The Knight Rider Game

Author: Azahar uddin

Problem Statement:

Write a gaming program to move a knight to the bottom right cell of a chess board from a given position. The rules of the game are

1.	Number of players = 2
2.	Each user chooses to place the knight in a cell of his choice through a simple command line interface
3.	User should not be allowed to place the knight on the winning position (at the start of the game).
4.	The Knights should move programmatically one after another (1 move for the first knight followed by 1 move for the second knight and so on) on the board.
5.	Each movement of the knight has to be valid knight movement according to Chess (The knight moves two squares horizontally and one square vertically or 1 square horizontally and 2 squares vertically)
6.	Display each movement on the command line by indicating the start and the end cell.
7.	The game ends when one of the knights reaches the END position (bottom right corner) or it can be a draw based on condition chosen by you.

Optionally, you can choose to ensure that each move is geared towards the corresponding knight winning (Reaching the end first).

The game has been named as Knight Rider (2 Player each having 1 knight).

Chessboard has been assumed to be 8 X 8 matrix board. Winning coordinate is 8,8.

All requirement points mentioned in Problem statement has been implemented.

Folder/File Structure:

- bin : It has runnable jar of the game: knightrider.jar

- config: It has config.properties file to configure language and language string xml. Language string xml is also in same folder.

- doc: It has documentation.

- src: Folder having source code. Comments have been provided in codes. If any portion is not clear please let me know.

- Log4j: Log4j plugin xml configuration file

- startgame.bat: Run this batch to start game. Before running this batch file please read readme.txt.

- readme.txt: Please read this file before launching startgame.bat.

Additionally,

- Game system intelligence will tell player whether knight with given starting position will ever result in reaching destination 8,8.

- At each movement game system intelligence will tell whether that move will result in quick win.

- Game system intelligence might offer help to Player for next movement based on situation of the game.

- Log4j plug-in has been used to generate log. Logs will be generated in log folder.

- There is result folder in which result log will be generated. Both log and result will be auto generated.

- Old log and result will be saved in Year-Month folder inside each folder.

- Interface has been provided to work in different languages. In cofig.properties change language and corresponding string_<lang>.xml. Place string_<lang>.xml in same folder.

- UML class diagram is provided with this document.

UML:

<<Java Class>>
© ChessBoard
com.mycompany.us.game.board

- ▫ m_PlayerChessMenMap: HashMap<Integer,HashMap<String,List<IChessMan>>>
- ˢᴼF log: Logger

- ● getOccupiedCoordList():HashMap<String,ICoordinates>
- ©ᶜ ChessBoard(HashMap<Integer,HashMap<String,List<IChessMan>>>)
- ● getAllChessManList(int):HashMap<String,List<IChessMan>>
- ● getChessMenListByType(int,String):List<IChessMan>
- ● move(int,IChessMan,ICoordinates,int,int):List<ICoordinates>
- ▪ validateAndUpdateKnightCoords(int,IChessMan,ICoordinates,int,int):List<ICoordinates>
- ▪ validateKnightMove(int,IChessMan,ICoordinates,int,int):List<ICoordinates>
- ● getKey(int,int):String

<<Java Interface>>
① IBoard
com.mycompany.us.game.board

- ● move(int,IChessMan,ICoordinates,int,int):List<ICoordinates>
- ● getAllChessManList(int):HashMap<String,List<IChessMan>>
- ● getChessMenListByType(int,String):List<IChessMan>
- ● getOccupiedCoordList():HashMap<String,ICoordinates>
- ● getKey(int,int):String

#m_iBoard ┤ 0..1

<<Java Class>>
© KnightMuzzle
com.mycompany.us.game.board

- ▫ m_AllMoves: LinkedHashMap<Integer,List<ICoordinates>>
- ˢᴼF log: Logger
- ▫ m_paths: LinkedList<LinkedList<ICoordinates>>
- ▫ m_iMove: int
- ▫ m_strType: String

- ● getAllMoves():LinkedHashMap<Integer,List<ICoordinates>>
- ● getCurrentCoord():ICoordinates
- ©ᶜ KnightMuzzle()
- ● setCoordinates(int,IBoard,int,int,boolean):String
- ● getType():String
- ● setType(String):void
- ● getNextMoveHelp(IBoard):String

<<Java Interface>>
① IChessMan
com.mycompany.us.game.board

- ● setCoordinates(int,IBoard,int,int,boolean):String
- ● getType():String
- ● setType(String):void
- ● getCurrentCoord():ICoordinates
- ● getAllMoves():LinkedHashMap<Integer,List<ICoordinates>>
- ● getNextMoveHelp(IBoard):String

-m_iChessMan 0..1

-m_OccupiedCoordsList

<<Java Class>>
© ChessmanCoordinates
com.mycompany.us.game.chessgame.coordinates

- ▫ m_iCoordinateDimension: int
- ▫ m_ArrayCoordinates: int[]
- ▫ m_iPlayer: int

- ©ᶜ ChessmanCoordinates(int,int)
- ©ᶜ ChessmanCoordinates(IChessMan,int,int,int)
- ● setRowCoordinate(int):void
- ● setColCoordinate(int):void
- ● getRowCoordinate():int
- ● getColCoordinate():int
- ● getCoordinates():int[]
- ● setCoordinates(IChessMan,int,int,int):void

0..*

-m_CurrentCoord 0..1

<<Java Interface>>
① ICoordinates
com.mycompany.us.game.chessgame.coordinates

- ● setRowCoordinate(int):void
- ● setColCoordinate(int):void
- ● getRowCoordinate():int
- ● getColCoordinate():int
- ● getCoordinates():int[]
- ● setCoordinates(IChessMan,int,int,int):void
- ● setPlayer(int):void
- ● getPlayer():int
- ● setChessMan(IChessMan):void
- ● getChessMan():IChessMan

-m_knightMoveValidtor 0..1

<<Java Class>>
© KnightMoveValidator
com.mycompany.us.game.chessgame.movevalidator

- ˢᴼF log: Logger

- ● validateMove(int,IChessMan,ICoordinates,int,int):List<ICoordinates>
- ©ᶜ KnightMoveValidator()

<<Java Interface>>
① IMoveValidator
com.mycompany.us.game.chessgame.movevalidator

- ● validateMove(int,IChessMan,ICoordinates,int,int):List<ICoordinates>

<<Java Class>>
© GameConst
com.mycompany.us.game.chessgame.utils

- ○ˢ KNIGHT_RIDER_INPUT_SEPERATOR: String
- ○ˢ KNIGHT_PLAYER_TYPE: String
- ○ˢ MULTIPLE_PLAYER: String
- ○ˢ STRINGS: String
- ○ˢ CONFIG_FILE: String
- ○ˢ FILE_SEPERATOR: String
- ○ˢ NODE_MESSAGES: String
- ○ˢ LANG: String
- ○ˢ CONT: String
- ○ˢ TRUE: String
- ○ˢ FALSE: String
- ○ˢ LAST_MOVE: int
- ©ᶜ GameConst()

<<Java Class>>
© StringIdentifierConst
com.mycompany.us.game.chessgame.utils

- ○ˢ COPY_RIGHT: String
- ○ˢ WELCOME_MSG: String
- ○ˢ PLAYER_NOTE: String
- ○ˢ WHO_WILL_WIN: String
- ○ˢ PLAYER: String
- ○ˢ IS_AT: String
- ○ˢ STOP_MSG: String
- ○ˢ ENTER_COORDINATE: String
- ○ˢ INCORRECT_COORDINATES: String
- ○ˢ INCORRECT_ROW: String
- ○ˢ INCORRECT_COL: String
- ○ˢ INCORRECT_MOVE: String
- ○ˢ INCORRECT_SETUP: String
- ○ˢ ABORT: String
- ○ˢ WON_MSG: String
- ○ˢ BROKE_MSG: String
- ○ˢ OVER: String

<<Java Class>>
© Utils
com.mycompany.us.game.chessgame.utils

- ˢᴼF log: Logger

<<Java Class>>
© TwoPlayerTwoKnightGame
com.mycompany.us.game.knightrider

Utils (partial)

- ○ NEW_LINE_SEPERATOR: String
- ○ DIR_PATH: String
- □ m_strStringPath: String
- ○ LANG: String
- ○ err: List<String>

- ● Utils()
- ● getSplitString(String,String):String[]
- ● knightStringValidator(String[]):boolean
- ● knightCoordinateValidator(String):int
- ● insertResultLog(Logger,String,boolean):void
- ● ConfigSetup(String,String):boolean
- ● getStringFilePath():String
- ● createGraph(int,int):IGraph
- ● getPath(int,int,int,int):LinkedList<LinkedList<ICoordinates>>
- ● createGraphForCoords(IGraph,ICoordinates):void

StringIdentifierConst (partial)

- ○ BYE: String
- ○ PLAYER_MOVE: String
- ○ COORDINATES: String
- ○ NO_MOVES: String
- ○ MOVE: String
- ○ INAPPROPRIATE_MOVE: String
- ○ NEXT_PROBABLE_MOVE: String
- ○ CORRECT_BUT_NOT_FAST_MOVE: String
- ○ HELP_SYSTEM: String
- ○ NO_HELP: String
- ○ NEXT_MOVE: String
- ○ EXCEPTION_FALIED: String
- ○ PROPERTY_DOES_NOT_EXIST: String

- ● StringIdentifierConst()

TwoPlayerTwoKnightGame (partial)

- ◇ m_iPlayer: int
- ● m_ResultLogger: Logger
- ◇ m_playerChessMenMap: HashMap<Integer,HashMap<String,List<IChessMan>>>
- □ m_bloop: boolean
- ● log: Logger

- ● TwoPlayerTwoKnightGame()
- ● TwoPlayerTwoKnightGame(boolean)
- ● setUp():void
- ● play():void
- ■ helpMethod(List<IChessMan>):void
- ■ validateMove(int,int,int,Scanner,List<IChessMan>):int
- ■ playerHelpMove(int,List<IChessMan>):void
- ■ generateMoveInfo(int,List<IChessMan>):void
- ■ gameOver(int):void
- ● stopGame():void
- ■ printMoves(boolean):void

<<Java Class>>
Ⓖ GameStrings
com.mycompany.us.game.file.strings

- □ m_StringValueMap: HashMap<String,String>
- ● log: Logger

- ◇ GameStrings()
- ● getStringValueMap():HashMap<String,String>
- ● getInstance():GameStrings
- ■ LoadDictionary(NodeList):void
- ● LoadFile(String):boolean
- ● getString(String):String
- ● setString(String,String):void
- ● setStringValueMap(HashMap<String,String>):void
- ● printDictionary():void

-m_GameStrings
0..1

-m_GameStrings
0..1

<<Java Interface>>
Ⓘ IGraph
com.mycompany.us.game.chessgame.utils

- ● getGraphObject():HashMap<HashMap<String,ICoordinates>,LinkedList<HashMap<String,ICoordinates>>>
- ● addEdge(ICoordinates,ICoordinates):boolean
- ● getList(ICoordinates):LinkedList<ICoordinates>
- ● getPath(ICoordinates,ICoordinates):LinkedList<LinkedList<ICoordinates>>
- ● printGraph():void

<<Java Class>>
Ⓖ CoordinatesGraph
com.mycompany.us.game.chessgame.utils

- □ m_Graph: HashMap<HashMap<String,ICoordinates>,LinkedList<HashMap<String,ICoordinates>>>
- ● log: Logger

- ◇ CoordinatesGraph()
- ● getGraphObject():HashMap<HashMap<String,ICoordinates>,LinkedList<HashMap<String,ICoordinates>>>
- ● addEdge(ICoordinates,ICoordinates):boolean
- ■ genKey(ICoordinates):String
- ■ getList(LinkedList<HashMap<String,ICoordinates>>):LinkedList<ICoordinates>
- ● getList(ICoordinates):LinkedList<ICoordinates>
- ● getPath(ICoordinates,ICoordinates):LinkedList<LinkedList<ICoordinates>>
- ■ getPathCoordinates(ICoordinates,ICoordinates,LinkedList<ICoordinates>,LinkedList<ICoordinates>,LinkedList<LinkedList<ICoordinates>>):void
- ● printGraph():void

<<Java Class>>
Ⓖ TestPrint
com.mycompany.us.game.chessgame.utils

- ● log: Logger

- ● TestPrint()
- ● printAllPlayerCoordinates(List<List<ChessmanCoordinates>>):void
- ● printAllPlayerMoves(Logger,int,IBoard):void
- ● printCoordinates(Logger,int,List<ICoordinates>):void
- ● printPaths(LinkedList<LinkedList<ICoordinates>>):void
- ● getPathString(LinkedList<ICoordinates>):String