/\***Project #9: Sudoku Matrix**

Write a program that randomly prints a Sudoku board (9x9 matrix) which is divided into 3 submatrices

(3x3 each). Each sub matrix contains numbers ranging from 1 to 9.

Fill numbers into the whole board in ways so that there is no duplicate numbers on each row, each

column and each 3x3 block. Here’s an example:\*/

/\* Solution:\*/

import java.lang.\*;

public class Sudoku

{

int[] mat[];

int N;

int SRN;

int K;

Sudoku(int N, int K)

{

this.N = N;

this.K = K;

Double SRNd = Math.sqrt(N);

SRN = SRNd.intValue();

mat = new int[N][N];

}

public void fillValues()

{

fillDiagonal();

fillRemaining(0, SRN);

removeKDigits();

}

void fillDiagonal()

{

for (int i = 0; i<N; i=i+SRN)

fillBox(i, i);

}

boolean unUsedInBox(int rowStart, int colStart, int num)

{

for (int i = 0; i<SRN; i++)

for (int j = 0; j<SRN; j++)

if (mat[rowStart+i][colStart+j]==num)

return false;

return true;

}

void fillBox(int row,int col)

{

int num;

for (int i=0; i<SRN; i++)

{

for (int j=0; j<SRN; j++)

{

do

{

num = randomGenerator(N);

}

while (!unUsedInBox(row, col, num));

mat[row+i][col+j] = num;

}

}

}

int randomGenerator(int num)

{

return (int) Math.floor((Math.random()\*num+1));

}

boolean CheckIfSafe(int i,int j,int num)

{

return (unUsedInRow(i, num) &&

unUsedInCol(j, num) &&

unUsedInBox(i-i%SRN, j-j%SRN, num));

}

boolean unUsedInRow(int i,int num)

{

for (int j = 0; j<N; j++)

if (mat[i][j] == num)

return false;

return true;

}

boolean unUsedInCol(int j,int num)

{

for (int i = 0; i<N; i++)

if (mat[i][j] == num)

return false;

return true;

}

boolean fillRemaining(int i, int j)

{

if (j>=N && i<N-1)

{

i = i + 1;

j = 0;

}

if (i>=N && j>=N)

return true;

if (i < SRN)

{

if (j < SRN)

j = SRN;

}

else if (i < N-SRN)

{

if (j==(int)(i/SRN)\*SRN)

j = j + SRN;

}

else

{

if (j == N-SRN)

{

i = i + 1;

j = 0;

if (i>=N)

return true;

}

}

for (int num = 1; num<=N; num++)

{

if (CheckIfSafe(i, j, num))

{

mat[i][j] = num;

if (fillRemaining(i, j+1))

return true;

mat[i][j] = 0;

}

}

return false;

}

public void removeKDigits()

{

int count = K;

while (count != 0)

{

int cellId = randomGenerator(N\*N)-1;

int i = (cellId/N);

int j = cellId%9;

if (j != 0)

j = j - 1;

if (mat[i][j] != 0)

{

count--;

mat[i][j] = 0;

}

}

}

public void printSudoku()

{

for (int i = 0; i<N; i++)

{

for (int j = 0; j<N; j++)

System.out.print(mat[i][j] + " ");

System.out.println();

}

System.out.println();

}

public static void main(String[] args)

{

int N = 9, K = 20;

Sudoku sudoku = new Sudoku(N, K);

sudoku.fillValues();

sudoku.printSudoku();

}

}

/\*Output:

A screenshot of a computer

Description automatically generated\*/