

```
1 import java.io.*;
2 import java.util.Arrays;
3
4 class NQueen {
5
6     static int N = 8;
7
8     static void printSolution(int board[][])
9     {
10         int N = board.length;
11         for(int i = 0; i < N; i++)
12         {
13             for(int j = 0; j < N; j++)
14                 System.out.printf("%2d ", board[i][j]);
15
16             System.out.printf("\n");
17         }
18     }
19
20     static boolean isSafe(int row, int col,
21         int slashCode[],
22         int backslashCode[],
23         boolean rowLookup[],
24         boolean slashCodeLookup[],
25         boolean backslashCodeLookup[])
26     {
27         if (slashCodeLookup[slashCode[row][col]] ||
28             backslashCodeLookup[backslashCode[row][col]] ||
29             rowLookup[row])
30             return false;
31
32         return true;
33     }
34
35     static boolean solveNQueensUtil(
36         int board[], int col, int slashCode[],
37         int backslashCode[], boolean rowLookup[],
38         boolean slashCodeLookup[],
39         boolean backslashCodeLookup[])
40     {
41         int N = board.length;
42
43         if (col >= N)
44             return true;
45
46         for(int i = 0; i < N; i++)
47         {
48             if (isSafe(i, col, slashCode, backslashCode,
49                 rowLookup, slashCodeLookup,
50                 backslashCodeLookup))
51             {
52                 board[i][col] = 1;
53                 rowLookup[i] = true;
54                 slashCodeLookup[slashCode[i][col]] = true;
55                 backslashCodeLookup[backslashCode[i][col]] = true;
56
57                 if (solveNQueensUtil(
```

```

58     board, col + 1, slashCode,
59     backslashCode, rowLookup,
60     slashCodeLookup,
61     backslashCodeLookup))
62     return true;
63
64     board[i][col] = 0;
65     rowLookup[i] = false;
66     slashCodeLookup[slashCode[i][col]] = false;
67     backslashCodeLookup[backslashCode[i][col]] = false;
68 }
69 }
70
71 return false;
72 }
73
74 static boolean solveNQueens()
75 {
76     int board[][] = new int[N][N];
77
78     // Helper matrices
79     int slashCode[][] = new int[N][N];
80     int backslashCode[][] = new int[N][N];
81
82     boolean[] rowLookup = new boolean[N];
83
84     boolean slashCodeLookup[] = new boolean[2 * N - 1];
85     boolean backslashCodeLookup[] = new boolean[2 * N - 1];
86
87     for(int r = 0; r < N; r++)
88         for(int c = 0; c < N; c++)
89         {
90             slashCode[r] = r + c;
91             backslashCode[r] = r - c + 7;
92         }
93
94     if (solveNQueensUtil(board, 0, slashCode,
95         backslashCode, rowLookup,
96         slashCodeLookup,
97         backslashCodeLookup) == false)
98     {
99         System.out.printf("Solution does not exist");
100         return false;
101     }
102
103     printSolution(board);
104     return true;
105 }
106
107 public static void main(String[] args)
108 {
109     solveNQueens();
110 }
111 }

```