CSC 204 Lab 14: Using Custom Classes

In this lab, you'll practice using the GeoPoint to specify geographic coordinates for locations on the globe and use the class to calculate distances across the globe.

Lab Preparation

To prepare for this lab, read through this lab carefully.

Materials Needed

Be sure you have the following on hand during the lab.

- This sheet
- Your course text
- The GeoPoint.class file from http://theochem.mercer.edu/csc204

Lab Setup

Create a new project and copy over the GeoPoint.class file from the class website. Place this in a new Lab 8 project in Eclipse and make sure it can be found by any program you write.

Part 1: Geographic Coordinates

Geographics coordinates (latitudes and longitudes) are typically specified in two ways

1. degrees, minutes, seconds format
2. decimal degrees format

Both of these methods require a way to store the position relative to the equator and the prime meridian; a direction (N/S for latitudes, E/W for longitudes) is therefore also required. In the decimal degrees format, one may specify the direction with a sign (west of the prime meridian and south of the equator is negative).

Part 2: Creating Geographic Objects

1. The GeoPoint class has several constructors for entering location data, but by far the easiest one to use simple uses the latitude and longitude decimal degrees notation. Below I have created an instance of the GeoPoint object for Mercer.

        GeoPoint mercer = new GeoPoint( 'N', 32.830389, 'W', 83.648552 );

Here are some additional Coordinates for some selected cities in the United States

Montgomery, AL,    N 32.233377  W 86.208528           (same as  32.233377,-86.208528)
Port Orange, FL,  N 29.080251,  W 81.045962               (same as 29.080251,-81.045962)
Part 3: Calculating Distances

If we assume that the earth is perfectly spherical, then it is not too difficult to come up with an equation to calculate the distance between two locations on a sphere travelling along the surface of the sphere. That is exactly how the `distanceInMiles` method of the `GeoPoint` class works. Given two GeoPoint objects (like the one for macon we created above), the distance between the two locations can be easily calculated. Assume you have two GeoPoint objects named `location1` and `location2`.

```java
double distance = location1.distanceInMiles(location2)
```

Part 4: Your Turn

Imagine you have friends in Montgomery that are planning on picking you up on their way to the beach (their parents own a house in Port Orange; it is very close to Daytona). You are going to write and submit to me via e-mail a program that will:

1. Compute and print out the mileage between Montgomery and Macon
2. Compute and print out the mileage between Macon and Port Orange
3. Print out the total mileage for the trip
4. Assuming an average speed of 65 mph, compute and print out how long the trip will take between each destination and how long it will take to make the entire trip.

Finally, get on MapQuest and compare your values with their travel time and distance.

Deliverables

Send me a copy of your programs to me as an e-mail attachment. The subject of your e-mail should be LAB 14 SUBMISSION – YOUR NAME. The body of your e-mail should include the travel mileage and time provided by MapQuest, the mileage and travel time calculated by your program, and your explanation of why the times may differ.