LK VI

Projekt: BABRoutenplaner

**Vorbemerkung**

In diesem Projekt werden die Kompetenzen und fachlichen Inhalte, die Schülerinnen und Schüler im Leistungskurs Informatik in den beiden Unterrichtsvorhaben Q1-IV: „Datenbanken“ und Q1-V: „Graphen und Bäume“ erworben haben, miteinander verknüpft und vertieft. Mithilfe der vorgegebenen Klasse DataBaseConnector erhalten sie die Möglichkeit, in Java-Programmen auf die Daten von Datenbanken zuzugreifen und so in der Praxis sehr weit verbreitete kontextbezogene Anwendungen mit Datenbankanbindung zu entwickeln. Der Unterricht sollte projektartig angelegt werden, d.h. die Schülerinnen und Schüler sollen in Gruppen und arbeitsteilig weitgehend selbstständig arbeiten mit dem Ziel, ein fertiges Produkt, in diesem Fall einen funktionierenden Routenplaner für Autobahnen, zu entwickeln. Aufgabe der Lehrerin oder des Lehrers ist es, den notwendigen Input und die einzuhaltenden Rahmenbedingungen zu geben und in die Steuerung des Projektes einzugreifen, wenn nicht zielgerichtet genug gearbeitet wird. Im Sinne der agilen Programmierung sollte die Programmentwicklung in Schichten und in Modulen erfolgen, die vollständig modelliert, implementiert und getestet werden, bevor die nächste Erweiterung begonnen wird. Bei konsequenter Einhaltung dieses Weges besteht eine große Chance, das Projekt mit einem lauffähigen Programm abzuschließen, auch wenn nicht alle Funktionalitäten, die anfangs festgelegt wurden, enthalten sind. Neben der Gruppenarbeit muss es regelmäßige Besprechungen mit der ganzen Lerngruppe geben, in der Zwischenergebnisse präsentiert, Schnittstellen abgesprochen und Fragen geklärt werden können.

Die Autobahn-Datenbank wird von der Lehrerin oder vom Lehrer vorgegeben. Das hat den Nachteil, dass die Schülerinnen und Schüler an deren Entwurf nicht beteiligt sind, bietet aber die Möglichkeit mit einer großen Zahl echter Daten zu arbeiten, die nicht erst mühselig erfasst werden müssen. Dennoch sollte nach Entscheidung für dieses Projekt der Kurs eine sinnvolle Struktur für eine Autobahndatenbank erarbeiten, bevor die fertige Datenbank vorgegeben wird. Im Folgenden wird ein Weg zum Ablauf des Projektes beschrieben und mögliche Lösungen angegeben, die selbstverständlich nicht verbindlich sind.

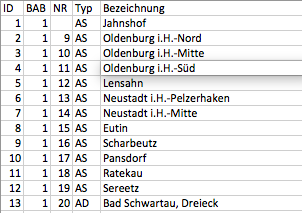
**Projektbeschreibung**

In diesem Projekt soll ein Routenplaner für das deutsche Autobahnnetz entwickelt werden. Nach Eingabe einer beliebigen Autobahn-Auffahrt und einer -Abfahrt wird der kürzeste Weg – sofern vorhanden - zwischen den beiden Anschlussstellen angezeigt.

**Struktur der Datenbank**

Die in diesem Projekt verwendete Datenbank *Autobahnen* besteht aus den beiden Tabellen *BABAnschluesse* und *Entfernungen*.

Die Tabelle *BABAnschluesse* hat folgende Struktur:



*ID*: eindeutiger Index des Datensatzes

*BAB*: Nummer der Autobahn

*NR*: Nummer der Anschlussstelle. Ist nicht in jedem Datensatz vorhanden.

*Typ*: AS: Auf- oder Abfahrt

AK: Autobahnkreuz

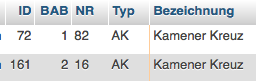
AD: Autobahndreieck

AN: Autobahnanfang oder Ende

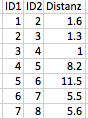
BG: Bundesgrenze

*Bezeichnung*: Name der Anschlussstelle

Für Autobahnkreuze und die meisten Dreiecke gibt es zwei Datensätze in der Tabelle.



Die Tabelle *Entfernungen* enthält die Entfernung in km zwischen zwei benachbarten Anschlussstellen und hat folgende Struktur:



Dabei sind ID1 und ID2 die Identifikationsnummern(ID) von zwei benachbarten Anschlussstellen in der Tabelle *BABAnschluesse* und *Distanz* die Entfernung in km zwischen den beiden Anschlussstellen .

Damit liegt der Datenbank *Autobahnen* folgendes Datenbankschema zugrunde:

BABAnschluesse(ID, BAB, Nr, Typ, Bezeichnung)

Strecke(↑ID1, ↑ID2, Distanz)

**Verwendete Software**

Entwicklungsumgebung: Bluej

Datenbank: mySQL unter Apache (installiert mit XAMPP) <https://www.apachefriends.org/de/>

Klassenbibliotheken:

* Abiturklassen zum Graphen: graph.jar (s. Download LK-Q1.V.2)
* Abiturklassen zur Datenbankanbindung: dbConnect.jar (s. Download LK-Q1-VI.1)

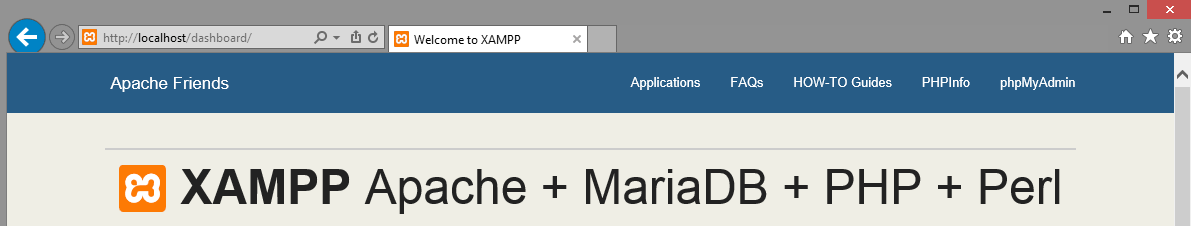
Klassen (s. Download LK-Q1.V.2):

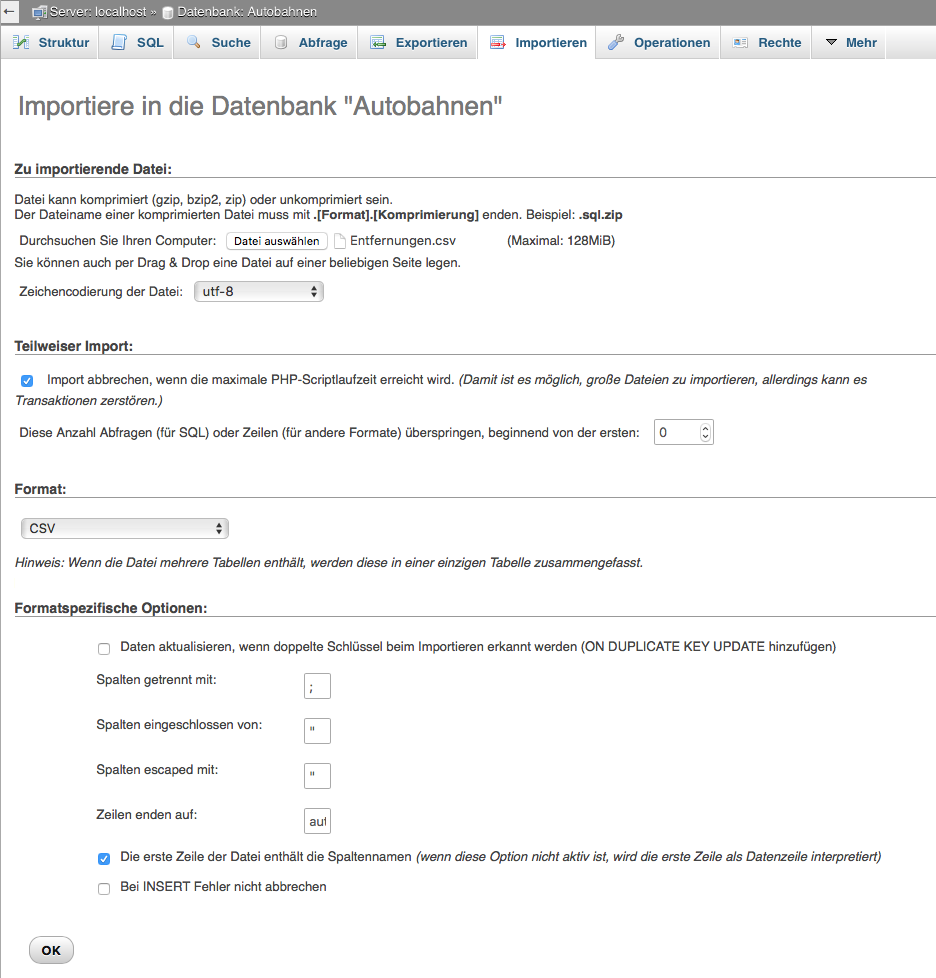
* DijkstraVertex.java
* PrioritaetsSchlange.java
* Dijkstra.java

**Projektphasen**:

1. Einrichtung der Datenbank
2. Anwendung der Klasse DatabaseConnector
3. SQL-Abfragen an die Datenbank
4. Modellierung der Benutzungsoberfläche
5. Modellierung des Implementationsdiagramms
6. Implementation der Software
   1. Laden der Datenbank in einen Graphen
   2. Berechnung des kürzesten Weges zwischen Auf- und Abfahrt
   3. Ausgabe der Route
7. Reflexion und Erweiterungsmöglichkeiten
8. **Einrichtung der Datenbank**

* Apache und mySQL im XAMPP-Control-Panel starten (ggf. Ports in Firewall freigeben)
* XAMPP-Startseite in Browser aufrufen mit [*http://localhost*](http://localhost)



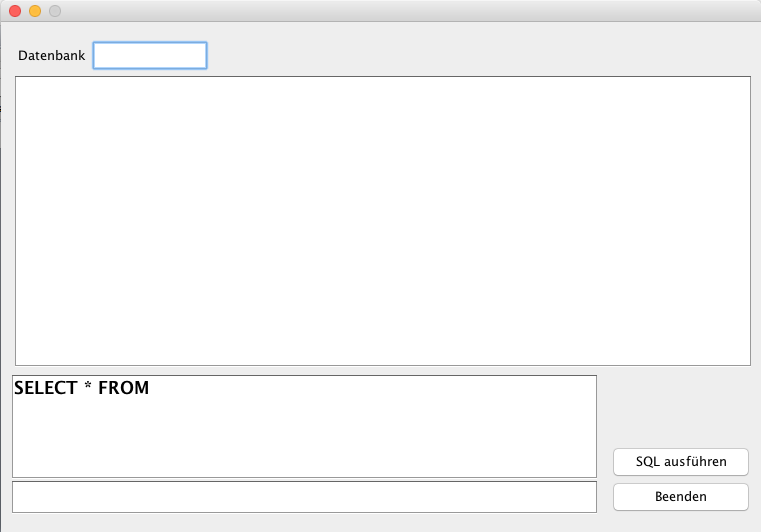
* phpMyAdmin aufrufen
* Datenbank *Autobahnen* in phpMyAdmin anlegen
* SQL-Datei *Autobahnen.sql*, die beide Tabellen enthält, in die Datenbank importieren oder die Tabellen *BABAnschluesse.csv* und *Entfernungen.csv* in die Datenbank importieren.
* Einstellungen für den Import der csv-Dateien
* Die rot umrahmten Textfelder müssen entsprechend der Abbildung ausgefüllt werden.
* Die Namen der importierten Tabellen werden in *BABAnschluesse bzw. Entfernungen* umbenannt.
* Die Datenbankstruktur der Tabellen wird entsprechend dem o.a. Datenbankschema angepasst.

Ist die Datenbank fertig eingerichtet, lassen sich mithilfe von phpMyAdmin erste SQL-Abfragen an sie richten.

1. **Anwendung der Klasse DataBaseConnector**

Die Verbindung zwischen einem Java-Programm und einer SQL-Datenbank kann mithilfe der Klassenbibliothek DBConnector hergestellt werden. Diese enthält die beiden Klassen DatabaseConnector und QueryResult, deren Objekte Datenbanken im Netz öffnen und SQL-Abfragen an sie richten können. Eine ausführliche Dokumentation befindet sich in den Materialien zu diesem Unterrichtsvorhaben. Bevor die Schülerinnen und Schüler mit der Entwicklung des Routenplaners beginnen, sollten sie sich mit der Anwendung dieser Klassenbibliothek vertraut machen. Dazu bietet es sich an,ein Java-Tool zu entwickeln, das SQL-Abfragen an eine Datenbank richten und die Ergebnisse anzeigen kann. Eine Benutzungsoberfläche, z.B. mit folgender Struktur, kann von der Lehrkraft vorgegeben werden:

Name der Datenbank



Fehlermeldungen

Ergebnistabelle

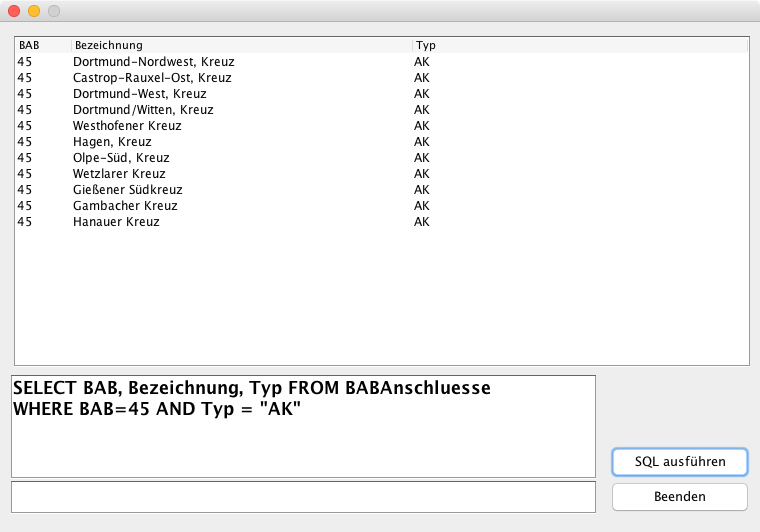
SQL- Abfrage

Beispiel:

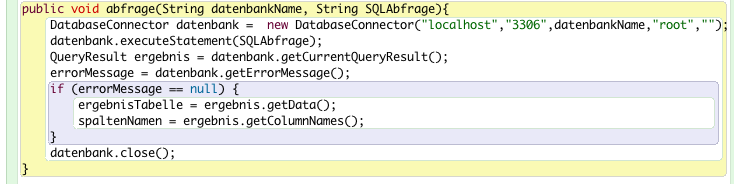
Die SQL-Abfrage:

SELECT BAB, Bezeichnung, AS\_Typ FROM BABAnschluesse WHERE BAB=45 AND AS\_Typ = “AK“

liefert als Ergebnis alle Autobahnkreuze der A45.



Die folgende Methode abfrage verdeutlicht die Anwendung der DBConnector-Klassen.



Die Variablen, ergebnisTabelle, errorMessage und spaltenNamen sind Attributvariablen der Klasse, die die Methode enthält.

1. **SQL-Abfragen an die Datenbank**

Um den Aufbau eines Graphen aus der Datenbank vorzubereiten, werden in dieser Phase zur Übung einige SQL-Abfragen von den Schülerinnen und Schüler entwickelt.

**Anmerkung**

Bei den folgenden Aufgabenstellungen umfasst der Begriff „Anschlussstellen“ alle Auf- und Abfahrten sowie Autobahnkreuze- und Dreiecke. Auch einige Tankstellen, an denen es eine Wendmöglichkeit gibt, sind in der Datenbank enthalten.

**Aufgabe**

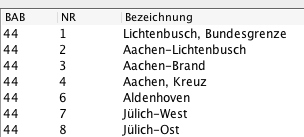
Entwickeln Sie eine SQL-Abfrage, die

1. die Autobahnnummer sowie die Nummern und die Bezeichnungen aller Anschlussstellen der A44 zurückgibt.
2. die Anzahl der Anschlussstellen der A1 zurückgibt.
3. die Längen aller Autobahnen berechnet und ausgibt.
4. die Bezeichnungen der beiden zu der AS Nottuln benachbarten Anschlussstellen und deren Entfernungen liefert.

Lösungen:

a) SELECT BAB, AS\_NR, Bezeichnung NR FROM BABAnschluesse

WHERE BAB=44



b) SELECT COUNT(ID) AS Anzahl FROM BABAnschluesse

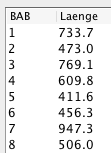
WHERE BAB = 1

Macintosh HD:Users:klausdingemann:Desktop:Bildschirmfoto 2015-11-20 um 09.31.50.png

c) SELECT BAB, SUM(Distanz) AS Laenge FROM BABAnschluesse,

Entfernungen WHERE BABAnschluesse.ID = Entfernungen.ID1

GROUP BY BAB



d) SELECT Bezeichnung, Distanz

FROM BABAnschluesse, Entfernungen

WHERE (BABAnschluesse.ID = Entfernungen.ID2 AND

BABAnschluesse.ID IN

(SELECT ID2 FROM BABAnschluesse, Entfernungen

WHERE Bezeichnung = "Nottuln"

AND BABAnschluesse.ID = Entfernungen.ID1)) OR

(BABAnschluesse.ID = Entfernungen.ID2 AND

BABAnschluesse.ID IN

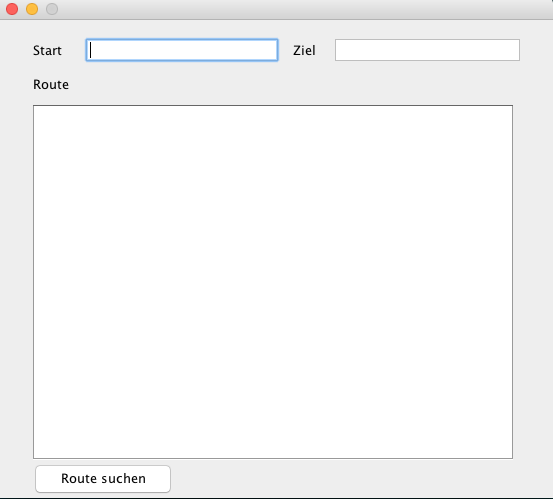
(SELECT ID1 FROM BABAnschluesse, Entfernungen

WHERE Bezeichnung = "Nottuln"

AND BABAnschluesse.ID = Entfernungen.ID2))



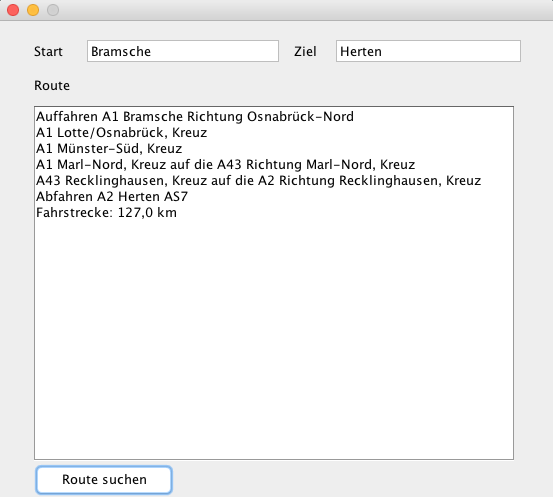
1. **Modellierung der Benutzungsoberfläche**

Die Benutzungsoberfläche sollten die Schülerinnen und Schüler weitgehend selbstständig entwerfen. Die Lehrkraft sollte darauf drängen, zunächst eine möglichst einfache Lösung zu konzipieren. Für einen Routenplaner gibt es im Internet viele Beispiele. Das Fenster muss jeweils ein Textfeld für die Eingabe der Startauffahrt und der Zielabfahrt enthalten sowie einen Textbereich für die Ausgabe der Route. Darüber hinaus wird ein Knopf benötigt, mit dem der Suchprozess gestartet wird.

Auf weitere wünschenswerte Funktionalitäten wie z.B. das Speichern oder das Drucken der Route sollte für die erste Programmversion verzichtet werden.

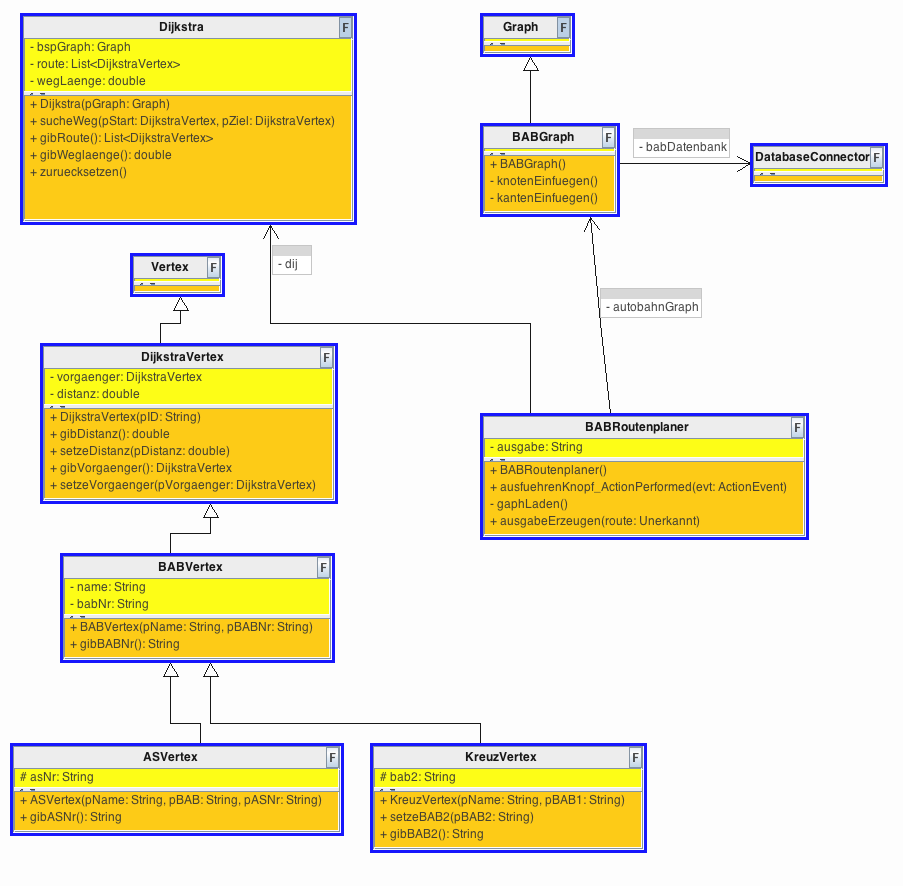
In dieser Phase können auch erste Überlegungen angestellt werden, wie die Ein- und Ausgabe der Daten gestaltet werden soll. Damit der Routenplaner einen Weg findet, müssen die Namen der Start- und die Zielanschlussstelle exakt so eingegeben werden, wie sie in der Datenbank gespeichert sind. Alternativ wäre es auch möglich, jeweils die Autobahnnummer und die Nummer der Anschlussstelle einzugeben. Komfortable Möglichkeiten wie die Auswahl aus einem Menü nach Eingabe weniger Buchstaben sollte für spätere Erweiterungen des Projekts zurückgestellt werden. Auch für die Ausgabe der Route gibt es mehrere Möglichkeiten. Die einfachste Lösung ist die Angabe aller Abfahrten, Kreuze und Dreiecke, die der Fahrer oder Fahrerin vom Start zum Ziel passiert. Damit wird bei längeren Fahrten die Liste sehr lang und enthält viele unnötige Informationen. Es ist völlig ausreichend, wenn jeweils nur das nächste Autobahnkreuz oder -dreieck angegeben wird sowie die Richtung, wenn die Autobahn gewechselt werden muss.





Selbstverständlich sollte wie beim zweiten Beispiel auch die Länge der Fahrtstrecke angegeben werden.

1. **Modellierung des Implementationsdiagramms**



Das Implementationsdiagramm für den Routenplaner enthält die wichtigsten Klassen und deren Methoden. Nicht alle verwendeten Klassen, die in den Klassenbibliotheken enthalten sind, wurden aufgeführt. Außerdem wurde auf die Attribute der Klasse Routenplaner verzichtet, die auf die Objekte der Benutzungsoberfläche verweisen.

**BABGraph**

Der Konstruktor dieser Klasse erzeugt unter Verwendung der Klasse DatabaseConnector den Graphen des Autobahnnetzes aus der Datenbank. Die Methode knotenEinfuegen erzeugt für jede Anschlussstelle, die in der Tabelle Autobahnen der Datenbank enthalten ist, einen Knoten des Graphen. Dabei ist zu beachten, dass für Kreuze oder Autobahndreiecke, die u.U. für verschiedene Autobahnen in der Datenbank enthalten sind, nur ein Graphknoten erzeugt werden darf. Die Methode kantenEinfuegen erzeugt für jeden Datensatz der Tabelle Entfernungen der Datenbank eine Kante im Graphen.

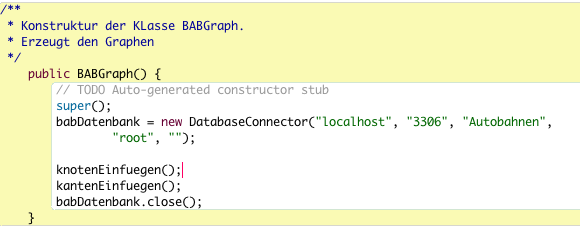
**BABRoutenplaner**

Der Konstruktor dieser Klasse erzeugt die Benutzungsoberfläche der Anwendung und ruft die Methode graphLaden auf, die ein Objekt der Klasse BABGraph erzeugt und damit den Graph des Autobahnnetzes konstruiert. Außerdem wir ein Objekt der Klasse Dijkstra erzeugt, um den Dijkstra-Algorithmus auf den Autobahngraphen anwenden zu können. Die Methode ausfuehrenKnopf\_actionPerformed wird durch Betätigung des Knopfes „Route suchen“ aufgerufen. Sie berechnet mithilfe der Methode sucheWeg des Dijkstra-Objektes den kürzesten Weg zwischen den in den beiden Textfeldern eingegebenen Anschlussstellen und gibt anschließend den mit der Methode ausgabeErzeugen formatierten Text zur Beschreibung der Route im Textbereich aus.

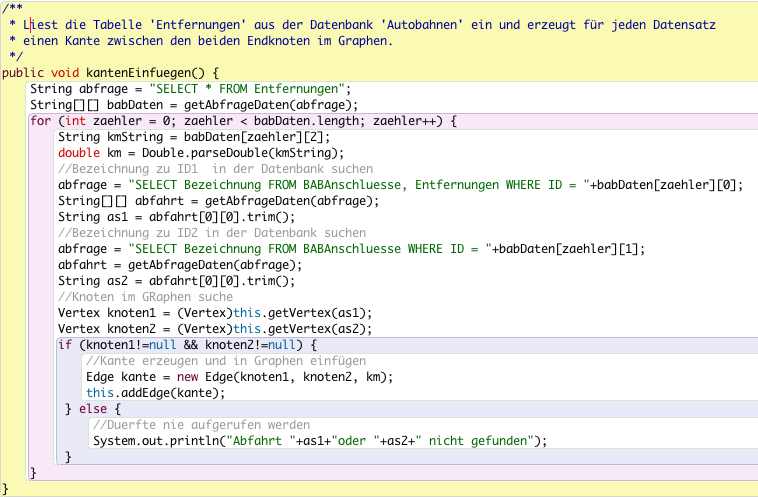
Die Funktionalität aller anderen Klassen und deren Methoden ist schon aus vorherigen Unterrichtsvorhaben bekannt oder erschließt sich unmittelbar aus dem Implementationsdiagramm. Die Klasse Dijkstra kann unverändert aus den Materialien zum Unterrichtsvorhaben LKQ-1.V entnommen werden.

1. Implementation der Software
   1. Laden der Datenbank in einen Graphen

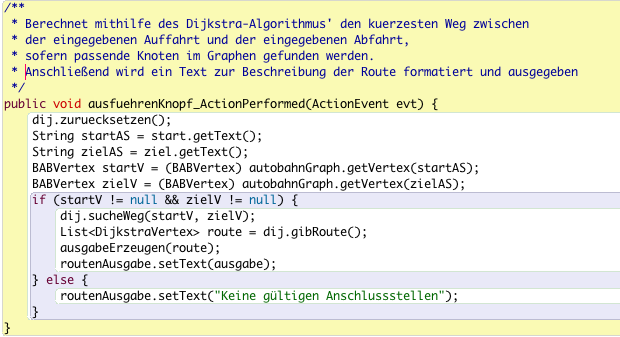
Der Graph des Autobahnnetzes ist eine Instanz der Klasse BABGraph.





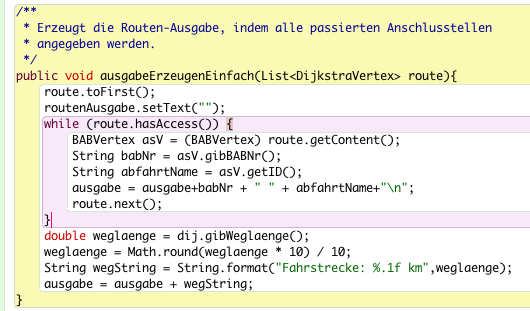


* 1. Berechnung des kürzesten Weges zwischen Auf- und Abfahrt



* 1. Ausgabe der Route

Die einfachste Form zur Beschreibung der Route besteht darin, alle Abfahrten, Kreuze und Dreiecke anzugeben, die auf der Route vom Start zum Ziel passiert werden. Die folgende Methode erzeugt diese Form der Ausgabe:



Übersichtlicher ist die Wegbeschreibung, wenn nur die Auffahrt, die Abfahrt sowie alle passierten Autobahnkreuze und –dreiecke mit der jeweiligen Fahrtrichtung und den notwendigen Autobahnwechseln angegeben werden. (s. Abbildung unter 4.). Der Algorithmus für diese Variante ist dem Quelltext der Klasse BABRoutenplaner zu entnehmen.

1. Reflexion und Erweiterungsmöglichkeiten

Abschließend werden sowohl das Ergebnis des Projektes als auch das methodische Vorgehen im Unterricht reflektiert. Insbesondere werden vom konkreten Projekt abstrahierend ein allgemeines Schema sowie Kriterien zur erfolgreichen Realisierung von Softwareprojekten erarbeitet.

Folgende Aspekte könnten in der Reflexion zum Beispiel angesprochen werden.

1. Erwartete und erreichte Funktionalität des Programms
2. Teamarbeit und Kooperation
3. Schwierigkeiten und sinnvolle Hilfen
4. Gestaltung der Entwicklungsphasen
5. Dokumentation des Projekts
6. Erweiterungsmöglichkeiten

Wenn noch Zeit verbleibt, kann die Software noch um unterschiedliche Funktionalitäten erweitert werden. Es besteht aber auch die Möglichkeit, dass Schülerinnen und Schüler diese Erweiterungen z.B. im Rahmen einer Facharbeit vornehmen.

Mögliche Erweiterungen:

* Verbesserung der Eingabe durch ein Auswahlmenü
* Staus, die in einer Tabelle der Datenbank stehen, bei der Routenplanung zu berücksichtigen
* Fahrtdauer unter Berücksichtigung von Geschwindigkeitsbegrenzungen (Tabelle in der Datenbank) berechnen
* Schnellste Route finden