**Week 11: Write a java program to implement Floyd’s algorithm for the all pairs shortest path problem.**

import java.util.Scanner;

public class AllPairShortestPath

{

private int d[][];

private int n;

public static final int INFINITY = 999;

public AllPairShortestPath(int n)

{

d = new int[n + 1][n + 1];

this.n = n;

}

public void ShortestPath(int adjacencymatrix[][])

{

for (int i = 1; i <= n; i++)

{

for (int j= 1; j<= n; j++)

{

d[i][j] = cost[i][j];

}

}

for (int k = 1; k <= n; k++)

{

for (int i = 1; i <= n; i++)

{

for (int j= 1; j<= n; j++)

{

if (d[i][k] + d[k][j]

< d[i][j])

d[i][j] = d[i][k]

+ d[k][j];

}

}

}

for (int i = 1; i <= n; i++)

System.out.print("\t" + i);

System.out.println();

for (int i = 1; i <= n; i++)

{

System.out.print(i + "\t");

for (int j= 1; j<= n; j++)

{

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

}

System.out.println();

}

}

public static void main(String... arg)

{

int cost[][];

int n;

Scanner scan = new Scanner(System.in);

System.out.println("Enter the number of vertices");

n = scan.nextInt();

cost = new int[n + 1][n + 1];

System.out.println("Enter the Weighted Matrix for the graph");

for (int i = 1; i <= n; i++)

{

for (int j= 1; j<= n; j++)

{

cost[i][j] = scan.nextInt();

if (i == j)

{

cost[i][j] = 0;

continue;

}

if (cost[i][j] == 0)

{

cost[i][j] = INFINITY;

}

}

}

System.out.println("The Transitive Closure of the Graph");

AllPairShortestPath obj= new AllPairShortestPath(n);

obj.ShortestPath(cost);

scan.close();

}

}