COMPSCI4039 - Programming IT
AE2: Connect Four

There is a FAQ at the end of this document, please read it.

Introduction

Connect Four is a popular two-player connection game in which the players take turns dropping coloured discs from the top into a vertically suspended grid. The objective is to be the first to form a horizontal, vertical, or diagonal line of four of one's own discs. Full rules can be found here.

In this assessment, you are being asked to program a simplified version of this game. The game will be played on a 6x7 grid, and players will take turns to drop their symbols (X or O) into the columns. The first player to connect four of their symbols in a row (horizontally, vertically, or diagonally) wins the game. Note that while the classic game uses coloured discs, this version will use X and O as the symbols for the two players.

You will create the classes and logic necessary to run a game of Connect Four.

This assessment is split into multiple sections. Partial marks will be awarded, so you do not need to complete the project in full to be awarded marks. The maximum number of marks that can be awarded for each section is shown to help you manage your time with this assessment.

Marking Scheme

Marks will be awarded for:

- Functional correctness
- Sound OO design
- Clean, tidy, and commented code

The marks are awarded out of 25. Remember that you do not need to complete the project to be awarded marks. Partial marks will be awarded for demonstrating an attempt to address the task.

Part 1: The Player and Disc Classes

Each player will have a name and a symbol (X or O). The Disc class should represent a disc dropped by a player and should include:

- The symbol of the disc (X or O).
- The position of the disc in the grid.

Each player class should have a reference to the grid they will play the game with, their name, and their symbol. Make these attributes private. The constructor for the Player class should take in the name of the player and the symbol.

Each player will have a method to take turns. This method will be called from the main controlling logic of the game.

[5 marks]

Part 2: The Grid Class

You are to specify a Grid class. This class should include:

- A 2D array to represent the grid.
- Methods to drop a disc into a column.
- Methods to check for a win condition (four symbols in a row).

The grid constructor should initialize a 6x7 grid. The Grid class should have methods to:

- Drop a disc into a specified column.
- Check if the grid is full.
- Check for a win condition after each move.

[5 marks]

Part 3: Playing the Game

When creating Player objects, the users should be prompted for their names and symbols (X or O). Players have a takeTurn method, which should be called from a main game loop within the main class. This method should prompt the user to input the column number where they want to drop their disc.

Within the main class, add logic to allow players to take turns by implementing the takeTurn method. The game should announce an update to the player after each turn, indicating whether the move was valid, and if the game has been won or is still ongoing.

[5 marks]

Part 4: Visualizing the Game

The Grid class should have a method, toString, that will allow the current state of the grid to be displayed to the console. The minimum information necessary is to show the grid with the discs in their current positions.

The toString method should invoke the toString representation of each cell in the grid. Each row should be printed with String.format to ensure proper alignment.

[5 marks]

Part 5: Enhancements

Expand on what you have implemented by adding additional features such as:

- A method to undo the last move.
- A method to save and load the game state.

These enhancements are optional and will be awarded extra marks if implemented correctly. There is no need to do both of these suggestions: either one, or something else, would be enough to get the extra marks. You should put in an extended comment in your code to explain what your enhancement is.

[5 marks]

FAQ

1. **Is it important to make flashy graphics for my board?**
   No. Functionality is what we're after here.
2. **Should I comment my code?**
   Yes. Your code is the only thing you will submit. We reserve the right to penalize because we cannot understand, even if the code works. Provide clear comments.
3. **Should I provide unit tests?**
   You can get 100% of the marks without them. However, writing tests will help you to get things working quicker. If you do write them, feel free to submit them. However, you are not expected to write unit tests.
4. **If I can't get something to work, what should I do?**
   We can sometimes award marks for you telling us (in comments) how you might get something to work so if something isn't working, tell us why you think that is.
5. **How long should I spend on this assessed exercise?**
   Hard to say. However, we find every year that students spend way too long on AEs.
6. **Will you post an example solution?**
   Yes. Once we have all the submissions and have completed the marking. This might be a while after the deadline (people get extensions for being ill etc.) so please be patient! We will devote an examples class to going over our model solution.