import java.util.Scanner;

class MatrixMultiplicationTask implements Runnable {

    private int[][] matrixA;

    private int[][] matrixB;

    private int[][] result;

    private int row;

    private int col;

    // Constructor to initialize the task

    public MatrixMultiplicationTask(int[][] matrixA, int[][] matrixB, int[][] result, int row, int col) {

        this.matrixA = matrixA;

        this.matrixB = matrixB;

        this.result = result;

        this.row = row;

        this.col = col;

    }

    @Override

    public void run() {

        int sum = 0;

        for (int k = 0; k < matrixB.length; k++) {

            sum += matrixA[row][k] \* matrixB[k][col];

        }

        result[row][col] = sum; // Store the computed value in the result matrix

    }

}

public class MultithreadedMatrixMultiplication {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Input dimensions for Matrix A and Matrix B

        System.out.print("Enter number of rows for Matrix A: ");

        int rowsA = scanner.nextInt();

        System.out.print("Enter number of columns for Matrix A (same as rows of Matrix B): ");

        int colsA = scanner.nextInt();

        System.out.print("Enter number of columns for Matrix B: ");

        int colsB = scanner.nextInt();

        // Initialize matrices

        int[][] matrixA = new int[rowsA][colsA];

        int[][] matrixB = new int[colsA][colsB];

        int[][] result = new int[rowsA][colsB];

        // Input elements of Matrix A

        System.out.println("Enter elements of Matrix A:");

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

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

                matrixA[i][j] = scanner.nextInt();

            }

        }

        // Input elements of Matrix B

        System.out.println("Enter elements of Matrix B:");

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

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

                matrixB[i][j] = scanner.nextInt();

            }

        }

        // Validate dimensions for matrix multiplication

        if (colsA != matrixB.length) {

            System.out.println("Matrix multiplication not possible: columns of Matrix A must equal rows of Matrix B.");

            return;

        }

        // Create threads for each element of the result matrix

        Thread[][] threads = new Thread[rowsA][colsB];

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

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

                threads[i][j] = new Thread(new MatrixMultiplicationTask(matrixA, matrixB, result, i, j));

                threads[i][j].start(); // Start the thread

            }

        }

        // Wait for all threads to complete

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

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

                try {

                    threads[i][j].join(); // Wait for the thread to finish

                } catch (InterruptedException e) {

                    e.printStackTrace();

                }

            }

        }

        // Display the resulting matrix

        System.out.println("Resultant Matrix after Multiplication:");

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

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

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

            }

            System.out.println();

        }

        scanner.close();

    }

}