Assignment GPS with Unit Tests CSIS-1410

Learning Objectives:

- Practice Composition
- Declare classes based on UML class diagrams
- Use Java documentation
- Practice Unit Tests

Turning in:

Create a jar **including the source code** and turn it in via Canvas

Description:

- Declare the following three classes: Gps, GpsCoordinates, GpsApp.
- Write unit tests for class Gps
- **Use javadoc to document your code**.

  Include doc comment for each of the classes Gps and GpsCoordinates.

  Include doc comments for each of the public methods and constructors in GpsCoordinates and Gps - except for getters and setters

  Notice how your doc comments provide helpful information as you call the methods in your program.

Ad GpsCoordinates:

Base your class declaration on the class diagram below. Do not add or remove any public methods or constructors.
All the output happes in GpsApp. There should be **NO** print statements in class GpsCoordinates.

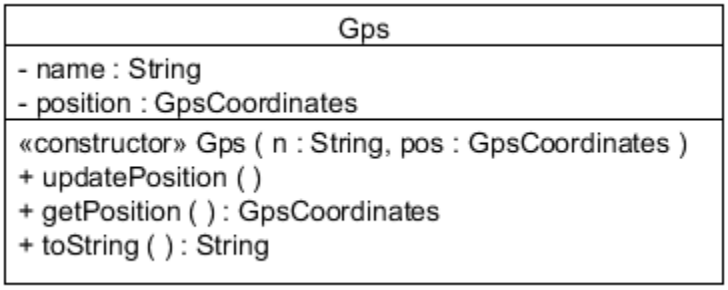
GpsCoordinates
- latitude : double - longitude : double
«constructor» GpsCoordinates (lat: double, lon : double) + getLatitude () : double + setLatitude (lat : double) + getLongitude () : double + setLongitude (lon : double) + toString () : String

Ad toString:

The method toString should return a String that includes the latitude followed by the longitude. Both latitude and longitude should display 6 digits after the decimal point. (see output)

Ad Gps:

Base your class declaration on the class diagram below. Do not add or remove any public methods or constructors.
All the output happes in GpsApp. There should be **NO** print statements in class Gps.



Ad toString:

The method toString should return a String that includes the name followed by the position. (see output)
Make sure to include the @Override annotation

Ad updatePosition:

The method updatePosition should set the field position to the current GPS position.
We simulate the movement of the GPS and the reading of the current GPS position by calculating a random numbers between -2.5 and 2.5 (exclusive) and adding it to the longitude. Then it calculates another random number form the same range and adds it to the latitude.

Write jUnit tests for class Gps (to keep the scope of the assignment in check you don't need to write any unit tests for the class GpsCoordinate)
Make sure to create the unit tests in a separate source folder within the same project

Ad GpsApp:

This class includes the main method. Inside main do the following:

- Create an instance of GpsCoordinates and assign it the value of Salt Lake City (Decimal WGS84)
  Print the result (including a label)
- Create an instance of Gps based on the SLC coordinates. Print the result.
- Update the position 3 times and print the results

Make your output look like the sample output below. (except for the randomly updated values)

Output:

```
Gps coordinates of SLC: 40.760671, -111.891122

myGps: Garmin: 40.760671, -111.891122

position update1: 38.260671, -114.391122
position update2: 37.760671, -114.891122
position update3: 38.260671, -114.391122
```