import java.util.Arrays;

import java.util.Random;

public class Sudoku {

public static final int[][] VALID\_BOARD\_9X9 = { { 4, 3, 5, 8, 7, 6, 1, 2, 9 }, { 8, 7, 6, 2, 1, 9, 3, 4, 5 },

{ 2, 1, 9, 4, 3, 5, 7, 8, 6 }, { 5, 2, 3, 6, 4, 7, 8, 9, 1 }, { 9, 8, 1, 5, 2, 3, 4, 6, 7 },

{ 6, 4, 7, 9, 8, 1, 2, 5, 3 }, { 7, 5, 4, 1, 6, 8, 9, 3, 2 }, { 3, 9, 2, 7, 5, 4, 6, 1, 8 },

{ 1, 6, 8, 3, 9, 2, 5, 7, 4 } };

public static final int GRID\_9X9 = 9;

public static final int GAME\_MODE\_EXPERT = 75;

public static final int GAME\_MODE\_MEDIUM = 60;

public static final int GAME\_MODE\_EASY = 50;

public static final int GAME\_MODE\_EASYER = 4;

public static final String SET\_VALUE\_9X9 = "123456789";

public static final int DEFAULT\_TOLERANCE = 5;

private int[][] puzzle;

private Random random = new Random();

private int[][] copyOf(int[][] original) {

int[][] copy = new int[original.length][];

for (int i = 0; i < original.length; i++) {

copy[i] = Arrays.copyOf(original[i], original[i].length);

}

return copy;

}

private int[][] swapRows(int[][] board, int row1, int row2) {

for (int j = 0; j < board.length; j++) {

int temp = board[row1][j];

board[row1][j] = board[row2][j];

board[row2][j] = temp;

}

return board;

}

private int[][] swapCols(int[][] board, int col1, int col2) {

for (int i = 0; i < board.length; i++) {

int temp = board[i][col1];

board[i][col1] = board[i][col2];

board[i][col2] = temp;

}

return board;

}

private int[][] swapRowsAndCols(int[][] board) {

int range = board.length == GRID\_9X9 ? 7 : 5;

int rowsInGrid = board.length == GRID\_9X9 ? 3 : 2;

int colsInGrid = 3;

for (int a = 0; a < range; a += rowsInGrid) {

int row[] = getTwoRanNum(a, rowsInGrid);

swapRows(board, row[0], row[1]);

}

for (int a = 0; a < range; a += colsInGrid) {

int[] col = getTwoRanNum(a, colsInGrid);

swapCols(board, col[0], col[1]);

}

return board;

}

private int[][] swapGrids(int[][] board) {

int firstgrid = 1 + random.nextInt(3);

int secondgrid = 1 + random.nextInt(3);

int numRowsInGrid = board.length == GRID\_9X9 ? 3 : 2;

if ((firstgrid == 1 && secondgrid == 2) || (firstgrid == 2 && secondgrid == 1)) {

for (int i = 0; i < numRowsInGrid; i++) {

swapRows(board, i, i + numRowsInGrid);

}

} else if ((firstgrid == 2 && secondgrid == 3) || (firstgrid == 3 && secondgrid == 2)) {

for (int i = numRowsInGrid; i < numRowsInGrid \* 2; i++) {

swapRows(board, i, i + numRowsInGrid);

}

} else if ((firstgrid == 1 && secondgrid == 3) || (firstgrid == 3 && secondgrid == 1)) {

for (int i = 0; i < numRowsInGrid; i++) {

swapRows(board, i, i + (numRowsInGrid \* 2));

}

}

return board;

}

private int[][] swapNums(int[][] board) {

int[] num = getTwoRanNum(1, board.length);

for (int i = 0; i < board.length; i++) {

for (int j = 0; j < board.length; j++) {

if (board[i][j] == num[0]) {

board[i][j] = num[1];

} else if (board[i][j] == num[1]) {

board[i][j] = num[0];

}

}

}

return board;

}

private int[] getTwoRanNum(int min, int tolerance) {

int a[] = new int[2];

a[0] = min + random.nextInt(tolerance);

a[1] = min + random.nextInt(tolerance);

return a;

}

private int[][] createBoard(int[][] board) {

for (int i = 0; i < 10; i++) {

swapRowsAndCols(board);

swapGrids(board);

swapNums(board);

}

return board;

}

private int[][] createPuzzle(int[][] board, int mode) {

this.puzzle = copyOf(board);

int numOfEmptyBlock = getNumberOfEmptyBlock(board, mode);

for (int i = 0; i < numOfEmptyBlock; i++) {

int[] rowcol = getTwoRanNum(0, board.length);

this.puzzle[rowcol[0]][rowcol[1]] = 0;

}

return copyOf(this.puzzle);

}

private int getNumberOfEmptyBlock(int[][] board, int mode) {

int numOfEmptyBlock = 0;

int numOfBlock = board.length \* board[0].length;

if (GAME\_MODE\_EASYER <= mode && mode <= GAME\_MODE\_EXPERT) {

numOfEmptyBlock = (int) Math.floor((mode \* numOfBlock) / 100);

} else {

numOfEmptyBlock = (int) Math.floor((GAME\_MODE\_MEDIUM \* numOfBlock) / 100);

}

int tolerance = (int) Math.floor(((numOfBlock - numOfEmptyBlock) \* 5) / 100);

numOfEmptyBlock += random.nextInt(tolerance + 1);

return numOfEmptyBlock;

}

public boolean check(int[][] board) {

boolean isCorrect = true;

int numOfRowsInGrid = 3;

final String setValues = SET\_VALUE\_9X9;

for (int i = 0; i < board.length; i++) {

String set = setValues;

for (int j = 0; j < board.length; j++) {

set = set.replace("" + board[i][j], "");

}

if (!set.isEmpty()) {

isCorrect = false;

return isCorrect;

}

}

for (int j = 0; j < board.length; j++) {

String set = setValues;

for (int i = 0; i < board.length; i++) {

set = set.replace("" + board[i][j], "");

}

if (!set.isEmpty()) {

isCorrect = false;

return isCorrect;

}

}

for (int hg = 0; hg < board.length; hg += numOfRowsInGrid) {

for (int vg = 0; vg < board[0].length; vg += 3) {

String set = setValues;

for (int i = hg; i < (hg + numOfRowsInGrid); i++) {

for (int j = vg; j < vg + 3; j++) {

set = set.replace("" + board[i][j], "");

}

}

if (!set.isEmpty()) {

isCorrect = false;

return isCorrect;

}

}

}

return isCorrect;

}

public int[][] getNewPuzzle(int grid, int gameMode) {

if (grid == GRID\_9X9) {

return createPuzzle(createBoard(VALID\_BOARD\_9X9), gameMode);

}

return createPuzzle(createBoard(VALID\_BOARD\_9X9), gameMode);

}

public int[][] resetPuzzle() {

return puzzle;

}

private void printArray(int[][] a) {

for (int i = 0; i < a.length; i++) {

for (int j = 0; j < a[i].length; j++) {

System.out.print(a[i][j] + "\t");

}

System.out.println();

}

System.out.println();

}

}