```
import java.util.Scanner;

public class NQueens {

    // Function to check if a queen can be placed at position (row, col)
    static boolean isSafe(int board[][], int row, int col, int n) {
        // Check the same column
        for (int i = 0; i < row; i++) {
            if (board[i][col] == 1) {
                return false;
            }
        }

        // Check the upper-left diagonal
        for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
            if (board[i][j] == 1) {
                return false;
            }
        }

        // Check the upper-right diagonal
        for (int i = row, j = col; i >= 0 && j < n; i--, j++) {
            if (board[i][j] == 1) {
                return false;
            }
        }

        return true;
    }

    // Function to solve N-Queens using Backtracking
    static boolean solveNQueens(int board[][], int row, int n) {
        // If all queens are placed, return true
        if (row == n) {
            return true;
        }

        // Try placing the queen in all columns of the current row
        for (int col = 0; col < n; col++) {
            // Check if placing the queen at board[row][col] is safe
            if (isSafe(board, row, col, n)) {
                // Place the queen
                board[row][col] = 1;
            // Recur to place the rest of the queens
            if (solveNQueens(board, row + 1, n)) {
                return true;
            }

            // If placing the queen doesn't lead to a solution, backtrack
            board[row][col] = 0;
        }
    }

    // If the queen can't be placed in any column in this row, return false
    return false;
}

// Function to print the board
static void printBoard(int board[][], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            System.out.print(board[i][j] + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    // Input the size of the board (n)
    System.out.print("Enter the size of the board (n): ");
    int n = sc.nextInt();

    // Create the board and initialize it with 0
    int[][] board = new int[n][n];

    // Input the initial board configuration
    System.out.println("Enter the initial board configuration (0 for empty, 1 for queen):");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            board[i][j] = sc.nextInt();
        }
    }

    // Try to solve the N-Queens problem
    if (solveNQueens(board, 0, n)) {
            System.out.println("Solution found:");
            printBoard(board, n);
        } else {
            System.out.println("No solution exists!");
        }

        sc.close();
    }
}
```