My project: (woo!)

3 classes:

Iroadtrip, WorldGraph, NodeDistance, nameChanger

IRoadTrip:

Contains main and all those other methods in the skeleton.

The constructor builds 2 hashmaps, corresponding to capDist.csv and state_name.tsv

The state_name one pairs are the name of the country and its country code

The capDist one pairs two country codes and the distance between their capitals

We then run through borders.txt to build the graph of the world, a new WorldGraph.

getDistance() checks whether the two countries inputted are adjacent, if not, we return -1

findPath() runs the djikstra method in WorldGraph to get a hashmap of all the finalized vertices. Then it finds the path backwards from the destination country and puts it in an arrayList, formatting the output correctly on the way. If we don't find a path between the two countries, we output an empty list

capitalize() – the only method I added to IRoadTrip, just capitalizes the first letter of each word in a string

acceptUserInput() – pretty simple, gets correct input from the user, exits if they type EXIT. If the input is correct, prints the path

WorldGraph – graph class for use in dijkstra's. the graph is a hashmap that stores a String and another hashmap that stores a String and an Integer. The first string is the source country and the 2nd is the destination country. The integer is the weight of the edge.

Methods:

addEdge() – adds edge to the graph, input is two strings and the distance between them. If the second string is empty, we just add an unconnected node to the graph.

areAdjacent() – checks if two countries share an edge together

edgeWeight() – gets the edge weight of an edge between two adjacent countries

containsCountry() – returns true if the graph contains the country, false if otherwise

Dijkstra() – runs dijkstra's algorithm on the graph, source country and destination country are passed through as parameters. Returns a hashmap with all of the finalized vertices. Returns when the vertex of the destination country is finalized. If we don't find a path, we return an empty hashmap.

NodeDistance – class for use in the Dijkstra method of WorldGraph. Contains 3 class variables – name of vertex, its distance from the source, and the vertex it came from. Class implements comparable so we can put it into a heap.

NameChanger – a class made for correcting the differences between countries in the 3 data files.

Contains 1 method –

Alias() – takes in a string and outputs the standardized country name. to standardize the names the method:

- makes the string lowercase
- remove leading and trailing whitespace
- removes everything in parentheses
- removes a slash and everything to the right of it
- removes a comma and moves everything before a comma to the end of the string
- removes “the”
- if the country is one that doesn’t fit the standard correction(such as Swaziland/Eswatini) , the method check it against an array of hard-coded edge cases to correct it
- if the country is one that we want to specifically exclude(such as Greenland), the method returns an empty string