```java
package dfs;

import java.util.Scanner;

public class NQueenProblem {
    Scanner s=new Scanner(System.in);
    final int N =s.nextInt();
    void printSolution(int [][]board)
    {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++)
                System.out.print(" " + board[i][j]
                        + " ");
            System.out.println();
        }
    }
    boolean isSafe(int [][]board, int row, int col)
    {
        int i, j;
        for (i = 0; i < col; i++)
            if (board[row][i] == 1)
                return false;
        for (i = row, j = col; i >= 0 && j >= 0; i
--, j--)
            if (board[i][j] == 1)
                return false;
        for (i = row, j = col; j >= 0 && i < N; i++,
j--)
            if (board[i][j] == 1)
                return false;
        return true;
    }
    boolean solveNQUtil(int [][]board, int col)
    {
        if (col >= N)
            return true;

        for (int i = 0; i < N; i++) {
            if (isSafe(board, i, col)) {
                board[i][col] = 1;
                if (solveNQUtil(board, col + 1))
```

```
40                      return true;
41                  board[i][col] = 0; // BACKTRACK
42              }
43          }
44          return false;
45      }
46
47      void solveNQ()
48      {
49          int [][]board = { { 0, 0, 0, 0 },
50                  { 0, 0, 0, 0 },
51                  { 0, 0, 0, 0 },
52                  { 0, 0, 0, 0 } };
53
54          if (!solveNQUtil(board, 0)) {
55              System.out.print("Solution does not exist
   ");
56              return;
57          }
58
59          printSolution(board);
60      }
61      public static void main(String []args)
62      {
63          NQueenProblem Queen = new NQueenProblem();
64          Queen.solveNQ();
65      }
66  }
```