2. Engine: This class runs the game by calling the getMove() method from the Player subclasses, keeps track of the board, and checks for end of game conditions.
3. Node: This class represents a single square on the game board and is mainly used for display purposes.
4. Player: This class is an interface for the different types of players so that getting the move from the players can be a generic call.
5. Human: This class is used for a human playing the game and handles command line input and illegal move choices.
6. Thoughtful: This class runs the thoughtful ai (discussed below).
7. Naive: This class runs the naïve ai (discussed below).
8. Strategy: This class holds all of the information for the strategy in the game. It makes decisions based off of the rules of the game.
9. Rules: This class holds the information about the rules of the game.

Thoughtful AI

The thoughtful ai uses a strategy class to decide what move to pick. The strategy class has three steps. First checks if the agent can win the game, then checks if the opponent can win the game, finally, it decides to use a strategic positioning move. The strategy class checks for winning moves by using a rule class that is made up of two rules in the tic tac toe game. The first rule is that if the the agent gets three spaces in a row it wins. This is checked by representing each row, column, and diagonal as a string of 0's for free spaces, 1's for player 1 and 2's for player 2, and matching it to a winning string of the players number. Below is a trace from a game where the agent is player 1 and finds a winning move in the third column:

```
Looking for a winning move by checking win condition.
Looking for winning move.
Checking 221
Checking 010
Checking 201
Checking 202
Checking 210
Checking 101
Winning move found in third column.
Picking a winning move.
```

Since the ai is player 1 it is looking for one of three winning strings: 110, 101, or 011, and it finds one in the third column. The second rule is if the opponent gets three spaces in a row it will win. Since it is illegal to pick a square that has already been chosen, choosing that square will stop the opponent from winning. The ai does this in the same way it checks for its own winning move but this time it looks for a pattern matching the opponents player number, 2, which can be seen in the trace below:

```
Looking for a blocking move by checking win condition for opponent.
Looking for opponent winning move.
Checking 201
Checking 010
Checking 200
Checking 202
Opponent winning move found in first column.
Picking a blocking move.
```

Since the opponent is player 2 it finds the pattern in the first column and picks the opponents winning move so that the opponent cannot and therefore does not win. If the thoughtful ai finds that it cannot make a winning move and the opponent cannot make a winning move it will make a strategic move. When the strategy class makes a decision about which type of move the agent should make, it returns its choice. The thoughtful agent gets the strategy move and searches its knowledge base to find the correct move to use. The knowledge base is sorted into winning, blocking, and strategic moves. The knowledge base is represented as strings in a text file. Each state is saved on three lines:

1. they type of knowledge represented
2. the game board
3. the optimal move

When the ai is sorting through the knowledge base it will only look at the saved board if it is the same type of knowledge as it is looking for (ex: if it is looking for a winning move it will skip the knowledge about blocking or strategy). The game board is represented as a 9-character string where free spaces are 0's, opponent moves are !'s and the players moves are ~'s. The optimal move is a number related to the index on the board of the optimal move where the board is represented with numbers 0-8. Below is an example board and how it would be saved in memory if the player was X's:

```
  | X |O
--------
  | X |O
---------
X|   |

w
0~!0~!~00
7
```

The w is for a winning move because if the ai selects position 7 it will win the game.

Naive AI

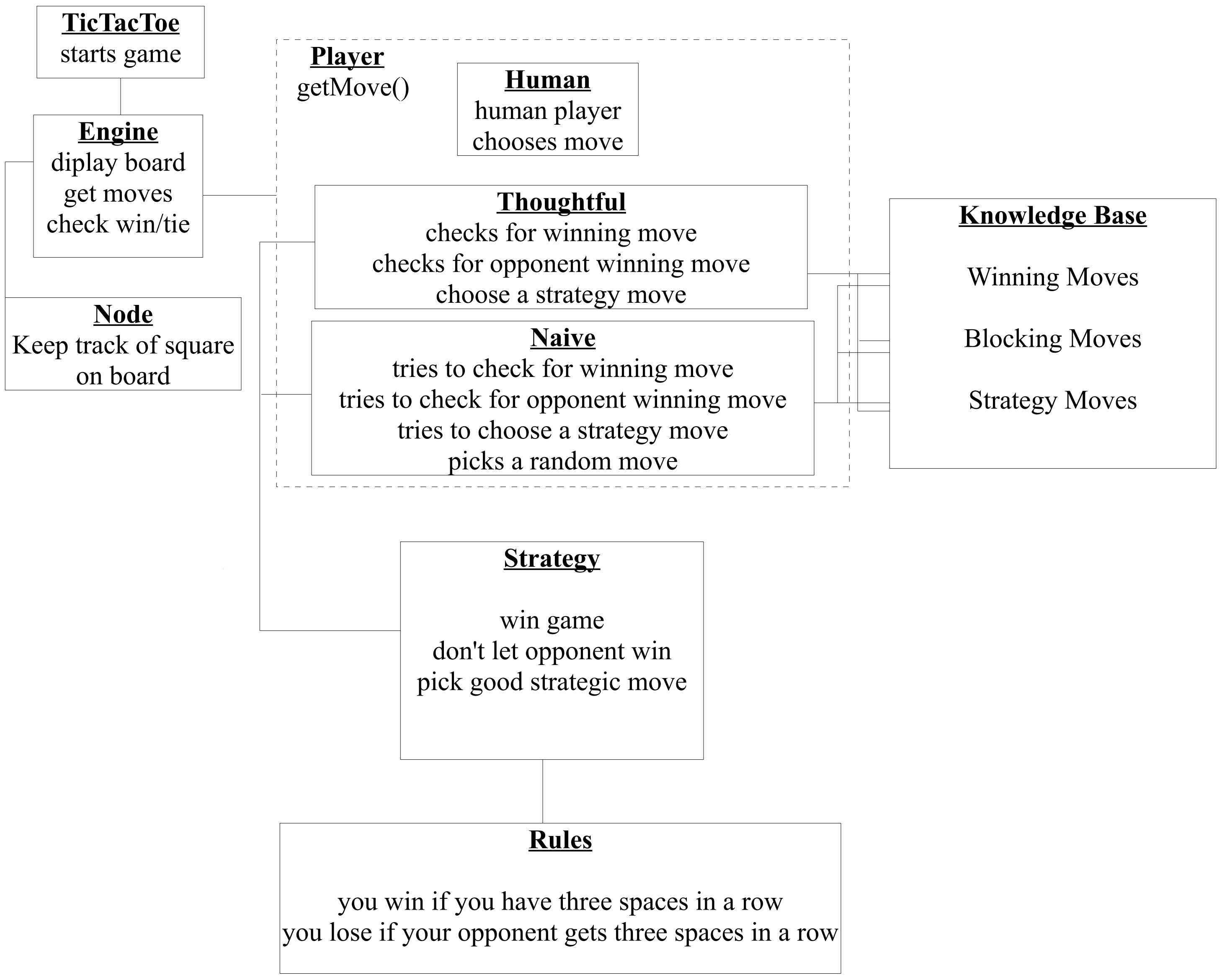
The naïve ai uses the same exact technique as thoughtful ai. It uses the same strategy class, and the strategy class uses the same rules class. It also uses the same knowledge base as thoughtful ai. The difference between naïve and thoughtful is that naïve has a chance to not look for good moves. Even naïve players of tic tac toe know the rules, but sometimes they either don't see a good move or forget to check for an opponent's good move. Naive ai has a random chance to skip some or all of the strategic steps. If it can make a winning move, but does not check for a winning move, then it will incorrectly classify the type of move it needs to make and as a result it will not find the current board in the knowledge base. It will then choose randomly. This goes the same for not checking for opponent winning moves or strategic moves. Below is an example of not checking for an opponent's (X's) winning move:

```
   | O | X
-----------
   | X |
-----------
 O |   | X

Attempting to look for a winning move by checking win condition.
Looking for winning move.
Checking 021
Checking 010
Checking 201
Checking 002
Checking 210
Checking 101
Checking 011
Checking 112
Attempting to look for a blocking move by checking win condition for opponent.
Picking a strategy move.

 O | O | X
-----------
   | X |
-----------
 O |   | X
```

Normally the ai would look to see if the opponent had a winning move after looking to see if it could win, but in this case it “forgot” to check and picked a random move instead.


TicTacToe
starts game
Engine
diplay board
get moves
check win/tie
Node
Keep track of square on board
Player
getMove()
Human
human player chooses move
Thoughtful
checks for winning move
checks for opponent winning move
choose a strategy move
Naive
tries to check for winning move
tries to check for opponent winning move
tries to choose a strategy move
picks a random move
Knowledge Base
Winning Moves
Blocking Moves
Strategy Moves
Strategy
win game
don't let opponent win
pick good strategic move
Rules
you win if you have three spaces in a row
you lose if your opponent gets three spaces in a row

Thoughtful vs. Naive Experiment

This experiment was run with thoughtful ai as player 1 and naïve ai as player 2. In this experiment thoughtful ai won 75% of the time and tied the other 25%. Whenever the thoughtful ai won it was because the naïve ai failed to look to see if the opponent could win or failed to pick a strategy move and was caught in a no win situation like the one below:

```
O |    |
------------
   | X | O
-------------
X |    | X
```

In this situation O's cannot block both of X's winning moves so X will win. In the games that ended up as a tie the naïve ai either did not fail to check for a type of move, failed to pick a good strategy move but recovered later, or picked a random move that happened to be the best move. This was what I was expecting the experiment to look like. The naïve ai misses looking for an opponent's winning move almost every game and the thoughtful ai shouldn't make any mistakes so the thoughtful ai should be winning most of the games it plays. Below is an example of the experiment being run, my comments are inside of the parenthesis.

Thoughtful(X's) vs. Naive(O's):

```
  |  |
-----------
  |  |
-----------
  |  |

Looking for a winning move by checking win condition.
Looking for winning move.
Checking 000          (this is the first row)
Checking 000          (second row)
Checking 000          (third row)
Checking 000          (first column)
Checking 000          (second column)
Checking 000          (third column)
Checking 000          (diagonal left to right)
Checking 000          (diagonal right to left)
Looking for a blocking move by checking win condition for opponent.
Looking for opponent winning move.
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
Picking the best strategic move.
```

(The thoughtful ai looked for a win first, then it looked for a move that would cause the opponent to win, finally it chose to pick a strategic move when neither of the rules it checked applied.)

```
   |   |
-----------
   | X |
-----------
   |   |

Attempting to look for a winning move by checking win condition.
Looking for winning move.
Checking 000
Checking 010
Checking 000
Checking 000
Checking 010
Checking 000
Checking 010
Checking 010
Attempting to look for a blocking move by checking win condition for opponent.
Looking for opponent winning move.
Checking 000
Checking 010
Checking 000
Checking 000
Checking 010
Checking 000
Checking 010
Checking 010
Picking the best strategic move.
```

(The naïve ai successfully checked to see if it could win the game and if the opponent could win the game. Neither of the rules it checked applied so it successfully chose a strategic move.)

```
   |   |
-----------
   | X |
-----------
 O |   |

Looking for a winning move by checking win condition.
Looking for winning move.
Checking 000
Checking 010
Checking 200
Checking 002
Checking 010
Checking 000
Checking 010
Checking 012
```

```
Looking for a blocking move by checking win condition for opponent.
Looking for opponent winning move.
Checking 000
Checking 010
Checking 200
Checking 002
Checking 010
Checking 000
Checking 010
Checking 012
Picking the best strategic move.
```
(The thoughtful ai went through its strategy of checking for a winning move, checking for a move to block the opponents win, and finally choosing a strategic move again.)

```
   |   | X
-----------
   | X |
-----------
 O |   |
```

```
Attempting to look for a winning move by checking win condition.
Looking for winning move.
Checking 001
Checking 010
Checking 200
Checking 002
Checking 010
Checking 100
Checking 010
Checking 112
Attempting to look for a blocking move by checking win condition for opponent.
Picking the best strategic move.
```
(The naïve ai failed to look for the opponents winning move this round, but there was not a winning move to block. It did successfully check for a win and choose a strategic move though.)

```
 O |   | X
-----------
   | X |
-----------
 O |   |
```

```
Looking for a winning move by checking win condition.
Looking for winning move.
Checking 201
Checking 010
Checking 200
Checking 202
Checking 010
Checking 100
Checking 210
Checking 112
Looking for a blocking move by checking win condition for opponent.
Looking for opponent winning move.
Checking 201
Checking 010
Checking 200
Checking 202
Opponent winning move found in first column.
Blocking opponents winning move.
```

(The thoughtful ai found the opponents winning move with rule two so the second step in its strategy told the ai to pick a blocking move and it successfully blocked the opponent from winning.)

```
 O |   | X
-----------
 X | X |
-----------
 O |   |

Attempting to look for a winning move by checking win condition.
Attempting to look for a blocking move by checking win condition for opponent.
Looking for opponent winning move.
Checking 201
Checking 110
Opponent winning move found in second row.
Blocking opponents winning move.
```

(The ai failed to check for a winning move, but did not fail to check for a blocking move. It found the opponents winning move with rule two so the second step in its strategy told the ai to pick a blocking move and it successfully blocked the opponent from winning.)

```
 O |   | X
-----------
 X | X | O
-----------
 O |   |

Looking for a winning move by checking win condition.
Looking for winning move.
Checking 201
Checking 112
Checking 200
Checking 212
Checking 010
Checking 120
Checking 210
Checking 112
Looking for a blocking move by checking win condition for opponent.
Looking for opponent winning move.
Checking 201
Checking 112
Checking 200
Checking 212
Checking 010
Checking 120
Checking 210
Checking 112
Picking the best strategic move.
```
(The thoughtful ai received the strategy choice from the strategy class because the rules from the other two steps did not apply.)

```
 O |   | X
-----------
 X | X | O
-----------
 O | X |
```

```
Attempting to look for a winning move by checking win condition.
Looking for winning move.
Checking 201
Checking 112
Checking 210
Checking 212
Checking 011
Checking 120
Checking 210
Checking 112
Attempting to look for a blocking move by checking win condition for opponent.
Picking the best strategic move.
```
(The naïve ai failed to check for the opponents winning move and incorrectly classified the best move as a strategic move. This caused the ai to pick the incorrect move.)

```
 O |   | X
-----------
 X | X | O
-----------
 O | X | O
```

```
Looking for a winning move by checking win condition.
Looking for winning move.
Checking 201
Checking 112
Checking 212
Checking 212
Checking 011
Winning move found in second column.
Picking a winning move.
```
(The thoughtful ai found a winning move. The first rule returned true so the first step in the strategy class returned the instruction to pick a win move.)

```
//////////////////////////////
Player 1 wins!

 O | X | X
-----------
 X | X | O
-----------
 O | X | O

//////////////////////////////
```

Questions

1. Why did you pick your last move?
In the trace the agent records what it is "thinking." It could go to when it records which move it is choosing and what comes before would explain why it chose what it did.

```
Looking for a winning move by checking win condition.
Looking for winning move.
Checking 201
Checking 112
Checking 212
Checking 212
Checking 011
Winning move found in second column.
Picking a winning move.
```

Before it said it was picking a winning move it says it found a winning move in the second column and before that is said it was looking for a winning move. So it would answer the question with: I was looking for a winning move and found one in the second column.

2. What made you decide there was a winning move in the second column?
The trace shows how the ai is reasoning about the state of the game. It says it is looking for a winning move and checking numbers. The numbers represent the players locations on the board, 1 for player 1, 2 for player 2, and 0 for no player. When the trace says a move was found it is because it found a string that matched what it was looking for and a rule passed.

```
Looking for a winning move by checking win condition.
Looking for winning move.
Checking 201
Checking 112
Checking 212
Checking 212
Checking 011
Winning move found in second column.
Picking a winning move.
```

This shows the ai finding a winning move in the second column. The ai checks the rows, then columns, and then the diagonals when it is checking its two rules. By tracing backwards it can see that when it checked the second column it matched one of the strings it was looking for for a winning move. So it would answer the question with: I found a winning move string in the second column while checking the first rule.

Differences

In project two the agents focused on using rules of the game to make decisions about the game. In this project the ai used strategy to choose which rules to apply to the situation or which order to check the rules. The overall architecture of the two projects is still very similar. The naïve is the same as thoughtful but it has a chance to "forget" to do something. In this project the naïve would forget to go through each step in the strategy. In project two, if naïve forgot to check for a rule it would just miss the move, but in this project it would incorrectly categorize the move it should make. By using a strategy class it is easier to trace why the naïve ai made a bad move by comparing what category they chose to what it should have been.