# Knights’s Tour Design Document

## Classes

chess.Tour

chess.Knight

chess.Warnsdorff

## Strategies for finding the next move in a tour

1. Warnsdorff’s rule (no explanation required)

# Tour

## Responsibilities

The tour class models the state of the tour. It initiates a ‘tour’, checks if there is a next move, and moves the knight to the best move using Warnsdorff to find it,

## Fields

Location board: contains the upper x and y bounds of the board.

Knight knight: the knight on the tour.

## Important methods

Boolean hasNext()

Given the moves determined by the knight, it determines if there are any moves to take by checking if the list of moves if empty. If this is true, the function will return false and vice versa.

Location next()

Using Warnsdorff, this function determines the best move for the knight to take and moves the night to the location and returns said location

# Knight

## Responsibilities

The knight class determined the possible moves of the knight either given a location or from the knights location.

## Fields

Int[] xChange: an array in ints that represent the x translations that when paired with the corresponding y translation, make one of the knights 8 possible moves

Int[] yChange: an array in ints that represent the y translations that when paired with the corresponding x translation, make one of the knights 8 possible moves

List <Location> moves: a list that contained all the moves of the knight from its location

Location location: the location of the knight

Location board: contains the upper x and y bounds of the board.

HashSet <Location> visited: a set of all the locations the knight visited

## Important methods

Void getMoves()

Gets all the possible moves from the knights location, taking into account board size and places visited. It places these moves in Knight.moves.

List<Location> testMoves()

Does the same as getMoves(Location loc) but takes a location as a parameter and outputs the list of moves directly

# Warnsdorff

## Responsibilities

Finds one of the best possible moves for the knight using Warnsdorff’s Rule.

## Fields

N/A

## Important methods

Location warnsRule (Knight knight)

Given a knight, it looks at each possible move the knight has and using testMoves() counts how many moves there are from each potential move the knight can take. Then it outputs a move with the least number of moves.