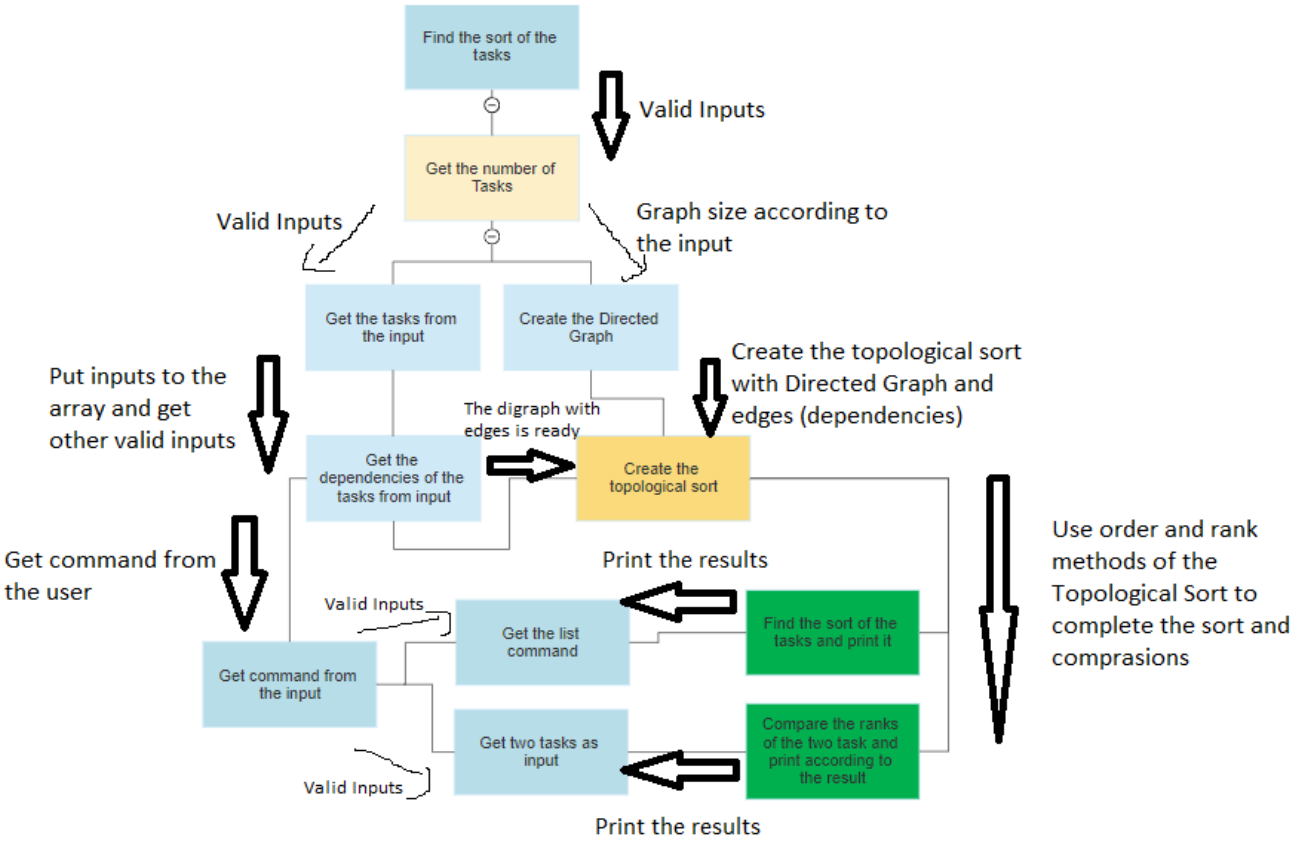
Fuat Yiğit Koçyiğit – 16429085948
CMPE343 Section 2
06.12.2021

CMPE343 Programming Homework 3 REPORT

Problem Statement and Code Design

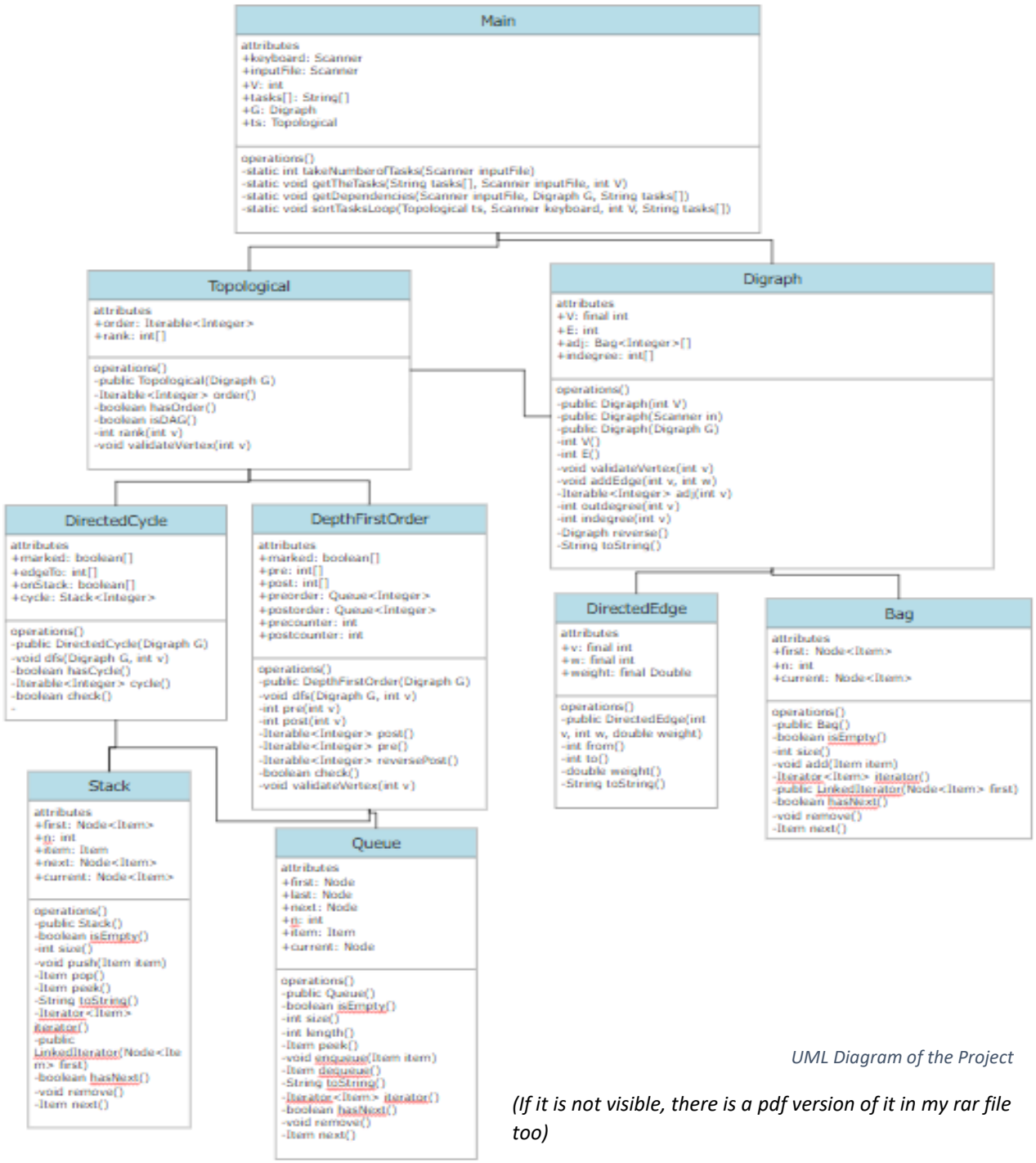
In this assignment, there are some tasks given. Some of the tasks are independent but other tasks are depending on one another, and it is needed to complete the dependent tasks on different weeks. It is expected to create an algorithm that finds what order should we do the given tasks. The program needs to calculate which tasks should be done in weeks with respecting the depended tasks.

To create the program, I used Directed Graphs to put the tasks and their dependencies in it and used Topological Sort algorithm to find the order of the tasks. The Directed Graph and Topological Sort have been implemented with the Digraph and Topological classes, respectively. Then, in the main class, the input has been taken from the input file and put to the according graph and array (to topological sort them. Arr ay is for holding the names of the tasks).



Structure Chart of the Project

(If it is not visible, there is a pdf version of it in my rar file too)



UML Diagram of the Project

(If it is not visible, there is a pdf version of it in my rar file too)

Implementation, Functionality and Performance Comparison

In the code, there is 9 different classes. 1 main class and 8 implementation classes. I implemented the topological sort and directed graph with the Topological and Digraph classes, respectively. The implementation classes have methods such as Digraph, addEdge, Topological, sort and rank. **Digraph** methods are used to create a graph to use. There are 3 different constructor method for Digraph class. One of them is designed to take the input and create the graph, other one is to create a graph in given size (V) and the last one is for copying a digraph. **addEdge** method is used to add new directed edges to the directed graph with taking 2 integer parameters (v and w). The **Topological** constructor takes a digraph as a parameter and then calculates the order according

to the topological sort and the ranks of every vertices. **Order** method of the topological sort returns the founded topological sort if the digraph is DAG. **Rank** method of the topological sort takes one integer parameter and returns it's rank as a result.

Then, I created the main class. In main class, there are 4 methods which are takeNumberofTasks, getTheTasks, getDependencies, and sortTasksLoop. **takeNumberofTasks** method takes the number of tasks from the user or input file in order to create a directed graph for it. **getTheTasks** method takes the names of every task from the user or input file and stores them to a string array for using it in the next steps. **getDependencies** gets the depended tasks from the user or input file and adds them to the directed graph with the *addEdge* method. **sortTasksLoop** starts a command loop. It includes 3 commands, 0 is exit, 1 is list and 2 is check order. *Check order* takes 2 strings from the user and compares their ranks with the rank method for printing which one is first in the order. *List command* prints the current topological sort to the user and the tasks' weeks according to the ranks of the tasks and *0* ends the program with system.exit(0).

Testing

To test my codes, I used the sample input that is given. The interface of my program and the output was: (*Figure 1, Figure 1.2 and Figure 1.3* is the screenshots of the same run)

Since there isn't a performance comparison, I did not measure the runtimes.

```
Please add the dependencies of the tasks with the task ID's:
Task 4 (phase2) is depended to 3 (phase1)
Task 5 (phase3) is depended to 4 (phase2)
Task 6 (phase4) is depended to 5 (phase3)
Task 9 (presentation) is depended to 8 (demo)
Task 8 (demo) is depended to 10 (design)
Task 5 (phase3) is depended to 7 (reporting)
Task 6 (phase4) is depended to 0 (project1)
Task 6 (phase4) is depended to 1 (project2)
Task 6 (phase4) is depended to 2 (project3)
```

Figure 1.2

```
Main (1) [Java Application] C:\Program Files\Java\jdk-14.0.2
Enter the number of tasks:
The number of tasks is 11

Please enter the task with ID 0:
Task 0 is project1
Please enter the task with ID 1:
Task 1 is project2
Please enter the task with ID 2:
Task 2 is project3
Please enter the task with ID 3:
Task 3 is phase1
Please enter the task with ID 4:
Task 4 is phase2
Please enter the task with ID 5:
Task 5 is phase3
Please enter the task with ID 6:
Task 6 is phase4
Please enter the task with ID 7:
Task 7 is reporting
Please enter the task with ID 8:
Task 8 is demo
Please enter the task with ID 9:
Task 9 is presentation
Please enter the task with ID 10:
Task 10 is design
```

Figure 1

```
Enter choice (0: Exit, 1: List schedule, 2: Check order):
2
Enter first task:
project1
Enter second task:
project2
You should do project1 before project2.
Enter choice (0: Exit, 1: List schedule, 2: Check order):
2
Enter first task:
phase2
Enter second task:
project1
You should do project1 before phase2.
Enter choice (0: Exit, 1: List schedule, 2: Check order):
2
Enter first task:
wronginput1
Enter second task:
wronginput2
The given task has not been found.
Enter choice (0: Exit, 1: List schedule, 2: Check order):
0
The program has been stopped.
```

Figure 1.3

Final Assessments

- The trouble points that I experienced in this assignment was implementing the topological sort algorithm and.
- The implementation/modifying the algorithms and creating the user interface/taking inputs correctly were the most challenging parts for me because it was hard to create a topological sort from inputs.
- I learned how to use the directed graphs and topological sort and how it can be very helpful for even in our daily lives.