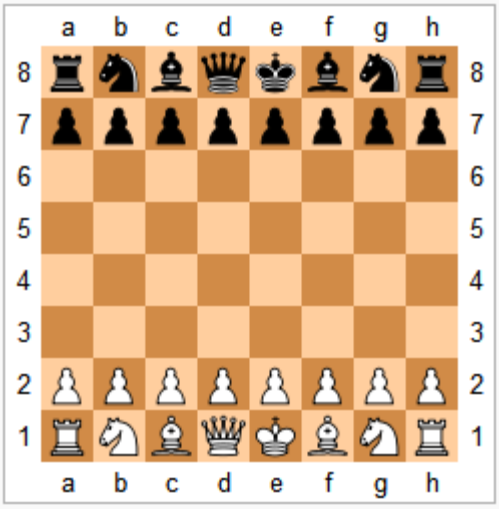
Knight Moves



Initial position: first row: rook, knight, bishop, queen, king, bishop, knight, and rook; second row: pawns

You have a chessboard of size 8 x 8. The rows are numbered from 1 to 8. The columns are numbered from "a" to "h". In a cell located at row R1 and Column C1, a knight is starting his journey. The knight wants to go to the cell located at row R2 and Column C2. Move the knight from its current cell to this destination cell with minimum number of moves.

A forward move increases the row.
A backward move decreases the row
A right move increases the column.
A left move decreases the column.

As a reminder, a knight's jump moves him 2 cells along one of the axes, and 1 cell along the other one. In other words, if a knight is at (4, f), it may move to (2, e), (2, g), (3, h), (5, h), (6, g), (6, e) (5, d) and (3, d). Or if the knight is at (2, b), it may move to (1, d), (3, c), (3, a) with the other five possible locations being off the board.

You have been given the complete `ChessLocation` class shown below:

```
public class ChessLocation
{
    private int  row;       //  1 .. 8
    private String col;     //  "a" .. "h"

    /**
     * Constructor for objects of class ChessLocation
     */
    public ChessLocation(int r1, String s)
    {
        row = r1;
        col = s;
    }

    public int getRow()    { return row; }

    public String getCol() { return col; }

    public boolean equals(Object obj)
    {
        ChessLocation temp = (ChessLocation)obj;
        return row == temp.getRow() && col.equals(temp.getCol());
    }
```

Borrowed from 2011 U Central Florida Local Contest

```
    public int hashCode()
    {
        Integer temp = new Integer(row);
        return col.hashCode() + temp.hashCode();
    }

    public String toString()
    {
        return getRow() + ", " + getCol();
    }
}
```

Your task is to complete the nine incomplete methods in the `KnightMoves` class. These nine methods include the eight move methods (`forwardTwoThenRight, forwardTwoThenLeft, forwardOneThenRightTwo, forwardOneThenLeftTwo, backwardTwoThenRight, backwardTwoThenLeft, backwardOneThenRightTwo, backwardOneThenLeftTwo`) and the `minimumNumMovesTo` method. For the eight move methods, if the method moves the Knight off the chess board, return `null`.

Sample code:

Using the declaration:

```
ChessLocation temp = new ChessLocation(4, "d");
ChessLocation corner = new ChessLocation(8, "a");
```

the following table indicates the `ChessLocation` return by indicated method.

The method	Returns the value
KnightMoves.forwardTwoThenRight(temp)	ChessLocation(6, "e")
KnightMoves.forwardTwoThenLeft(temp)	ChessLocation(6, "c")
KnightMoves.forwardOneThenRightTwo(temp)	ChessLocation(5, "f")
KnightMoves.forwardOneThenLeftTwo(temp)	ChessLocation(5, "b")
KnightMoves.backwardTwoThenRight(temp)	ChessLocation(2, "e")
KnightMoves.backwardTwoThenLeft(temp)	ChessLocation(2, "c")
KnightMoves.backwardOneThenRightTwo(temp)	ChessLocation(3, "f")
KnightMoves.backwardOneThenLeftTwo(temp)	ChessLocation(3, "b")
KnightMoves.forwardOneThenLeftTwo(corner)	null
KnightMoves.forwardTwoThenRight (corner)	null

And

Using the declaration:

```
KnightMoves game = new KnightMoves(new ChessLocation(1, "a"));
```

the following table indicates the `int` returned by the `minimumNumMovesTo` method.

The method	Returns the value
game.minimumNumMovesTo(new ChessLocation(2, "c")))	1
game.minimumNumMovesTo(new ChessLocation(2, "b")))	4

Borrowed from 2011 U Central Florida Local Contest