Project Design:

Loading Data: attractions.csv ⟶ HashMap<String, String> attractions
road.csv ⟶ ArrayList<Road> (POJO: Road.java)

Build CityList: RoadList<Road> ⟶ CityList<String>

User Input: String starting_city.
String ending_city
ArrayList<String> attractions

Find the shortest path: Using the Dijkstra Algorithm to find the starting_city to all other attractions.
then choose an attraction which is the closest one to starting_city as a new starting_city.
⋮
Applying the algorithm again and again until all attractions have been chosen.
Finally, calculate the last attraction to ending_city, and find the shortest path.

UML:

RoadTrip
+ totalCost: Integer
+ attractions: HashMap<String, String>
+ roadsList: ArrayList<Road>
+ cityList: ArrayList<String>
- nodes: List<Vertex>
- edges: List<Edges>
- loadingData( )
- buildCityList( )
+ route(starting_city: String, ending_city: String, attractions: List<String>): List<Vertex>
- addLane(laneId: String, sourceLocNo: int, destLocNo: int, weight: int)
- printPath(List<Vertex>, path)

Running Time of route( ): Loop for add nodes: $O(|V|)$
Loop for add edges: $O(|E|)$
Loop for running algorithm with size of attractions times ($a$): $O(a)$
Inside this loop, running the dijkstra algorithm: $O(|V|^2)$

Whole running time: $O(|V|) + O(|E|) + O(a|V|^2) \longrightarrow O(a|V|^2)$ ($a$ = size of attractions)