Exercise 5:
Searching Game Trees (Part II)
Due Date: 24.08.06, 17:00

The purpose of this exercise is to refactor the classes used to create the *connect four* game (exercise 4) so that they will make implementation of any future similar game easy. The family of future games that need to be supported is two-player, turn-based games that can be played using search trees.

Specifically:

1. Refactor the classes so that a new two-player, turn-based game involving search trees can be supported easily (i.e. reuses code). Note that there is no need to support *now* all such games, just give the framework to easily create new implementations.
    a. Create a diagram of the classes‘ relations as the ones shown in class (note that there is no need to be strict about the format/symbols of the diagram)
    b. Explain in a few bullets the changes you have made
2. Refactor the connect-four code so it fits the new framework
3. Create an implementation of Tic-Tac-Toe

The aim is to use as much modularity and code reuse, while maintaining simplicity and a code structure that is easy to understand and maintain. Remember that the design should support easily creating many game implementations in the future, not just connect four and tic-tac-toe.

<ins>What to submit:</ins>

1. The class relationship diagram after refactoring (hardcopy)
2. Short explanation of the changes made (hardcopy)
3. Refactored code (.java files) of both games (connect four and tic-tac-toe) as a .zip file and hard copy