The classes I use

-**parse(String data)** is used to split data input as ArrayList by comma

-**parseCities()** is used to read from **roads.csv** and extract unique city names by putting them into a set. I use **uniqueCities** to store city names. And put **city pair** and **distance** into an **ArrayList<String[]>** called **cities**.

-**parseAttraction()** reads from **attractions.csv** and put attraction and location into **attractionsHashtable** because access hash table is fast.

-**graph()** is used to store read from **uniqueCities** set and store every city as a **Vertex** into a **ArrrayList<Vertex>** called **v**. Then, read each item in **cities** and find matched city vertex in **v.**

Finally, **addNeighbors** to each Vertex in **v.**

-Find() is used to find matched city name in **v,** once it finds the value, return index.

-**menu()** is used to gain user input. It uses **validCity()** to check if the city entered by user is in uniqueCities set. Also, when user input **attractionPlace**, the function will check if attractionPlace is contained in **attractionsHashtable**. If there is a match in **attractionsHashtable**, add city name to **attractionCities.**

-route() get **String start, String end and ArrayList<String> attractions**, it will use **compute()** to find shortest path like this

graph() //initialize ArrayList<Vertex> v

Compute(startCity,attraction city1)

v.clear()//reset v

graph()

Compute(attraction city1,attraction city2)

v.clear()

graph()

Compute(attraction city3,attraction city4)

v.clear()

…

Compute(attraction cityn, endingCity)

It will loop through **attractions** and record previous city name as **prev.**

Each time it finishes compute, it will reset **this.Predecessor** in **Vertex**

Therefore, the time complexity would be O((n+2)* V^2 lg(V))

Because n is number of cities in attractions, adding starting and ending cities, therefore, n+2

-**dpq()** implementation of Dijkstra Shortest Path

-**trace()** trace vertex through **getPredecessor()** put in ArrayList **path**