Objektorientierte Programmierung (E-B3-OPP)

Praktikumsaufgabe 1:
Geografische Koordinaten

HAW Hamburg
Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

1 Allgemeine Hinweise

1.1 Primäre Lernziele

- Einfache Klassen mit Variablen und Methoden
- Ausführbare Klassen
- Statische Elemente

1.2 Vorbereitung

- Bereiten Sie die Aufgaben soweit wie möglich vor dem Praktikum vor.
- Versuchen Sie unbedingt, die Aufgaben zunächst *allein* zu lösen. Tauschen Sie sich bei Problemen mit Ihrer Gruppe oder mit anderen Kommilitonen aus.
- Erzeugen Sie beim Praktikumstermin ein Java-Projekt, das Ihre Gruppe zur Abnahme präsentiert.

1.3 Erfolgreiche Teilnahme und Abnahme

- Es besteht Anwesenheitspflicht bei *allen* Praktikumsterminen.
- Die erfolgreiche Abnahme *aller* Praktika ist Voraussetzungen zur Teilnahme an der Laborprüfung.
- Der Code *muss* den in der Vorlesung aufgestellten Programmierrichtlinien entsprechen.
- Zur erfolgreichen Abnahme müssen die theoretischen Grundlagen und Hintergründe der im Praktikum behandelten Konzepte erläutert werden können.

2 Geografische Koordinaten und Entfernungen

2.1 Breiten- und Längengrade

Geografische Koordinaten $g = (lat, lon)$ werden durch die *Breite* (*latitude*) und *Länge* (*longitude*) in Grad dargestellt.

- Breitengrade verlaufen parallel zum Äquator. Der Wertebereich umfasst -90° („südliche Breite“) am Südpol bis 90° („nördliche Breite“) am Nordpol.
- Längengrade laufen durch den Nordpol und den Südpol. Der Wertebereich umfasst -180° („westliche Länge“) bis 180° („östliche Länge“). Der sogenannte Nullmeridian (Längengrad 0°) verläuft durch den Ort Greenwich in England.

Was	*Bezeichner*	*Verlauf*	*Entspricht*	*Wertebereich*
Breite	Latitude *lat*	Parallel zum Äquator	y-Koordinate	[-90°, 90°]
Länge	Longitude *lon*	Durch Nord- und Südpol	x-Koordinate	[-180°, 180°]

Hinweise:

- Beachten Sie, dass hierbei im Vergleich zu herkömmlichen Koordinaten die x- und y-Komponenten vertauscht sind.
- In *Google Maps* erhalten Sie die geografischen Koordinaten, indem Sie auf einen Ort klicken.

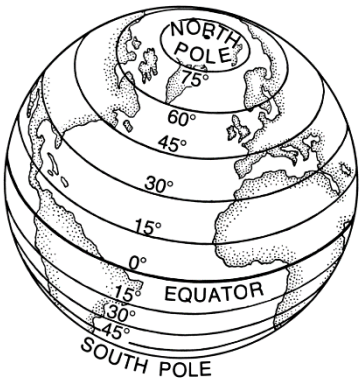


Abbildung 1: Breitengrade (Wikipedia[1])

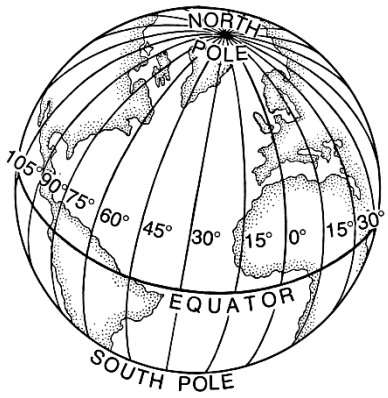


Abbildung 2: Längengrade (Wikipedia[2])

2.2 Lokale Entfernungsberechnung

Wir wollen im Folgenden die Entfernung zweier Orte auf der Erdoberfläche bestimmen. Liegen zwei Orte relativ dicht beieinander, kann man näherungsweise die Erdkrümmung vernachlässigen:

1. Bestimme die Abstände in Richtung der Breiten- und Längengrade in km.
2. Berechne hieraus den direkten Abstand durch den Satz des Pythagoras

[1] https://commons.wikimedia.org/wiki/File:Latitude_(PSF).png (public domain)
[2] https://commons.wikimedia.org/wiki/File:Longitude_(PSF).png (public domain)

Teilt man den Erdumfang in 360°, so entspricht je 1° in etwa 111,3 km. Da der Abstand benachbarter Breitengrade überall auf der Erde gleich groß ist (Abb. 1), entspricht ein Unterschied von 1° Breite jeweils 111,3 km. Der Abstand zweier Breitengrade lat_1 und lat_2 in km beträgt daher:

$$\Delta y = 111{,}3 \cdot |lat_1 - lat_2|$$

Der Abstand benachbarter Längengrade hängt vom Breitengrad der Orte ab. Am Äquator (0° Breite) entspricht der Abstand 111,3 km. Zu den Polen hin laufen die Längengrade aufeinander zu und der Abstand wird kleiner (Abb. 2). Am Nord- und Südpol schneiden sich die Längengrade, sodass dort der Abstand 0 km entspricht. Dies wird in folgender Formel durch den Kosinus ausgedrückt, der am Äquator (0° Breite) 1 und an den Polen (±90° Breite) 0 ergibt. Als Argument wird der Mittelwert der Breitengrade beider Orte verwendet:

$$\Delta x = 111{,}3 \cdot \cos\left(\frac{lat_1 + lat_2}{2}\right) \cdot |lon_1 - lon_2|$$

Insgesamt ergibt sich für den Abstand *d* in km zweier geografischer Orte g_1 und g_2:

$$d = \sqrt{\Delta x^2 + \Delta y^2}$$

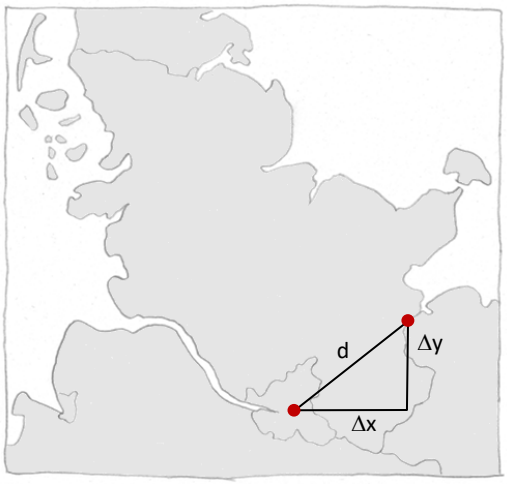


Abbildung 3: Entfernung zwischen Hamburg und Lübeck

2.3 Entfernungsberechnung auf der Erdkugel

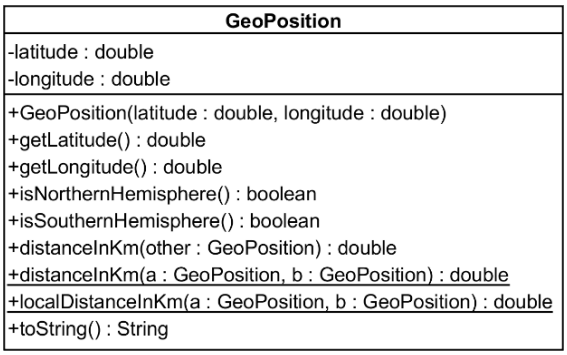
Für eine genauere Berechnung der Entfernung zweier Orte auf der Erdoberfläche sei ohne Herleitung nachfolgende Formel gegeben[3]. Hierbei entspricht der Faktor 6378,388 km dem Erdradius.

$$d = 6378{,}388 \cdot \arccos(\sin lat_1 \cdot \sin lat_2 + \cos lat_1 \cdot \cos lat_2 \cdot \cos(lon_2 - lon_1))$$

[3] https://www.kompf.de/gps/distcalc.html

3 Klasse *GeoPosition*

Es ist eine Klasse *GeoPosition* zur Repräsentation geografischer Koordinaten zu erstellen. Die Klasse wird grundlegend durch das nebenstehende UML-Symbol beschrieben. In dem Kasten unterhalb des Klassennamens sind die Attribute mit jeweiligem Datentyp angegeben. Durch ein Minuszeichen wird gekennzeichnet, dass der Modifier *private* zu verwenden ist.



Im unteren Abschnitt sind die Methoden mit Parameterlisten sowie nach dem Doppelpunkt dem Datentyp ihres Rückgabewertes aufgelistet. Unterstrichene Elemente sind Klassenmethoden.

3.1 Eclipse und Projektstruktur

a) Erstellen Sie in Eclipse ein Java-Projekt namens *Praktikum*, wobei Sie unter *„Project layout“* die Option *„Use project folder as root for sources and class files“* wählen. Erzeugen Sie im Projekt das Paket *lab1.geoPosition*.

b) Fügen Sie dem Projekt die aktuellste *JUnit*-Bibliothek hinzu.

c) Fügen Sie zum Paket die gegebene Test-Klasse *TestGeoPosition* hinzu. Sie können hierfür die Datei *TestGeoPosition.java* aus dem Windows-Explorer per „Drag&Drop“ in das Paket im Eclipse *Package Explorer* ziehen.

3.2 Attribute, Konstruktoren und Getter

Requirement 1 — Die Klasse besitzt private Variablen namens *latitude* und *longitude* zur Speicherung des Breitengrades bzw. Längengrades. Beide Variablen sind vom Typ *double*.

Requirement 2 — Es gibt einen Konstruktor mit zwei *double*-Parametern, der den Attributen die übergebenen Parameterwerte zuweist. (Tipp: Nutzen Sie in Eclipse den Menüpunkt „*Source / Generate …*“, um die Methode automatisch erzeugen zu lassen.)

Requirement 3 — Es gibt Getter-Methoden, die den Wert von *latitude* bzw. *longitude* zurückgeben. (Lassen Sie auch diese Methoden automatisch in Eclipse erzeugen.)

Deklarationen:

- `public GeoPosition(double latitude, double longitude)`
- `public double getLatitude()`
- `public double getLongitude()`

3.3 Abfrage der Hemisphäre

Requirement 4 — Es gibt Methoden zur Abfrage, ob sich eine geografische Position auf der nördlichen bzw. auf der südlichen Erdhalbkugel befindet.

Deklarationen:

- `public boolean isNorthernHemisphere()`
- `public boolean isSouthernHemisphere()`

3.4 Abstände zwischen zwei geografischen Orten

Requirement 5 — Es gibt zwei Klassenmethoden zur Berechnung des Abstandes („Luftlinie“) zwischen zwei als Parameter übergebenen Orten. Der Abstand wird jeweils in Kilometern zurückgegeben. Die Methode *localDistanceInKm()* verwendet zur Berechnung das lokale Verfahren nach Abschnitt 2.2. Die Methode *distanceInKm()* verwendet die genauere Berechnung nach Abschnitt 2.3.

Requirement 6 — Es gibt eine nicht-statische Methode *distanceInKm()*, die den Abstand zu einem als Parameter übergebenen Ort berechnet. Die Berechnung erfolgt nach Abschnitt 2.3.

Hinweise:

- Nutzen Sie für mathematische Funktionen die Methoden der Klasse *Math*.
- Beachten Sie, dass die trigonometrischen Funktionen Winkel in Radiant (nicht Grad) erwarten.

Deklaration:

- `public static double localDistanceInKm(GeoPosition a, GeoPosition b)`
- `public static double distanceInKm(GeoPosition a, GeoPosition b)`
- `public double distanceInKm(GeoPosition other)`

3.5 Methode *toString()*

Requirement 7 — Die Klasse besitzt eine Methode *toString()*, die eine wie folgt formatierte Zeichenkette zurückgibt: `(<latitude>, <longitude>)`. (Tipp: Lassen Sie das Grundgerüst dieser Methode automatisch in Eclipse erzeugen.)

Deklarationen:

- `public String toString()`

3.6 Unit-Tests

Requirement 8 — Stellen Sie sicher, dass alle gegebenen Unit-Tests fehlerfrei laufen.

4 Abstände

Erzeugen Sie eine ausführbare Klasse *GeoApp* (d.h. ein ausführbares Programm), das den Abstand der HAW Hamburg und den in der nachfolgenden Tabelle aufgelisteten Orten auf Konsole ausgibt. Verwenden Sie zur Abstandsbestimmung jeweils das lokale sowie das genaue Verfahren und übertragen Sie die Ergebnisse in die Tabelle. Tragen Sie zudem ein, um wie viel Prozent das lokale Verfahren vom genauen Abstand abweicht.

Ort	*Breitengrad*	*Längengrad*	*Entfernung km (genau)*	*Entfernung km (lokal)*	*Abweichung %*
HAW Hamburg	53,557078	10,023109	-	-	-
Eiffelturm	48,858363	2,294481			
Palma de Mallorca	39,562553	2,661947			
Las Vegas	36,156214	-115,148736			
Copacabana	-22,971177	-43,182543			
Waikiki Beach	21,281004	-157,837456			
Surfer's Paradise	-28,002695	153,431781			

Bestimmen Sie auf gleiche Weise die Entfernung der HAW zu den Polen und zum Äquator.

Ort	*Breitengrad*	*Längengrad*	*Entfernung km (genau)*	*Entfernung km (lokal)*	*Abweichung %*
Nordpol					
Äquator					
Südpol					

5 Einhaltung der Programmierrichtlinien (Qualität)

Stellen Sie sicher, dass alle in der über EMIL zur Verfügung gestellte Checkliste zur Softwarequalität aufgeführten Qualitätskriterien erfüllt sind.

6 Theoretische Fragen

1. Wie unterscheiden sich *primitive Variablen* und *Referenzvariablen*?
2. Wie unterscheiden sich *Instanzvariablen*, *Objektvariablen*, *Attribute* und *lokale Variablen*?
3. Was wäre die Folge, wenn Sie die Variable *latitude* mit dem Schlüsselwort *static* versehen?
4. Es gibt zwei Methoden mit Namen *distanceInKm()*. Erläutern Sie den Unterschied.