UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS61B **M. Brudno**
Summer 2000

Project #3[*]

Due: Thursday, August 3, 2000

Your last project is the implementation of a game. Game playing is one of the oldest areas of research in computer science. Almost as soon as computers became programmable, chess programs were being written for them. This project is to implement a game called Connect 4.

Connect 4

Connect 4 is a simple game, some what resembling Tic-Tac-Toe. It is played by two players, on a 7 by 6 board, which is stood upright. The players alternate moves, and when it is a player's move he drops a piece of his color into one of the columns. The piece falls until it hits either the bottom of the board or a piece which was put in previously. The game is over when one of the players gets 4 consecutive pieces in any row, column or diagonal (hence the name). If the whole board is filled up without either player getting 4 in a row the game is declared a draw.

Your objective

In this project you are to implement the game. A lot of the code has been provided for you, including the whole graphical user interface (GUI). You do not need to understand exactly how the GUI works. If you look at the main method of the Connect4GUI class you will see how to use the relevant functions. The following are the classes given to you as part of this project and what you have to do with each.

connect4.java: You do not need to touch this class. It provides a method to parse parameters and create players. At the end it creates an instance of c4Game and runs it. This is the main executable.

c4Game.java: This class implements the loop of the game. Its job is to ask each player for their move and display it. It also must determine when the game is over and quit gracefully. You have to write the constructor and the run method.

Player.java: This is an abstract class that you should read but not touch. All players are to inherit from it. The Player class has an abstract method `int move(int i)` which gets the last move played by the opponent and returns the column in which to play the next move.

[*]The GUI was written by Kaolin Stockinger

HumanPlayer.java: represents a human player. Responsible for handling input by the user. You will fill in the play method which chooses a move according to the user's input.

ComputerPlayer.java: represents a computer player. You should implement the computer's strategy using at least the minimax and alpha-beta algorithms. The strategy could be a thread, and if you wish to write it as such we have provided a Timer object which will interrupt your strategy once the allowed time is up.

How good does your ComputerPlayer need to be? It mainly needs to correctly implement Minimax search with alphabeta pruning to a limited depth and use an appropriate evaluation function.

Board.java: Represents a Connect4 board. The ComputerPlayer should use the Board class to represent its internal state. You will implement a number of methods in Board including:

- `boolean lastmovewon(byte player)` determines as efficiently as possible whether the last move played by player won the game. One can then determine whether the game has been won by querying `lastmovewon` after every move.
- `void moved(byte player, int colnum)` and `void takeback(int colnum)` place a move in a particular column and take one back respectively in the Board state. These routines will not actually put the pieces in the graphical representation of the board. That is discussed in Section .
- `double eval(byte player)` implements a heuristic evaluation function from player's perspective. An evaluation function allows a player to approximate the *value* of a particular board position without doing a complete search. `eval` should return 1 if the player won with the previous move, 0 if he will lose next move and something in between in all other cases.
- `void display()` prints out some textual representation of the board using Xs and Os.

You should fill in the methods to fit the comments.

Graphical User Interface (GUI)

This is the GUI. You do not have to write any GUI. You do not need to touch this code. You should not touch this code, until everything else works. Then, just maybe, you can touch this code.

I've tried to make this code as *easy* to use as I possibly could. Let me give you an example:

```
public class Test {
  public static void main(String[] args) {
    Connect4GUI myGUI=new Connect4GUI();
    myGUI.paintBoard();
  }
}
```

That's all you need to do to get a 'classic-style' Connect4 board to pop up on the screen. If you want to change the colors or some other options, there are several methods available to you.

Colors	Strings	Dimensions
setBackgroundColor(Color)	setPLAYER1(String)	setBoardSize(Dimension)
setBaseColor(Color)	setPLAYER2(String)	setFrameSize(Dimension)
setBoardColor(Color)	setFrameTitle(String)	
setNameColor(Color)		
setPLAYER1Token(Color)		
setPLAYER2Token(Color)		

Now, if all these options are too much for you, but you don't quite like the colors that you start off with, I also added a method called `setTheme(int)` . Theme 0 is the original, 1 I call modern, 2 jungle, and 3 monochrome. So... if you wanted a 10x12 board instead of the normal 7x6 (don't know why you would, but I aimed to please), and you wanted my favorite color scheme, you'd simply...

```
public class Test {
  public static void main(String[] args) {
    Connect4GUI myGUI=new Connect4GUI();
    myGUI.setBoardSize(new Dimension(10,12));
    myGUI.setTheme(1);
    myGUI.paintBoard();
  }
}
```

So... the next question is... how do you drop a piece in? Well, the following should answer that!

```
public class Test {
  public static void main(String[] args) {
    Connect4GUI myGUI=new Connect4GUI();
    myGUI.paintBoard();
    myGUI.setPiece(Connect4GUI.PLAYER1,4);
    myGUI.setPiece(Connect4GUI.PLAYER2,4);
    myGUI.setPiece(Connect4GUI.PLAYER1,3);
    myGUI.setPiece(Connect4GUI.PLAYER2,4);
    myGUI.setPiece(Connect4GUI.PLAYER1,1);
  }
}
```

That's all it takes to drop 5 tokens on the board!

I suppose your last question is, "This is all great, but how do I get input from it for my `HumanPlayer` class?"

Well, all you need to do is implement the `java.awt.event.ActionListener` interface, and fill out the `actionPerformed(ActionEvent)` method.

```
public interface ActionListener extends EventListener {
  void actionPerformed(ActionEvent e);
}
```

The class `java.awt.event.ActionEvent` has a method `getActionCommand` which returns a String with the current command name in it. If the user clicks in column 2, that queues an ActionEvent whose command name is `"COL:2"`.

Deliverables

- The directory `~cs61b/hw/proj3`. Please do not change any of the class names, interface names, method names, or parameter lists to the public methods we specify above. This is to make it more straight-forward for us to test your code in an automated manner. Code which does not precisely match the specification will be heavily frowned upon.
- You also need to hand in a Makefile which compiles all code you submit.
- Include a README which details the general structure of the game and your strategy (including your search method and your evaluation function).
- To submit your result, use the command '`submit proj3`'. Each partnership should turn in exactly one project (from either partner). Make *sure* that both of your names and logins are on everything you turn in. You will turn in nothing on paper.

Extra Credit

There will be extra credit available (up to 5 pts) to those groups which enhance the basic design of this project. Some possible options are

- An enhanced GUI
- An interruptable, iterative deepening strategy that efficiently uses its opponents think time.
- A networkable version so that two players on different machines can play against each other.
- Anything else that strikes your fancy

You should not expect to receive extra credit for trivial modifications. If you implement some feature which you believe to be worthy of extra credit, you can submit it using `submit extra-credit` on the same deadline that the project is due. Note that extra credit points will be given totally at the whim of the instructor, no objective scale will be used.

Tournament

There will be a tournament of your ComputerPlayer implementation for those who are interested with as yet undisclosed prizes. The tournament is completely for fun. It will not directly affect your grade. More details will be forthcoming.