Project 1—Connect Four

Due: 7/24/2015 (Sunday)

Playing 'Four' is like playing Chess, but harder & more strategic" - James Brotsos

Overview

You will be writing a simulated Connect Four aka "Four" game that will enable you to 1) play versus another human, 2) play against a dumb computer, 3) play against a smart computer and 4) watch the computer play against itself.

Specifications

You should have a main menu that allows a user to choose 5 different options:

1. 2 player game
2. 1 player game vs dumb computer
3. 1 player game vs smart computer
4. Simulated game with computer playing itself
5. Quit

A connect four board has 6 rows and 7 columns. You must determine if a user is a winner by checking to see if there are four tokens in a row going up & down, left to right and both diagonal ways. Once a winner is determined or a tie happens, an announcement should be shown to the console and the main menu should be shown again.

After each play, the Connect 4 grid (a two-dimensional array) should be printed with an updated game play.

[human vs human]

The program will read in input from the console that must be some form of coordinates in a Connect Four grid. You can decide how that input format should be. If the coordinate is invalid (either the slot is taken, the slot doesn't exist on the board, or the slot isn't at the bottom of the row), the user should skip a turn or be prompted to place a token in a valid location.

[human vs dumb computer]

The dumb computer goes second and will place a random token in a valid spot. It is impossible for a dumb computer to put a token in an invalid spot.

[human vs smart computer]

The smart computer goes second and will try to either beat you or not let you win. It doesn't put tokens into random spots but instead will check to see the best spot on the board to place a token.

[computer vs computer]

Should be smart vs. dumb and smart should win.

Getting Started

You only need one file for this program and do not need to use objects if you don't want to (but you can).

You need to first create a menu that allows users to select an option until they quit. You will need to read in input from the console to choose which version of the game they will play. Once you start playing, you will also need to read input from the console that contains the coordinates. This will involve splitting the input String and parsing it into different variables to be able to determine the integer coordinates.

It is good practice to start off by drawing a conditional "map" first to see the structure of the program. This also allows other people to see what your code should be doing. Start simple and continue to build on it. Get the menu to work, then get the grid to print a blank board and then get tokens to display.

Your program should be checking if it's a valid choice, if there is a winner, and if the board is full after each play. Start with left to right and then up & down. The diagonal is the most complicated algorithm so do that one last.

Grading

Requirement	Points
Main menu that loops until Quit	10
Handle invalid commands	10
Determine winner 4 left to right	10
Determine winner 4 up & down	10
Determine winner 4 diagonally	10
Determine illegal plays	10
Determine if its a tie	10
Human vs Dumb Computer	10
Human vs Smart Computer	5
Computer vs computer	5
Comments for every method	5
Work on it solo	5
Total	**100**

Checkpoints

Working on a software project is easier if you have intermediate goals to measure your progress against. If you can show a certain amount of progress by each checkpoint. If you are ahead of the schedule, even better!

- Checkpoint 1—**7/22**: main loop that has menu and asks for input and will print the blank grid to start
- Checkpoint 2—**7/24**: able to determine if input is valid location on grid
- Checkpoint 3—**7/26**: able to determine if user won going up & down, and left to right.

Main Menu

The main menu must allow a user to input a choice between 1 and 5 on what game they want to play. The main menu will appear after every game until a user selects 5 to Quit. A call to a method that prints and reads the Main menu should be called from a while loop in your program

Handle Invalid Commands

When a user inputs their coordinates for the Connect Four board, they need to input it on the console. An example would be:

[3,4]

You then need to parse that line so you can add the token to the appropriate place. However, a user might input:

[x,y]

Or any junk. You need to be able to handle that. It involves checking to see if the String read is a valid integer before converting it to an integer. The goal is that a user could put any junk into their coordinates and it doesn't crash the program.

Determine Winner

A winner is determined by having 4 in a row across left to right, 4 in a row up and down, 4 going up and to the right diagonally and 4 going up and to the left diagonally. When verifying, you must make sure you stop at the boundaries of the grid.

Determine Illegal Plays

A legal play is when the user selects a spot on the Connect Four grid that is legal. It has to be at the bottom of the row or on top of another token. It also has to be a valid coordinate on the grid (not out of bounds).

Comments

To get credit for comments, every method should have a comment header. It must have this format:

```
/*
 * <method name>- <short description of method>
 * @param: <list of parameter types, names and their purpose>
 * @return: <return type>
 *  <more detailed description of the method>
 */
```

And must be located directly before the method declaration. **If you are missing any headers, you will get 0 points for comments. Your comments prove that you did your work and that you know what is happening in the method.**

Work on it solo

To receive credit for the working on it solo, you cannot work on it with anyone, including a parent, relative or tutor. You can ask anyone for "advice", I encourage that but you cannot copy and paste code from **ANY** resource. If you copy code from the Internet or a classmate, you can potentially get a 0 for the program.

Extra Features

- Error handling, cover all errors possible using try & catch
- Store results in a file and add option to in the main menu to print past winners. Add a menu option to allow users to see past winners.
- Try to make this in GUI using JPanel Class.