/\* Name: Om Kanwar

\* Date: 05/01/18

\* Class: COMP285 Section 2

\* Assignment: This class initializes the constructor for the City and

\* helps implement the cityPath class.

\*

\*/

public class City {

private String name;

private boolean visited;

public City(String name) {

this.name = name;

visited = false;

}

void hasVisited() {

this.visited = true;

}

public boolean visit() {

return this.visited;

}

String getName() {

return this.name;

}

}

\* Name: Om Kanwar

\* Date: 05/01/18

\* Class: COMP285 Section 2

\* Assignment: Assignment #3

\* Summary: This file navigates the paths of the origin city entered to the

\* adjacent cities.

\*/

public class cityPath{

private City sourceCity;

private City[] destCity = new City[10];

int numCities = 0;

public cityPath(City sourceCity) {

this.sourceCity = sourceCity;

this.sourceCity.hasVisited();

}

public void destCity(City destCity) {

this.destCity[numCities] = destCity;

this.numCities++;

}

public City getSourceCity() {

return this.sourceCity;

}

/\*

public void printAllDest() {

for(int i = 0; i < numCities; i++) {

System.out.println(destCity[i].getName());

}

}

\*/

public City nextCity() {

for (int i = 0; i < numCities; i++) {

if(destCity[i].visit() == false) {

destCity[i].hasVisited();

return destCity[i];

}

}

return null;

}

}

/\*

\* Name: Om Kanwar

\* Date: 05/01/18

\* Class: COMP285 Section 2

\* Assignment: USAir Flight Path Program

\* Summary: This is the main driver class of the program. This file implements

\* the files cityPath and City and uses the functions and the variables

\* written within those files to find a path from a user inputted source city

\* to a user inputted destination city.

\*/

import java.io.\*;

import java.util.\*;

public class Driver {

public static void main(String[] args) {

//Initialize array lists and hashmap to take in data from files

ArrayList<String> cities = getCities();

ArrayList<String> cityNames = getFlights();

HashMap<String, String> hash = createHashMap();

while(true) {

//simple functions to get the source city and the destination city

String startingCity = getStartingCity(cities);

String destinationCity = getDestCity(cities);

//creates a deque data structure as instance of the cityPath class

Deque<cityPath> stack = new ArrayDeque<cityPath>();

// Push the starting city

stack.push(searchCity(startingCity, cityNames));

// Sets the starting city to be hasVisited

stack.peek().getSourceCity().hasVisited();

String value = "";

try {

while (true) {

// Break from while loop if city at top of stack has path to

// destination

value = hash.get(stack.peek().getSourceCity().getName() + "," + destinationCity);

if (value != null) {

break;

}

// Next unvisited city

City nextUnvisited = stack.peek().nextCity();

// Pop if all cities have been visited

if (nextUnvisited == null) {

stack.pop();

}

// Push next unvisited city on the stack if not popped

else {

stack.push(searchCity(nextUnvisited.getName(), cityNames));

stack.peek().getSourceCity().hasVisited();

}

}

} catch (EmptyStackException e) {

System.out.println("Sorry, USAir doesn't fly from " + startingCity + " to " + destinationCity + "\n");

}

catch (NullPointerException e) {

// Print out error message if origin city has no flights sprouting

// from it

System.out.println("Sorry, there are no flights from this origin city to this destination city");

continue;

}

//outline route requested

String requestedRoute = "The requested route is to fly from " + startingCity + " to " + destinationCity + ".";

System.out.println(requestedRoute);

// Create new arraylist to hold organize flight paths using stack

ArrayList<String> flightPath = new ArrayList<>();

//used to get each city on the flight path

while (stack.size() > 0) {

flightPath.add(stack.removeLast().getSourceCity().getName());

}

String stuff = "";

//total cost of flight path

int totalCost = 0;

// iterate through flightPaths to get costs of flights

for (int i = 0; i < flightPath.size()-1; i++) {

String cost = hash.get(flightPath.get(i) + "," + flightPath.get(i+1));

totalCost += Integer.parseInt(cost);

System.out.println(flightPath.get(i) + " to " + flightPath.get(i+1) + " $" + cost);

}

// variable to hold final cost of flight path

String finalCost = hash.get(flightPath.get(flightPath.size()-1) + "," + destinationCity);

totalCost += Integer.parseInt(finalCost);

String finalPath = flightPath.get(flightPath.size()-1) + " to " + destinationCity + " $" + finalCost;

System.out.println(finalPath);

System.out.println("Total cost: $" + totalCost);

}

}

// function loads in cities from text file

public static ArrayList<String> getCities() {

ArrayList<String> cities = new ArrayList<>();

try {

Scanner sc = new Scanner(new File("cities.txt"));

String city = "";

while (sc.hasNextLine()) {

cities.add(sc.nextLine());

}

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("cities.txt file not found, please make sure cities.txt file is within scope of directory and try again.");

System.exit(1);

}

return cities;

}

// function loads in flight paths and costs from text file

public static ArrayList<String> getFlights() {

ArrayList<String> cityNames = new ArrayList<String>();

Scanner sc = null;

try {

sc = new Scanner(new File("flights.txt"));

} catch (FileNotFoundException e) {

e.printStackTrace();

System.out.println("flight.txt file not found, please make sure the correctly named file is wihtin the scope of the directory and try again.");

System.exit(1);

}

String[] flightLines;

while (sc.hasNextLine()) {

flightLines = sc.nextLine().split(",");

cityNames.add(flightLines[0].trim() + "," + flightLines[1].trim());

}

return cityNames;

}

// function updates current city to be checked for flight path

public static cityPath searchCity(String nextUnvisited, ArrayList <String> cityNames) {

cityPath currentCity = new cityPath(new City(nextUnvisited));

for (int i = 0; i < cityNames.size(); i++) {

String[] line = cityNames.get(i).split(",");

if (nextUnvisited.equals(line[0].trim())) {

currentCity.destCity(new City(line[1].trim()));

}

}

return currentCity;

}

// function asks user for input for origin city and returns origin city

public static String getStartingCity(ArrayList<String> cities) {

Scanner in = new Scanner(System.in);

String startingCity = "";

boolean hasCity = false;

do {

System.out.println("Please enter an origin city:");

startingCity = in.nextLine();

for (int i = 0; i < cities.size(); i++) {

if (startingCity.equals(cities.get(i))) {

return startingCity;

}

}

System.out.println("USAir doesn't serve this city. Try again");

} while (!hasCity);

return null;

}

// function asks user for destination city and returns destination city

public static String getDestCity(ArrayList<String> cities) {

Scanner in = new Scanner(System.in);

String startingCity = "";

boolean check = false;

do {

System.out.println("Please enter a destination city:");

startingCity = in.nextLine();

for (int i = 0; i < cities.size(); i++) {

if (startingCity.equals(cities.get(i))) {

return startingCity;

}

}

System.out.println("USAir doesn't serve this city. Try again");

} while (!check);

return null;

}

// hash map holds a key with the cost of each flight

public static HashMap<String, String> createHashMap() {

HashMap<String, String> hash = new HashMap<>();

Scanner sc = null;

try {

sc = new Scanner(new File("flights.txt"));

} catch (FileNotFoundException e) {

e.printStackTrace();

System.exit(1);

}

String[] hashLines;

while (sc.hasNextLine()) {

// split lines in each text file at each comma and each argument

// becomes a subsection of the entire line

hashLines = sc.nextLine().split(",");

hash.put(hashLines[0].trim() + "," + hashLines[1].trim(), hashLines[2].trim());

}

return hash;

}

}