Nqueen.java(Backtracking)

```
public class NQueenProblem
{
	final int N = 8;
		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) == true)
					return true;
				board[i][col] = 0;
			}
		}
		return false;
	}
	boolean solveNQ()
	{
```

```
                int board[][] = new int[8][8];

                if (solveNQUtil(board, 0) == false)
                {
                        System.out.print("Solution does not exist");
                        return false;
                }

                printSolution(board);
                return true;
        }
        public static void main(String args[])
        {
                NQueenProblem Queen = new NQueenProblem();
                Queen.solveNQ();
        }
}
// This code is contributed by RD,PK,RC(NOT AD)
```

Output -

```
1  0  0  0  0  0  0  0
 0  0  0  0  0  0  1  0
 0  0  0  0  1  0  0  0
 0  0  0  0  0  0  0  1
 0  1  0  0  0  0  0  0
 0  0  0  1  0  0  0  0
 0  0  0  0  0  1  0  0
 0  0  1  0  0  0  0  0
```