**Ex2:Directed Weighted Graphs**

**Task:**

We want to create a program that when given a json file such as G1.json (which you can find in this git project)

A gui will open on the screen that will show the graph in addition the user will have an option to load a different json graph file, save the current graph(if he made any changes) change the graph which include an option to: add a node (point), remove node, add edge, remove edge.

In addition the user will be able to run algorithms on the graph:

IsConnected: *Returns true if and only if (if) there is a valid path from each node to each  
other node. NOTE: assume directional graph*

shortestPathDist: *Computes the length of the shortest path between src to dest*

shortestPath: *Computes the the shortest path between src to dest - as an ordered List of nodes:  
src--> n1-->n2-->...dest*

Center: *Finds the NodeData which minimizes the max distance to all the other nodes.  
Assuming the graph isConnected, elese return null*

Tsp: *Computes a list of consecutive nodes which go over all the nodes in cities.  
the sum of the weights of all the consecutive (pairs) of nodes (directed) is the "cost" of the solution -  
the lower the better.*

**How the algorithm works?:**

The algorithm contain 6 classes:

**Ex2(main class)**-contain the main function that will run the program as well as 3 function-

Getgrapg-given a json file will create a MyDirectedWeightedGraph that contain all the pointed and edged contain in the json file graph

Getgrapgalgo-given a json file will create a MyDirectedWeightedGraphAlgorithm which contain a graph that have all the points and edges of the json file

Rungui-given a json file will create a gui that present a graph and allow the user to make a lot of actions on the graph as stated in TASK above  
**MyGeoLocation**-each point have a mygeolocation which will keep track of where is the point position on the graph

**MyEdgeData-**the edges of the graph each edge have a src(the starting node) dest(the node he is heading) and

Weight(the edge weight)

**Node**-the points each point have a MyGeoLocation to keep track of his location on the graph and an id

**MyDirectedWeightedGraph**- the graph which contains a hashmap<integer,node> p-the points are saved in an hashmap given the node id as a key will give the node as a value.

Hashmap<Node,Hashmap<Integer,MyEdgeData> e-the edges the key will be the starting node and the value will be hashmap<interger,MyEdgeData> the key will be the edge destination and the value will be MyEdgeData which will contain all the data regarding the edge.

**MyDirectedWeightedGraph-**the algorithms the class contain a MyDirectedWeightedGraph graph all the algorithm are going to run on the given graph

The classes are implements of the given interfaces:

GeoLovation

EdgeData

NodeData

DirectedWeightedGraph

DirectedWeightedGraphAlgorithm

**How the algorithm works?:**

Each classes in the algorithm is used as a field in other class graphalgorithm require a graph the graph require nodes and edges and nodes require a geolocation

The most important class here is MyDirectedWeightedGraph as this is the class which create a graph for us to work on without that class we wont be able to do anything this class is implemented by 2 hashmaps when the value of 1 of the given hashmaps are another hashmap the reason the class is implemented that way is to allow the used to get the most important function that will build and present the graph in O(1) (which mean almost instantly) without need to worry about every little action to take forever.

the algorithms that are presented in MyDirectedWeightedGraphAlgo are mostly known algoritms such as dijktra heres a link for more info about the algorithm:

<https://www.freecodecamp.org/news/dijkstras-shortest-path-algorithm-visual-introduction/>

Node MyEdgeData and MyGeoLocation are all very simple classes that while taking important part in the process are relatively easy to understand and thus the last class I will like to discuss will be Ex2:

Ex2 contain the function to create the gui a json file is being sent to getgrapg will create a graph given the info in the json file then the graph will be sent to getgraphalgo which will simply add the graph to a new graphalgo so algorithm could be performed on the graph and in the end the graphalgo will be sent to rungui to make all the information visible on the screen for the simple user to enjoy.

**How to run?:**

Running the program is extremely simple all that need to be done is to active the jar file in the command line using the command "java -jar Ex2.jar" then a request to a path to your json graph file will be requested after such a file will be provided a gui interface will pop on the screen and the user will be able to use the program freely the gui itself is also very simple to understand in the left upper side of the program there is a menu option with all the function that have mentioned in TASK for the used to use.

**Performance**

The program is able to present up to 100000(and possibly more) in a matter of seconds

**Important for the checker of this task!! Read before checking the program:**

Unfortunately I wasn’t able to complete everything in time the functions:

Save,center,tsp will not work