NQueensProblem

|  |
| --- |
| **package** day16\_17;  **public** **final** **class** NQueensProblem {  **public** **static** **void** main(String[] args) {  **int** size = 8;  **boolean**[][] board = **new** **boolean**[size][size];  NQueensProblem nQueensProblem = **new** NQueensProblem();  **if** (!nQueensProblem.nQueen(board, size, 0)) {  System.***out***.println("No solution found :( ");  }  }  **private** **boolean** nQueen(**boolean**[][] board, **int** size, **int** row) {  **if** (row == size) {  printBoard(board, size);  **return** **true**;  } **else** {  **boolean** foundSolution = **false**;  **for** (**int** col = 0; col < size; col++) {  **if** (isValidCell(board, size, row, col)) {  board[row][col] = **true**; // Place the queen  foundSolution = nQueen(board, size, row + 1) || foundSolution; // Recursive call  board[row][col] = **false**; // Backtrack  }  }  **return** foundSolution;  }  }  **private** **boolean** isValidCell(**boolean**[][] board, **int** size, **int** row, **int** col) {  // check column  **for** (**int** i = 0; i < row; i++) {  **if** (board[i][col]) {  **return** **false**;  }  }  // check upper left diagonal  **for** (**int** i = row, j = col; i >= 0 && j >= 0; i--, j--) {  **if** (board[i][j]) {  **return** **false**;  }  }  // check upper right diagonal  **for** (**int** i = row, j = col; i >= 0 && j < size; i--, j++) {  **if** (board[i][j]) {  **return** **false**;  }  }  **return** **true**;  }  **private** **void** printBoard(**boolean**[][] board, **int** size) {  **for** (**int** i = 0; i < size; i++) {  **for** (**int** j = 0; j < size; j++) {  System.***out***.print(board[i][j] ? "Q " : "- ");  }  System.***out***.println();  }  System.***out***.println();  }  } |