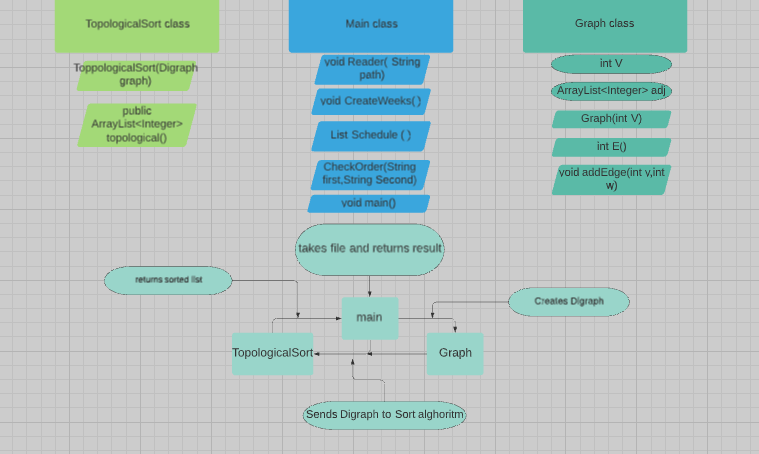
## Problem Statement and Code Design:

Problem: In this task, the problem is creating a task scheduler to schedule tasks of a software company manager. The tasks have prerequisites Therefore, we need to be careful about the order of the task while coding. Since there are only three teams only three jobs can be done in one week. Finally, the dependent tasks must be in different weeks. The first task Creating an array to keep the list of the task to call during the listing operation end of the program. The second task Creating the graph from the input file. The graph is a directed graph therefore, created graph should be a directed graph. I represented the graph with an adj Array list to keep the edges. The third task is sorting the task according to their prerequisites and splitting them into weeks. The sorting algorithm works in the following steps: compute indegree for each vertex. Pick all the edges that have indegree 0 and add them to the queue. Then remove a vertex from the queue decrease in-degree by 1 for all its neighbor nodes. if the in-degree of a neighbor node becomes zero then add it to the queue. Basically, we are starting from the outside of the graph and ranking the edges. Then according to their rank calculate an order.

## Structure chart:



## Implementation and functionality:

Main class: it is the starting point of the program. It includes scanners that read the file and take input to evaluate the result of the sort, Directed Graph object, and topological sort object.

Void Reader(String file): it takes a file as an input then reads line by line. The first line of the file represents the Directed Graph size and other Strings represent tasks and other integers represent edges. It puts Strings into a week array then creates edges into the graph with edge inputs.

Public void reader(input file){

firstline = Scanner.readline(); Digraph g = new Digraph(firstline);

for(int i =0;I<firstline;I++) { data =Scanner.realine();

data =Scanner.realine(); while(data not null) {

tasks[i]=data; } g.addEdge(data1,data2)

data =Scanner.realine();}

void CreateWeeks(): after Sorting the input data, we need to evaluate the order of the task and split them into weeks. This method exactly does that job. It fills up the 2d String array with sorted tasks. Now the head of the department is ready to work with a perfect schedule.

ListSchedule(): it prints the week's array with a for a loop. It is the method that runs after the user triggers the program by pressing 1.

listSchedule(){

Weeks[ ][ ]= tasks;

Foreach (week in Weeks) {print week;

Foreach(task in week){ print task; } } }

CheckOrder ( String first, String Second): it is a compare method that compares the places of two String in the week's array. If the first task is in an early week it says you should do the first task first. If the first task is not in an early week it says you should do the second task first. If both tasks are in the same week it says you should do both of the tasks in the week.

Main(): it calls the crucial method that the program needs while running and has a loop that takes input from the user and most of the time react according to that input.

Digraph class: It represents the directed Graph class. It includes three methods that help it to create edges and return the adj list and return the size of the graph.

TopologicalSort class: This class calculates the topological sort and returns the order of the topological sort as an array list. It takes a DAG graph as input then calculates the order of the edges.

TopologicalSort (Digraph graph): it takes a graph as input and initializes the graph input for the sort. It is a constructor method.

Public ArrayList<Integer> topological(): it sorts the edges of the given graph based on their in degrees, for instance, if the indegree is zero, it is a starting vertex.

Public ArrayList<Integer> topological(): it sorts the edges of the given graph based on their in degrees, for instance, if the indegree is zero, it is a starting vertex.

Public ArrayList<Integer> topological(){

metin içeren bir resim

Açıklama otomatik olarak oluşturulduarrayList list =Sort(graph.adj());

return list;

}

## Testing:

A testing class has been created to test the program for different inputs. The class takes a list of inputs from a text file named testinput.txt. you can see the inside of the test file on the right side. The program works as expected. In the first set of the tests, I tested whether the program gives the correct output for different weeks or not. In the second set of tests, I tested the first test with different data. In the third set of tests, I tested whether it works for the tasks on the same week or not. In the last test, I tested the tasks for different weeks which the first task is must come after the second task. Finally, I test the program if the user inputs a number bigger than two. The tests are successful. The bug was if the user input a number which is bigger than 2 and I solved it.

## Final Assessment:

The trouble in this assessment was the graph was hard adapted as a topological order because the indegree of the program was not clear enough.

The most challenging part was getting the correct result while turning the sorted array List into weeks because the tasks that have the same degree create a problem.

The assignment did not consume a huge amount of my time this time. I like that part of the assignment.