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.
10. Trace: This class represents the trace of a single move in the game. It keeps the turn number, the type of move chosen, the correct type of move (might be different from the one chosen), the position of the chosen move, and a message.

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

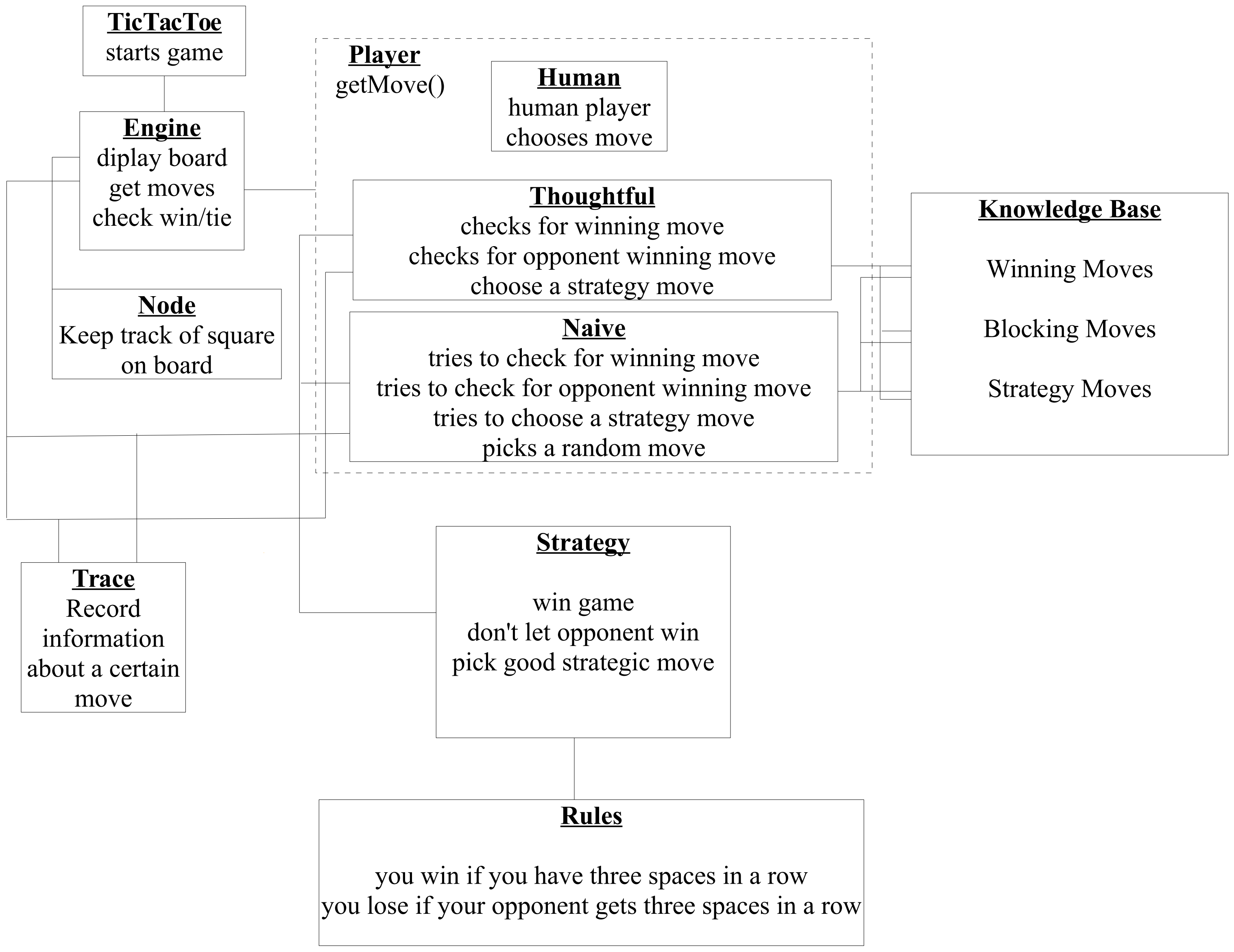
The naïve ai uses the same exact technique as thoughtful ai. It uses the same strategy class, and the strategy class uses the same rules class. It also uses the same knowledge base as thoughtful ai. The difference between naïve and thoughtful is that naïve has a chance to not look for good moves. Even 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
Trace
Record information about a certain move
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

Questioning

Questions can be answered by searching through the traces recorded during the game. Each agent keeps track of the moves it makes during the game and specific details of that move as an array of trace objects. A trace object stores the turn number, the type of move the agent thought it should make, the correct type of move it should make (this can be different from what type it chose), the position on the board it chose, and a message about the move. Questions are asked by first choosing which player you are going to ask a question and then which turn you want to ask about. When the player is chosen that players trace is retrieved. Then the the specific trace that corresponds to the turn chosen is retrieved. Once the correct trace is selected it can be asked three questions: "which", "why", and "wrong". "Which" stands for "Which position did you select?" When the agent is asked this question it responds with "I placed my move in [position on board move was placed]." The position is the index on the board that was selected. Instead of placing the number index there a string is placed that relates to the position on the board. For example, if the position selected was 0 it would return "I placed my move in top left corner." The strings are stored in an array where the positions on the board correspond to a natural language description of the position. "Why" stands for "Why did you select the move you did?" When the agent is asked this question it responds with "[knowledge used to make type of move classification]" "I decided the best move was a [type of move] move." For example, if the agent was asked why on a turn it made the winning move in the second column it would return "Winning move found in second column." "I decided the best move was a winning move." "Wrong" stands for "What did you do wrong on this turn?" When the agent is asked this question it responds with one of two answers. First it checks to see if the type of move it chose is the same as the type of move it actually should be. If it is the same it will return "I did nothing wrong." If the two are different it will return "I decided the move was a [type of move chosen] move, but it was actually a [real move type] move." "I classified the move incorrectly and therefore made a bad move." This can only be seen with the naïve agent because the thoughtful agent never makes bad moves (or it shouldn't).

Thoughtful vs. Naive Experiment

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

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

Player 1 turn: 1
Looking for a winning move by checking win condition.
Looking for winning move.
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
Checking 000
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.

   |   |
-----------
   | X |
-----------
   |   |

Player 2 turn: 1
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.

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

Player 1 turn: 2
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.

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

Player 2 turn: 2
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 a random move.

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

Player 1 turn: 3
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.

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

Player 2 turn: 3
Attempting to look for a winning move by checking win condition.
Attempting to look for a blocking move by checking win condition for opponent.
Picking the best strategic move.

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

Player 1 turn: 4
Looking for a winning move by checking win condition.
Looking for winning move.
Checking 201
Checking 110
Winning move found in second row.
Picking a winning move.
//////////////////////////////
Player 1 wins!

 O |   | X
-----------
 X | X | X
-----------
 O | O |
//////////////////////////////
Which player do you want to ask a question?
1
Which turn are you interested in?
4
What type of question do you want to ask?
why

Winning move found in second row.
I decided the best move was a winning move.
(The thoughtful agent found a winning move so it classified the move as a winning move)

Would you like to ask another question?
yes
Which player do you want to ask a question?
2
Which turn are you interested in?
3
What type of question do you want to ask?
why

I did not find any winning or blocking moves.
I decided the best move was a strategic move.
(The naïve agent failed to find any winning or blocking moves so it classified the move as a strategic
move)

Would you like to ask another question?
yes
Which player do you want to ask a question?
2
Which turn are you interested in?
3
What type of question do you want to ask?
wrong

I decided the best move was a strategic move, but it was actually a blocking move.
I classified the move incorrectly and therefore made a bad move.
(The agent incorrectly classified the move, it should have classified it as a blocking move. It realizes
classifying the move incorrectly is bad)

Would you like to ask another question?
yes
Which player do you want to ask a question?
2
Which turn are you interested in?
1
What type of question do you want to ask?
which
```

I placed my move in bottom left corner.
(The agent translates it's move into an easy to understand location on the board)

Would you like to ask another question?
yes
Which player do you want to ask a question?
1
Which turn are you interested in?
3
What type of question do you want to ask?
why

Opponent winning move found in first column.
I decided the best move was a blocking move.
(The thoughtful agent found an opponents winning move so it classified the move as a blocking move to stop the opponent from winning)

Would you like to ask another question?
yes
Which player do you want to ask a question?
1
Which turn are you interested in?
1
What type of question do you want to ask?
wrong

I did nothing wrong.
(The thoughtful agent did not make a mistake on this turn so it says so)

Would you like to ask another question?
no

<u>Result:</u>
The questions show how the naïve agent can make mistakes in classifying moves. The agents use a knowledge base of moves to play the game and if they do not classify the move correctly they will not find the case in the knowledge base. The naïve agent has a chance to forget to do every step in its strategy so it can incorrectly classify a move sometimes.