159.272 Programming Paradigms

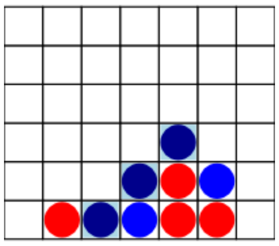
Assignment 2

Playing a game: Four in a Line

15% Weighting
Due 10 May 2020

Playing games interactively is a common computational activity. We can use abstraction to aide in the design of games by separating the game logic from the actual user interaction required to play the game. No matter the type of user interface provided to play the game (text-based, desktop environment, web browser), the underlying logic remains the same.

In this assignment, you will complete Java implementation of Four in a Line (also known as Connect Four, Captain's Mistress, Four Up, Plot Four, Find Four, Fourplay, Four in a Row), a two-player connection game in which the players first choose a color and then take turns dropping colored discs into a seven-column, six-row, vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The object of the game is to connect four of one's own discs of the same color next to each other vertically, horizontally, or diagonally before your opponent. Four in a Line is a strongly solved game. The first player can always win by playing the right moves.



See Maths is Fun to practice the game, or read the wikiHow entry for tips on how to win. The Java code for this assignment consists of three Java files (assignment2_source.zip):

- **`Game.java` (this is the entry point to the program (i.e., main method))**
- **`FourinLine.java`**
- **`GameTree.java`**
- **`Piece.java`**
- **`Player.java`**

The game logic is contained in the module **FourinLine.java**. Please download all three files and include them in your Eclipse project.

Instructions:

1. Download the zip project from Stream
2. Import the project into your IDE
3. Complete the functions in FourInLine.java. Replace the methods with “ throw new RuntimeException("Missing implementation”.

4. **Game.java** defines the main class of the application with the main method. Run and test your changes in the main method defined in Game.java. The final function call in the main method starts the game. Right click and run the main class as a Java application as you did at the tutorials.
5. There are a test class under /tests/ and text files in /testdata/ to help you test your application. A full complete application should result in all test being passed. Keep running the tests as you develop your solution. Note that tests are not exhaustive, and you should thoroughly test/play the implemented game.

There is a README file within the project folder with instructions on how to run the program in different environments (in case you prefer to run your application from Terminal/Command Line

Task

Your main task is to define the functions in this module using the provided data types.

The other two modules have been provided for you. **GameTree.java** constructs a game tree and uses the minimax strategy with limited lookahead to determine the best move for the computer to make. The **main** method is in **Game.java** module, so to run the program you need to run your **Game.java** object (*right-click -> run as Java application*). This will start a new game and will show you the following message on the screen:

```
Welcome to four-in-line
Is Red player to be human or computer?
```

GameTree.java constructs a game tree and uses the minimax strategy with limited lookahead to determine the best move for the computer to make. You should work through the code and *make an effort* to understand how these concepts have been implemented.

Your second task considers the estimate method in **GameTree.java** which takes a Player object and a **GameState** object and returns an integer value, which "rates" the result of the current game configuration, for that player

Your task is to "make this function smarter". The vanilla version returns a score for the given player and the given configuration, based on whether or not the player has won the game. Can you make this function smarter? This function can be modified to account for strategic concerns, e.g. placing pieces in the centre columns gives you more options, clumping your pieces together can provide more ways to make four in a line. The important point is that you can modify this function without changing *any* of the other code, and end up with a smarter program.
Think about what you would need to do to alter this program behavior using the concepts that you have learnt in this course.

What to submit

When you have a finished version, submit your Eclipse project (Export as an Eclipse Archive file) for marking. Make sure that you export the entire project and not only the java classes.

Marking

- up to 12 marks for a correct working version of **FourinLine.java**, modulated by the use of good programming style, and
- up to 3 marks for an updated version of **GameTree.java**, modulated by the sophistication of your improvements.

Submit your work on Stream by **9 June 2020**

There is a penalty for late submissions. The penalty is 10% deducted from the total possible mark for every day delay in submission (one day late – 90%, two days – 80% etc).

You are expected to manage your source code, this includes making frequent backups. “The cat ate my source code” is not a valid excuse for late or missing submissions. Consider managing your code in a *private* repository (bitbucket or similar). Private is essential here to avoid plagiarism, we reserve the right to deduct marks from your submission if your repository was public.