TCSS 435 — Artificial Intelligence & Knowledge Acquisition
Autumn 2014 — Homework Assignment 2

Due Date: Thursday, Oct. 23

Guidelines

Homework should be electronically submitted to the instructor by the end of the day on the due date. A submission link is provided on the course Canvas page for this assignment.

Assignment Description

In this assignment, you'll take on the role of strategy designer for a computer player in the game of Connect Four. You will *not* need to implement the basic adversarial search algorithms of minimax and alpha-beta search. Instead, you will design and implement the utility estimate function, which encapsulates game strategy, used by these algorithms.

Implementation Specifications

The Connect Four application source code is available on the course Canvas page for this assignment. To create a new computer player to the game, you must create a new class that extends either `MinimaxPlayer` or `AlphaBetaPlayer` (extending `AlphaBetaPlayer` is recommended for performance). Your new class must provide the following three methods: a constructor with one integer parameter, the `evaluateUtility` method, and the `getName` method. Skeleton code is provided below. You are free to choose the name of your class and your player.

```
public class AGreatPlayer extends AlphaBetaPlayer {
	public AGreatPlayer(int maxDepth) {
		super(maxDepth);
	}

	public double evaluateUtility(ConnectFourBoard board) {
		return 0;
	}

	public String getName() {
		return "A Great Player";
	}
}
```

The constructor is passed the maximum depth of the search tree (as selected via the GUI), which should be directly passed to the parent class' constructor. Attempting to make your computer player more comparative by altering the search depth will not be rewarded.

The `getName` method should return the name of your player. This name is used to identify your player in the game setup dialog and during play.

The `evaluateUtility` method should return a utility "score" for the Connect Four board parameter. This method implements the strategy used by the your player. Since you will be graded on the competitiveness of your player (details below), your player's strategy will need to be more sophisticated than the strategy used by the included `MinimaxPlayer` or `AlphaBetaPlayer`. Take some time to consider what features of a board are good and bad for your player, then design code that can quantify those features. The state of the board can be fully inspected (see the `ConnectFourBoard` class). The `Disc` object that represents your player's pieces is available in the inherited field `myDisc`. The opponent's `Disc` is available via `myDisc.getOpponent()`. The range of values returned by the `evaluateUtility` function is not important, since the values are only used for comparison. Consequently, within whatever range you wish to use, high values indicate the board is favorable to your player while low values denote an unfavorable board.

Once your computer player is complete, you do not need to alter any other provided code to include your player in the Connect Four application. The application automatically searches for and includes all player classes. Unless your execution environment is somewhat unusual, the application will be able to incorporate your new player without any intervention from you. If for some reason your player isn't available within the application, contact the instructor for assistance (DYI enthusiasts might try tweaking the middle of the `ConnectFour.createAndShowGUI` method first).

Grading

The software development component of this assignment is relatively minor. Therefore, you will be graded primarily on the competitiveness of the computer player you design. Your player will be pitted against the `MinimaxPlayer`, `AlphaBetaPlayer`, and `MysteryPlayer` at search depths of 4 and 7. Only the Java bytecode for `MysteryPlayer` is provided, so you cannot inspect the utility function used by that player. In each game, your player will use the same search depth as the opponent. You will receive maximum points if your player can consistently defeat all three of the above opponents. Fewer points will be awarded for defeating two of the three, only one of the three, and finally none of the opponents. To fairly judge the capabilities of your player, matches with include several games in both the first-player and second-player positions.

Good luck.

Connect Four Tournament

After this assignment is complete, a grand tournament among all players who wish to participate will be held. To facilitate this activity, do not modify any provided class, such as `ConnectFourBoard`. Whatever helper methods your player utilizes should reside within your `Player` subclass.

Deliverables

The following items should be submitted to the instructor. Failure to correctly submit assignment files will result in a 10% grade penalty.

1) Source code for your Connect Four player class.
2) Any supporting files, e.g., source code for additional classes, used by your player class.

Do not submit unmodified provided files, e.g., `Minimax.java` and `ConnectFourBoard.java`. Do not include any extraneous files, such as Eclipse IDE files, bytecode files, or subversion files.