I created eight classes, one for each of the data files (Airport, Airlines and Routes), a Problem class, Node class, ReadFile class, RouteFinding class and a Main class. The data file classes each create objects of the data files, i.e., Airport, Route and Airline objects, with getters for each of the object attributes as defined by the columns in the files. The Problem class represents the route-finding problem to be solved, and its constructor takes in the starting and destination states as Strings. It has a method to check if a state is the destination and another method to generate all the airports that are reachable from a current state. The Node class initializes a node object with an Airport state, a parent node, a Route object to represent the action that was taken to get to the state and an initial path cost of zero which increases with every action taken. In this context, the path cost represents the number of flights taken to get to the destination, which is the optimization criteria for this program. The Node class has a method which generates the final solution path of actions taken to arrive at the destination. The ReadFile class essentially reads from each of the data files and assigns the values to respective hashmaps to allow easier accessing of the objects. It also has a method to read the input file and it stores the data in the file into source and destination String variables to be further used by the Problem class as starting and destination states. The RouteFinding class implements a breadth first algorithm on a Problem object passed into the method and returns the solution path. It also has a method which uses the solution path returned by the breadth first search method to write the path results to a file. The class also has a method to retrieve the file name from an absolute path where necessary, to aid in the naming of the corresponding output file. The Main class consists of a main method that runs the entire program.

Due to my extensive use of the hashmap data structure, I have become more familiar with its use and contexts where it would be the most best suited data structure to use. I have also been able to practice reading and writing from large data files and discovered that in some cases, extra care must be taken to deal with data that may have discrepancies or other features that may negatively affect the way one reads for them. For instance, a case where each line is being split on a comma, but some commas occur in unwanted fields. I have also learned how to adapt the breadth first search algorithm from a more lenient Python version to a Java version which requires that I modify the code to suit the different data types.

References

*This function is used to get the last element on a file path. - Java File Path IO*. (n.d.). Retrieved September 29, 2022, from http://www.java2s.com/example/java/file-path-io/this-function-is-used-to-get-the-last-element-on-a-file-path.html