HSB
Hochschule Bremen
City University of Applied Sciences

Fakultät 4
Elektrotechnik und Informatik
Prof. Dr. Ing. Heiko Mosemann
Flughafenallee 10
28199 Bremen

0421-5905-5601
Heiko.Mosemann@hs-bremen.de

Bremen, November 2022

# *Aufgabenstellung Entwicklungsarbeit PROG im WiSe 22/23*

Liebe Menschen,

die Prüfungsform zum Modul PROG ist die Entwicklungsarbeit. Die Prüfungsordnung (Abschnitt II: Prüfungsformen § 7 Arten der Prüfungsleistungen, Studienleistungen) sagt:

Eine Entwicklungsarbeit besteht in der Erstellung und Demonstration einer Computer-Software einschließlich der zugehörigen Dokumentation. Die Dokumentation umfasst in der Regel folgende Dokumente:
- die Aufgabenstellung,
- die Anforderungsdefinition,
- den Entwurf,
- das Quellprogramm,
- die Testdokumentation,
- Benutzungshinweise und
- ein Anwendungsbeispiel.

Die Prüfungsleistung kann als Aufsichtsarbeit gestaltet werden; die Regelungen zu 1 gelten dann entsprechend.

Die Entwicklungsarbeit ist eine Einzelarbeit.

Geben Sie Ihre Entwicklungsarbeit als PDF über diesen Dateiaustauschdienst spätestens bis zum **27.01.2023 (23:59 MEZ)** ab.

Benennen Sie Ihre Datei sinnvoll (PROG_EA_Name_Vorname.pdf).

Die Demonstration Ihrer Entwicklungsarbeit wird am letzten Vorlesungstermin sein.

# Aufgabenstellungen

Sind und Zweck dieser Entwicklungsarbeit ist die praktische Umsetzung der in der Vorlesung behandelten Themen.

**Sie dürfen keine grafische Oberfläche (GUI) programmieren. Weder AWT, Swing, JavaFX noch andere GUI-Toolkits sind erlaubt!**

Es folgt eine Aufzählung grundsätzlicher Dinge, die Sie bei der Erstellung des Codes beachten sollen. Diese Liste erhebt keinen Anspruch auf Vollständigkeit.

- Programmieren Sie objektorientiert.
- Vermeiden Sie redundanten Code.
- Schreiben Sie modulare, überschaubare Methoden.
- Berücksichtigen Sie auch das Gelernte aus dem Modul ‚Grundlagen der Informatik'.
- Setzen Sie Design by Contract ein und kommentieren Sie Ihre Methoden mit Vor- und Nachbedingungen.
- Schreiben Sie sinnvolle Kommentare und Dokumentationskommentare.
- Benennen Sie Ihre Bezeichner sinnvoll und verständlich.
- Setzen Sie Konstanten ein, da Literale im Code nur als Konstanten verwendet werden sollten. Ausnahmen sind die in der Spezifikation angegebenen Literale wie z.B. ‚null', ‚true' und ‚false' und Formeln mit allgemein bekannten Werten, z.B. die 2 bei PI/2.
- Halten Sie die Dinge einfach und verständlich. Die Performance steht nicht im Vordergrund, sondern sauberer Code.

Sie sollten es vermeiden, nach ‚Lösungen' im Internet zu suchen. Sie werden sicherlich etwas finden, aber extrem selten genügt dieser Code meinen Vorstellungen von gutem Code. Sie müssen die Anforderungen eines Studiums erfüllen, und im Netz ist das Niveau oft deutlich niedriger. Aus meiner Sicht lernen Sie besser, wenn Sie sich an dem Skript, die empfohlene Literatur, die Dokumentation der Java-API und die Java-Spezifikation halten. Besonders die Spezifikation ist sehr hilfreich!

**1. Teilaufgabe**

Sie finden die Datei ‚LadeStationen.csv' in der AULIS-Gruppe zu diesem Modul. Die Daten habe ich von esri Deutschland. Es handelt sich um 21566 Datensätze über Ladestationen für E-Autos in Deutschland. Laden Sie die Daten aus der oben genannten Datei ein. Sie dürfen dafür nur die Streams aus der Java API verwenden. Organisieren Sie die Daten in einer Collection. Legen Sie die dafür notwendigen Klassen an. Eine Zeile der Datei enthält die Werte der Attribute Betreiber, Straße, Hausnummer, Postleitzahl, Ort, Bundesland, Breitengrad, Längengrad und Anschlussleistung. In den Ursprungsdaten sind mir einige fehlerhafte Daten aufgefallen, achten Sie besonders auf Plausibilität der Daten Breitengrad und Längengrad, da diese Daten für die Aufgabenstellung sehr relevant sind. Das Trennzeichen in der csv-Datei ist das Semikolon. Geben Sie die Bearbeitungszeiten des Einlesens der Datei und der Überprüfung der Daten an. Machen Sie sinnvolle Ausgaben. Hier ein Beispiel möglicher Ausgaben:

Lade Daten aus Datei: ./src/res/LadeStationen.csv
Eingelesene Zeilen: 21566 (28 Millisekunden)

Trage Ladestation nicht ein. Breiten- oder Längengrad nicht korrekt:
GeoPosition{breitenGrad=47.55352, laengenGrad=9703.0}
Trage Ladestation nicht ein. Breiten- oder Längengrad nicht korrekt:
GeoPosition{breitenGrad=52399.0, laengenGrad=13.06617}
…
Trage Ladestation nicht ein. Breiten- oder Längengrad nicht korrekt:
GeoPosition{breitenGrad=51447.0, laengenGrad=10.57188}
Trage Ladestation nicht ein. Breiten- oder Längengrad nicht korrekt:
GeoPosition{breitenGrad=51212.0, laengenGrad=10.46145}

Anzahl valider Ladestationen: 21500 (41 Millisekunden)

**2. Teilaufgabe**
Sortieren Sie die eingelesenen Ladestationen nach Postleitzahl (niedrige Zahlen stehen oben) und innerhalb der selben Postleitzahlen nach Leistung (höhere Leistung steht oben). Implementieren Sie eine sinnvolle Ausgabe der sortierten Daten. Geben Sie die Bearbeitungszeiten der Sortierung aus.

Hier ein Beispiel möglicher Ausgaben:
Anzahl Ladestationen: 21500 (in 43 Millisekunden sortiert)
…
EnBW mobility+ AG und Co.KG , Ammersche Landstraße 27-30,99974 Mühlhausen, 150.0
EnBW mobility+ AG und Co.KG , Ammersche Landstraße 27-31,99974 Mühlhausen, 150.0
Langenhan Mühlhausen GmbH, Am Brühl 3,99974 Ammern, 44.0
Stadtwerke Mühlhausen GmbH, Brunnenstraße 97,99974 Mühlhausen, 22.0
Stadtwerke Mühlhausen GmbH, Windeberger Landstr. 73,99974 Mühlhausen, 22.0
Thüringer Energie AG, Gedeplatz 0,99976 Lengenfeld unterm Stein, 30.0
Thüringer Energie AG, Karlstraße 26,99988 Heyerode, 30.0
Thüringer Energie AG, Mühlhäuser Str. 18,99991 Großengottern, 86.0
Thüringer Energie AG, Hauptstraße 46,99991 Altengottern, 30.0

**3. Teilaufgabe**
Dem Programm werden zwei Argumente übergeben. Das erste Argument ist die Epsilonumgebung und soll eine natürliche Zahl sein. Die Epsilonumgebung ist der Mindestabstand von Ladestationen. Das zweite Argument ist die maximale Entfernung zwischen Ladestationen und soll eine natürliche Zahl sein.

Nach dem Start des Programms überprüfen Sie die Argumente auf Plausibilität. Das Programm informiert die nutzende Person über fehlerhafte Argumente.

Sind die Argumente korrekt, dann durchlaufen Sie die in Teilaufgabe 2 sortierte Liste und löschen alle Stationen, die sich innerhalb der Epsilonumgebung in Bezug auf eine Referenzstation befinden.

Sie starten bei der ersten Ladestation in der Liste (Referenz) und prüfen die Lage dieser Station zu den Lagen aller anderen Stationen (Vergleich) in der Liste. Dazu verwenden Sie den Breiten- und Längengrad. Setzen Sie die Haversine-Formel ein, die die Entfernung

zweier Punkte auf einer Kugel unter Verwendung ihrer Längen- und Breitengrade berechnet.

Ist der Abstand der Vergleichstation kleiner oder gleich der Epsilonumgebung zur Referenzstation wird die Vergleichsstation aus der Liste gelöscht. Haben sie die Liste für die erste Referenzstation vollständig durchlaufen, dann wird die nächste Station in der Liste zur Referenzstation.

Sie können auch schon ab Teilaufgabe 1 Collections einsetzen, die beim Eintragen der Ladestationen diese Anforderungen erfüllen. Ich habe hier die sehr einfache Vorgehensweise beschrieben.

Implementieren Sie sinnvolle Ausgaben und geben Sie die verbrauchte Zeit an.

Hier ein Beispiel möglicher Ausgaben:
Programmaufruf mit den Argumenten: 25 250
Lade Daten aus Datei: ./src/res/LadeStationen.csv
Prüfe auf Redundanz mit Epsilon = 25
Anzahl Ladestationen nach Löschung redundanter Stationen: 378 (Millisekunden: 2256)
SachsenEnergie AG, Weißeritzstraße 0A,1067 Dresden, 175.0
SachsenEnergie AG, Markt 3Z,1313 Strehla, 75.0
SachsenEnergie AG, Am Fiebig 1,1561 Thiendorf, 175.0
Allego GmbH, Im Industriegebiet 1,1683 Nossen, 93.0
SachsenEnergie AG, Dresdner Str. 17,1773 Altenberg, 43.6
SachsenEnergie AG, Schmilka 20Z,1814 Bad Schandau, 44.0
SachsenEnergie AG, Am Markt 8,1824 Elstra, 30.0
ASS Automobile GmbH, Knappenstr. 2,1968 Senftenberg, 110.0
SachsenEnergie AG, August-Bebel-Platz 8,2627 Weißenberg, 30.0
H.-J. Paul Fahrzeuge GmbH u Co, Hauptstraße 100,2727 Neugersdorf, 13.2
SachsenEnergie AG, Schlossplatz 0,2929 Rothenburg / Oberlausitz, 75.0
EnBW mobility+ AG und Co.KG , Sachsendamm 32,2943 Weißwasser/O.L., 300.0
EnBW mobility+ AG und Co.KG , Muskauer Platz 3,3042 Cottbus, 300.0
…

**4. Teilaufgaben**
In dieser Teilaufgabe implementieren Sie einen Graphen, mit dem ein Wegenetz der Ladestationen aufgebaut wird. Das zweite Argument ist die maximale Entfernung zwischen Ladestationen. Gehen Sie die Liste durch und erzeugen Sie für jede Ladestation einen Knoten im Graphen. Ist der Abstand zwischen zwei Knoten kleiner oder gleich der maximalen Entfernung, tragen Sie eine Kante zwischen diesen Knoten ein.

Sie dürfen keine Bibliotheken wie JGraphT verwenden, sondern sollen den ADT Graph selbst implementieren.

Man kann über die Epsilonumgebung die Anzahl der Ladestationen in der Liste reduzieren. Nutzen Sie das, um die Anzahl der Knoten im Graphen für Ihre Tests überschaubar zu halten.

Sie haben im Modul ‚Grundlagen der Informatik' Graphen als abstrakte Datentypen (ADT) kennen gelernt und wie man diese implementiert. Man verwendet dazu die impliziten

Datenstrukturen Adjazenz-Liste oder Adjazenz-Matrix. Bewerten Sie diese beiden Möglichkeiten für sich und entscheiden Sie sich für die für diese Aufgabe geeignete Lösung. Geben Sie den Graphen ansprechend aus und informieren Sie wieder über den Zeitverbrauch.

Hier ein Beispiel möglicher Ausgaben:
Programmaufruf mit den Argumenten: 50 400
…
Konstruiere Graph mit maximaler Entfernung von 400
Anzahl Ladestationen: 116 (Graph in 72 Millisekunden konstruiert)
…
86153 Augsburg -> 3448 Meuselwitz, 4420 Markranstädt, 6268 Querfurt, 6346 Bruchköbel, 7318 Saalfeld, 9111 Chemnitz, 9505 Litzendorf, 18184 Roggentin, 23795 Bad Segeberg, 23816 Leezen, 33181 Bad Wünnenberg, 34134 Kassel, 35037 Marburg, 35394 Gießen, 35781 Weilburg, 36037 Fulda, 37115 Duderstadt, 37293 Herleshausen, 37308 Heilbad Heiligenstadt, 50968 Siegburg, 54291 Trier, 54457 Wincheringen, 54689 Daleiden, 55590 Meisenheim, 56070 Koblenz, 57078 Siegen, 63762 Großosteim, 63897 Miltenberg, 66111 Saarbrücken, 66994 Dahn, 67157 Wachenheim, 70188 Stuttgart, 72393 Burladingen, 72525 Münsingen, 72669 Kressbronn, 73430 Aalen, 74076 Heilbronn, 74575 Schrozberg, 76131 Karlsruhe, 77694 Kehl, 78224 Singen, 79111 Freiburg, 79713 Bad Säckingen, 82319 Starnberg, 83022 Rosenheim, 83278 Traunstein, 84028 Landshut, 84234 Nagel, 84347 Pfarrkirchen, 85049 Ingolstadt, 86485 Biberbach, 87527 Sonthofen, 90616 Neuhof an der Zenn, 92242 Hirschau, 93047 Regensburg, 93444 Bad Kötzting, 96476 Bad Rodach,

48145 Münster -> 3448 Meuselwitz, 4420 Markranstädt, 4509 Delitzsch, 4758 Oschatz, 6268 Querfurt, 6346 Bruchköbel, 6429 Nienburg (Saale), 6502 Thale, 6895 Zahna-Elster, 7318 Saalfeld, 9111 Chemnitz, 9505 Litzendorf, 10115 Berlin, 14712 Rathenow, 15837 Baruth/Mark, 16831 Rheinsberg, 17166 Teterow, 18069 Rostock, 18184 Roggentin, 19053 Schwerin, 19258 Boizenburg / Elbe, 19322 Wittenberge, 20095 Hamburg, 21737 Wischhafen, 23188 Edewecht, 23552 Lübeck, 23730 Neustadt/Holstein, 23769 Burg auf Fehmarn, 23795 Bad Segeberg, 23816 Leezen, 24103 Kiel, 24395 Nieby, 24896 Treia, 25761 Büsum, 25924 Rodenäs, 26427 Bensersiel, 26757 Borkum, 27211 Bassum, 27356 Rotenburg, 29221 Celle, 29313 Hambühren, 29468 Bergen, 30161 Hannover, 31188 Holle, 31515 Steinhude, 32049 Herford, 33034 Brakel, 33181 Bad Wünnenberg, 34134 Kassel, 35037 Marburg, 35394 Gießen, 35781 Weilburg, 36037 Fulda, 37115 Duderstadt, 37293 Herleshausen, 37308 Heilbad Heiligenstadt, 38350 Helmstedt, 40210 Düsseldorf, 44579 Castrop-Rauxel, 46325 Borken, 47559 Kranenburg, 47623 Kevelaer, 48455 Bad Bentheim, 49733 Haren, 50226 Frechen, 50968 Siegburg, 54457 Wincheringen, 54689 Daleiden, 55590 Meisenheim, 56070 Koblenz, 57078 Siegen, 63897 Miltenberg, 66111 Saarbrücken, 66994 Dahn, 67157 Wachenheim, 73430 Aalen, 74076 Heilbronn, 74575 Schrozberg, 76131 Karlsruhe, 77694 Kehl, 84234 Nagel, 90616 Neuhof an der Zenn, 96476 Bad Rodach,

**5. Teilaufgabe**
In der letzten Teilaufgaben implementieren Sie eine Suche von einem Start-Ort zu einem Ziel-Ort. Sie fragen die nutzende Person nach der Eingabe einer Postleitzahl (über die Konsole) für den Start-Ort. Sie müssen sicherstellen, dass diese Postleitzahl zu einem Knoten im Graphen gehört. Dann erfragen Sie den Ziel-Ort. Geben Sie das Ergebnis und

die Berechnungszeit aus. Es geht hier nur um einen möglichen (nicht unbedingt den kürzesten) Weg vom Start zum Ziel.

Hier ein Beispiel möglicher Ein- und Ausgaben:
Bitte die Postleitzahl (muss im Graphen sein) des Startknotens eingeben:
48145
Bitte die Postleitzahl des Zielknotens eingeben:
19053
Suche einen Pfad Start/Ziel: 48145, 19053
Diese Postleitzahl ist von 48145 erreichbar: 19053 (Berechnungszeit 129 Millisekunden)

**Fazit**
Mit dieser Aufgabenstellung haben wir alle Kapitel der Vorlesung behandelt. Die Aufgabe ist sehr praxisnah und legt den Grundstein für eine Suche nach dem kürzesten Pfad in einem Wegenetz. Solche Probleme gibt es in der Praxis immer wieder. Ich habe bewusst auf die Berechnung des kürzesten Pfades verzichtet, da dies aus meiner Sicht für eine Entwicklungsarbeit im ersten Semester zu umfangreich ist (nicht alle von Ihnen haben Vorerfahrung im Programmieren). Gerne können Sie für sich selbst die Suche nach dem kürzesten Weg implementieren (aber nicht in der Entwicklungsarbeit abgeben!). Dazu nutzen Sie bitte den A*-Algorithmus.