```
public class SudokuSolver {

    public static void main(String[] args) {
        int[][] board = {
            {5, 3, 0, 0, 7, 0, 0, 0, 0},
            {6, 0, 0, 1, 9, 5, 0, 0, 0},
            {0, 9, 8, 0, 0, 0, 0, 6, 0},
            {8, 0, 0, 0, 6, 0, 0, 0, 3},
            {4, 0, 0, 8, 0, 3, 0, 0, 1},
            {7, 0, 0, 0, 2, 0, 0, 0, 6},
            {0, 6, 0, 0, 0, 0, 2, 8, 0},
            {0, 0, 0, 4, 1, 9, 0, 0, 5},
            {0, 0, 0, 0, 8, 0, 0, 7, 9}
        };

        if (solveSudoku(board)) {
            printBoard(board);
        } else {
            System.out.println("No solution exists");
        }
    }

    // Function to print the Sudoku board
    public static void printBoard(int[][] board) {
        for (int[] row : board) {
            for (int num : row) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }

    // Function to check if it's safe to place num in the cell (row, col)
    public static boolean isValid(int[][] board, int row, int col, int num) {
        // Check if num is not in the current row
        for (int x = 0; x < 9; x++) {
            if (board[row][x] == num) {
                return false;
            }
        }

        // Check if num is not in the current column
        for (int x = 0; x < 9; x++) {
            if (board[x][col] == num) {
                return false;
            }
        }
```

```
    // Check if num is not in the current 3x3 subgrid
    int startRow = row - row % 3;
    int startCol = col - col % 3;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[startRow + i][startCol + j] == num) {
                return false;
            }
        }
    }

    return true;
}

// Function to solve the Sudoku using backtracking
public static boolean solveSudoku(int[][] board) {
    int[] empty = findEmptyLocation(board);
    if (empty == null) {
        return true; // Puzzle solved
    }

    int row = empty[0];
    int col = empty[1];

    for (int num = 1; num <= 9; num++) {
        if (isValid(board, row, col, num)) {
            board[row][col] = num;
            if (solveSudoku(board)) {
                return true;
            }
            board[row][col] = 0; // Backtrack
        }
    }

    return false;
}

// Function to find an empty location in the Sudoku board
public static int[] findEmptyLocation(int[][] board) {
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            if (board[i][j] == 0) {
                return new int[] { i, j };
            }
        }
    }
```

```
        return null;
    }
}
```