import org.chocosolver.solver.Model;

import org.chocosolver.solver.Solver;

import org.chocosolver.solver.variables.IntVar;

public class Sudoku {

public static void main(String[] args) {

int i, j, k;

// 1. Create a Model

Model model = new Model("my first sudoku problem");

// 2. Create variables

/\* the board which is 9 X 9 \*/

/\* (0, 0) is the top left position and (8, 8) is the bottom right position \*/

/\*each cell is an integer variable taking their value in [1, 9] \*/

IntVar[][] bd = model.intVarMatrix("bd", 9, 9, 1, 9);

/\* the nine rows \*/

/\* each row is an array of 9 integer variables taking their value in [1, 9] \*/

IntVar[] r0 = model.intVarArray("r0", 9, 1, 9);

IntVar[] r1 = model.intVarArray("r1", 9, 1, 9);

IntVar[] r2 = model.intVarArray("r2", 9, 1, 9);

IntVar[] r3 = model.intVarArray("r3", 9, 1, 9);

IntVar[] r4 = model.intVarArray("r4", 9, 1, 9);

IntVar[] r5 = model.intVarArray("r5", 9, 1, 9);

IntVar[] r6 = model.intVarArray("r6", 9, 1, 9);

IntVar[] r7 = model.intVarArray("r7", 9, 1, 9);

IntVar[] r8 = model.intVarArray("r8", 9, 1, 9);

/\* the nine columns \*/

/\* each column is an array of 9 integer variables taking their value in [1, 9] \*/

IntVar[] c0 = model.intVarArray("c0", 9, 1, 9);

IntVar[] c1 = model.intVarArray("c1", 9, 1, 9);

IntVar[] c2 = model.intVarArray("c2", 9, 1, 9);

IntVar[] c3 = model.intVarArray("c3", 9, 1, 9);

IntVar[] c4 = model.intVarArray("c4", 9, 1, 9);

IntVar[] c5 = model.intVarArray("c5", 9, 1, 9);

IntVar[] c6 = model.intVarArray("c6", 9, 1, 9);

IntVar[] c7 = model.intVarArray("c7", 9, 1, 9);

IntVar[] c8 = model.intVarArray("c8", 9, 1, 9);

/\* the nine blocks or boxes \*/

/\* each box is an array of 9 integer variables taking their value in [1, 9] \*/

IntVar[] b0 = model.intVarArray("b0", 9, 1, 9);

IntVar[] b1 = model.intVarArray("b1", 9, 1, 9);

IntVar[] b2 = model.intVarArray("b2", 9, 1, 9);

IntVar[] b3 = model.intVarArray("b3", 9, 1, 9);

IntVar[] b4 = model.intVarArray("b4", 9, 1, 9);

IntVar[] b5 = model.intVarArray("b5", 9, 1, 9);

IntVar[] b6 = model.intVarArray("b6", 9, 1, 9);

IntVar[] b7 = model.intVarArray("b7", 9, 1, 9);

IntVar[] b8 = model.intVarArray("b8", 9, 1, 9);

// 3. Post constraints

/\* post constraints for the given hints or clues \*/

model.arithm (bd[0][2], "=", 2).post();

model.arithm (bd[1][1], "=", 8).post();

model.arithm (bd[1][4], "=", 3).post();

model.arithm (bd[1][7], "=", 7).post();

model.arithm (bd[2][0], "=", 3).post();

model.arithm (bd[2][3], "=", 5).post();

model.arithm (bd[2][5], "=", 4).post();

model.arithm (bd[3][7], "=", 2).post();

model.arithm (bd[3][8], "=", 8).post();

model.arithm (bd[4][0], "=", 8).post();

model.arithm (bd[4][1], "=", 3).post();

model.arithm (bd[4][4], "=", 1).post();

model.arithm (bd[5][1], "=", 4).post();

model.arithm (bd[5][3], "=", 7).post();

model.arithm (bd[5][6], "=", 3).post();

model.arithm (bd[5][7], "=", 5).post();

model.arithm (bd[5][8], "=", 1).post();

model.arithm (bd[6][1], "=", 7).post();

model.arithm (bd[6][4], "=", 5).post();

model.arithm (bd[6][5], "=", 6).post();

model.arithm (bd[6][8], "=", 4).post();

model.arithm (bd[7][2], "=", 3).post();

model.arithm (bd[8][2], "=", 5).post();

model.arithm (bd[8][3], "=", 4).post();

model.arithm (bd[8][5], "=", 1).post();

model.arithm (bd[8][6], "=", 6).post();

/\* for the nine box variables \*/

/\* each box variable is associated with appropriate cell positions in board \*/

/\* for example, b0 [0] is equal to board [0][0] and b0 [8] is equal to board [3][3] \*/

/\* b1 [0] is equal to board [0][3] and b1 [8] is equal to board [2][5] \*/

/\* b2 [0] is equal to board [0][6] and b2 [8] is equal to board [2][8] \*/

/\* Continuing in this way, b8 [8] is equal to board [8][8] \*/

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[i][j], "=", b0[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[i][3+j], "=", b1[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[i][6+j], "=", b2[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[3+i][j], "=", b3[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[3+i][3+j], "=", b4[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[3+i][6+j], "=", b5[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[6+i][j], "=", b6[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[6+i][3+j], "=", b7[i \* 3 + j]).post();

for ( i = 0; i < 3; i++)

for ( j = 0; j < 3; j++)

model.arithm (bd[6+i][6+j], "=", b8[i \* 3 + j]).post();

/\* for the nine row variables \*/

/\* each row variable is associated with appropriate cell positions in board \*/

for ( j = 0; j < 9; j++)

model.arithm (bd[0][j], "=", r0[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[1][j], "=", r1[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[2][j], "=", r2[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[3][j], "=", r3[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[4][j], "=", r4[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[5][j], "=", r5[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[6][j], "=", r6[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[7][j], "=", r7[j]).post();

for ( j = 0; j < 9; j++)

model.arithm (bd[8][j], "=", r8[j]).post();

/\* for the nine column variables \*/

/\* each column variable is associated with appropriate cell positions in board \*/

for ( i = 0; i < 9; i++)

model.arithm (bd[i][0], "=", c0[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][1], "=", c1[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][2], "=", c2[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][3], "=", c3[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][4], "=", c4[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][5], "=", c5[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][6], "=", c6[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][7], "=", c7[i]).post();

for ( i = 0; i < 9; i++)

model.arithm (bd[i][8], "=", c8[i]).post();

/\* post global constraint alldiff for the nine rows \*/

model.allDifferent(r0).post();

model.allDifferent(r1).post();

model.allDifferent(r2).post();

model.allDifferent(r3).post();

model.allDifferent(r4).post();

model.allDifferent(r5).post();

model.allDifferent(r6).post();

model.allDifferent(r7).post();

model.allDifferent(r8).post();

/\* post global constraint alldiff for the nine columns \*/

model.allDifferent(c0).post();

model.allDifferent(c1).post();

model.allDifferent(c2).post();

model.allDifferent(c3).post();

model.allDifferent(c4).post();

model.allDifferent(c5).post();

model.allDifferent(c6).post();

model.allDifferent(c7).post();

model.allDifferent(c8).post();

/\* post global constraint alldiff for the nine boxes \*/

model.allDifferent(b0).post();

model.allDifferent(b1).post();

model.allDifferent(b2).post();

model.allDifferent(b3).post();

model.allDifferent(b4).post();

model.allDifferent(b5).post();

model.allDifferent(b6).post();

model.allDifferent(b7).post();

model.allDifferent(b8).post();

// 4. Solve the problem

Solver solver = model.getSolver();

solver.showStatistics();

solver.showSolutions();

solver.findSolution();

// 5. Print the solution

for ( i = 0; i < 9; i++){

for ( j = 0; j < 9; j++){

System.out.print(" ");

/\* get the value for the board position [i][j] for the solved board \*/

k = bd [i][j].getValue();

System.out.print(k );

}

System.out.println();

}

}

}