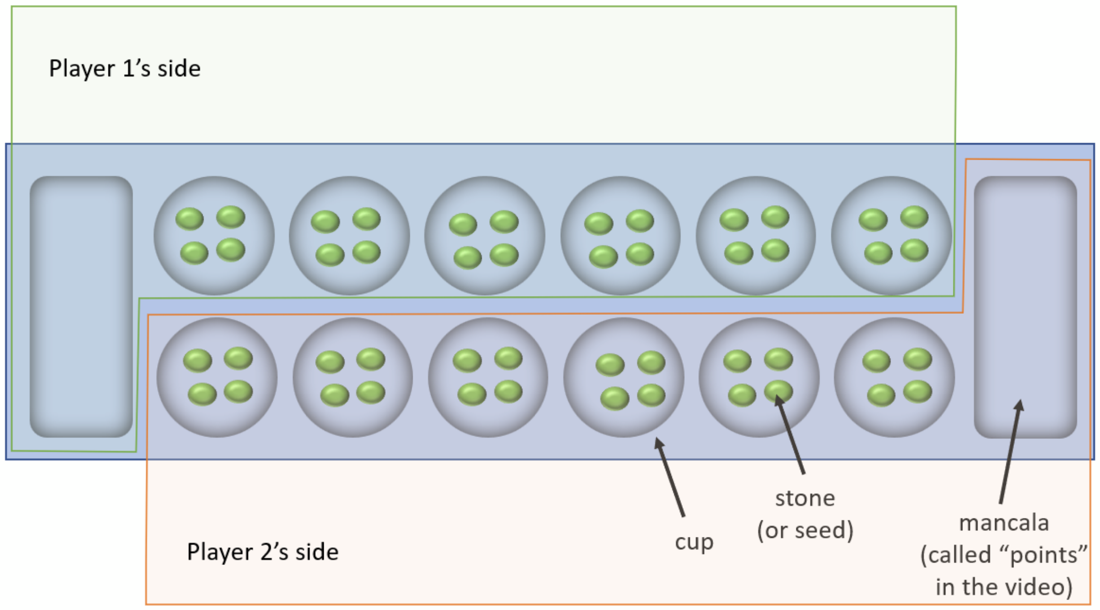
Programming Assignment 3
Mancala

CSC460 Introduction to Artificial Intelligence & Machine Learning
Endicott College • Dr. Hank Feild
Spring 2022

Name(s):	Luke Jodice

Overview

In this programming assignment, you will generalize and extend the Minimax code we worked on in class (available here) and implement a new game that uses it: Mancala. Mancala is a family of games and there are *many* variations. The particular variation you are responsible for in this programming assignment is the one described in this video, with one very important exception: you do not need to model the case of additional turns; that is, if either player dispenses their last stone in their mancala, they do not get another turn as is specified in the video (that's something we can't handle in our current Minimax implementation). For reference, here is a Mancala board with the parts labeled.



The game must support two modes of operation for the AI: (1) Minimax with α-β pruning and (2) depth-limited Minimax with αβ-pruning down to depth *d*, as specified by the user. For option (2), you will have to come up with a reasonable evaluation function. You may obtain the user's choice between the options via command line arguments or through an interaction with the user once the program starts but before game play starts.

To represent the board between turns, I suggest using a string formatter to produce something like this where all of the cups are numbered for easy reference and represented in a way that looks kind of like a cup on a real board; they can handle up to two-digit numbers representing the number of stones in the cup:

```
              Computer's side

       1     2     3     4     5     6
       |     |     |     |     |     |
     ( 3)  ( 5)  ( 0)  (10)  ( 2)  ( 2)
(13)                                    ( 9)
     ( 2)  ( 1)  ( 2)  ( 0)  ( 1)  ( 0)
       |     |     |     |     |     |
       6     5     4     3     2     1

                  Your side
```

You may work with up to one other person on this programming assignment.

Learning Outcomes

By the end of this assignment, you should be able to:

- generalize the Minimax class to work with multiple games
- implement depth-limited Minimax with α-β pruning
- implement a user-vs-AI game with multiple modes of operation for the AI
- implement an accurate simulation of Mancala
- implement a reasonable evaluation function for Mancala

Specifications

Before you submit, go through and check off all of the specs your submission meets.

Your submission should meet each of the following criteria. It must pass all specifications in order to be contribute to your targeted grade bundle at the end of the semester:

- ☑ ~~program compiles and/or runs without error~~
- ☑ ~~the source code is well organized and commented~~
- ☑ ~~each source code file includes a header indicating who wrote or contributed to the code~~

- [x] ~~code from any external sources, or code that relied on outside influence, includes a citation in a comment just above the corresponding code~~
- [x] ~~the code is built off of the codebase we built in class~~
- [x] ~~there is an implemented, working Mancala (game) class~~
    - [x] ~~it implements at least one appropriate evaluation function~~
    - [x] ~~it includes a command line interface that allows the user to specify the depth for depth-limited Minimax (0 is used to indicate that full Minimax should be used)~~
    - [x] ~~the game starts with four stones in each of the cups on the board (not in the mancalas at the end)—there's a total of 48 stones~~
    - [x] ~~the game ends when all the cups are empty for on the side of the board corresponding to the player whose turn it is, at the start of their turn~~
    - [x] ~~the winner is determined by which player has more stones in their mancala, or a draw if both have 24~~
- [x] ~~there is an implemented, working TicTacToe (game) class~~
- [x] ~~there is an implemented, working Minimax class~~
    - [x] ~~it works with any game (that is, it isn't tied to a single game)~~
    - [x] ~~it implements α-β pruning~~
    - [x] ~~it supports depth-limited Minimax~~
- [x] ~~the submission includes a `README.md` file~~
    - [x] ~~it describes how to compile (if Java) and run your program~~
    - [x] ~~it describes what command line arguments, if any, are available~~
    - [x] ~~it includes a section that describes the evaluation function you implemented and why you think it is appropriate and effective~~
- [x] ~~all output produced by the program is nicely formatted~~
- [x] ~~the board is displayed before each of the user's turn~~
- [x] ~~the board makes clear how many stones are in each cup and mancala, and which side corresponds to the computer and human~~
- [x] ~~the board includes a key to indicate what the user must enter to specify a cup on their side of the board (numbers or letters)~~
- [x] ~~the AI's move is indicated with some sort of print out~~
- [x] ~~user moves are validated; the user should continue to be (re)prompted until they specify one of the six cups on their side of the board that has at least one stone~~

Grading

As stated in the syllabus:

> *Programming assignments; started in class and completed at home. These can be done individually or in pairs.*
>
> - *Pass: the program compiles, runs mostly as expected, the output is formatted well, the source code includes a header and in-line comments, the code is well organized, and all assignment-specific criteria are met*

As also outlined in the syllabus, if your submission does not pass, you will have an opportunity to make corrections and resubmit if you choose to use a redo token.

Submissions

Make a copy of this document and fill out your and your partner's name (if applicable). Check off each of the specifications that your submission meets. Download a version of this as a PDF and include it in the root directory of your project (next to the README). Compress your project directory (zip or tar-gz) and submit it on Canvas.