# CSE 142, Autumn 2018
# Programming Assignment #8: GeoLocation (10 points)
### Due Tuesday, November 27th, 9:00 PM

This assignment will give you practice with being a client of one class and being the implementer of a second class. The classes store location information for various places of interest. The target application is something like Google Maps where a user can make requests such as asking for restaurants that are near a certain location. This assignment should not take as long to complete as the other assignments, so it is worth 10 points instead of the usual 20. You will turn in *three* files as described below.

## Part A: Location Data (2 points)

The code we are writing will be more interesting to use if we have a lot of data to work with. Companies like Google have invested significant resources to identify places of interest and to record their name, their address, and their location along with some tags that are relevant for searching. We don't have the resources of a company like Google, but we have a lot of students. So we will use a crowdsourcing approach by having each student provide information for 5 places of interest in the general Seattle area. You can include any place that you think a UW student would be likely to go including, for example, restaurants near the airport.

For each location, we want the following information:

- Name of the place of interest
- Street address (or similar description) for the place (not a complete address, just enough information to find it if you were near that location)
- One or more search tags for this place
- The latitude of this place
- The longitude of this place

For example, suppose you want to include the Jack in the Box near campus in our data file. You know its name and address and can come up with some tags. To find its latitude and longitude, you can go to this web page:

http://universimmedia.pagesperso-orange.fr/geo/loc.htm

Or you can get the information directly from Google Maps by following the instructions here:

http://www.wikihow.com/Get-Longitude-and-Latitude-from-Google-Maps

You can add whatever search tags you think are appropriate, but Google provides a list of standard tags that would be useful to apply to your entries:

https://developers.google.com/places/documentation/supported_types

You should put the information together on separate lines, as in:

```
Jack in the Box
4749 University Way
restaurant, fast food
47.66476
-122.31335
```

You are to choose 5 locations and to come up with a similar 5-line entry for each location. The locations should not be private residences and should not include offensive descriptions. That means your file will have 25 lines of data. You can use any text editor you like or you can use jGRASP, but if you use an editor like Microsoft Word, be sure to save it as plain text. You are allowed to include a blank line between entries to make it easier to distinguish them while editing. A sample file called `places.txt` has been provided on the course website to demonstrate what your file should look like.

Turn in a file called `myplaces.txt` with the data for your five places.

## Part B: GeoLocationClient.java (2 points)

In this part of the assignment you will write client code to manipulate some GeoLocation objects. The `GeoLocation` class is being provided to you, so you don't have to write it, but you will need to download it from the course website. You will instead be writing code that constructs and manipulates three `GeoLocation` objects. Write and turn in a class called `GeoLocationClient` in a file called `GeoLocationClient.java`.

The popular TV series Breaking Bad made use of geographic location information. The Walter White character buried millions of dollars at a particular location in the desert outside of Albuquerque, New Mexico. He then bought a lottery ticket to help him remember that his stash was buried at a latitude of 34 degrees, 59 minutes, 20 seconds and a longitude of -106 degrees, 36 minutes, 52 seconds. Your client program will compute the distance between Walter's stash and the local FBI building and a local studio known as ABQ Studios (where Breaking Bad was filmed). The coordinates of these locations are as follows:

| **Location** | **Latitude** | **Longitude** |
|---|---|---|
| Walter's Stash | 34.988889 | -106.614444 |
| ABQ Studios | 34.989978 | -106.614357 |
| FBI Building | 35.131281 | -106.61263 |

**You must represent each location with a `GeoLocation` object in your program**. You should use the latitude and longitude values in this log to construct the three `GeoLocation` objects. You should then print the three objects and call the `distanceFrom` method twice to get the desired output. Please note that the latitude/longitude information in the first three lines of output has to be produced by calls on the `toString` method of the `GeoLocation` class and the values in the final two lines of output have to be produced by calls on the `distanceFrom` method of `GeoLocation`.

Your program should produce exactly the following output:

```
the stash is at latitude: 34.988889, longitude: -106.614444
ABQ studio is at latitude: 34.989978, longitude: -106.614357
FBI building is at latitude: 35.131281, longitude: -106.61263
distance in miles between:
    stash/studio = 0.07548768123801672
    stash/fbi    = 9.849836190409732
```

## Part C: PlaceInformation.java (6 points)

For this part of the assignment, you will write a class called `PlaceInformation` that stores information about a place of interest. It should have the following public methods:

```
public PlaceInformation(String name, String address, String tag,
                        double latitude, double longitude)
public String getName()
public String getAddress()
public String getTag()
public String toString()
public GeoLocation getLocation()
public double distanceFrom(GeoLocation spot)
```

The first three "get" methods simply return the values that were provided when the object was constructed. The `toString` method should return the name followed by the location details inside parentheses (location details should formatted the same as in the `toString` method for `GeoLocation`). The `getLocation` method should return a reference to a `GeoLocation` object representing the same location represented by this object. The `distanceFrom` method should return the distance between this object and the parameter as calculated by the `distanceFrom` method in the `GeoLocation` class.

Although the constructor takes a latitude and longitude, **you should store this information inside the `PlaceInformation` object using a `GeoLocation` object**.Remember that in writing your class, you don't want to include code that appears elsewhere. For example, your `GeoLocation` object knows how to compute a distance, so you

should not be repeating the code for computing distances in your `PlaceInformation` class. You should instead be asking the `GeoLocation` object to perform the computation. Be sure to properly encapsulate your object with private fields.

This class is similar to the `GeoLocation` class, so you can use it as a model for how to write your own class. The GeoLocation class will also provide a useful example of how to comment your class. Each method should be commented and the class itself should have a general comment.

Turn in a file called `PlaceInformation.java`. We are providing a client program called `PlaceInformationClient.java` that can be used to test your class.