```
//N Queen program using branch and bound algorithm
/*N queens have to be put on N*N chess matrix safely, so that none of
queens could come in other queens' regions.*/
/*Output:
- Q - -
- - - Q
Q - - -
- - Q -

- - Q -
Q - - -
- - - Q
- Q - - */
import java.util.*;
public class NQueen
{
        public static final int N = 4;
        private static boolean isSafe(char mat[][], int r, int c)
        {
                for (int i = 0; i < r; i++)
                        if (mat[i][c] == 'Q')
                                return false;
                for (int i = r, j = c; i >= 0 && j >= 0; i--, j--)
                        if (mat[i][j] == 'Q')
                                return false;
                for (int i = r, j = c; i >= 0 && j < N; i--, j++)
                        if (mat[i][j] == 'Q')
                                return false;
                return true;
        }

        private static void nQueen(char mat[][], int r)
        {
                if (r == N)
                {
                        for (int i = 0; i < N; i++)
                        {
                                for (int j = 0; j < N; j++)
                                        System.out.print(mat[i][j] + "
");
                                System.out.println();
                        }
                        System.out.println();

                        return;
                }
                for (int i = 0; i < N; i++)
                {
                        if (isSafe(mat, r, i))
                        {
                                mat[r][i] = 'Q';
                                nQueen(mat, r + 1);
                                mat[r][i] = '-';
                        }
                }
        }
        public static void main(String[] args)
        {
                char[][] mat = new char[N][N];
                for (int i = 0; i < N; i++) {
                        Arrays.fill(mat[i], '-');
                }
                nQueen(mat, 0);
        }
}
```