CS3 Project 2: Connect Four (Spring 2023)

Project description

In this project, we'll write a program for the game Connect Four, and have a tournament between our automated players. For information on the game, see http://en.wikipedia.org/wiki/Connect_Four.

Getting started

The project code will be organized into one top-level folder (which we open in vscode), with three subfolders containing Java source files for the corresponding packages. Two of the subfolders, connectFour and sample, should be copied from the starter code, here: https://drive.google.com/drive/folders/1EOWhSbkHcShGsF7VUMEeX-jtkMMU2Goe?usp=sharing. The third subfolder should be your first name in lower case, which is where you'll put your own Java files.

Required elements

1. Make a Board class that implements the Grid interface. It must use private field(s) for the state of the game, including an int[][]. It must have at least one public constructor, and at least one public method that updates the state (so your program can make moves). It must override the toString method to return a human-friendly String representation of the current position.
2. Make a Game class that includes a main method that prompts the user to choose the players, plays the game while printing the moves and positions, checking whether the game is over after each move, and eventually printing the result. If any player makes an illegal move (column full or out of range), that player loses. If the current player just got four in a row (horizontally, vertically, or diagonally), that player wins. If the entire grid is full without any four in a row, the game is a draw.
3. Make a HumanPlayer class that implements the Player interface. It must have a public constructor that takes a parameter of type Scanner and prompts for the user's name (which it should return from getPlayerName). For each move, it should prompt the user to enter the column number (0-6). It should not allow the user to enter an illegal move.
4. Make an AutomatedPlayer class that implements the Player interface. It must only choose legal moves for legal positions (column 0-6 only if not full), must move reasonably quickly (one second or less per move on a standard laptop), and for full credit it must play significantly better than the SampleRandomPlayer. It must have a public constructor that takes no parameters, and the string returned by the getPlayerName method must include your name. This is the player that will represent you in the tournament.

Collaboration

You may help each other with design, testing, and debugging, however you should not share code. Also, the tournament will be more interesting with different approaches to the automated players.

Suggestions and optional elements

Add a second constructor to your Board class that takes a Grid as a parameter, so your automated player can convert an arbitrary Grid into a temporary instance of your own Board class. Then add some public "helper" methods to your Board class, such as int numberOfMovesPlayed(), int playerNumberToMove(), int firstEmptyRowInColumn(int column), boolean playerHasFourInARow(int playerNumber), boolean isGameOver(), void playInColumn(int column, int playerNumber). When checking for four in a row, carefully consider each of the four possible directions: horizontal, vertical, diagonal up, diagonal down.

Make a TestPlayer class that takes an int[] as a constructor parameter and makes those moves in sequence without even looking at the grid, so you can test your Board and Game classes by playing TestPlayer instances against each other that you construct to check various four-in-a-row situations as well as illegal moves and draw. You can of course do the same kind of testing with HumanPlayer, but it's nice to have tests that you can keep running automatically until they all pass.

What to turn in (as a link to a Google drive folder with the following files)

Turn in only the Java source files in your own package subfolder, plus your reflection document.

- Board.java
- Game.java
- HumanPlayer.java
- AutomatedPlayer.java
- Reflection: One paragraph about your experience on the project, and one paragraph about your approach to making your automated player play well.

Grading rubric (25 points total)

- Readable code following Java conventions, use of int[][], thoughtful reflection: 5 points
- Game class, picking players, playing game, reporting moves and result: 5 points
- Board class, managing state, detecting wins and draws, toString: 5 points
- HumanPlayer class, managing name, prompting until move is legal, returning move: 5 points
- AutomatedPlayer class, getPlayerName, legal moves, plays quickly, plays well: 5 points

Example start and end of an interactive game:

```
Please choose player 1 (random, human, automated): automated
Please choose player 2 (random, human, automated): human
Please enter your name: Arthur
Starting game between player 1 (Arthur's Auto) and player 2 (Arthur)
… skipping to the last move …
It is player 2's move
Arthur, which column (0-6)? 0
Move 16: player 2 moved in column 0
Current position:
5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
3 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
2 | 0 | 1 | 2 | 0 | 0 | 0 | 0 |
1 | 2 | 2 | 2 | 2 | 1 | 0 | 1 |
0 | 2 | 1 | 1 | 2 | 1 | 1 | 1 |
   --- --- --- --- --- --- ---
    0   1   2   3   4   5   6
Player 2 (Arthur) wins!
Result: 2
```