CS207 Consolidation week Assignment

- Consolidation week is the week commencing 10th Jan 2022. (You may undertake this assignment prior to then if you wish.) This assignment is worth 10% of the mark for the class.
- Submit your solution (in zipped form) on MyPlace by deadline **10pm Thursday 13th January 2022**. Code should not be altered after the submission deadline. The submission link will appear under Consolidation Week on MyPlace. Late submission on MyPlace of up to 7 days, with penalty, is permitted – see separate file on MyPlace under Consolidation Week for details.
- Demonstrate your work (at **individually allocated times** to be specified later) in first two weeks of semester 2 – demo sessions commence Monday 9am week 1 (17th Jan 2022).
- Demonstration is **compulsory** to obtain marks for this assignment.
- Help through zoom will be available for those who wish it during a couple of times within the Consolidation week– times will be publicised in January on MyPlace
- Help also available via the MyPlace General Forum for the class.

Task

Download and unzip file **GraphFiles.zip** from MyPlace.
You are supplied with:

- a simple interface DiGraphADT.java
- file DiGraphEdgeList.java which provides an edge list implementation of the interface
- an incomplete program Topological.java which reads in a text file, builds a graph from it and displays a topological ordering of nodes.
- a text file Edges.txt containing some data for a graph for use in the topological ordering program provided for part 3.

Part 1:

Produce an implementation of the DiGraphADT interface which uses an adjacency matrix representation. There should be a single constructor which takes the number of nodes as argument.

Part 2:

Produce an implementation of the DiGraphADT interface which uses an adjacency list representation. There should be a single constructor which takes the number of nodes as argument.

Part 3:

Complete method topologicalOrder() in class Topological, such that it implements the algorithm described on page 13 of MyPlace item 10.5. This is the version which makes use of an additional queue to aid efficiency. (Note: Queue is an interface in the Java Collections Framework and so although Queue should be used for the static type of the relevant variable, a concrete type - from the Java Collections framework - which implements this interface should be used for its dynamic type.)

Part 4:

Produce another application which takes in a digraph from a text file, as takes place in class Topological, but then determines whether the digraph is acyclic or not. You may adapt the topological ordering algorithm to determine this: if the queue becomes empty but not all nodes have been 'visited' then the graph contains a cycle.

(continued over)

Notes:

For the purposes of this exercise, you may assume that the node numbers passed to various methods are valid, i.e. you don't need to include checks or raise exceptions to deal with a node number which is out of range being passed as an argument.

Parts 1, 2 and 3 are independent of each other, so for example part 3 can be attempted without having completed earlier parts. It is recommended that you tackle part 3 prior to tackling part 4.

You should ensure that you test your work appropriately. At the time of the demonstration you will be issued with a Driver class to use when demonstrating your work. You should be able to describe how your code works.

Marking Guidelines:
Part 1 25%
Part 2 25%
Part 3 25%
Part 4 25%

I.Ross 25-11-21