**CS 1302 – Homework 8**

This homework counts as 3 homeworks. We will also extend this assignment in later homework.

**Overview**

1. You have been provided a file, *airports.txt* which lists the airport code, latitude, longitude, city, and state for approximately 1200 airports in the US. A small portion of the file is shown below (note, the header line is not in the file):

Airport Code Latitude Longitude City State

ANB 33.58 85.85 Anniston AL

AUO 32.67 85.44 Auburn AL

BHM 33.57 86.75 Birmingham AL

CKL 32.9 87.25 Centreville AL

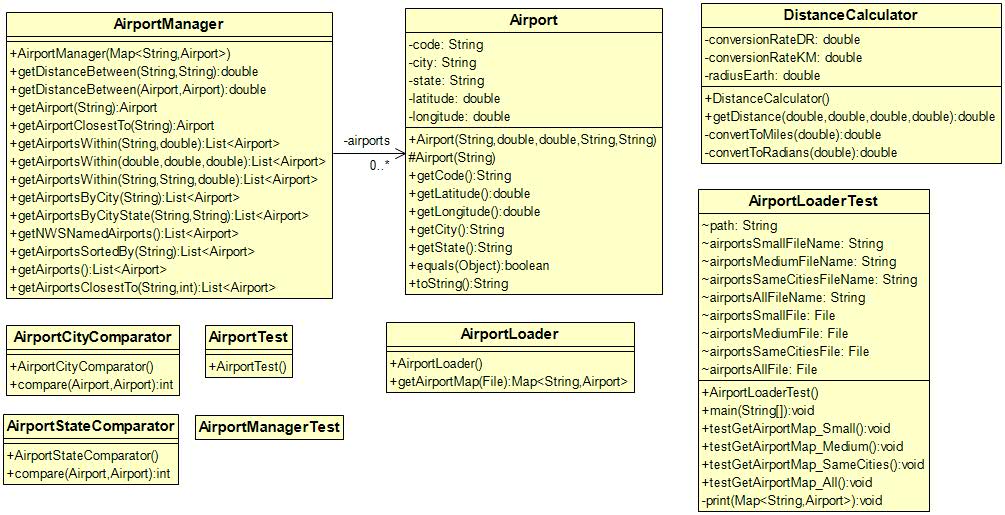
DHN 31.32 85.45 Dothan AL

OZR 31.28 85.72 FortRucker AL

1. You will write the following classes (and shown in class diagram below)

* *Airport* class
* *AirportManager* class that holds a map of *Airports*
* *AirportLoader* class which has a single method to read a file of airports and return a map of *Airport* objects.
* Two comparators (*AirportCityComparator* & *AirportStateComparator*) and two test classes (*AirportTest*-1 method, *AirportManagerTest*-14+ methods).
* The two classes on the right, below, *DistanceCalculator* and *AirportLoaderTest* are written and provided to you.

Documentation for the classes is found on the HW page.



1. You have also been provided three additional files for testing:

* *airportsSmall.txt* which contains 6 airports.
* *airportsMedium.txt* which contains 89 airports.
* *airportsSameCities* which contains 19 airports, some with the same city, but different stat, and some with the same state, but different city.

1. For testing, you will need to know the distance between cities for some methods. I’ve provided you a spreadsheet, *distance calculator.xlsx* that will calculate the distance between two pairs of latitude & longitude. This online calculator will work also:

<http://www.meridianoutpost.com/resources/etools/calculators/calculator-latitude-longitude-distance.php>

**Steps to Complete**

1. Create a project in Eclipse, then create a package named *prob1.* Next, download *hw8\_code.zip*, unzip, and drag the files into the *prob1* package. Make sure you have the documentation pulled up (link on HW page)
2. Write the *Airport* class to represent the data for a single airport.
3. Create an *AirportTest* class and write tests for the *equals* method.
4. Run tests and evaluate.
5. Open the *AirportLoaderTest* class that was provided. Review the code to see how the *getAirportMap* method is used, and to get a feel for the tests.
6. Write the *AirportLoader* class
7. Run the *AirportLoaderTest* class and evaluate the output.
8. Create the *AirportManager* class and write code to implement the instance variable and constructor.
9. Create the *AirportManagerTest* class and write a test for the constructor. You can probably borrow code from *AirportLoaderTest*.
10. Run tests and evaluate.
11. Write this method: **public** Airport getAirport(String code)
12. Write tests for this method in the *AirportManagerTest* class
13. Run tests and evaluate.
14. Write this: **public** List<Airport> getAirports()
15. Write a test for this method in the *AirportManagerTest* class
16. Run test and evaluate.
17. Do the following:
18. (Read, no action required) Open the *DistanceCalculator* class which has a static method with this signature:

**public** **static** **double** getDistance(**double** lat1, **double** long1, **double** lat2, **double** long2)

This method calculates the distance in miles between two pairs of latitude and longitude. It is already written for you. You will use this in later steps.

1. Write a method in the *AirportManager* class with this signature:

**public** **double** getDistanceBetween(Airport airport1, Airport airport2)

1. Write a test for this method in the *AirportManagerTest* class
2. Run test and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** **double** getDistanceBetween(String code1, String code2)

which is the same as the previous method except that the inputs are Airport *codes* instead of *Airport* objects.

1. Write a test for this method in the *AirportManagerTest* class
2. Run test and evaluate.
3. Do the following:
4. Write the *AirportCityComparator* class.
5. Write the *AirportStateComparator* class.
6. Do the following:
7. Write a method with this signature:

**public** List<Airport> getAirportsByCity(String city){

which returns a list of the Airports whose “City” is *city.*

1. Write test(s) for this method in the *AirportManagerTest* class.
2. Run tests and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** List<Airport> getAirportsByCity(String city){

which returns a list of the Airports whose “City” is *city.*

1. Write test(s) for this method in the *AirportManagerTest* class.
2. Run tests and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** List<Airport> getAirportsByCityState(String city, String state){

which returns a list of the Airports whose “City” is *city* and “State” is *state.*

1. Write test(s) for this method in the *AirportManagerTest* class.
2. Run tests and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** List<Airport> getNWSNamedAirports(){

which returns a list of the Airports whose code follows the National Weather Service (NWS) naming system. NWS codes have an ‘X’ as the third character. For example: LAX is Los Angeles’ airport.

1. Write test(s) for this method in the *AirportManagerTest* class.
2. Run tests and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** List<Airport> getAirportsSortedBy(String sortType)

which returns a list of Airports that are sorted according to *sortType:*

|  |  |
| --- | --- |
| *sortType* | Meaning |
| “City” | Sorted by city, then state. For example, if two cities are the same, then the one with the “smaller” state would be first. |
| “State” | Sorted by state, then city. For example, if two states are the same, then the one with the “smaller” city would be first. |

Hints – You need two *Comparator*s.

1. Write test(s) for this method in the *AirportManagerTest* class.
2. Run tests and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** Airport getAirportClosestTo(String code)

which returns the *Airport* which is closest to the airport corresponding to *code*.

Hint: Use the Airport *equals* method so that you don’t try to calculate the distance between the input Airport and itself.

1. Write a test for this method in the *AirportManagerTest* class
2. Run test and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** List<Airport> getAirportsWithin(String code, **double** withinDist)

which returns a list of *Airports* which are within *withinDist* of the *Airport* with *code*.

1. Write tests for this method in the *AirportManagerTest* class
2. Run tests and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** List<Airport> getAirportsWithin(String code1, String code2, **double** withinDist)

Returns a list of *Airports* which are within *withinDist* of both the *Airports* that correspond to *code1* and *code2*. For example, find all airports that are within 100 miles of both the Valdosta and Moody airports.

Hints:

1. Call the overloaded getAirportsWithin(String code, **double** withinDist) twice, once with each *code* and the distance. This will yield two lists. Then, put them together for the answer.
2. Using, *airportsMedium.txt,* this is a reasonable test: getAirportsWithin("ANB", "VLD", 150) which returns 3 cities.
3. Write tests for this method in the *AirportManagerTest* class. Hint: you can use the same airports from the tests for the previous method.
4. Run tests and evaluate.
5. Do the following:
6. Write a method with this signature:

**public** List<Airport> getAirportsClosestTo(String code, **int** num)

Returns a list of the *num* Airports that are closest to the Airport with *code*. For example, if *num=3*, the find the 3 closest *Airports* to *code*. No hints. This will take some thought!

Using, *airportsSmall.txt*, these are good tests.

getAirportsClosestTo("ANB", 1);

getAirportsClosestTo("ANB", 2);

getAirportsClosestTo("ANB", 3);

1. Write test(s) for this method in the *AirportManagerTest* class.
2. Run tests and evaluate.
3. Do the following:
4. Write a method with this signature:

**public** List<Airport> getAirportsWithin(**double** withinDist, **double** lat, **double** lon){

Returns a list of Airports that are within *withinDist* to the latitude and longitude, *lat* and *lon* respectively.

Hint: This solution is simple, but requires a bit of thought. Don’t reinvent the wheel. Create a “dummy” airport with the lat and lon, and then call an overloaded

1. Write test(s) for this method in the *AirportManagerTest* class.
2. Run tests and evaluate.

**Submission**

Zip your *prob1* folder into a file named: *hw8\_lastname.zip* and submit on Blazeview by the due date.

