NQueensProblem

```java
package day16_17;



public final class NQueensProblem {
    public static void main(String[] args) {
        int size = 8;
        boolean[][] board = new boolean[size][size];

        NQueensProblem nQueensProblem = new NQueensProblem();
        if (!nQueensProblem.nQueen(board, size, 0)) {
            System.out.println("No solution found :( ");
        }
    }

    private boolean nQueen(boolean[][] board, int size, int row) {
        if (row == size) {
            printBoard(board, size);
            return true;
        } else {
            boolean foundSolution = false;
            for (int col = 0; col < size; col++) {
                if (isValidCell(board, size, row, col)) {
                    board[row][col] = true; // Place the queen
                    foundSolution = nQueen(board, size, row + 1) || foundSolution; //
Recursive call
                    board[row][col] = false; // Backtrack
                }
            }
            return foundSolution;
        }
    }

    private boolean isValidCell(boolean[][] board, int size, int row, int col) {
        // check column
        for (int i = 0; i < row; i++) {
            if (board[i][col]) {
                return false;
            }
        }

        // check upper left diagonal
        for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
            if (board[i][j]) {
                return false;
            }
        }

        // check upper right diagonal
        for (int i = row, j = col; i >= 0 && j < size; i--, j++) {
            if (board[i][j]) {
                return false;
            }
        }
        return true;
    }
```

```
    private void printBoard(boolean[][] board, int size) {
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                System.out.print(board[i][j] ? "Q " : "- ");
            }
            System.out.println();
        }
        System.out.println();
    }
}
```

RatInMaze.java | Task1.java | Task3.java | Task4.java | Person.java | NQueensProbl... | Task5.java

```
36              if (board[i][col]) {
37                  return false;
38              }
39          }
40
41          // check upper left diagonal
42          for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
43              if (board[i][j]) {
44                  return false;
45              }
46          }
47
48          // check upper right diagonal
49          for (int i = row, j = col; i >= 0 && j < size; i--, j++) {
50              if (board[i][j]) {
51                  return false;
52              }
53          }
54          return true;
55      }
56
57      private void printBoard(boolean[][] board, int size) {
58          for (int i = 0; i < size; i++) {
59              for (int j = 0; j < size; j++) {
60                  System.out.print(board[i][j] ? "Q " : "- ");
61              }
62              System.out.println();
63          }
64          System.out.println();
```

Markers | Properties | Terminal | Console | Coverage

<terminated> NQueensProblem [Java Application] C:\Users\Nikita\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v202107

```
- - - Q - - - -
Q - - - - - - -
- - Q - - - - -
- - - - - Q - -
- Q - - - - - -
- - - - - - Q -
- - - - Q - - -
```