Ann-Vanessa Lartey

60932024

Intermediate Computer Programming

Individual Project 1

David Ebo Adjepon-Yamoah

30th September 2022

The problem was to write a programme which outputs a series of flights that takes a passenger from a start city to a destination city, given a start city and destination city. The input of this program is a text file, and so is the program's output.

I started by putting down the multiple algorithms that will assist me in creating the classes needed. I decided to use the breadth-first search algorithm to search through the cities. This will search through the states or cities and find the shortest path from the start state to the goal state. After research, I discovered that breadth-first search is more optimal than depth-first search, hence my decision to use the breadth-first search algorithm.

I first created an Airports class containing the constructors representing each column in the airports cvs file. I also created an Airlines class containing the constructors representing each of the columns in the airlines cvs file. I then created a Routes class containing the constructors representing each of the columns in the routes cvs file. I created a class called readFiles.java that reads each of the csv files and inputs all the elements into a hashmap so they can be easily accessed. While reading the files, I made each reader method split the file based on a comma and place the elements into an ArrayList. To find the routes, I created an ArrayList of strings, destination, in my readRouteFile method, which represents all the airport IDs of the destinations. I created a hashmap start\_destMap that represents the start and destination airport IDs. In this hashmap, the airport ID in the initial airport file created is the key, and the destination ArrayList is the value of the hashmap.

From this project, I learnt that object-oriented programming is a lot to deal with. A simple search algorithm requires a lot of time and effort to ensure that the program works correctly and efficiently with the best optimality possible.

References

stackoverflow.com

Javatpoint.com

w3schools.com