Computer Science 204
Assignment #3
Geographic Distance Locator
Due Date: Monday, February 24th, 2003
40 Points

Objective

The purpose of this assignment is to extend our familiarity with the string class and also using classes. In this assignment you will use a class designed by me called GeoPoint which stores geographic coordinates and calculates distances to other geographic locations.

Assignment Summary

Your program should prompt the user for two geographic coordinates in degree, minutes, seconds format including directions from the prime meridian. Any number of spaces will delimit the angular and directional components from each other and a comma will delimit the latitude from the longitude. Either latitude or longitude can come first. The following are sample inputs for the program:

23 45 32 N, 82 12 23 W
24 4 2 w, 3 34 17 S
32 32 25 E, 12 12 23 n
45 32 23 s, 34 43 21 w

Note that all numbers are integers and any number of spaces can differentiate between the degrees, minutes, and seconds. The program does not have to check if the data are input correctly. We will do that another time. Your job is to use the String and Character class methods to parse the input so that it can be placed into a GeoPoint object. Once you have both locations stored in a GeoPoint object, you can call the distanceInMiles method from the GeoPoint class to calculate the distance between the two points.

Notes

1. In solving this problem, you are required to follow the Software Development Method described in class. Using any word processor, think about and type up the following before starting your coding:

   (a) A description of the problem to be solved
   (b) A specification describing your input and output
   (c) An algorithm – an outline in plain English of how to solve your problem. (The real question here is how are you going to parse the input to break it up into the coordinate components.)
Refer to the case study on page 63 of your text for an example. This document is due on Monday, February 17, 2003 by 5:00 p.m..

2. Your source file should be named Distance.java and should be in a Prog3 directory on your account.

3. Use proper indentation and documentation throughout. Provide comments for all variables and important statements. See pages 65 – 67 for an example of a nicely commented program.

4. Indent all statements within the main body four spaces inward.

5. There are several constructors for the GeoPoint class. You will want to use the following:

    GeoPoint();  // just creates a GeoPoint Object

    GeoPoint( char latitudeDirection, int latitudeDeg, int latitudeMin, int latitudeSec, char longitudeDirection, int longitudeDeg, int longitudeMin, int longitudeSec)

It is imperative that when you call the GeoPoint constructor with directional data that you put the latitude data first. While your program can read the data in either format, your call to the constructor must have the latitude first. You may utilize the toString method of the GeoPoint class to print your geometric coordinates.

6. You may copy the GeoPoint.java file to your local directory with the command:

    cp /pub/pounds/CSC204/Progs/Prog3/GeoPoint.java .

7. Just to make things interesting, here are the coordinates for several world cities:

| Mercer University – 32 50 27 N, 83 37 57 W | London, UK – 51 30 0 N, 0 10 0 W |
| Atlanta, GA – 33 45 10 N, 84 23 37 W | Los Angeles, CA – 34 3 15 N, 118 14 28 W |
| Baghdad, Iraq – 33 20 0 N, 44 26 0 E | Moscow, Russia – 55 45 0 N, 37 42 0 E |
| Bangkok, Thailand – 13 44 0 N, 100 30 0 E | New York, NY – 40 45 6 N, 73 59 39 W |
| Brisbane, Australia – 27 30 0 S, 153 1 0 E | Ouagadougou, Burkina Faso – 12 22 0 N, 1 31 0 W |
| Capetown, South Africa – 35 55 0 S, 18 22 0 E | Paris, France – 48 52 0 N, 2 20 0 E |
| Chicago, IL – 41 52 28 N, 87 38 22 W | Quito, Ecuador – 0 13 0 S, 78 32 0 W |
| Dili, East Timor – 8 57 0 S, 125 58 0 E | Rio de Janeiro, Brazil – 22 53 43 S, 43 13 22 W |
| Fairbanks, Al – 64 48 0 N, 147 51 0 W | Rome, Italy – 41 48 0 N, 12 36 0 E |
| Hong Kong, Hong Kong – 22 15 0 N, 114 10 0 E | Santiago, Chile – 33 27 0 S, 70 42 0 W |
| La Paz, Bolivia – 16 30 0 S, 68 9 0 W | Singapore, Singapore – 1 18 0 N, 103 50 0 E |
Hand In
Create a typescript file using the script command. Type the following commands for your script:

- cat -n Distance.java (to display your file)
- javac Distance.java (to compile your file)
- java Distance (type twice to run your program with two different sets of sample input)
- exit (to exit script)

Print out your typescript file by typing the command trueprint -P lab204 typescript (if you are in lab 204). Hand in your typescript file stapled to the back of your write-up. Your program is considered on-time as long as your script is run by 5:00 PM on the due date.

Include your signature on your front page write-up. This signature affirms that you have written all code that you are submitting and have followed the Honor Code in completing this assignment.

Sample Run

*********************************************************
* GEOGRAPHIC DISTANCE                       *
* CALCULATOR                                *
*********************************************************

Please enter geographic coordinates for point 1: 32 50 27 N, 83 37 57 W

Please enter geographic coordinates for point 2: 21 18 22 N, 157 51 35 W

The distance from

32 50’ 27" N, 83 22’ 3" W
to
21 18’ 22" N, 157 8’ 25" W

across the surface of the planet is 4544 miles.

Revision Policy
All assignments handed in on or before the due date that you do not receive full credit for in either implementation or documentation are eligible for revision.