Connect Four

Connect four is a two-player game. The objective of the game is to connect four markers in sequence, in either a row, column or diagonal, while preventing your opponent from doing the same. The game is won when a player makes four markers in a sequence. The game is played by placing a marker in one of seven columns, and the marker is placed on the lowest row in that column. There are six rows in each column to make a play field of 7 x6. Players take turns placing markers.

Coordinator Commands

The following are the commands required to interface with the connect four coordinator. All commands will be sent to the engine via `System.in` in java, and all responses sent from the engine to the coordinator should be done via `System.out.println` or `System.out.print`. Note that a new line character needs to be at the end of each command, so using the first option is probably easiest. Each command is case sensitive.

Sent From the Coordinator

- `name` – request for the engines name. The reply should be the name of the engine. The name should be whatever name you want to give it, followed by your student number, that is, `enginename-cXXXXXXX`.
- `isready` – A command used to synchronise the engine with the coordinator. The reply should be `readyok`.
- `position startpos <moves>` – The current order of moves played from the start position. For example, `position startpos 0123456`. Only the numbers 0 6 will ever be used. This command will always be followed by the `isready` command. The only operation allowed to be performed by the engine between receiving this command is to update the internal board with the moves, no searching is allowed.
- `go ftime x stime y` – Tells the engine to start calculating. The x string will be the time remaining for the first player in milliseconds. The `y` string will be how long the second player has left. The reply should be `bestmove z v` where `z` is the column number, 0 6, that the engine wants to place its next marker in. If it is invalid it will be an automatic forfeit. `v` is the value of the evaluation function after that move. All searching is done here.
- `perft x` – Tells the engine to count how many nodes are in the entire search tree when expanded to depth `x` from the current position. The reply should be `perft x y` where `x` is the same value passed in, and `y` is the number of nodes in the search tree when it is expanded to depth `x`. This should be a separate algorithm to your minimax algorithm, mostly used for debugging and performance testing (hence the name) purposes and be ready to take in any depth.
- `quit` – Tells the engine to quit. Reply should be `quitting`.

Sent From Your Engine

- `readyok` – The response to any `isready` command only when the engine is ready.
- `bestmove z v` – The response to the go command. `z` is the column the engine wants the next marker in, labeled 0 – 6. `v` is the value of the evaluation function after that move.
- `info` – If used, must be after a `go` command from the coordinator, and before the `bestmove` response from the engine. It can be followed by a number of optional strings. Each string must be paired with the corresponding data. Not a requirement to use `info`, it is mainly used for debugging purposes.
  - `depth x` – The depth searched by the engine so far.
  - `nodes x` – The number of nodes searched by the engine so far.
  - `score x` – The current best evaluation by the engine so far.
  - `String x` – Any string the engine wishes to display. If there is a string command, the rest of the line will be interpreted as the string corresponding to the string command.
- `quitting` – The response to the command `quit`. The engine should exit at this point.

Coordinator.jar

A file called `connect_4_coordinator.jar` has been supplied. It is created against java 1.8. It will be used to verify your engine is working correctly and implements everything mentioned in the coordinator commands section. To use the file, refer to your IDE's documentation to see how to run a jar file in it, and adjust the command line arguments for it. If you wish to use it on the command line to run it, I would advise using an ANSI complaint terminal, although this is not necessary. To run the file from the command line you need to use `java -jar connect_4_coordinator.jar` along with the arguments to pass to the program. The arguments are:

- (required) `e0 <path>` where `<path>` is the directory containing the file called Interface which includes the main method of your engine.
- (optional) `e1 <path>` where `<path>` is the directory containing a file called Interface which includes the main method of another engine. If this is not used the first engine will be used again, so it is easy to set up a self game for the engine.
- (optional) `time x` where `x` is the amount of time in milliseconds each engine is allowed to use. If this is not used, `x` will be 120000 (2 minutes).
- (optional) `perft x` where `x` is the depth the engine should calculate the `perft` function to.