CS 350 Task 3 Hints

Angle

```
double normalize(double angle)
```

make the angle reside on the interval [0,360)

```
Angle(double angle)
```

store the angle

```
Angle add(Angle angle)
```

add the angle to this angle and return the sum angle

```
double getValue()
```

return the angle

```
Angle reciprocate()
```

return a new angle with 180 degrees added

```
Angle subtract(Angle angle)
```

subtract the angle from this angle and return the difference angle

CoordinatesScreen

```
CoordinatesScreen(int x, int y)
```

store the coordinates

```
CoordinatesScreen add(CoordinatesScreen coordinates)
```

add the coordinates to these coordinates and return the sum coordinates

```
CoordinatesScreen getHalf()
```

return the coordinates divided by two and rounded

```
int getX()
```

return the x coordinate

```
int getY()
```

return the y coordinate

```
boolean isVisible()
```

return the visibility state

```
void isVisible(boolean isVisible)
```

set the visibility state

```
CoordinatesScreen subtract(CoordinatesScreen coordinates)
```

subtract the coordinates from these coordinates and return the difference coordinates

A_LatitudeLongitude

```
int convertToDegrees(double nmea)
```

return the degrees component of the NMEA encoding. (Hint: use algebra)

```
int convertToMinutes(double nmea)
```

return the minutes component of the NMEA encoding. (Hint: use algebra)

```
double convertToNauticalMiles(int degrees, int minutes, double seconds)
```

we're not doing this one

```
double convertToNMEA(int degrees, int minutes, double seconds)
```

convert DMS to NMEA by the equation given

```
double convertToSeconds(double nmea)
```

return the seconds component of the NMEA encoding. (Hint: use algebra)

A_LatitudeLongitude(int degrees, int minutes, double seconds)

store the DMS

double calculateDistanceMeters(A_LatitudeLongitude target)

calculate the distance in nautical miles and convert to meters

double calculateDistanceNauticalMiles(A_LatitudeLongitude target)

calculate the distance between the two coordinates and convert it to nautical miles. (Hint: use algebra)

double convertToNMEA()

call the convertToNMEA() with these coordinates

int getDegrees()

return the degrees

int getMinutes()

return the minutes

Latitude

Latitude(double nmea)

call the super constructor

Latitude(int degrees, int minutes, double seconds)

call the super constructor

Latitude add(Latitude latitude)

convert both latitudes to NMEA and create a new one with the sum

Latitude subtract(Latitude latitude)

convert both latitudes to NMEA and create a new one with the difference

Longitude

Longitude(double nmea)

call the super constructor

Longitude(int degrees, int minutes, double seconds)

call the super constructor

Longitude add(Longitude latitude)

convert both longitudes to NMEA and create a new one with the sum

Longitude subtract(Longitude latitude)

convert both longitudes to NMEA and create a new one with the difference

CoordinatesDelta

CoordinatesDelta(double x, double y)

store the coordinates

CoordinatesDelta add(CoordinatesDelta coordinates)

add the coordinates to these coordinates and return the sum coordinates

Angle calculateBearing(CoordinatesDelta target)

use the differences in x and y coordinates to build a triangle to get the angle. (Hint: use trig)

double calculateDistance(CoordinatesDelta target)

use the Pythagorean Theorem to calculate the distance between the coordinates. (Hint: use trig)

CoordinatesDelta calculateTarget(Angle bearing, double distance)

use cos and sin to calculate new coordinates from these coordinates at an angle and radius

double getX()

return the x coordinate

```
double getY()
```

return the y coordinate

```
CoordinatesDelta subtract(CoordinatesDelta coordinates)
```

subtract the coordinates from these coordinates and return the difference coordinates

CoordinatesWorld

```
CoordinatesWorld build(int latitudeDegrees,  int latitudeMinutes,  double latitudeSeconds,
                       int longitudeDegrees, int longitudeMinutes, double longitudeSeconds)
```

build new coordinates by building the intermediate objects

```
double convertMetersToNauticalMiles(double meters)
```

convert meters to nautical miles. (Hint: use algebra)

```
CoordinatesWorld(Latitude latitude, Longitude longitude)
```

store the coordinates

```
CoordinatesWorld add(CoordinatesWorld coordinates)
```

use the add methods on each coordinate component and build new coordinates

```
Angle calculateBearing(CoordinatesWorld target)
```

use the differences in NMEA values for latitude and longitude to build a triangle to get the angle. (Hint: use trig)

```
double calculateDistanceMeters(CoordinatesWorld target)
```

calculate nautical miles to the target and convert to meters

```
double calculateDistanceNauticalMiles(CoordinatesWorld target)
```

use the Pythagorean Theorem to calculate the distance between the NMEA values for latitude and longitude and convert to nautical miles

```
CoordinatesWorld calculateTarget(Angle bearing, double distance)
```

use cos and sine to determine the coordinates at the distance in nautical miles from these coordinates. (Hint: use trig)

`CoordinatesWorld calculateTarget(CoordinatesDelta delta)`

add the delta coordinates in nautical miles to the NMEA values for latitude and longitude

`Latitude getLatitude()`

return the latitude component

`Longitude getLongitude()`

return the longitude component

`CoordinatesWorld subtract(CoordinatesWorld coordinates)`

use the subtract methods on each coordinate component and build new coordinates

A_Shape

`A_Shape(CoordinatesWorld reference, CoordinatesDelta deltaStart, CoordinatesDelta deltaEnd)`

record the values

`A_Shape(CoordinatesWorld reference, CoordinatesDelta deltaStart, CoordinatesDelta deltaEnd, int index)`

record the values

`CoordinatesDelta getDeltaEnd()`

get the delta end value

`CoordinatesDelta getDeltaStart()`

get the delta start value

`int getIndex()`

get the index

`CoordinatesWorld getReference()`

get the reference coordinates

```
CoordinatesWorld getWorldStart()
```

calculate the target from the reference to the delta start

```
boolean hasIndex()
```

return whether there is an index

```
CoordinatesWorld interpolateWorld(double distance, boolean isFromAElseB)
```

use interpolateDelta to get the delta coordinates to feed into calculateTarget on the reference

```
void setIndex(int index)
```

set the index

ShapeLine

```
ShapeLine(CoordinatesWorld reference, CoordinatesDelta deltaStart, CoordinatesDelta deltaEnd)
```

call the super constructor and calculate the length

```
double getLength()
```

get the length

```
CoordinatesDelta interpolateDelta(double distance, boolean isFromAElseB)
```

calculate the bearing from delta start to delta end and calculate the target with the distance

```
boolean isOnPath(double distance)
```

return whether the distance is nonnegative to the distance

ShapeArc

we're not doing this class