```java
package leetcode.com.pickup1.hard;

/**
 * Created by tclresearchamerica on 6/1/16.
 * ****************************************************
 * Location:
 * https://leetcode.com/problems/sudoku-solver/
 * ****************************************************
 * Description:
 * Write a program to solve a Sudoku puzzle by filling the empty cells.
 * Empty cells are indicated by the character '.'.
 * You may assume that there will be only one unique solution.
 * A sudoku puzzle...
 * ...and its solution numbers marked in red.
 * ****************************************************
 * In hindsight:
 * 基本的逻辑没有问题,细节上面会有些瑕疵
 * 1.数字转字符:数字(0开始)+49
 * 2.字符转数字:字符('0'开始)-49
 * 3.因为数组下标写错了,所以debug进行了30分钟,完全消耗掉了,第二次做题的优势
 * 4.结束条件可以选一个全局变量,而不一定要用方法的返回值,这样代码会变得清晰许多isFound
 * 2016/09/05:
 * 1.忘记了做法,没有想到那三个数组和isFound的标志!!!-->
 * 2.忘了字符和数字间的转换
 */
public class No037_Sudoku_Solver {

    public static void main(String[] args) {
        int i = 0;
        i += 49;
        char c = (char) i;
        System.out.println(c);
        i = 1;
        System.out.println((char) (i + '0'));
    }

    boolean[][] rowVisit, colVisit, gridVisit;
    private boolean isFound = false;

    public void solveSudoku(char[][] board) {

        initial(board);
        helper(board, 0, 0);
    }

    public void initial(char[][] board) {
        rowVisit = new boolean[9][9];
        colVisit = new boolean[9][9];
        gridVisit = new boolean[9][9];
        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                if (board[i][j] != '.') {
                    int num = board[i][j] - '1';
                    int index = i / 3 * 3 + j / 3;
                    setFlg(i, j, index, num, true);
                }
            }
        }
    }
    private void helper(char[][] board, int row, int col) {
        if (row == 9) {
            isFound = true;
            return;
        }
        int index = row / 3 * 3 + col / 3;
        if ('.' == board[row][col]) {
            for (int i = 0; i < 9; i++) {
                if (rowVisit[row][i] == false && colVisit[col][i] == false &&
gridVisit[index][i] == false) {
                    setFlg(row, col, index, i, true);
                    board[row][col] = (char) (i + '1');
                    if (col == 8) {
                        helper(board, row + 1, 0);
                    } else {
                        helper(board, row, col + 1);
                    }
                    if (isFound) return;
                    board[row][col] = '.';
                    setFlg(row, col, index, i, false);
                }
            }
        } else {
            int num = board[row][col] - '1';
            setFlg(row, col, index, num, true);
            if (col == 8) {
                helper(board, row + 1, 0);
            } else {
                helper(board, row, col + 1);
            }
        }
    }
    private void setFlg(int row, int col, int index, int i, boolean flg) {
        rowVisit[row][i] = flg;
        colVisit[col][i] = flg;
        gridVisit[index][i] = flg;
    }
}
```