**Objective**:

Read in n, and an n by n adjacency matrix representing a directed graph and Determine whether the digraph contains a directed cycle. If it contains a directed cycle, print one or if it is a directed acyclic graph, print a topological ordering.

**Procedure:**

1. Create a java program in eclipse to detect the cycle and the topological order.

2. Input the graph given by the user.

3. Check whether the graph has cycle or not.

4. If the graph has a cycle then print the cycle.

5. Else if the graph does not have a cycle, then print the topological order for the input graph.

**Technology used:** Java

**Tool used:** Eclipse

**Source Code Filename:**

TopoSortORCyclePrint.java

Graphconstruction.java

**Explanation:**

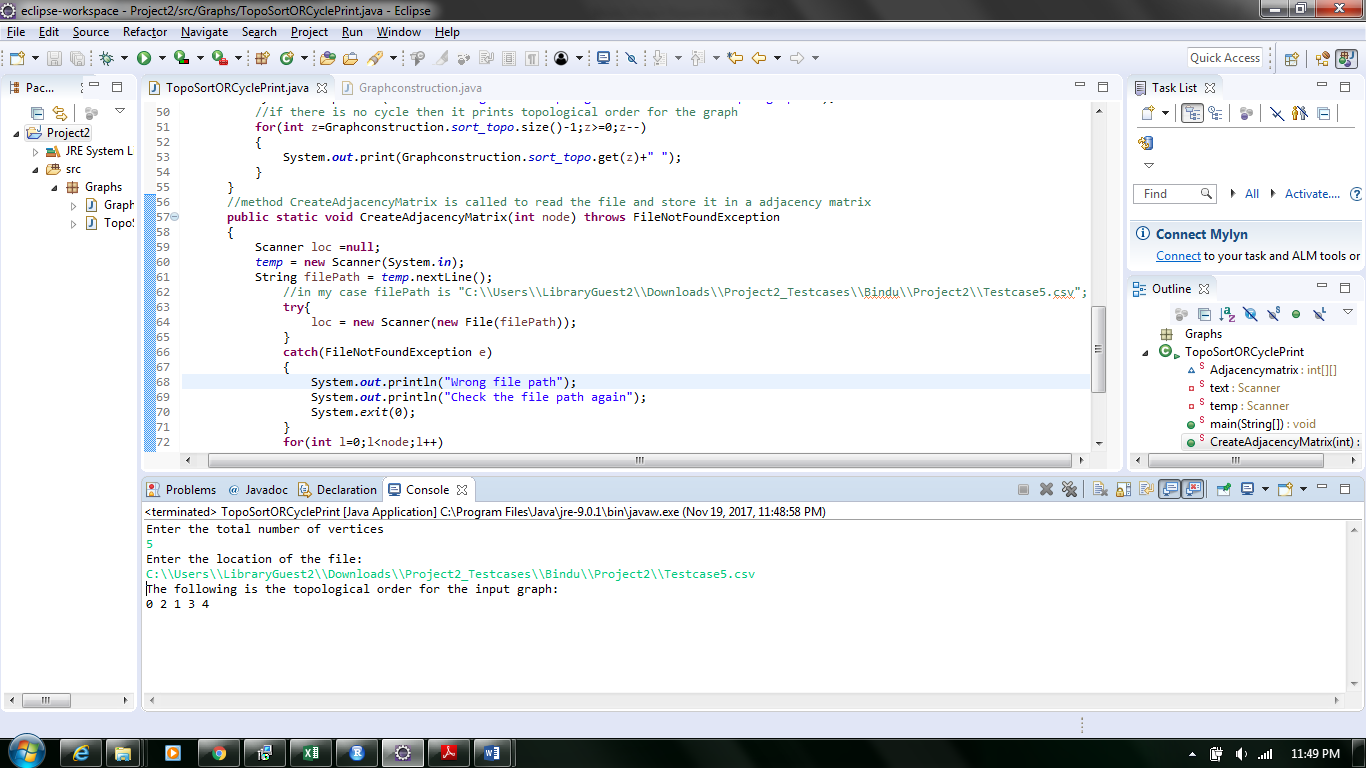
Topological ordering of a directed graph is a linear ordering of its vertices such that for every directed edge *uv* from vertex *u* to vertex *v*, *u* comes before *v* in the ordering.  A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph (DAG).

**Program Explanation:**

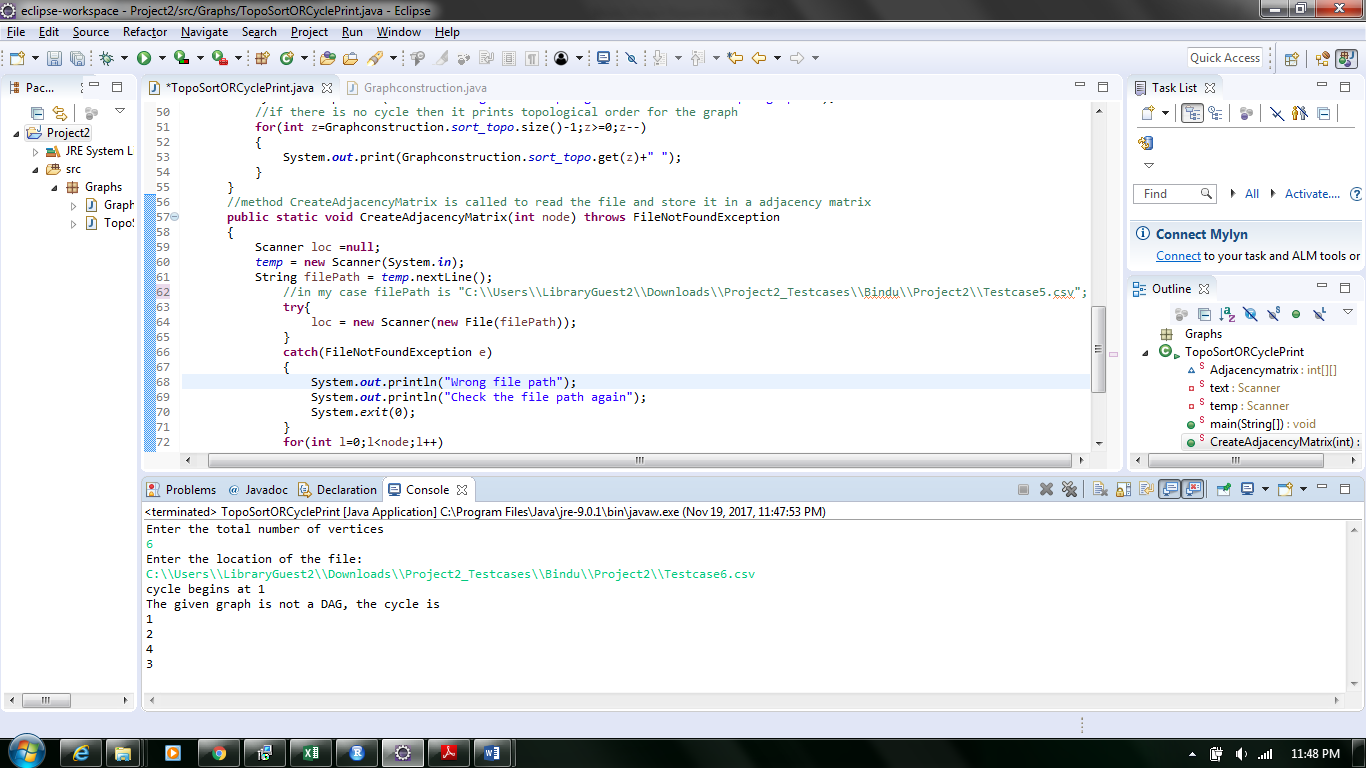
1. The program is to analyze if a given adjacency matrix has a cycle or not. If there is cycle, display that it has a cycle. If not, display the topographical order.
2. The program takes csv files as input.
3. The CreateAdjacencyMatrix method locates the file, read the file line by and line and split using the delimiter (,).
4. The values are stored in 2 dimensional array Adjacencymatrix [][].
5. The Graphconstruction creates arraylist input to store the values.
6. The check dfs is called to check whether there is a cycle or not.
7. If the node is not visited add the node to the arraylist input.
8. Mark the node as visited.
9. Check for the adjacent nodes to the current node and call check dfs.
10. Remove the nodes from the input one by one and store it in a variable var.
11. The checkinput method is called to check whether it contains cycle or not.
12. First the checkinput is called to check whether nodes forms a cycle by comparing the var and the nodes in the arraylist. If it matches then it returns true. If not it returns false.
13. If the check input method returns false, then var is added to the arraylist sort\_topo.
14. If there is a cycle then its prints the cycle using the arraylist input and prints the cycle.

**Screen Shots of output:**

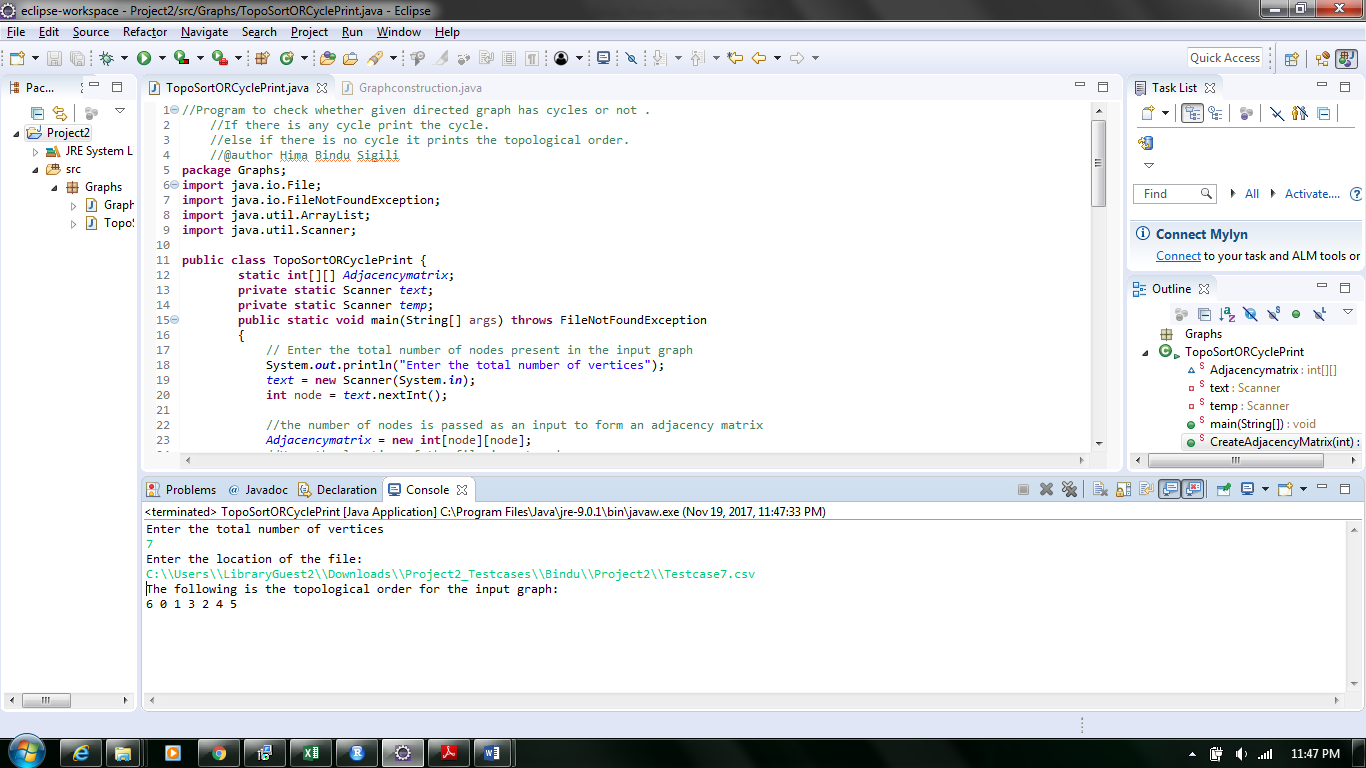
**Testcase5.CSV( Textcase1.txt file with n=5)**



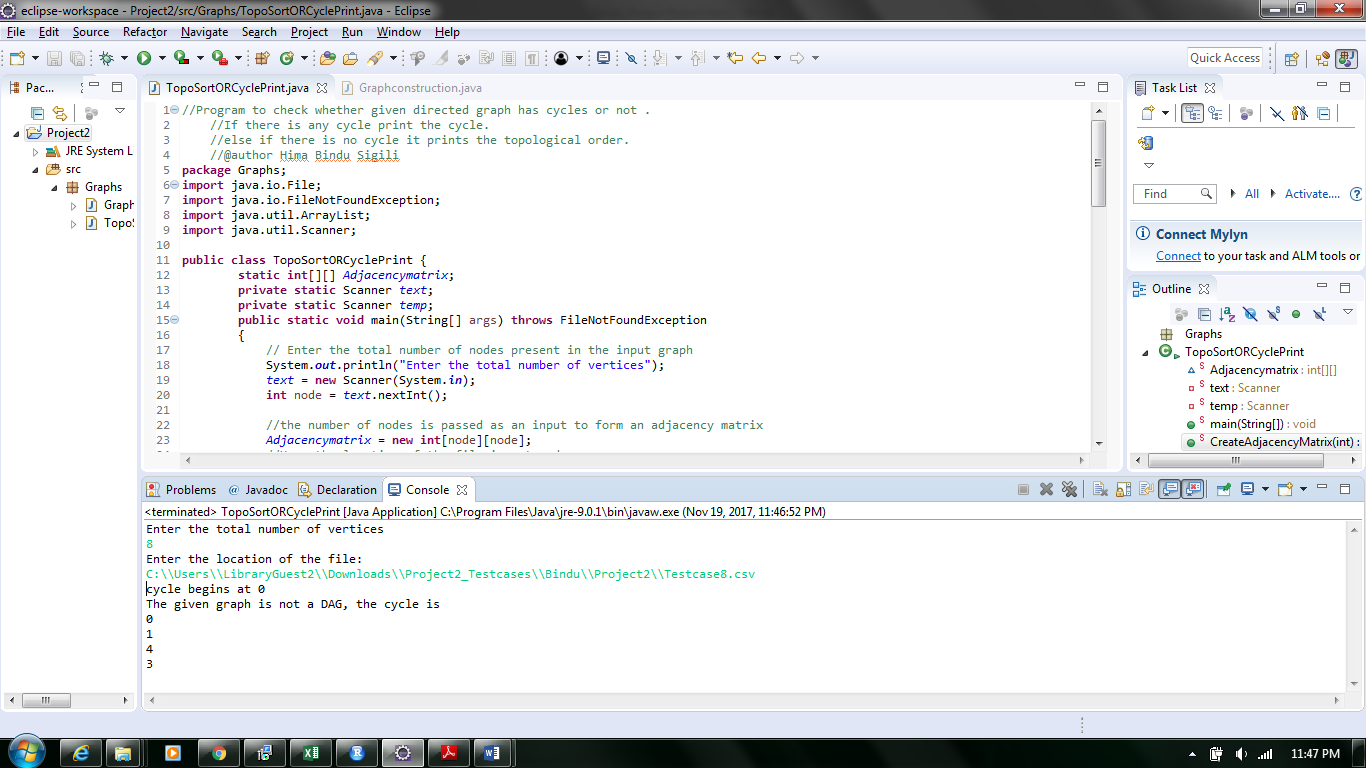
**Testcase6.CSV( Textcase1.txt file with n=6)**



**Testcase7.CSV( Textcase1.txt file with n=7)**



**Testcase8.CSV( Textcase1.txt file with n=8)**



**Time Complexities:**

The time complexity for the algorithm Ɵ (E+V). V is the number of nodes and E is the number of connections.