```java
import java.util.Scanner;

public class FloydWarshall {
    private int matrix[][];
    private int numVertices;
    public static final int INFINITY = 999;

    public FloydWarshall(int vertices) {
        matrix = new int[vertices + 1][vertices + 1];
        this.numVertices = vertices;
    }

    public void floydwarshall(int mat[][]) {
        for (int s = 1; s <= numVertices; s++) {
            for (int d = 1; d <= numVertices; d++) {
                matrix[s][d] = mat[s][d];
            }
        }
        for (int i = 1; i <= numVertices; i++) { // i = intermediate
            for (int s = 1; s <= numVertices; s++) {
                for (int d = 1; d <= numVertices; d++) {
                    if (matrix[s][i] + matrix[i][d] < matrix[s][d])
                        matrix[s][d] = matrix[s][i] + matrix[i][d];
                }
            }
        }

        for (int s = 1; s <= numVertices; s++) {
            System.out.print("\t" + s);
        }
        System.out.println();
        for (int s = 1; s <= numVertices; s++) {
            System.out.print(s + "\t");
            for (int d = 1; d <= numVertices; d++) {
                System.out.print(matrix[s][d] + "\t");
            }
            System.out.println();
        }
    }

    public static void main(String... arg) {
        int matrix[][];
        int numVertices;
        Scanner scan = new Scanner(System.in);
        System.out.println("Number of vertices:");
        numVertices = scan.nextInt();
        matrix = new int[numVertices + 1][numVertices + 1];
        System.out.println("Weighted matrix");
        for (int s = 1; s <= numVertices; s++) {
            for (int d = 1; d <= numVertices; d++) {
                matrix[s][d] = scan.nextInt();
                if (s == d) {
                    matrix[s][d] = 0;
                    continue;
                }
                if (matrix[s][d] == 0) {
                    matrix[s][d] = INFINITY;
                }
            }
        }
        System.out.println("The Transitive Closure of the Graph");
        FloydWarshall floydwarshall = new FloydWarshall
(numVertices);
        floydwarshall.floydwarshall(matrix);
        scan.close();
    }
}
```