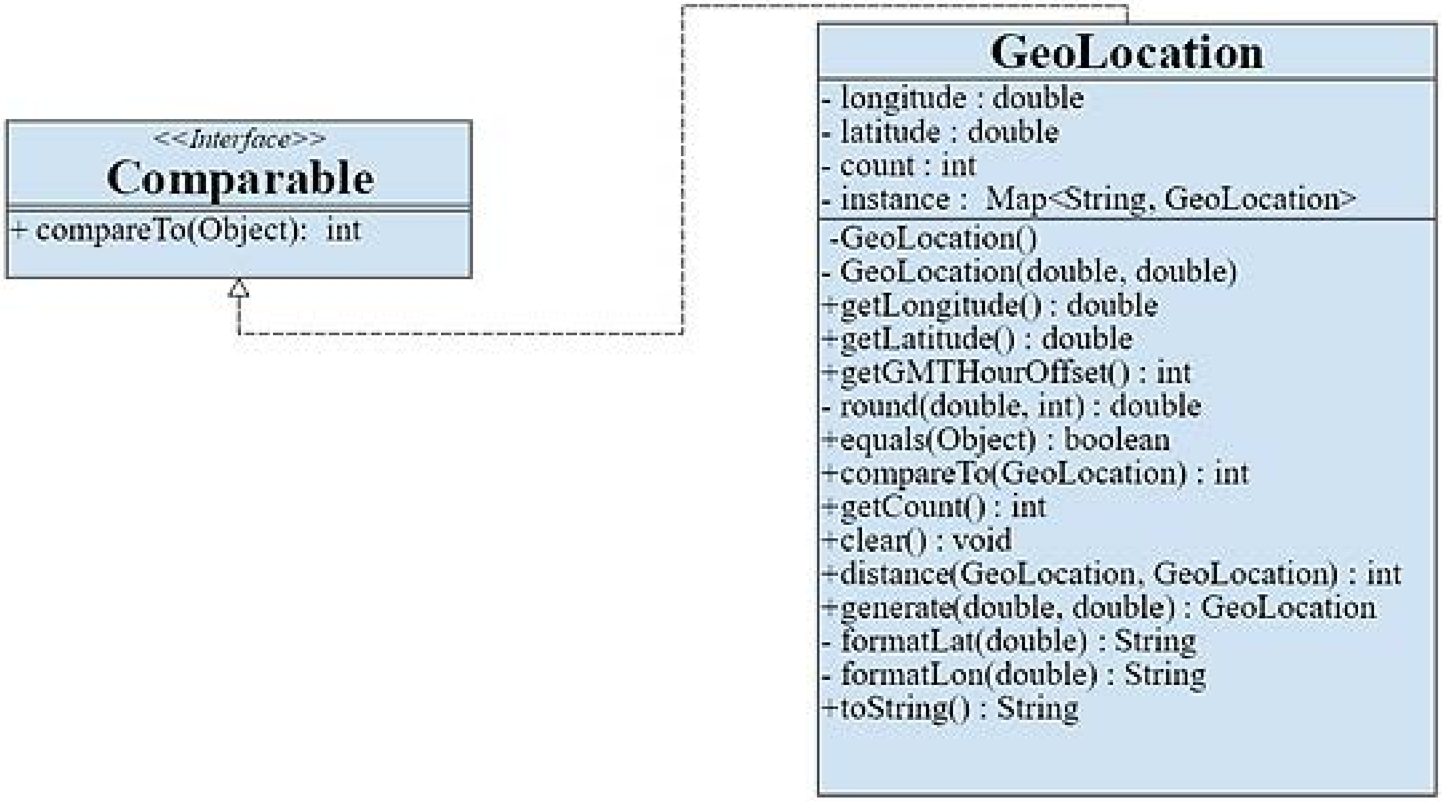
EECS 2030 SU Jay Cen

June 16, 2017 ID: 215145105

Assignment Report

When writing a class, programmers are not only to design but also to consider special privilege offer to client such as privacy leakage and access permission. In the GeoLocation class, the goal is to design features that have special performance. I used the Multiton design pattern in this class and I had overcome many obstacles. By definition, Multiton design pattern required multiple instances - for each contains an unique state, accessor that provide the state information, private constructors that prevent clients from creating instances on their own, and immutability, which prevent clients from modifying the state. I revised several times to meet the receipt for Multiton. This GeoLocation class have features that allow clients to use for computation of GMT hour offsets and the distance between two locations on the globe. Writing the methods for the computations, I created many helper methods to make it more convenience. The purpose of creating helper methods is to avoid duplication of code, and it's simple to debug when there are errors in the codes. Overall, I learned many structural syntax to meet certain requirements for special methods such as equals, compareTo, and toString.


<<Interface>>
Comparable
+ compareTo(Object): int
GeoLocation
- longitude : double
- latitude : double
- count : int
- instance : Map<String, GeoLocation>
-GeoLocation()
- GeoLocation(double, double)
+getLongitude() : double
+getLatitude() : double
+getGMTHourOffset() : int
- round(double, int) : double
+equals(Object) : boolean
+compareTo(GeoLocation) : int
+getCount() : int
+clear() : void
+distance(GeoLocation, GeoLocation) : int
+generate(double, double) : GeoLocation
- formatLat(double) : String
- formatLon(double) : String
+toString() : String

eecs2030.assignment

Class GeoLocation

java.lang.Object
    eecs2030.assignment.GeoLocation

All Implemented Interfaces:

`java.lang.Comparable<GeoLocation>`

```
public final class GeoLocation
extends java.lang.Object
implements java.lang.Comparable<GeoLocation>
```

Created by Jay Cen on 5/29/2017.

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`static void`	`clear()` reset the number of created objects to zero, and initiate the instance to an empty MapList
`int`	`compareTo(GeoLocation other)` Compare two GeoLocation objects numerically by hour offset , then by latitude
`static double`	`distance(GeoLocation location1, GeoLocation location2)` Compare and determine the shortest distance between the given locations and the radius of Earth, which is 6,371 km
`boolean`	`equals(java.lang.Object obj)` The method determines if the two locations are the same point;
`static GeoLocation`	`generate(double lonitude, double lat)` a factory method that return a GeoLocation object with the specified parameters
`static int`	`getCount()` It keeps track of the number of objects creation
`int`	`getGMTHourOffset()` Calculate the GMT hour offset depending on the globe's longitude.

`double`	`getLatitude()`
	return latitude

`double`	`getLongitude()`
	return longtitude

`java.lang.String`	`toString()`

Methods inherited from class java.lang.Object

`clone, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Method Detail

getLongitude

```
public double getLongitude()
```

return longtitude

Returns:

`the double type longitude`

getLatitude

```
public double getLatitude()
```

return latitude

Returns:

`the double type latitude`

getGMTHourOffset

```
public int getGMTHourOffset()
```

Calculate the GMT hour offset depending on the globe's longitude. The Hour offset is ranging between -12 and 11.

For Example
Longitude of -180 and 180 will return a value of -12
Longitude of 175 will return a value of 11;
Longitude of 0 will return a value of 0;

Returns:

`the GMT hour offset`

equals

`public boolean equals(java.lang.Object obj)`

The method determines if the two locations are the same point;

Overrides:

`equals in class java.lang.Object`

Parameters:

`obj - an object`

Returns:

`true if the two locations are at the same point on the globe; false otherwise`

compareTo

`public int compareTo(GeoLocation other)`

Compare two GeoLocation objects numerically by hour offset , then by latitude

Specified by:

`compareTo in interface java.lang.Comparable<GeoLocation>`

Parameters:

`other - the GeoLocation to be compared`

Returns:

the value 0 if the two GeoLocations have the same hour offset and latitude; the value -1 if the two GeoLocations have the same hour offset but this GeoLocation's latitude is less than the other GeoLocation's longitude **OR** the hour offset of this GeoLocation is less than that of GeoLocation; the value 1 if the two GeoLocations have the same hour offset but this GeoLocation's latitude\ is greater than that of GeoLocation.

getCount

`public static int getCount()`

It keeps track of the number of objects creation

Returns:

`the number of created objects stored`

clear

`public static void clear()`

reset the number of created objects to zero, and initiate the instance to an empty MapList

distance

```
public static double distance(GeoLocation location1,
                              GeoLocation location2)
```

Compare and determine the shortest distance between the given locations and the radius of Earth, which is 6,371 km

Parameters:

`location1 - a Geolocation object`

`location2 - another Geolocation object`

Returns:

`the shortest distance (in Km) between the locations and the radius of Earth`

generate

```
public static GeoLocation generate(double lonitude,
                                   double lat)
```

a factory method that return a GeoLocation object with the specified parameters

Parameters:

`lonitude - a double type`

`lat - a double type`

Returns:

`the creation of a robust object, GeoLocation, that will work for any input passed as arguments`

toString

```
public java.lang.String toString()
```

Overrides:

`toString in class java.lang.Object`

Returns:

`a String using the format of ( +000.0000, -00.0000)`