ECS629/759 Artificial Intelligence: Assignment 1

Due date: Fri 4 March 2016

The aim of this assignment is to write a Java class to play the game Gomoku, or five-in-a-row, on an 8×8 board, using alpha-beta search. The game is played as follows: two players (white and black) take turns at placing a stone of his/her colour on an unoccupied square on the board (white moves first). The first player to complete a continuous horizontal, vertical or diagonal line of 5 stones of his/her colour is the winner (scoring 2 points). The loser scores 0. If all squares are occupied and neither player has won, then each player gets 1 point.

Your program will be given a time limit of 10 seconds per move (on the ITL computers). Any program which exceeds this limit will immediately forfeit the game to its opponent. Similarly any program which raises an exception or makes an illegal move (out of range or already occupied) will lose immediately. There are no other restrictions on moves (gomoku experts may be aware that some tournaments have further restrictions).

Your player class must be an instance of the abstract class `GomokuPlayer`, so you need to write a class which extends `GomokuPlayer` and overrides the method `chooseMove()`:

```
public Move chooseMove(Color[][] board, Color myColour)
```

The first argument is the current state of the board (each cell contains Color.black, Color.white or null), the second argument identifies which colour you are playing (Color.black or Color.white), and the return value is the move that your algorithm selects. The `Move` object has two fields, `row` and `col`, corresponding to a move at `board[row][col]`, so both fields should be between 0 and 7. Use the constructor `public Move(int row, int col)` to set the fields. The name(s) of any class(es) you define must end with your student number (e.g. `Player120394567`). I have supplied an example of a trivial player in `RandomPlayer.java`.

A Java class (`GomokuReferee.class`) is provided for developing and testing your player. This includes a graphical interface which allows you to play against your programme or watch games between computer players. It automatically finds any player classes which are in the same directory as you call `java GomokuReferee` from). `GomokuReferee` takes the following optional arguments:
`log` — echo all moves to standard output for debugging or analysis of your player; games are selected manually using the GUI.
`batchTest` — run a tournament where all players play against each other, without using the GUI; all moves and results of each game, plus a summary of the tournament, are printed to standard output.
`limit` ***value*** — set the time limit per move to ***value*** (default 10.0 seconds).
`delay` ***value*** — set the minimum time between displaying successive moves to ***value*** (default 1.0 seconds).

Marks will be allocated as follows (subject to the usual late penalties):

- 60%: based on your total score achieved in games against your classmates, plus some additional fixed opponents (written by me)
- 40%: code and report describing how your program works. The report should be MAXIMUM 1 page, PDF format ONLY. Strategy, algorithms, originality, code quality and ability to follow these instructions will be assessed.

Using the link on your landing page, submit your assignment as a single zip file containing:

- source code (Java source code, filename(s) ending in `<yourStudentNumber>.java`)
- report (PDF format, filename ending in `.pdf`)
- Do NOT put any directory structure in the zip file
- Do NOT submit more than one player or more than one pdf file
- Do NOT submit a Word document

- Do NOT submit any files supplied by me (e.g. the referee class)
- Do NOT submit code written by anyone else

Part of the marking will be performed by a script, e.g.:

```
cp ../suppliedFiles/*.class .        # these are NOT from the zip file
unzip yourSubmission.zip
javac *.java
java GomokuReferee batchTest
evince *.pdf
```

Please make sure that your submission would not cause any errors with such a script. Note also that a human will be reading the code and cases of plagiarism will be reported to the Academic Registry.