MODULE 1 (B17)

Route map navigation using Dijkstra’s algorithm

[ASSIGNED TO : P.GUNA VARDHAN, CB.EN.U4AIE22145]

# Description of the module :

In this module, We are going to implement a program based on “Route Map Navigation ”in JAVA, using the concepts of “Dijkstra’s algorithm”. The user will select the starting and ending points of his/her route through numbers assigned to a pre built set of locations. There will be a table displayed in the terminal which shows the numbers corresponding to the pre built locations. When, starting and ending points of the route are given by user, The program evaluates the shortest path among the given two user inputted points and displays the output.

If the user have to go from A to D, There is no point in the program if the output directly displays “Go to D from A”. So, the code I built in such a way that, If there are other nodes such as B,C in the shortest path between A to D, The program displays, “Go to A. From there, Go to B. From there, Go to C. From there, Go to D.” So that, the route is indirectly being said which helps in navigating the user between the initial and end points.

The program relies on classes to function properly. The classes in the program are as follows :

## DijkstraAlgorithm :

This is the main part of the code. In short, In this class the working of Dijkstra algorithm is implemented in java. It has few functions, methods and constructors, Which are going to be explained in detail in the ‘Appendix’ section of this report.

## InstructionsForUser :

As the working of this code requires the user to be aware of which location/ landmark is assigned to which number, This class creates a table with 29 rows and 2 columns, and displays the location table with the help of the next class, which is inherited from this class.

## GetLandMarkName :

This class has only one method, Which takes an integer as argument and returns the real time in-campus location name corresponding to that number, The working of this and the previous class basically rely on each other.

## DisjkstraShortestPath :

This is the main class, in which the driver class is also present. In this we make user of all the three above classes. We first display the location table so that the user knows what numbers t enter. Then we create a huge tree such that each node in the tree is assumed as each location which corresponds to one node and one number among the 29 nodes hence the total number of pre assumed locations are 29. Also, the user inputs (starting and ending point locations) are given.

Then we will call few methods in the class 1 (DijkstraAlgorithm) to find the shortest path between given two points. We use the GetLandMarkName class to display the output in a user friendly manner.

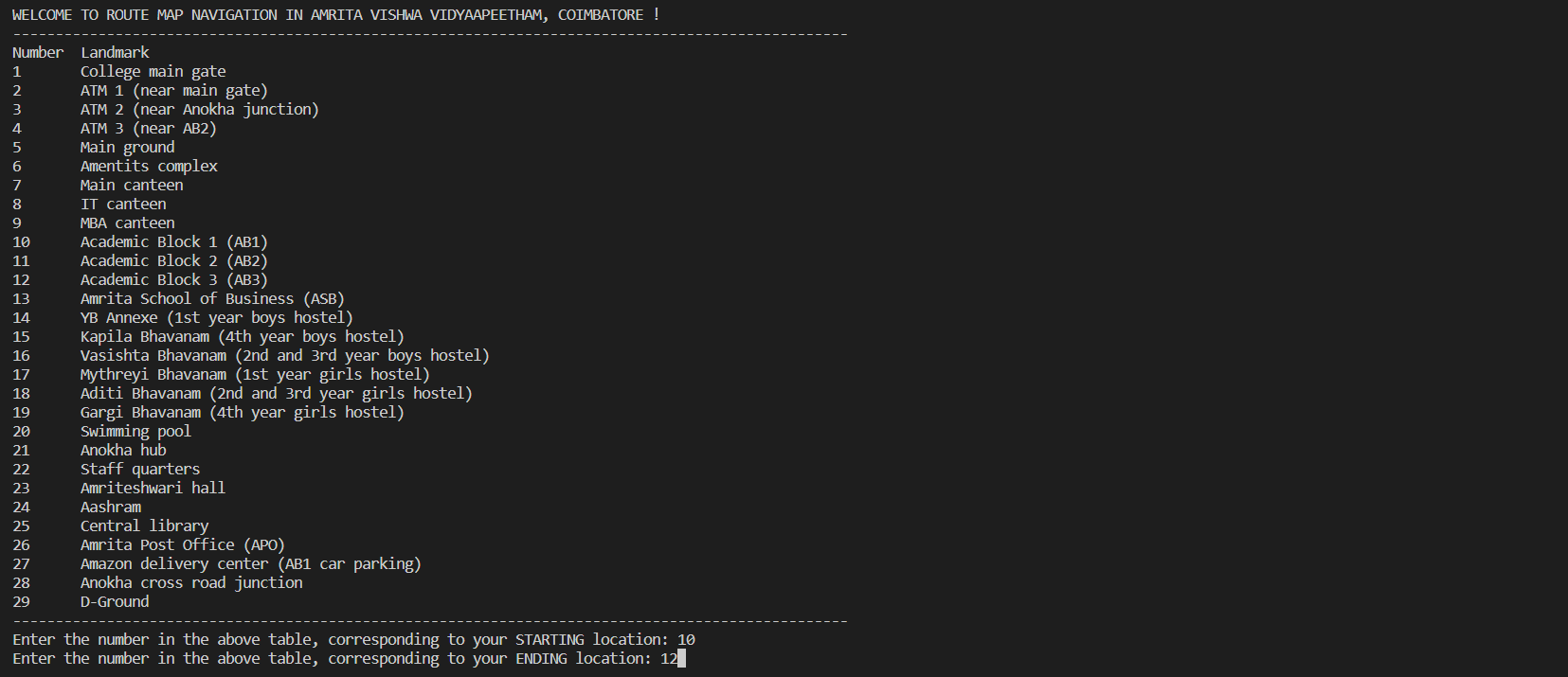
# Motivation for the module :

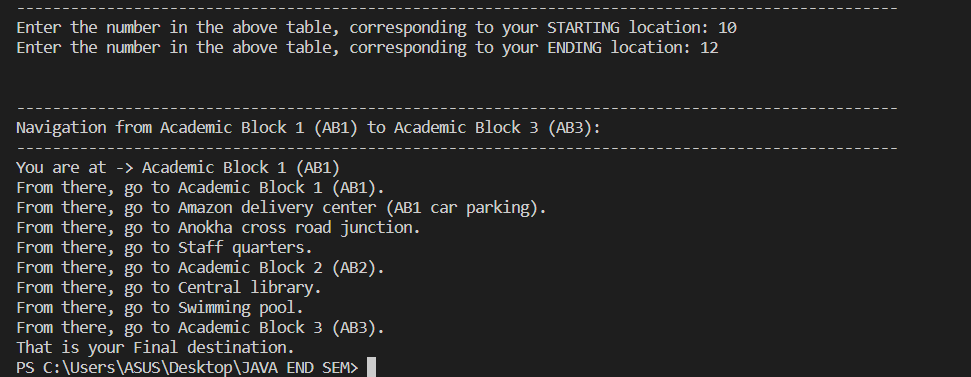
The motivation behind developing the "Route Map Navigation using Dijkstra's Algorithm" module is to provide users with a convenient and efficient way to navigate within a specified campus or location. This module aims to simplify the process of finding the shortest path between two points by utilizing the Dijkstra's algorithm.The main motivation for this module is to address the common problem of users getting lost or taking longer routes when navigating through complex routes or large locations. By implementing the Dijkstra's algorithm, we can calculate the shortest path between the user's starting and ending points, considering the weights (distances) associated with different paths.The module also provides a user-friendly interface by displaying a location table with assigned numbers, allowing users to easily select their desired starting and ending points. Additionally, the module incorporates the functionality to provide step-by-step instructions for the user, guiding them through the intermediate points on the shortest path.By implementing this module, users can benefit from efficient and accurate route planning within the campus or location, saving time and effort in navigating from one point to another. It enhances the overall user experience and reduces the chances of getting lost, especially in large and unfamiliar environments.Overall, the motivation for this module is to simplify navigation, provide accurate route planning, and enhance the user experience by utilizing the Dijkstra's algorithm for finding the shortest path in a route map navigation system.

# Relevance of the module in entire system :

This module is itself a complete program. It doesn’t rely on the modules done by other teammates. From start to end, Everything that requires to show the output with required conditions discusses above are implemented in this module alone. Other modules of the whole project just does the same thing using different algorithms. So, they are not interlinked with this module.

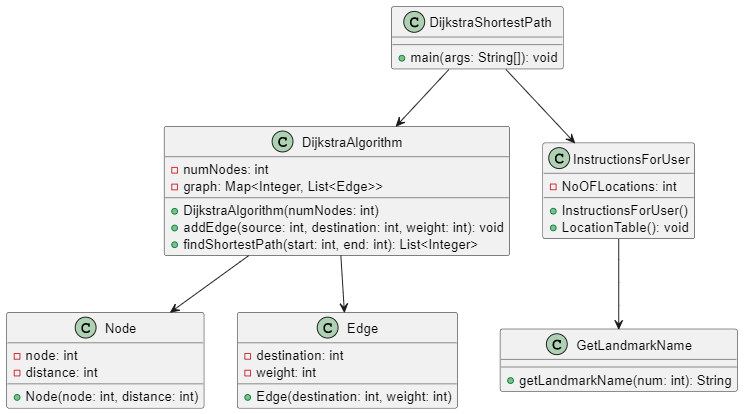
# Input and Output :





# Appendix :

## UML Diagram :



## Breakdown of methods in all the classes :

### DijkstraAlgorithm -> DijkstraAlgorithm :

This is a constructor which creates the graph (tree) on which we are going to apply the algorithm, every time an object of this class is created .

### DijkstraAlgorithm -> addEdge :

Using this method, We can add nodes to the tree. We will be using a total of 29 nodes and 36 connections between them in this navigation considering the landmarks in our campus. We have to specify the number corresponding to the starting and ending nodes of the road, and the weight (Which is calculated based on the real time distance [scale : 1 unit = 25 metres]), Based on which we can create nodes as required in the contrast of the university map.

### DijkstraAlgorithm -> findShortestPath :

This is the main method, Which takes the numbers corresponding to the starting and ending locations and gives the shortest path between them using the logic behind Dijkstra’s algorithm. (More about the algorithm in ppt and during presentation)

### DijkstraAlgorithm -> Node -> Node :

Used to create a data type called node, Which implements the functionality of nodes (landmarks) in the tree.

### DijkstraAlgorithm -> Edge -> Edge :

Used to create a data type called edge, Which implements the functionality of edges (roads) in the tree.

### InstructionsForUser -> InstructionsForUser :

This is a constructor used to print the welcome message every time an object is created for this class. It also pauses the whole compilation process for 1 second so that the user experience will be more immersive.

### InstructionsForUser -> LocationTable :

This method gets the real time locations mapped with the temporary numbers and helps in displaying them in form of a table so that the user gets to know what numbers to enter for getting the path between his/her desired locations.

### GetLandMarkName -> getLandmarkName :

This method has the huge switch case with 29 conditions which basically maps between the numbers assigned to nodes and the locations corresponding to those numbers.

### DjsktraShortesPath -> main function :

In this main function, we organize everything as discussed above, To justify the working mechanism on which the whole code relies.

MODULE PRESENTED BY : P.GUNA VARDHAN [CB.EN.U4AIE22145]