Object Orientation – Spring 2018 **Assignment 4**

Learning Objectives

After completing this assignment, you should be able to:

- define and use enumeration types;
- define interfcaes;
- implement interfaces in different ways;
- use interfaces as object references;
- understand the Java memory model;
- copy objects.

Two player board game

There are many two-player board games. One of the simplest and best known is **Tic-Tac-Toe**. There are many variants, see e.g. `https://en.wikipedia.org/wiki/Tic-tac-toe_variants`. Some of the more popular variants are **Connect Four** `https://en.wikipedia.org/wiki/Connect_Four` and **Treblecross** `https://en.wikipedia.org/wiki/Treblecross`.

You can implement a game of your choice in this assignment. Verify with the aid of the assistants if your game is suited if it is different from the three games mentioned above. In the rest of the assignment we assume that you implement Connect Four.

1 Program

The game must be implemented as an Object Oriented Program in Java. We discuss some aspects that are required to meet the learning objectives.

1.1 Board

First there is a board representing the state of the game. Basically this is a two dimensional array of fields. Add methods by need to simplify the implementation of the game. Typically candidates are:

play in a given column;

winning is the given position part of a winning situation, e.g., four on a row horizontal, diagonal or vertical;

copy make a copy of the current board for analysis of potential moves;

toString for printing the current board in ascii-art.

1.2 Fields

The fields in the board are either empty, or one of the two colors in the game. The traditional colors in connect four are red and yellow. Extend this enumeration type with a custom `toString` method yielding a ascii representation of the field (e.g., `.`, `X`, and `O`). Define a method `other` that yields the color of the other player. For the empty field this method is the identity function.

1.3 Player

Define an interface that is common for all players in the game. Apart from a method `play` this interface defines at least getters for the name and color of the player. Add methods by need.

1.4 Human Player

The human player will display the current state of the board in ascii to the user and reads the column where the user wants to play. Since this input and output of this kind of player is hard to predict from the structure of the program (some player happens to be a human), it is difficult to achieve a strict separation of model and IO here.

1.5 Computer Player

The computer player tries to generate a smart move algorithmically. Most of the two-player board games are solved, i.e. there is a winning strategy known. It is **not** required to implement this strategy for the game you have chosen. Some approximation is sufficient. Possible strategies are:

Rule-Based Strategy

A rule based strategy determines the next move on a set of rules. For example:

1. If I have a winning move, take it.
2. If the opponent has a winning move, block it.
3. If the opponent as a move that yields a fork (two winning moves), block it.
4. Prevent a own move that yields the opponent a winning move.
5. Make an allowed move preferably around the center of the board.

Such rules can be extended to become a winning strategy, but that is not required here.

MinMax search algorithm

From introduction AI you probably know the MinMax search algorithm that evaluates the next N moves. In principle this generates a tree of boards, a smarter implementation only needs the path to the current point in this tree. The trees for many games are too big to be evaluated completely, hence we add the constraint of N moves and make some approximation for the value of the leaves of the trees. Assign values to the board based on the possibilities to win or loose the game given the current value of the fields.

1.6 Game

The game has a board and an array of two players that get alternating turns to play. The game stops as soon as there is a winner, or the board is completely filled.

During the development the game can be played with two human players, but the final game requires one human and one computer player.

When you have spare time you can add features like choosing who has the first turn, reading the name of the human player, playing another game, quitting a game, getting a hint, and undo the last move.

Handing in

Hand in your Java files on Blackboard **before Sunday March 4, 23:59**.