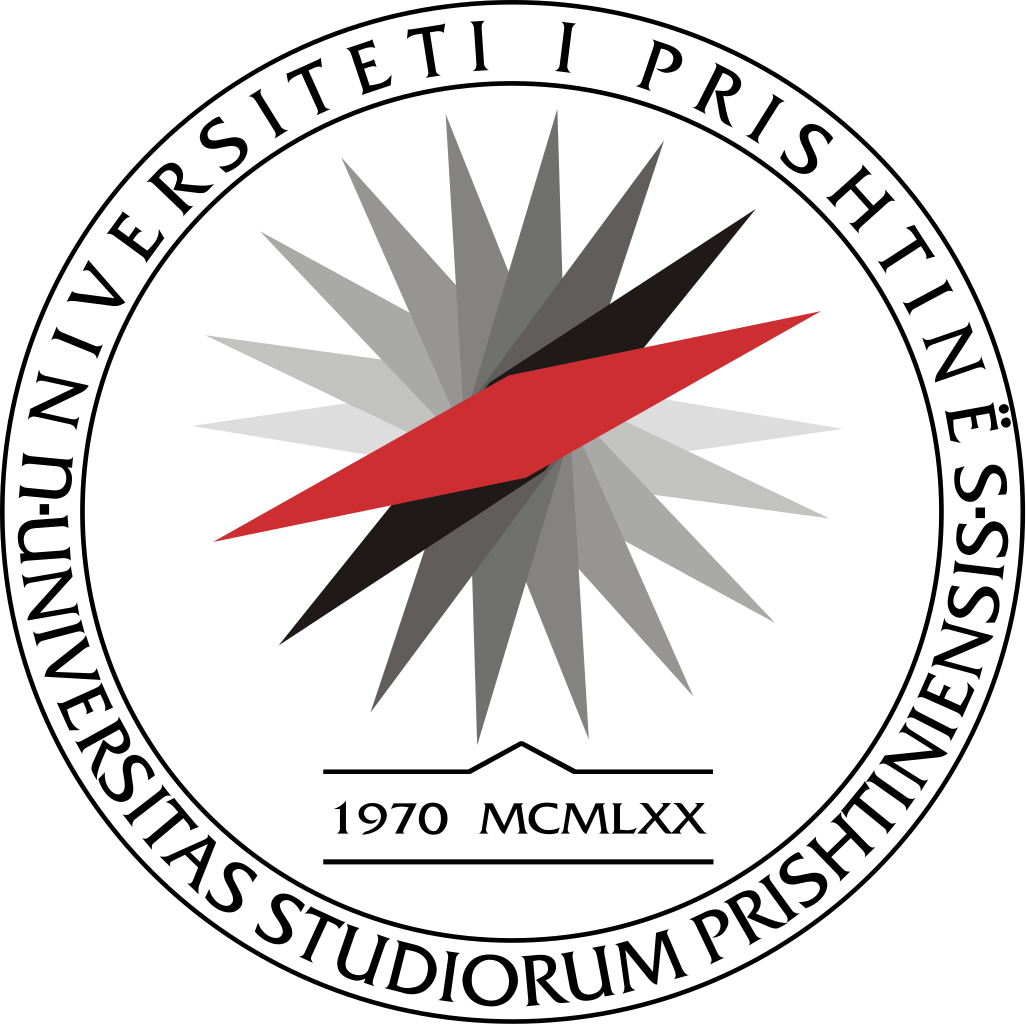
**UNIVERSITETI I PRISHTINËS**

**FAKULTETI I SHKENCAVE MATEMATIKE – NATYRORE**

**DEPARTAMENTI I MATEMATIKËS**

**PROGRAMI: Shkencë Kompjuterike**



Lënda: Siguria e te dhenave

Tema: Random Matrix Generator

Studentët:

* Lisa Ramizi

# Dokumentimi I kodit

Ky program gjeneron matrica të rastësishme dhe kryen shumëzimin e matricës. Ai përfshin komponentët e mëposhtëm:

1. **gjeneronA():**

* Gjeneron një matricë 4x4 A të mbushur me vlera të plota të rastësishme që variojnë nga 0 në 255.
* Rikthen matricën e gjeneruar A.

1. **Gjenerata e parë ():**

* Gjeneron një matricë 2x2 matricë\_1 të mbushur me vlera të plota të rastësishme që variojnë nga 0 në 255.
* Kthen matricën e krijuar të matricës\_1.

1. **Gjenerata e dytë (matrica\_1):**

* Gjeneron një matricë 3x3 matricë\_2 të mbushur me vlera të plota të rastësishme që variojnë nga 0 në 255.
* Kopjon vlerat nga matrica\_1 në pozicione specifike në matricën\_2.
* Rikthen matricën e gjeneruar matricën\_2.

1. **GjenerojMDS (matrica\_1, matrica\_2):**

* Gjeneron një MDS matricë 4x4 të mbushur me vlera të plota të rastësishme që variojnë nga 0 në 255.
* Kopjon vlerat nga matrica\_1 dhe matrica\_2 në pozicione specifike në MDS.
* Kthen matricën e gjeneruar MDS.

1. **findDeterminant(matrica):**

* Llogarit dhe kthen përcaktorin e matricës së dhënë.
* Mbështet matricat me madhësi 2x2, 3x3 dhe 4x4.

1. **shumëzon (a, b):**

* Kryen shumëzimin e matricës ndërmjet dy matricave a dhe b.
* Rikthen matricën që rezulton.

1. **main(String[] args):**

* Pika hyrëse e programit.
* Gjeneron matricën A duke përdorur generateA().
* Gjeneron matricën matricë\_1 dhe kontrollon përcaktuesin e saj. Rigjenerohet nëse përcaktorja është zero.
* Gjeneron matricën matricë\_2 bazuar në matricën\_1 dhe kontrollon përcaktuesin e saj. Rigjenerohet nëse përcaktorja është zero.
* Gjeneron matricën MDS bazuar në matricën\_1 dhe matricën\_2. Kontrollon përcaktorin dhe rigjeneron nëse është zero.
* Shumëzon matricën A me MDS dhe ia cakton rezultatin matricës B.

# Kodi

import java.util.Random;

public class RandomMatrix {

public static void main1(String[] args) {

System.***out***.println("Matrix A");

int[][] A = *generateMatrix1*(4, 4);

System.***out***.println();

int[][] matrix\_1 = *generateMatrix1*(2, 2);

if (*findDeterminant*(matrix\_1) == 0) {

matrix\_1 = *generateMatrix1*(2, 2);

}

int[][] matrix\_2 = *generateMatrixWithFixedValues*(matrix\_1);

if (*findDeterminant*(matrix\_2) == 0) {

matrix\_2 = *generateMatrixWithFixedValues*(matrix\_1);

}

System.***out***.println();

System.***out***.println("Matrix MDS");

int[][] MDS = *generateMDS*(matrix\_1, matrix\_2);

if (*findDeterminant*(MDS) == 0) {

MDS = *generateMDS*(matrix\_1, matrix\_2);

}

System.***out***.println();

System.***out***.println();

System.***out***.println("Matrix B = MDS \* A");

int[][] B = *multiplyMatrices*(MDS, A);

}

public static int[][] generateMatrix1(int rows, int columns) {

int[][] matrix = new int[rows][columns];

Random random = new Random();

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

System.***out***.println();

for (int j = 0; j < columns; j++) {

matrix[i][j] = random.nextInt(256);

System.***out***.printf("%-5d", matrix[i][j]);

}

}

return matrix;

}

public static int[][] generateMatrixWithFixedValues(int[][] matrix) {

int[][] newMatrix = new int[matrix.length + 1][matrix[0].length + 1];

Random random = new Random();

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

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

if (i == 1 && j < 2) {

newMatrix[i][j] = matrix[0][j];

} else if (i == 2 && j < 2) {

newMatrix[i][j] = matrix[1][j];

} else {

newMatrix[i][j] = random.nextInt(256);

}

System.***out***.printf("%-5d", newMatrix[i][j]);

}

}

return newMatrix;

}

public static int[][] generateMDS(int[][] matrix\_1, int[][] matrix\_2) {

int[][] matrix\_3 = new int[4][4];

Random random = new Random();

for (int i = 0; i < matrix\_3.length; i++) {

System.***out***.println();

for (int j = 0; j < matrix\_3[i].length; j++) {

if (i == 1 && j < 3) {

matrix\_3[i][j] = matrix\_2[0][j];

} else if ((i == 2 || i == 3) && j < 2) {

matrix\_3[i][j] = matrix\_1[i - 2][j];

} else {

matrix\_3[i][j] = random.nextInt(256);

}

System.***out***.printf("%-5d", matrix\_3[i][j]);

}

}

return matrix\_3;

}

public static int findDeterminant(int[][] matrix) {

int n = matrix.length;

int determinant = 0;

if (n == 2) {

determinant = matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

} else if (n == 3) {

determinant = matrix[0][0] \* (matrix[1][1] \* matrix[2][2] - matrix[1][2] \* matrix[2][1])

- matrix[0][1] \* (matrix[1][0] \* matrix[2][2] - matrix[1][2] \* matrix[2][0])

+ matrix[0][2] \* (matrix[1][0] \* matrix[2][1] - matrix[1][1] \* matrix[2][0]);

} else if (n == 4) {

determinant = matrix[0][0]

\* (matrix[1][1] \* (matrix[2][2] \* matrix[3][3] - matrix[2][3] \* matrix[3][2])

- matrix[1][2] \* (matrix[2][1] \* matrix[3][3] - matrix[2][3] \* matrix[3][1])

+ matrix[1][3] \* (matrix[2][1] \* matrix[3][2] - matrix[2][2] \* matrix[3][1]))

- matrix[0][1]

\* (matrix[1][0] \* (matrix[2][2] \* matrix[3][3] - matrix[2][3] \* matrix[3][2])

- matrix[1][2] \* (matrix[2][0] \* matrix[3][3] - matrix[2][3] \* matrix[3][0])

+ matrix[1][3] \* (matrix[2][0] \* matrix[3][2] - matrix[2][2] \* matrix[3][0]))

+ matrix[0][2]

\* (matrix[1][0] \* (matrix[2][1] \* matrix[3][3] - matrix[2][3] \* matrix[3][1])

- matrix[1][1] \* (matrix[2][0] \* matrix[3][1] - matrix[2][3] \* matrix[3][0])

+ matrix[1][3] \* (matrix[2][0] \* matrix[3][1] - matrix[2][1] \* matrix[3][0]))

- matrix[0][3]

\* (matrix[1][0] \* (matrix[2][1] \* matrix[3][2] - matrix[2][2] \* matrix[3][1])

- matrix[1][1] \* (matrix[2][0] \* matrix[3][2] - matrix[2][2] \* matrix[3][0])

+ matrix[1][2] \* (matrix[2][0] \* matrix[3][1] - matrix[2][1] \* matrix[3][0]));

}

return determinant;

}

public static int[][] multiplyMatrices(int[][] a, int[][] b) {

int m = a.length;

int n = b[0].length;

int[][] c = new int[m][n];

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

System.***out***.println();

for (int j = 0; j < n; j++) {

int sum = 0;

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

int product = a[i][k] \* b[k][j];

sum += product;

}

c[i][j] = sum;

System.***out***.printf("%-5d", c[i][j]);

}

}

return c;

}

public static void main(String[] args) {

System.***out***.println("Matrix A");

int[][] A = *generateMatrix1*(4, 4);

System.***out***.println();

int[][] matrix1 = *generateMatrix1*(2, 2);

if (*findDeterminant*(matrix1) == 0) {

matrix1 = *generateMatrix1*(2, 2);

}

int[][] matrix2 = *generateMatrix1*(3, 3);

matrix2[1][0] = matrix1[0][0];

matrix2[1][1] = matrix1[0][1];

matrix2[2][0] = matrix1[1][0];

matrix2[2][1] = matrix1[1][1];

if (*findDeterminant*(matrix2) == 0) {

matrix2 = *generateMatrix1*(3, 3);

matrix2[1][0] = matrix1[0][0];

matrix2[1][1] = matrix1[0][1];

matrix2[2][0] = matrix1[1][0];

matrix2[2][1] = matrix1[1][1];

}

System.***out***.println();

System.***out***.println("Matrix MDS");

int[][] MDS = *generateMatrix1*(4, 4);

MDS[2][0] = matrix1[0][0];

MDS[2][1] = matrix1[0][1];

MDS[3][0] = matrix1[1][0];

MDS[3][1] = matrix1[1][1];

MDS[1][0] = matrix2[0][0];

MDS[1][1] = matrix2[0][1];

MDS[1][2] = matrix2[0][2];

MDS[2][2] = matrix2[1][2];

MDS[3][2] = matrix2[2][2];

if (*findDeterminant*(MDS) == 0) {

MDS = *generateMatrix1*(4, 4);

MDS[2][0] = matrix1[0][0];

MDS[2][1] = matrix1[0][1];

MDS[3][0] = matrix1[1][0];

MDS[3][1] = matrix1[1][1];

MDS[1][0] = matrix2[0][0];

MDS[1][1] = matrix2[0][1];

MDS[1][2] = matrix2[0][2];

MDS[2][2] = matrix2[1][2];

MDS[3][2] = matrix2[2][2];

}

System.***out***.println();

System.***out***.println();

System.***out***.println("Matrix B = MDS \* A");

int[][] B = *multiplyMatrices*(A, MDS);

}

public static int[][] generateMatrix(int rows, int columns) {

int[][] matrix = new int[rows][columns];

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

System.***out***.println();

for (int j = 0; j < columns; j++) {

matrix[i][j] = (int) (Math.*random*() \* 256);

System.***out***.printf("%-5d", matrix[i][j]);

}

}

return matrix;

# Rezultati

