chess

Game : ISerializable

~ moves: ArrayLoist<String>
~ board: Board
~ whiteKing: ChessPiece
~ blackKing: ChessPiece
+ end: boolean
~ round: int
~ currentColor: Color
~ winner: String
~ command: String
~ coordinates: char[]
~ row: int
~ col: int
~ rowDest: int
~ colDest: int
~ invalidInput: boolean
~ invalidMove: boolean
~ possDest: boolean[][]
+ scanner: java.util.ScannerType
~ fw: FileWriter

+ main(argv: String[])
+ Game()
- getMoves(): ArrayList<String>
+ saveMoves(filename: String)
+ saveGame(fileName: String)
+ loadGame(fileName: String): Game
+ loadGameByMoves(fileName: String): Game
- loadGameByMoves(moves: ArrayList<String>): Game
- translateColumn(positition: int): char
- translateColumn(position: char): int

Board : ISerializable

+ Board()
+ getBoard(): ChessPiece[][]
+ getPosition(piece: ChessPiece): int[]
+ move(rowCur: int, colCur: int, rowDes: int, colDes: int)
+ toString(): String

<<enumeration>>
Color

WHITE
BLACK

piece

ChessPiece : ISerializable

~ type : ChessPieceType
~ color : Color

+ ChessPiece(type: ChessPieceType, color: Color)
+ getType(): ChessPieceType
+ getColor(): Color
+ toString(): String
+ equals(obj:Object): boolean
+ getPossibleDestinations(board: Board): boolean[][]
+ canMove(board: Board, row: int, col: int): boolean

Queen

+ Queen(color: Color)
+ toString(): String
+ canMove(board: Board, row: int, col: int): boolean

King

+ King(color: Color)
+ toString(): String
+ canMove(board: Board, row: int, col: int): boolean

Rook

+ Rook(color: Color)
+ toString(): String
+ canMove(board: Board, row: int, col: int): boolean

Bishop

+ Bishop(color: Color)
+ toString(): String
+ canMove(board: Board, row: int, col: int): boolean

Knight

+ Knight(color: Color)
+ toString(): String
+ canMove(board: Board, row: int, col: int): boolean

Pawn

+ Pawn(color: Color)
+ toString(): String
+ canMove(board: Board, row: int, col: int): boolean

<<enumeration>>
ChessPieceType

KING
QUEEN
ROOK
BISHOP
KNIGHT
PAWN