**KNIGTH’S TOUR PROGRAM DOCUMENTATION**

KnightsTour.java

ChessComponents.java

Location.java

1. **Location.java:** This is a class for creating the instances of the boxes that we place the knight on the chess table. It helps the program to keep the track of each move by storing each Location instance in an ArrayList.

It has two methods called getRow(), for returning the attribute “int row”, and getColumn(), for returning the attribute “int column”.

1. **ChessComponents.java:** This is the class, which contains the algorithm for solving the Knight’s Tour puzzle. The chessboard is represented as a two-dimensional integer array, finalGame[][], which has all 0 elements at the beginning. There is also another two-dimensional array, which has the accessibility values of specific boxes on a chessboard. Both arrays are 5x5.

There is a one-dimensional ArrayList for the location instances called “moves” for tracking the moves, and backtracking process.

As the constructer initializes the instance, the program places the first knight to the first box, which is randomly determined. It also records the first box as the first move, signs the finalGame [][], and increments the moveNumber by 1.

The first method is called isPossible(int horizontal, int vertical), which returns a Boolean variable after applying all 8 possible moves of a knight to the current location, and checks if the resulting position on the chessboard, and has never visited before.

The second method is nextMove(), which is a recursive method. This is the method, where the program finds the perfect box on the chessboard for the next move after checking all possible 8 moves with the help of isPossible(int horizontal, int vertical) method inside a nested loop. The array called numbers[] is the array for storing the accessibility value of each resulting box. There is also an array called move[] for storing the index number of all 8 moves. These index numbers will later be used for determining the next currentRow and currentColumn from combining a move from horizontalMoves[] and verticalMoves[] arrays. These arrays are designed to represent the horizontal and vertical position changes in each 8 possible moves of a knight.

Backtracking also occurs inside the nextMove() method. If program can’t find any valid next position, computer backtracks to the previous position, removes the last move from the “queens” list, set the last moves coordinates on the finalGame[][] to 0, decrease the moveNumber and recurs. Also, program adds the coordinates of the box, in which backtrack occurred to forbiddenRow[] and forbiddenColumn[] ArrayLists, so program won’t repeat itself during the recursion.

Third method is called placeKnight(), which is the main recursive method. It places the knight to the position determined by the nextMove() method, records the coordinated of the move to the moves[] list, resets the forbiddenRow[] and forbiddenColumn[] values, and recurse until the movenumber reach to 64, which means the knight visited all boxes.

The last method is printGame(), which simply uses a nested for loop to create the String game, which will represent the finalGame[][] array as the display of the final view of the chessboard.

1. **ViewerFrame.java:**
2. **KnightsTour.java:** This is the tester class of the program. Since some of the initial positions on the chessboard can give a valid solution to the Knight’s Tour puzzle, there are two int arrays named horizontal[] and vertical[] for combining a valid initial position when the same index number is entered. The index number is generated randomly. Then the ChessComponents object is initialized, and placeKnight() method is called.