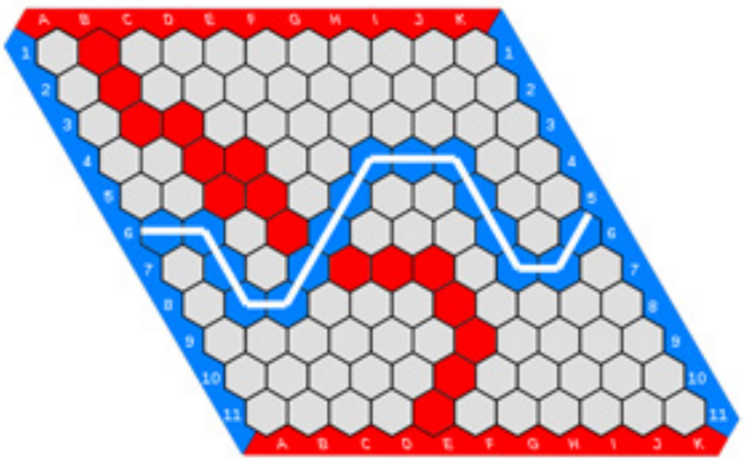
Introduction to Programming

End of year Exercise 2016

The Game HEX

Summary
Hand-out date: 14th March 2016
Hand-in date: 3rd May 2016 by 2pm
Hand-in method: Submission through DUO

Overview
Hex is a strategy board game played on a hexagonal grid (see example board below). Two players, red and blue, take it in turns to place one of their pieces on the board until one of the players has constructed a path of their own pieces between their two sides. In the example board below blue has constructed a path between their left and right sides of the board and has won the game.



A few comments on the game of Hex:

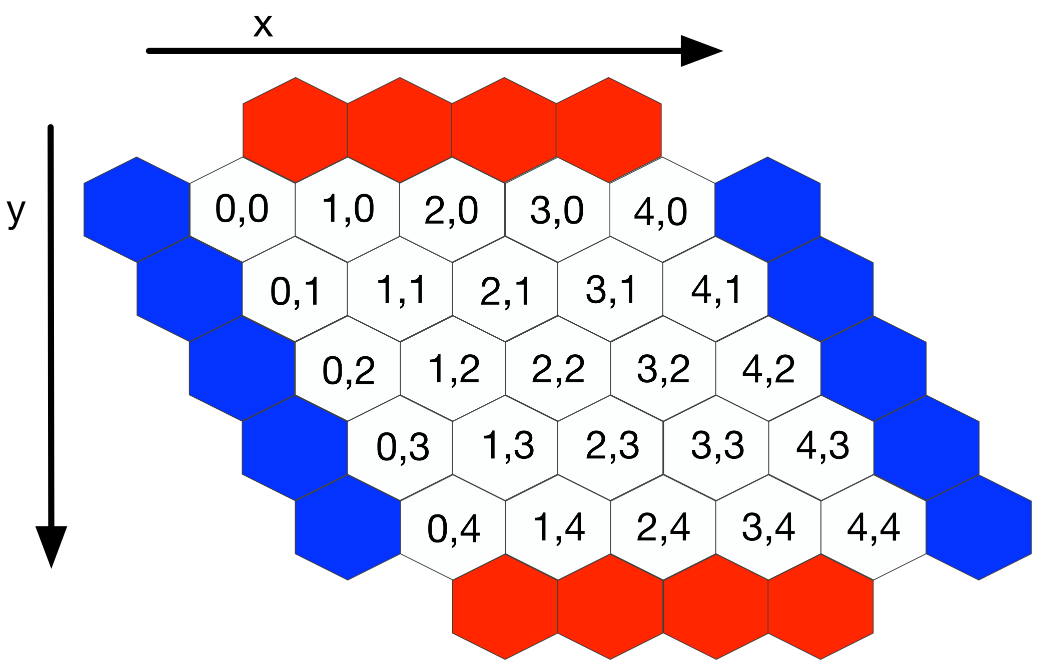
- Independently invented by Piet Hein (1942) and John Nash (1947)
- The game can only end in one of the two players winning
- The game can be played with any size board (n by m) though for fairness the game is normally played with a board of n by n cells.
- Common game sizes are 8 by 8, 9 by 9, 10 by 10 and 11 by 11.
- There is no know winning strategy, though it has been proven one exists

Aim of the assignment
The aim of the assignment is for you to write your own version of the game Hex. This will allow humans to play against humans, humans to play against a computer and computers to play against computers.
A set of interfaces will be provided which you will need to develop your code against.
The interfaces can be downloaded from DUO in the zip file HexGame.zip.

Board Layout

As there are a number of ways in which the board can be laid out. For consistency here is a specification of how the board is laid out for this work. (see figure below)

- Blue will try to make a line left to right
- Red will try to make a line top to bottom
- X values run left to right
- Y values run top to bottom



Developing your solution

In order to develop your game Hex you will need to write a number of class implementations. These can be broken down into the five parts described below. Parts 1 to 3 can be done independently of each other whilst parts 4 and 5 build on the first three parts.

Part 1: The board (20%)

The Board class is the core element of the game. The board is responsible for maintaining the current state of the game and indicating if the placement of a new piece is valid. The board is also responsible for determining if the game has been won (note that this is part 4 below and should be tackled once you have parts 1 to 3 working).

The assessment for this part will be as follows:

- The Board class compiles
- The Board class gives a valid response to each method that is called upon it
- You can create a new Board of any arbitrary size
- The Board class will prevent a piece being placed within a cell that is already occupied (by throwing a PositionAlreadyTakenException)
- The Board class will provide a correct 2-dimentional array of Piece (enumeration) representing the current state of the board

Requirements for submission:

- You must call your Board class Board.java and it must implement BoardInterface.java

Part 2: The Game (20%)

The GameManager class is responsible for managing a game as it is running. It has two methods the first of which allows you to set the Player class instance that will be red/blue and the second the size of the board (which is requires two integer values). The GameManager has only one other method playGame() which causes the GameManager to initiate and manage the game.

The game manager should request each Player object to make a move, by passing it the current state of the board. Take the Move the player makes and apply this to the board. After each move the GameManager should check to see if the game has been won or that the player has conceded. Once the game has been won/conceded the GameManager should inform each Player of the result.

The assessment for this part will be as follows:

- the GameManager compiles
- the GameManager gives valid responses to each method
- the GameManager follows the correct sequence of actions to allow a game to be played
- the GameManager prevents a Player from cheating
- the GameManager handles a Player conceding correctly
- the GameManager correctly informs Players of the game outcome

Requirements for submission:

- You must call your game manager class GameManager.java and it must implement GameManagerInterface.java

Part 3: The Human Player (20%)

This requires you to develop a HumanPlayer instance of the Player interface. The HumanPlayer will need to output the current state of the board (as passed to it by the GameManager) and ask the user to make their move. The move needs to be taken from the user and returned to the GameManager. The HumanPlayer class should allow the user to concede and also respond to illegal and bad moves.

The assessment for this part will be as follows:

- The ability of the HumanPlayer to display the current state of the board (this includes marks for appearance and looks)
- The ability of the HumanPlayer to take an input from the user for their next move
- The ability for the HumanPlayer to be either Red or Blue (set by the method setColour)

Requirements for submission:

- You must call your user interface class HumanPlayer.java and it must implement PlayerInterface.java. You must provide an implementation of the MoveInterface, called Move.java.

Part 4: Testing if a player has won (20%)

In order to determine if a player has won the game it is essential that the Board class can identify when a line of pieces exists between two opposite sides of the board. There are many ways to implement this and the choice is up to you as to how you achieve this.

The assessment for this part will be as follows:

- Correctly identifying if a player has won
- Returning the correct winner

Requirements for submission:

- You will implement the gameWon() method within the Board class and submit this as the same Board.java class as presented for part 1.

Part 5: Building a Computer Player (20%)

This part of the assignment is the extension part for you to show off your ability to develop good algorithms to solve problems in Java. You can use any approach you wish in determining the best moves to make.

The assessment for this part will be as follows:

- Making only valid moves i.e. only going for cells which are currently unoccupied
- The tournament. Each student's work, which contains a Computer player, will be entered into a tournament in which each ComputerPlayer will be pitted against an opponent with the winner going forward to the next round. Full marks will be given to the overall winner with marks being allocated to each other ComputerPlayer proportional to how well they do in the tournament.

Requirements for submission:

- You will submit a class called ComputerPlayer_<your id>.java where you replace <your id> with your login id. For example if your login id was aphq46 you would submit ComputerPlayer_aphq46.java. Note that your class will also need to be called ComputerPlayer_<your id>.

Plagiarism

You must not plagiarise your work. You may use program source code from the provided course examples or any other source **BUT** this usage must be acknowledged in the comments of your submitted file. Automated software tools (e.g. https://theory.stanford.edu/~aiken/moss/) may be used to initially detect cases of potential source code plagiarism in this assignment which will include automatic comparison against code from previous year groups. Attempts to hide plagiarism by simply changing comments/variable names will be detected.

You should have been made aware of the Durham University policy on plagiarism. Anyone unclear on this must consult the course lecturer prior to submission of this practical.