ISP Mandatory Assignment 1 - Connect Four

Your Task

In groups of 2-3 you are going to make your own implementation of the game "Connect Four"[1] (rules described below). The game should be implemented preferably in Java and otherwise in C#. It should be possible for a human player to play against the computer, and moreover the computer should make "reasonable" moves, i.e. it should be able to win sometimes. Therefore, letting the computer take e.g. random moves does not count as a valid solution to this exercise.
The computer's response time on a board with 7 columns and 6 rows should be "reasonable"[2]. On average, response time should not be longer than 10 seconds although few moves up to 30 seconds would be acceptable.

What is provided

A GUI-class (FourConnectGUI) has been implemented in Java for you to use, so you can spend more time on the game logic. The GUI-class calls functions defined in the interface IGameLogic (see Figure 1) and your task is to implement this interface (or an similar interface in C# if you choose to program in C#). You do not have to change anything in the GUI class if you just implement the interface according to the comments above the methods in the interface. If you program in C# you have to make a GUI yourself. You

Figure 1: The interactions between user, FourConnectGUI and IGameLogic

are provided with 4 files, FourConnectGUI, IGameLogic, GameLogic and ShowGame. The first two files are mentioned above. The GameLogic is a very small implementation of the interface, so you are able to see an empty game board when you run the main-function. You should of course extend this class. The last file contains the main-method to view the game. Here the game logic and GUI are initialised and the window where you see the game is setup and shown.

[1] Though there is a lot of downloadable versions of the game on the Internet, you should do the effort of making it yourself

[2] You could e.g. compare the time the computer takes to make a move with how long you would be prepared to wait for it when playing the game for entertainment purpose

In the implementation it is assumed that player1 is blue associated with the value 1 and player2 is red, associated with 2. This means that the evaluation function should return 1 if blue won and 2 if red won.

Running the game

The main method accepts 4 parameters from the command-line. The first two parameters are the players of the game, where the first option will be the starting player. Players can either be humans or game logics, it is not allowed, however, to have 2 human players. Game logics are implementations of the IGameLogic interface, and are loaded dynamically. E.g. if you implement the IGameLogic interface with a java class named 'MyFirstLogic' you only need to compile the MyFirstLogic.java file and run the game in the following way:

```
java ShowGame human MyFirstLogic
```

this command will create a default 7x6 game board , where a human is player1 and your logic is player2. If you would like to have your game logic be player1 you only have to switch the order in which you pass the parameters. The 'human' argument is a special keyword which activates mouse clicks for a human player. If you would like to play two logics against each other you just have to replace the keyword 'human' with the IGameLogic implementation of your choice.

The game parameters also allow you to define the size of the game board, e.g. if I would like to play 'gameLogic1' against 'gameLogic2' on a 8x9 game board I would run the game in the following way:

```
java ShowGame gameLogic1 gameLogic2 8 9
```

The rules

The board consist of a rectangle of x columns and y rows. Two players, Red and Blue, play against each other, taking turns after each other. In a turn the player takes a token in his own colour and chooses a column to put the token in. The token then falls from the top and down on the tokens already present in that column. When all rows in a column are full, none of the players are allowed to choose that column.
A player wins when he puts a token that causes him to have 4 (or more) of his own tokens in a line, either horizontally, vertically or diagonally. (see Figure 2).

There is a tie between the players when all places are filled up but none of the players has accomplished to get 4-in-a-row.

Hints on solving the exercise

The state space is too large to just make an implementation of the minimax-algorithm or the alpha-beta-search; if these algorithms are the only ones used, the running time gets too high for a game board bigger than approx. 4x4. Therefore an evaluation function is necessary. That is, you need to decide the value of a given board heuristically.

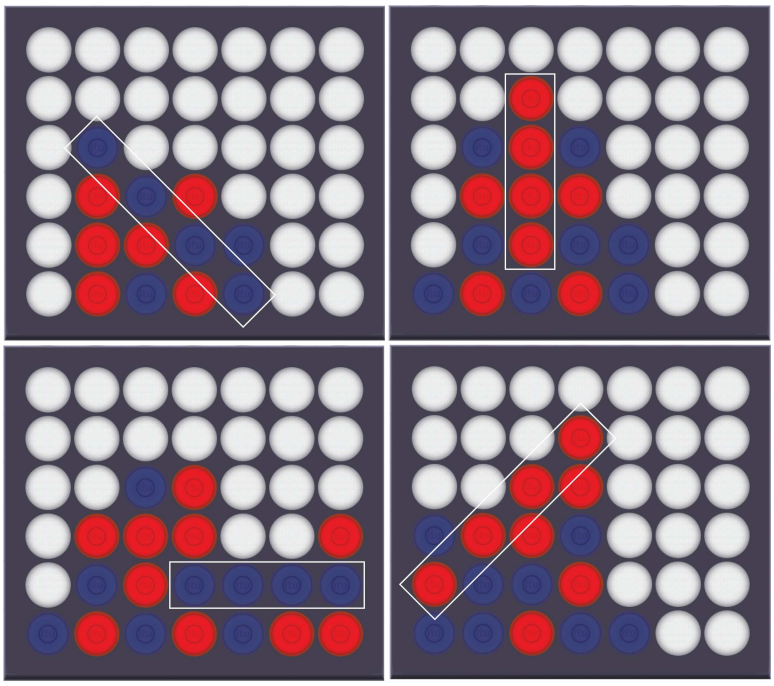

Figure 2: Winning situations

Practical advise

Experiment with the cut-off-depth of the algorithm together with your evaluation function to examine the possible trade-off between longer calculation time and "smarter" moves.
Develop your code incrementally: Be sure that the coins fall where you expect them to (just let the computer pick, say, the first non-full column) before proceeding. Then start with the minimax algorithm and be sure this (including the successor function) works on a small game board. Then you can proceed with the alpha-beta-search and later the evaluation function. Before applying the evaluation function in your main search, check that your evaluation function returns the expected utilities.
Finally, it is not recommended to write all the code in a single class.

Competition

After all groups have handed in their implementation of the game, we are going to have a tournament to find the best implementation. You can only participate if you have made your program in Java.
Two groups[3] will in turn play against each other twice and the winner will be the one who wins twice. In case of a tie, the winner will be the one who used the less moves to win. Then the winners of the first round goes on to meet each other (in pairs) and this continues until an overall winner has been found.

[3]In the first round, 3 groups might have to play against each other. In that case, all three groups will play against each other, i.e. there will be 3 x 2 matches

Each group should name their IGameLogic implementation after the name of the group, so that no two IGameLogics are called the same.

Hand-in

You should hand in the following:

- Compilable and runnable source code with useful/clarifying comments in Java or C#.
  - Running time of computers move in average less than 10 seconds.
- A report on 2-3 pages explaining your solution.
  - Explain evaluation and cut-off function.
  - Explain search algorithm.