```java
import java.util.*;

class Solution {
    char[][] board;
    // Arrays to indicate is the diagonal and offdiagonals of any cell has Queen
    boolean[] diagonal, offDiagonal, column;

    // O(1) Time complexity function to check if queen can be placed at [x,y] or not
    public boolean isValid(int x, int y, int n) {
        return !(diagonal[y-x+n-1] || offDiagonal[x+y] || column[y]);
    }

    // wrapper function checking and filling all the positions correctly
    public void helper(int row, int n, List<List<String>> ans) {
        if(row==n) {
            //O(n*n)
            List<String> base = new ArrayList<>();
            for(char[] r: board) {
                base.add(String.valueOf(r));
            }
            ans.add(base);
            return;
        }
        for(int j=0; j<n; j++) {
            if(isValid(row, j, n)==true) {
                // put a queen on row,j
                board[row][j] = 'Q';
                diagonal[j-row+n-1] = true;
                offDiagonal[row+j] = true;
                column[j] = true;
                // recursively call for next row queen
                helper(row+1, n, ans);
                // backtrack
                diagonal[j-row+n-1] = false;
                offDiagonal[row+j] = false;
                column[j] = false;
                board[row][j] = '.';
            }
        }
    }

```

```java
    // solve N queens
    public List<List<String>> solveNQueens(int n) {
        board = new char[n][n];
        // Required mapping = j-i+n-1
        diagonal = new boolean[2*n-1];
        // required mapping = i+j
        offDiagonal = new boolean[2*n-1];
        // required mapping = j
        column = new boolean[n];

        // fill the board with '.'
        for(int i=0; i<n; i++) Arrays.fill(board[i], val: '.');
        List<List<String>> ans = new ArrayList<>();

        // call the wraper function
        helper(row: 0, n, ans);
        return ans;
    }
}

public class NQueen {
    Run | Debug
    public static void main(String[] args) {
        int n = 4;
        System.out.println("\nThe N Queen is solved for: " + n);
        Solution solve = new Solution();
        List<List<String>> ans = solve.solveNQueens(n);
        for(List<String> sol: ans) {
            System.out.println();
            for(String row: sol){
                System.out.println(row);
            }
            System.out.println();
        }
    }
}
```