20.3 Program 2a: LatAndLong

Objectives

* Write a Java program to solve a problem by manipulating arrays.
* Divide code into methods that perform specific tasks.
* Format output.
* Develop programs using an IDE.
* Use Math class functions in a trigonometric function.

Background Reading

* ZyBook: 6. Arrays, 2.14 Random, 12.2 printf()

Instructions

* Write a complete Java application called **LatAndLong.java** that satisfies the program requirements below. Develop the code in an IDE (Eclipse or your favorite).
* Periodically submit your program for grading. This will help keep you from going too far off track.  
  When you are satisfied with the program (or are out of time and want some partial credit), upload the source file LatAndLong.java.
  + See the "Uploading source file to ZyBooks lab" video on our Blackboard course web site for help.
* Your program will be graded automatically against the requirements.
* You may submit as many times as necessary.
* The automatic grading program is very specific. If you feel you have the correct solution but are not receiving full credit, please
  + Carefully review the output -- you might need to scroll all the way to the right to find what is wrong with a particular output.
  + Verify you have the correct names for the program itself and all methods.
  + Check your calculations by hand: was there a logic error?
  + Review the requirements: did you miss a step? misinterpret a requirement?
  + If all these check out, contact the T.A. for assistance.

Program Requirements

(0) Write a method getIdentificationString() that returns a string containing the programming assignment and your name. DO NOT call this method. Just make sure it is there and working correctly. The automatic grader will verify it.

Ex.

Program 2, Patty Kraft

(1) Write a method called printArrays(Double[] array1, Double[] array2) that prints corresponding values of the arrays on the same line, formatted to the millionths, right-justified, and at least twelve characters wide.

48.858093 2.294694

-3.070000 37.349998

**Note** The System.out object contains a printf() method that we will use instead of print() or println(). Here's a quick description of printf(). Go to ZyBook section [printf](https://learn.zybooks.com/zybook/SDSUCS108KraftSummer2021/chapter/20/section/3) for full reference.  
The basic idea is: System.out.printf("String with placeholders using %", variable list to match placeholders);  
Example: System.out.printf("%8.2f%8.2f\n", doubleVal1, doubleVal2); would print out doubleVal1 at least eight characters wide (counting the decimal point) and with two digits following the decimal. Blanks make up any unused spaces. The above values with this formatting would print:

48.86 2.29

-3.07 37.35

(2) In the main method, create two one dimensional Double arrays (note the capital 'D' for the wrapper class) called latitude and longitude with the following values in the order given:

* Latitude: 48.858093, -3.070000, 27.173891, 51.501476, -22.9519
* Longitude: 2.294694, 37.349998, 78.042068, -0.140634, -43.2104

(3) In the main method, call printArrays() to print out latitude and longitude in table form.

(4) In the main method, call distanceFrom() to find the distance between the second and fourth coordinate pairs using the Law of Cosines and print it out following "Distance in kilometers: " A coordinate pair is a latitude and longitude at a corresponding index, e.g. latitude[0] and longitude[0].

//Using The Law Of Cosines To Determine Distance

/\*\*

\* Returns the distance in miles between this geo location and the given

\* other geo location

\*/

public static double distanceFrom(double x1, double y1, double x2, double y2) {

double lat1 = Math.toRadians(x1);

double long1 = Math.toRadians(y1);

double lat2 = Math.toRadians(x2);

double long2 = Math.toRadians(y2);

// apply the spherical law of cosines with a triangle composed of the

// two locations and the north pole

double theCos = Math.sin(lat1) \* Math.sin(lat2) +

Math.cos(lat1) \* Math.cos(lat2) \* Math.cos(long1 - long2);

double arcLength = Math.acos(theCos);

return arcLength \* EARTH\_RADIUS\_KM;

}

//where class member variable `public static final double EARTH\_RADIUS\_KM = 6371;`

(5) In the main method, sort the latitude array by smallest to largest using Arrays.sort(latitude).

(6) in the main method, output the arrays again. [Did the integrity of the data remain intact?]

Sample LatAndLong output:

48.858093 2.294694

-3.070000 37.349998

27.173891 78.042068

51.501476 -0.140634

-22.951900 -43.210400

Distance in kilometers: 7024.525221562596

-22.951900 2.294694

-3.070000 37.349998

27.173891 78.042068

48.858093 -0.140634

51.501476 -43.210400

(7) For absolutely no credit, find the global locations of these lat/long values. :)