Programming Lab: Week 16
2D arrays

Last week

- 2D arrays

Learning Objectives

- To develop problem solving skills
- To learn how to construct multi-dimensional arrays

Resources

- Lecture notes - 2D arrays
- Supplied Java code from Week 15

Portfolio Exercise

This exercise is based on the draughts code that we demonstrated in the Week 15 lecture. Recall that this maintains a two-dimensional array of integers, which maps onto a given position of a game of draughts.

You must build on the supplied code for a draughts board to achieve the following:

Start by printing the draughts board, as in the lecture notes. The application should ask for a *start location* on the board (X,Y location); if this contains *either* a black or white piece, then an *end location* for a move must be requested (i.e., the location where the piece must be moved to). **The application should then state whether or not this is a valid move**; if it *is* a valid move, then the board should be updated (and pieces removed, if appropriate) and printed, and another move requested.

Note that this only allows single "captures" or forward moves, but one player can enter multiple moves (e.g., to effect a "multiple capture" move, if available).

The following rules define what we mean by a "valid" move"

- Single pieces must move towards the opponent's side of the board.
- Pieces may not move off the board, but may be removed if captured.
- All moves are made on the diagonals.
- A piece may capture an opponent's piece by moving from a square adjacent to the opponent to a square beyond the opponent on the diagonal. After this move is made, the opponent's piece is removed from the board.
- A single piece may move to an adjacent empty square on the diagonal, so long as the move is made towards the opponent's side of the board.

Hints:

You should start by writing, within the board class, a single method,

```
CheckMove(int start_x, int_start_y, int end_x, int end_y)
```

which should return an integer value, depending on its analysis of the board:

-1: illegal move (e.g. not on diagonal, move off board, etc.)
0: piece at `start_x`, `start_y` moves to `end_x`, `end_y`, with no capture
1: piece at `start_x`, `start_y` moves to `end_x`, `end_y`, with capture of opposing piece

Each of these situations should be tested, in turn, within the method.

Get your code working for one colour first, then extend it to handle the other colour (the code will essentially be the same, but in the other "direction").

Extension Exercise

Include code to detect the end of a game (i.e., when all pieces of one colour have been removed from the board).

Include the facility for "Kings"; these are single pieces that have reached the far side of the board (ie. the first row of the opponent's side), and may move in any direction diagonally. How will you represent these on the screen?