```
public class NQueenBranchAndBound {
    static final int N = 8;
    static int[][] board = new int[N][N];
    static int[][] slashCode = new int[N][N];
    static int[][] backslashCode = new int[N][N];

    static boolean[] rowCheck = new boolean[N];
    static boolean[] slashCheck = new boolean[2 * N - 1];
    static boolean[] backslashCheck = new boolean[2 * N - 1];

    // Pre-fill slash and backslash codes
    static void initialize() {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                slashCode[i][j] = i + j;
                backslashCode[i][j] = i - j + (N - 1);
                board[i][j] = 0;
            }
        }
    }

    // Check if it's safe to place a queen at (row, col)
    static boolean isSafe(int row, int col) {
        return !rowCheck[row]
            && !slashCheck[slashCode[row][col]]
            && !backslashCheck[backslashCode[row][col]];
    }

    // Recursive utility to solve N Queens using Branch and Bound
    static boolean solveNQueens(int col) {
        if (col >= N) return true;

        for (int row = 0; row < N; row++) {
            if (isSafe(row, col)) {
                // Place the queen
                board[row][col] = 1;
                rowCheck[row] = true;
                slashCheck[slashCode[row][col]] = true;
                backslashCheck[backslashCode[row][col]] = true;

                // Recurse
                if (solveNQueens(col + 1)) return true;

                // Backtrack
```

```
                board[row][col] = 0;
                rowCheck[row] = false;
                slashCheck[slashCode[row][col]] = false;
                backslashCheck[backslashCode[row][col]] = false;
            }
        }
        return false;
    }

    // Print the solution
    static void printBoard() {
        for (int[] row : board) {
            for (int cell : row)
                System.out.print((cell == 1 ? "Q " : ". "));
            System.out.println();
        }
    }

    public static void main(String[] args) {
        initialize();
        int startCol = 0; // Try other values for multiple solutions (1 to 7)
        if (solveNQueens(startCol)) {
            printBoard();
        } else {
            System.out.println("No solution exists.");
        }
    }
}
```