Othello Program Design and Planning

Theme Graphics

The theme is based on the cartoon show *Rick and Morty* by Adult Swim. The two opposing players will play as the characters happy Rick Sanchez and toxic Rick Sanchez respectively.

In file images:



**Figure 1** Banner.jpg - A stylized title image that will be the banner of the game



**Figure 2** Rick.jpg - A portrait of happy Rick Sanchez that will be player 1’s game piece



**Figure 3** ToxicRick.jpg - A portrait of toxic Rick Sanchez that will be player 2’s game piece

Sources of base images:

1. <https://en.wikipedia.org/wiki/List_of_Rick_and_Morty_episodes>
2. <https://aminoapps.com/c/rick-and-morty-amino/page/item/druggie-l-421/3ZPL_MlCDI0gZx6z01bWJrK82Qo1ZNp0kN>

Methods and Fields

* display textboxes
* Update matches
* Maybe don’t have the methods do everything -> make them specific

**Class Othello**

New Fields

int[] matchScore

* The number of matches each player has won

Methods

private void newMatch ()

* Set up a new game.
* Set the player scores to 0, call initBoard().

private void initBoard ()

* Set up the board in its starting formation.

private boolean validMove (int row, int col)

* Given the coordinates of the slot, return whether if it is a valid move. If the move is not valid, display a textbox informing the user.
* Done by checking if the position in board is marked with an indicator.

private void updateBoard (int row, int col, int player)

* Given the coordinates of the piece just played and who played it, update board, which represents the game board. Display how many pieces were outflanked in a textbox.
* Done by calling updateValidMoves (row, col) and flank (row, col, player)

private int updateValidMoves (int row, int col)

* Given the coordinates of the piece just played, update board with the indicators for valid moves. Return the number of valid moves.

private int flank (int playerTurn)

* Outflank any appropriate enemy pieces: update graphics and board. Returns the number of disks outflanked. Display a textbox for the user telling him/her how many disks were outflanked.
* Calls all more specific flank methods.

private int flankHori (int playerTurn)

* Outflank any appropriate enemy pieces in the row, given the player who just played. Update the graphics and board. Return the number of disks outflanked.

private int flankVerti (int playerTurn)

* Outflank any appropriate enemy pieces in the column, given the player who just played. Update the graphics and board. Return the number of disks outflanked.

private int flankDiag (int playerTurn)

* Outflank any appropriate enemy pieces in the one or two diagonals, given the player who just played. Update the graphics and board. Return the number of disks outflanked.

private void updateScore (int points, int player)

* Given the number of pieces turned over and the player, update the scores in score.

private boolean checkWinner ()

* Add the player scores, if it is 64 exactly, the game is over. If there is a winner or a tie, update the match score, display a textbox for the user.
* The values in score are added. Winner is determined by comparing values in score.

private void setNextPlayer ()

* Update curPlayer, the current player, to the next player. Also change the displayed player to the next player.
* validMove
  + check if the space specified by the given row and column is a valid move
* outFlankHori
  + check if the move (specified by the given row and column) by the specified player result in any outflank horizontally
  + flip any disks if necessary
  + return the number of outflanked disks
* outFlankVert
  + check if the move (specified by the given row and column) by the specified player result in any outflank vertically
  + flip any disks if necessary
  + return the number of outflanked disks
* outFlankDiag
  + check if the move (specified by the given row and column) by the specified player result in any outflank diagonally
  + flip any disks if necessary
  + return the number of outflanked disks
* checkWinner
  + return the winner of the game