Didaktische Hinweise
Projekte im Bereich Datenbankenanwendung

Zielgruppe

Die Materialien und Projektideen richten sich an Schülerinnen und Schüler in der Qualifikationsphase, die einen Informatikkurs auf grundlegendem oder erhöhtem Niveau belegt haben. Das niedersächsische Kerncurriculum für die gymnasiale Oberstufe (Niedersächsisches Kultusministerium, 2017) sieht für die Qualifikationsphase die selbständige Arbeit der Schülerinnen und Schüler an einem größeren Projekt vor. Die hier vorgestellte Entwicklung einer Datenbankanwendung kann als Rahmen für ein solches Projekt dienen. Dabei werden Kompetenzen im Umgang mit Datenbanken mit Kompetenzen aus dem Bereich Algorithmik verknüpft.

Voraussetzungen und Lernziele

In Informatikkursen werden die Themen Algorithmik und Datenbanken häufig getrennt voneinander unterrichtet. So erlernen die Schülerinnen und Schüler mit *SQL* eine andere Art der Programmierung, wie es in den *Einheitlichen Prüfungsanforderungen* (EPA) *Informatik* (Kultusministerkonferenz, 2004) gefordert wird. Da sie bei der Verwendung von Datenbanken im Alltag aber nicht selbst SQL-Anweisungen formulieren müssen, ist für die Lernenden nicht ersichtlich, wo Datenbanken und SQL-Abfragen zum Einsatz kommen. Ziel der Projektvorschläge ist es daher, die beiden Welten der imperativen bzw. objektorientierten Programmierung einer Anwendung und der Erstellung von SQL-Abfragen zusammenzubringen. Durch die Konstruktion eines eigenen Programms, das eine Datenbank verwendet, sollen das Zusammenspiel der Konzepte und die Einsatzmöglichkeiten von Datenbanken transparenter werden.

Eine Hürde bei der Implementierung einer Datenbankanwendung, die die imperative und objektorientierte Programmierung mit der Erstellung von SQL-Anfragen verknüpft, ist häufig, dass für die Kommunikation mit einer Datenbank neue Klassen und Konzepte benötigt werden, die über die im Kerncurriculum ausgewiesenen Kompetenzen im Bereich Algorithmik hinausgehen.

Für die hier vorgestellten Projekte wurden daher Werkzeuge ausgewählt und eine Hilfsklasse erstellt, welche die Kommunikation mit einer Datenbank so weit vereinfachen, dass die im niedersächsischen Kerncurriculum (Niedersächsisches Kultusministerium, 2017) geforderten Kompetenzen ausreichen, um eine Datenbankanwendung in Java oder Processing zu erstellen. Dadurch wird die Implementierung von Datenbankenanwendungen in der Qualifikationsphase möglich, sodass die Lernenden einen Eindruck bekommen, was bei den Anwendungen, die sie z. B. täglich im Internet verwenden, im Hintergrund passiert. Bei geeigneter Wahl der Projekte können in diesem Zusammenhang außerdem Fragen aus dem Bereich *Datenschutz* diskutiert werden. Vorschläge für entsprechende Projekte finden Sie im Ordner *Aufgaben_Projektideen*.

Da ein Projekt, welches das Lernfeld *Algorithmen und Datenstrukturen* mit dem Modul *Datenbanken* aus dem Lernfeld *Informationen und Daten* verbindet, komplexer ist als die Implementierung einer Anwendung ohne Datenbankanbindung bzw. das Erstellen einer einzelnen SQL-Anfrage, sollten die Schülerinnen und Schüler in beiden Bereichen bereits grundlegende Kompetenzen erworben haben.

Welche Kompetenzen bei der Entwicklung einer Datenbankanwendung unter Verwendung der weiter unten vorgestellten Hilfsklasse *DBManagerSQLite* bzw. *DBManagerProcSQLite* benötigt und damit vertieft und gefördert werden, wird in den folgenden Abschnitten erläutert.

Kompetenzen aus dem Lernfeld Algorithmen und Datenstrukturen

Die Schülerinnen und Schüler sollten sicher im Umgang mit Zeichenketten sein, um die SQL-Anweisungen ohne große Frustration zusammensetzen und individuelle Daten, die über eine Benutzeroberfläche eingegeben werden, einbauen zu können. Des Weiteren erfordert die Verwendung der Hilfsklasse *DBManager…* den Umgang mit Klassen und Objekten. Da das Ergebnis der SQL-SELECT-Anweisung hier als zweidimensionale Reihung zurückgegeben wird, sollten die Schülerinnen und Schüler auch im Umgang mit dieser Datenstruktur bereits Erfahrungen gesammelt haben.

Hilfreich ist weiterhin, wenn die Schülerinnen und Schüler mit dem Erstellen einer Benutzeroberfläche z. B. als *JFrame* unter Verwendung von Textfeldern und Buttons für die Benutzereingabe vertraut sind. Dies lässt sich aber auch im Rahmen der Implementierung einer Datenbankanwendung erlernen und wird in einer entsprechenden Anleitung für die Schülerinnen und Schüler explizit erklärt.

Es folgt eine Übersicht über die benötigten Kompetenzen aus dem niedersächsischen Kerncurriculum für die Oberstufe (Niedersächsisches Kultusministerium, 2017).

Modul Grundlagen der Algorithmik:

Einführungsphase:

- alle im Kerncurriculum genannten Kompetenzen

Qualifikationsphase:

- … verwenden geeignete Variablentypen zur Speicherung von Werten.
- … unterscheiden zwischen lokalen und globalen Variablen.
- … unterscheiden zwischen primitiven Datentypen und Objektreferenzen.
- … verwenden Übergabeparameter und Rückgabewerte in Operationen.

Modul Klassen und Objekte:

Qualifikationsphase:

- […] implementieren Algorithmen unter Verwendung von gegebenen […] Klassen/Objekten.

Modul statische und dynamische Datenstrukturen:

Einführungsphase:

- … entwerfen und implementieren Algorithmen unter Verwendung elementarer Zeichenkettenoperationen.

Qualifikationsphase:

- … erläutern das Prinzip, mehrere Daten des gleichen Typs in Reihungen zu verwalten […].
- … entwerfen und implementieren Algorithmen unter Verwendung von ein- und **zweidimensionalen** Reihungen.

Kompetenzen aus dem Modul Datenbanken

Die Schülerinnen und Schüler sollten bereits einige SQL-Anfragen an eine Datenbank gestellt haben und mit der Darstellung des Ergebnisses als Tabelle vertraut sein. Dieses Bild der Rückgabe hilft bei der Vorstellung, wie das Ergebnis in der zweidimensionalen Reihung vorliegt. Eine Vertiefung hinsichtlich der Komplexität der SQL-Anfragen kann sicherlich auch noch im Rahmen des Projektes zur

Erstellung einer Datenbankanwendung erfolgen. Ob auch Kenntnisse im Umgang mit ER-Modellen oder UPDATE, INSERT und DELETE-Anweisungen benötigt werden, hängt von dem geplanten Projekt ab. Mindestvoraussetzung sind daher die folgenden im niedersächsischen Kerncurriculum (Niedersächsisches Kultusministerium, 2017) für die Qualifikationsphase genannten Kompetenzen:

- … erläutern den Aufbau relationaler Datenbanken unter Verwendung der Begriffe Datensatz, Attribut, Primärschlüssel, Fremdschlüssel und Tabelle.
- … formulieren einfache Abfragen und Verbundabfragen über mehrere Tabellen.
- … formulieren Abfragen an Datenbanken unter Verwendung von Aggregatfunktionen.

Kompetenzen aus dem Modul Datenschutz

Fragestellungen aus dem Bereich *Datenschutz* können durch ein geeignetes Projekt motiviert und im Rahmen des Projektes diskutiert werden. Vorkenntnisse sind hier nicht zwingend erforderlich. In der Qualifikationsphase werden die Schülerinnen und Schüler aber ggf. bereits erste Kenntnisse zum Thema *Datenschutz* aus der Einführungsphase mitbringen, da das Kerncurriculum hier fordert, dass die Schülerinnen und Schüler „rechtliche Rahmenbedingungen für den Umgang mit ihren persönlichen Daten, wie z. B. informationelle Selbstbestimmung und Datenschutzrichtlinien“ (Niedersächsisches Kultusministerium, 2017, S. 16) erläutern können. Die für die Qualifikationsphase geforderte Kompetenz „Die Schülerinnen und Schüler diskutieren die Chancen und Risiken der automatisierten Datenanalyse“ kann dann im Rahmen des Projektes erworben bzw. vertieft werden.

Die Werkzeuge

Im Folgenden wird genauer auf die verwendeten Werkzeuge und die Hilfsklassen *DBManagerSQLite* *DBManagerProcSQLite* eingegangen. Die Ausführungen hier sind als Hintergrundwissen für Lehrkräfte gedacht. Entsprechende Anleitungen für die Schülerinnen und Schüler zur Anwendung befinden sich im Ordner *Anleitungen*.

Die Wirtssprache Java/Processing

Typisch für eine Datenbankanwendung ist die Möglichkeit, verschiedene Daten einzugeben, zu denen dann je nach Zweck der Anwendung passende Daten aus der Datenbank herausgesucht werden.

Es bietet sich daher an, für die Anwendung eine Nutzeroberfläche (GUI) zu erstellen, die mithilfe verschiedener Dialogelemente eine komfortable Eingabe der Daten ermöglicht. In den Materialien wird die Erstellung einer entsprechenden GUI für eine *JFrame*-Anwendung in Java mithilfe des Java-Editors (Röhner, 2023) erläutert. Alternativ können Lerngruppen, die bislang mit Processing (Reas & Fry, 2024) gearbeitet haben, eine entsprechende Bibliothek und das passende Tool nutzen, um eine GUI für ein Processing-Projekt zu erstellen, z. B. die Bibliothek *G4P* von Peter Lager. Auch hierzu sind eine Anleitung und Beispiele in den Materialien vorhanden.

Um die Kommunikation mit der Datenbank zu kapseln und auf Konzepte abzubilden, mit denen die Schülerinnen und Schüler bereits vertraut sind, wurde die Java-Klasse *DBManagerSQLite* erstellt. Diese wird im Abschnitt *Die Hilfklasse DBManagerSQLite* genauer erläutert. Die Beispiele für Processing enthalten eine entsprechende Anpassung der Klasse zur Verwendung in Processing: *DBManagerProcSQLite*. Soll mit einer anderen Wirtssprache gearbeitet werden, kann die Hilfsklasse ggf. für diese Sprache von der Lehrkraft angepasst werden.

Das Datenbanksystem SQLite

Das Einrichten eines Datenbankservers ist häufig mit hohem administrativem Aufwand verbunden. In den vorliegenden Materialien und Beispielen wird daher eine *SQLite-Datenbank* (SQLite, 2024) verwendet. Der Vorteil besteht darin, dass es sich bei der Datenbank lediglich um eine Datei handelt, die lokal auf dem Rechner abgelegt werden kann. Eine Datei, die eine SQLite-Datenbank enthält, hat die Endung *.db*. Ein Datenbankserver wird nicht benötigt. Dies erspart der Lehrkraft aufwändige Installationsarbeiten.

Wenn die Datenbank lokal auf dem Rechner gespeichert wird, können die Lernenden jeweils ihre eigene Datenbank verwenden. Dies ermöglicht zum einen den Entwurf und die Implementierung individueller Datenbanken. Zum anderen können die Schülerinnen und Schüler, auch wenn eine einheitliche Datenbank für alle zur Verfügung gestellt wird, die Daten in ihrer lokalen Kopie verändern, ohne dass es Konflikte mit den Änderungen anderer Schülerinnen und Schüler gibt. Bei der Verwendung einer zentralen Datenbank auf einem Datenbankserver beschränkt sich der Unterricht in der Regel auf SQL-SELECT-Anweisungen, um zu vermeiden, dass die Lernenden unkontrolliert Datensätze löschen oder verändern. Bei der Verwendung einer lokalen SQLite-Datenbank können sie gefahrlos Änderungen vornehmen. Geht beim Bearbeiten der Daten doch einmal etwas schief, kann die Datei mit der ursprünglichen Datenbank einfach wieder auf den Rechner kopiert werden.

Die Einschränkungen, die hinsichtlich Mehrbenutzerbetrieb und Transaktionsverwaltung bestehen, sind für die Schule zu vertreten. Der Funktionsumfang ist für die Komplexität, die Anwendungen in der Schule erreichen, vollkommen ausreichend. In der Regel werden die Schülerinnen und Schüler eine Anwendung nur lokal auf ihrem Rechner testen und nutzen. In den hier vorgestellten Projekten geht es schließlich nicht darum professionelle Anwendungen zu entwickeln, sondern das Zusammenspiel der verschiedenen Programmierkonzepte zu verstehen.

Falls die Schülerinnen und Schüler zunächst mit einem anderen Datenbanksystem z. B. mit einer *MySQL-Datenbank* gearbeitet haben und für die Implementierung der Datenbankanwendung auf eine *SQLite-Datenbank* umsteigen, sollten sie darauf hingewiesen werden, dass sich die jeweiligen *SQL-Dialekte* in einigen Details unterscheiden können und die SQL-Anweisungen daher immer separat aber im gleichen Datenbanksystem getestet werden sollten. Beispielsweise wird der Datentyp *date* von einer SQLite-Datenbank nicht unterstützt. Ein Datum, z. B. der Geburtstag, wird daher als Zeichenkette in der Form *YYYY-MM-DD* gespeichert. Auf die Zeichenkette können dann wie für den Datentyp *date* die Operationen „kleiner“ < und „größer“ > angewendet werden, um zwei Daten zu vergleichen. Eine Übersicht über die Umsetzung von SQL im SQLite-Datenbanksystem erhält man z. B. auf der Seite https://www.tutorialspoint.com/sqlite/index.htm (Mohtashim, 2024).

Um eine SQLite-Datenbank zu erzeugen, die Struktur einer SQLite-Datenbank zu visualisieren oder SQL-Anweisungen zu testen, bietet sich das Programm *DB Browser for SQLite* (Piacentini, 2023) an.

Soll ein anderes Datenbanksystem verwendet werden, können die Materialien, insbesondere die Hilfsklasse für die Kommunikation mit der Datenbank entsprechend angepasst werden. Es wird dann ein entsprechender Datenbanktreiber für das verwendete Datenbanksystem benötigt.

Die Hilfsklasse DBManagerSQLite

Die Klasse *DBManagerSQLite* kapselt die für die Kommunikation mit der Datenbank notwendigen Mechanismen, so dass die aus dem Unterricht bekannten Konzepte ausreichen, um eine Datenbankanwendung in Java zu erstellen. Die hier vorgestellte Idee der Kapselung aller datenbankspezifischen Methodenaufrufe lässt sich natürlich auch auf andere objektorientierte Programmiersprachen übertragen. Für Lerngruppen, die mit Processing arbeiten, ist in den Beispielen eine entsprechende Klasse *DBManagerProcSQLite* enthalten, die eine Anpassung der Klasse *DBManagerSQLite* an Processing darstellt.

Die Klasse *DBManagerSQLite* stellt die Verbindung zu einer *SQLite-Datenbank her.* Damit die Klasse *DBManagerSQLite* genutzt werden kann, muss zuvor die Bibliothek für den Datenbanktreiber eingebunden werden. Die aktuelle Version des *SQLite-JDBC-Treibers* kann auf der folgenden Seite heruntergeladen werden: https://github.com/xerial/sqlite-jdbc#usage (Saito, 2024). Es handelt sich bei dem Treiber um *jar*-Dateien, die an beliebiger Stelle gespeichert werden können. Der Speicherort muss anschließend im Klassenpfad hinterlegt werden. Für die für die Schule entwickelte Programmierumgebung *Java-Editor* (Röhner, 2023) wird das Vorgehen zum Einbinden der *jar*-Datei ausführlich in den Materialien für die Schülerinnen und Schüler beschrieben.

Im Folgenden wird der Aufbau der Klasse *DBManagerSQLite* als Hintergrundwissen für die Lehrkraft genauer erläutert. Die Klasse sowie eine Dokumentation im Java-Doc Format befinden sich im Ordner *Lehrer → Hilfsklasse*. Für die reine Verwendung der Klasse, wie es für die Lernenden gedacht ist, ist es nicht notwendig sich mit der Implementierung auseinanderzusetzen.

Implementierung der Konstruktoren

Für die Kommunikation mit einer Datenbank muss ein *Connection*-Objekt für die Verbindung und ein *Statement*-Objekt für das Senden von SQL-Anweisungen erzeugt werden. Dies übernimmt die Klasse *DBManagerSQLite*, so dass der Java- spezifische Teil der Kommunikation mit der Datenbank für die Schülerinnen und Schüler in einer Blackbox gekapselt ist.

Da die Lernenden bei der Verwendung einer SQLite-Datenbank jeweils lokal auf ihrem Rechner arbeiten, wird in der Regel nur eine Anwendung auf die Datenbank zugreifen. Daher werden in der Klasse *DBManagerSQLite* das *Connection*- und das *Statement*-Objekt zu Beginn vom Konstruktor erzeugt und für die gesamte Laufzeit der Anwendung genutzt. Greifen viele Anwendungen parallel auf die Datenbank zu, müssen *Connection*- und *Statement*-Objekt nach jeder Verwendung geschlossen und bei Bedarf neu erzeugt werden, da eine Datenbank nur eine begrenzte Anzahl von Verbindungen zur Verfügung stellt.

Die Klasse *DBManagerSQLite* stellt die zwei folgenden Konstruktoren zur Verfügung. Der erste Konstruktor ist parameterlos. Er stellt die Verbindung zu der Datenbank her, die in der Klasse voreingestellt ist. In der vorliegenden Implementierung wird eine Verbindung zur ebenfalls beiliegenden Datenbank *schule_erweitert* hergestellt. Der zweite Konstruktor erhält den Namen einer Datenbank als Parameter. Die Endung *.db* wird vom Konstruktor automatisch angehängt. So kann eine Verbindung zu einer beliebigen SQLite-Datenbank hergestellt werden, die in dem Programm-Verzeichnis gespeichert ist. Wenn alle Schülerinnen und Schüler eine einheitliche andere Datenbank verwenden sollen, kann die Standard-Belegung der Variablen `dbName` in der Implementierung der Klasse *DBManagerSQLite* auch von der Lehrkraft geändert werden, sodass der parameterlose Konstruktor automatisch die Verbindung zu der passenden Datenbank erzeugt.

Implementierung der Methoden

Um eine SQL-SELECT-Anweisung an die Datenbank zu senden, stellt die Klasse *DBManagerSQLite* die Methode *sqlAnfrageAusfuehren* zur Verfügung. Diese erwartet als Eingabe eine Zeichenkette, welche eine SQL-SELECT-Anweisung enthält.

Die Antwort der Datenbank wird in Java als Objekt der Klasse *ResultSet* zurückgeliefert. Diese Klasse wurde speziell für Ergebnisse von Datenbankanfragen erstellt und bietet eine Reihe datenbankspezifischer Funktionen. So kann das *ResultSet* z. B. die Attribute in dem jeweiligen Datentyp liefern, in dem sie in der Datenbank gespeichert sind, die Datensätze werden erst nach und nach aus der Datenbank abgerufen und es können zusätzliche Metadaten über die Tabelle, wie der Name und Datentyp der Attribute abgefragt werden. Der Zugriff auf die einzelnen Datensätze eines *ResultSets* erfordert jedoch das Erlernen neuer spezieller Konzepte und Methodenaufrufe. Dies soll hier vermieden werden.

Wenn die Schülerinnen und Schüler in einem Datenbankbrowser direkt eine SQL-SELECT-Anweisung eingeben, erhalten sie die Rückgabe in Form einer Tabelle mit den ausgewählten Spalten und einer Zeile für jeden gefundenen Datensatz. Eine Tabelle lässt sich gut in einer zweidimensionalen Reihung abbilden, die den Schülerinnen und Schülern als Datenstruktur bekannt ist. Die Methode *sqlAnfrageAusführen* überträgt daher alle Datensätze, die das *ResultSet* zu der übergebenen SQL-SELECT-Anweisung liefert, in eine zweidimensionale Reihung vom Typ Zeichenkette. Der erste Index wird dabei als Index der Zeile und der zweite als Index der Spalte interpretiert. Um die Metainformation über die Namen der Attribute zu erhalten, enthält die erste Zeile keinen Datensatz, sondern die Namen der Attribute als Spaltenüberschriften. Wurde bei der SELECT-Anweisung ein ALIAS angegeben, so wird dieser als Spaltenüberschrift verwendet. Enthält ein Datensatz für ein Attribut in der Datenbank den Wert *null*, so wird in der zweidimensionalen Reihung die Zeichenkette „null“ an der entsprechenden Stelle eingetragen.

Um den Schülerinnen und Schülern den Umgang mit dem Ergebnis einer Abfrage mit ihnen bekannten Konzepten zu ermöglichen, wird also auf einen Teil der von Java vorgesehenen Funktionalität verzichtet. Die beiliegenden Aufgabenbeispiele und Projektideen, kommen jedoch problemlos ohne diese Funktionalitäten aus. Die einzige Einschränkung, die sich eventuell bemerkbar macht, ist der einheitliche Datentyp Zeichenkette für alle Attribute. Da die Ausgabe in der Benutzeroberfläche ohnehin meist als Text erfolgt, erscheint diese Einschränkung aber akzeptabel. Die gängigsten Datentypen, in denen die Daten in der Datenbank vorliegen, werden neben Zeichenketten Zahlen sein. Diese lassen sich bei Bedarf aus einer Zeichenkette parsen, sollte die Information doch einmal in einer Variablen vom Typ Ganzzahl oder Fließkommazahl benötigt werden. Java stellt dafür die statischen Methoden *Integer.parseInt* bzw. *Float.parseFloat* zur Verfügung. Dieses Problem tritt auch auf, wenn eine Zahl über ein Textfeld erfragt wird, so dass das Parsen den Schülerinnen und Schülern unter Umständen bereits bekannt ist.

Während das Ausführen einer SQL-SELET-Anweisung in Java über die Methode *executeQuery* läuft, steht für UPDATE-, INSERT- und DELETE-Anweisungen die Methode *executeUpdate* zur Verfügung. Dementsprechend stellt auch die Klasse *DBManagerSQLite* hierfür gesonderte Methoden zur Verfügung. Um die Zuordnung zu vereinfachen, gibt es drei Methoden: *datensatzAendern*, *datensatzEinfuegen* und *datensatzLoeschen*. Die drei Methoden sind jedoch gleich aufgebaut und unterscheiden sich nur in ihrem Namen. Alle drei Methoden erwarten die entsprechende SQL-

Anweisung als Zeichenkette. Der Rückgabewert entspricht dem Rückgabewert der Methode *executeUpdate*. Es handelt sich um eine Ganzzahl, die die Anzahl der geänderten, eingefügten bzw. gelöschten Datensätze angibt. Wurde kein Datensatz geändert, eingefügt oder gelöscht, da z.B. die WHERE-Klausel für keinen Datensatz erfüllt war oder der Datensatz, der eingefügt werden soll, keinen gültigen Schlüssel enthält, ist der Rückgabewert 0. Ergänzt wurde der Rückgabewert -1. Dieser zeigt an, dass die SQL-Anweisung nicht ausgeführt werden konnte, da sie z. B. syntaktisch nicht korrekt war und deshalb eine *SQLException* ausgelöst hat.

Umgang mit Exceptions

Alle Methodenaufrufe, die aus dem Paket *java.sql* stammen, können eine *SQLExceptions* auslösen, wenn beispielsweise die Datenbank nicht erreichbar ist oder eine SQL-Anweisung syntaktische Fehler enthält. Die *SQLExceptions* werden innerhalb der Methoden der Klasse *DBManagerSQLite* abgefangen. Die entsprechende Fehlermeldung wird als Text ausgegeben. Im Java-Editor erscheint sie z. B. im Fenster *Meldungen.* Die Schülerinnen und Schüler müssen sich somit bei der Implementierung nicht um das Abfangen der Exceptions kümmern, sie erhalten aber die entsprechende Information, wenn eine Exception auftritt.

Übersicht über die Konstruktoren und Methoden

Es folgt eine Übersicht über die Konstruktoren und Methoden der Klasse *DBManagerSQLite:*

Konstruktor	**Beschreibung**
`DBManagerSQLite()`	erzeugt ein DBManagerSQLite-Objekt für die Datenbank schule_erweitert und stellt eine Verbindung zur Datenbank her
`DBManagerSQLite(String dbName)`	erzeugt ein DBManagerSQLite- Objekt für die angegebene Datenbank und stellt eine Verbindung zur Datenbank her

Rückgabewert	**Methode**	**Beschreibung**
`int`	`datensatzAendern(String sql)`	führt die übergebene SQL-Update-Anweisung aus
`int`	`datensatzEinfuegen(String sql)`	führt die übergebene SQL-Einfüge-Anweisung aus
`int`	`datensatzLoeschen(String sql)`	führt die übergebene SQL-Lösch-Anweisung aus
`String[][]`	`sqlAnfrageAusfuehren(String sqlAnfrage)`	führt die übergebene SQL-Anfrage aus

Die Methoden im Detail

sqlAnfrageAusfuehren

`public String[][] sqlAnfrageAusfuehren(String sqlAnfrage)`
führt die übergebene SQL-Anfrage aus

Parameter:

`sqlAnfrage` - Die SQL-Anfrage, die ausgeführt werden soll, als Zeichenkette.

Rückgabewert:

Das Ergebnis der SQL-Anfrage als zweidimensionale Reihung vom Typ Zeichenkette. Interpretiert man den ersten Index als Zeilen- und den zweiten als Spatennummer, enthält die erste Zeile der Reihung die Überschriften der Spalten. Danach folgt pro Datensatz eine Zeile mit den entsprechenden Werten. Diese werden unabhängig von den Datentypen der Datenbank als Zeichenkette gespeichert. Enthält eine Zelle in der Datenbank den Wert null, wird die Zeichenkette "null" in das entsprechende Feld der zweidimensionalen Reihung geschrieben.
Schlägt der Versuch, die SQL-Anfrage zu stellen, fehl, enthält die zweidimensionale Reihung nur ein Feld mit dem Inhalt "Fehler".

datensatzEinfuegen

`public int datensatzEinfuegen(String sql)`
führt die übergebene SQL-Einfüge-Anweisung aus

Parameter:

`sql` - Die SQL-Anweisung, die ausgeführt werden soll, als Zeichenkette.

Rückgabewert:

Die Anzahl der betroffenen Datensätze. Der Rückgabewert ist 0, wenn die Anweisung keine Änderung in der Datenbank bewirkt hat. Der Rückgabewert -1 zeigt an, dass ein Fehler aufgetreten ist.

datensatzAendern

`public int datensatzAendern(String sql)`
führt die übergebene SQL-Update-Anweisung aus

Parameter:

sql - Die SQL-Anweisung, die ausgeführt werden soll, als Zeichenkette

Rückgabewert:

Die Anzahl der betroffenen Datensätze. Der Rückgabewert ist 0, wenn die Anweisung keine Änderung in der Datenbank bewirkt hat. Der Rückgabewert -1 zeigt an, dass ein Fehler aufgetreten ist.

datensatzLoeschen

`public int datensatzLoeschen(String sql)`
führt die übergebene SQL-Lösch-Anweisung aus

Parameter:

sql - Die SQL-Anweisung, die ausgeführt werden soll, als Zeichenkette

Rückgabewert:

Die Anzahl der betroffenen Datensätze. Der Rückgabewert ist 0, wenn die Anweisung keine Änderung in der Datenbank bewirkt hat. Der Rückgabewert -1 zeigt an, dass ein Fehler aufgetreten ist.

Anpassungen der Klasse *DBManagerProcSQLite*

Um in Processing (Reas & Fry, 2024) eine Verbindung zu einer SQLite-Datenbank herzustellen, wird die Bibliothek *BezierSQLib* von Florian Jenett benötigt. Der Konstruktor der darin enthaltenen Klasse *SQLite* übernimmt bereits einige Aufgaben, wie z. B. das Herstellen einer Verbindung zur Datenbank.

Anders als die Konstruktoren der Klasse *DBManagerSQLite* benötigen die Konstruktoren der Klasse *DBManagerProcSQLite* als Übergabeparameter einen Verweis auf den aufrufenden Sketch. Die entsprechende Referenz ist in der setup-Methode des Sketches in `this` enthalten.

Die Klasse *SQLite* aus der Bibliothek *BezierSQLib* stellt eigene Methoden zum Ausführen von SQL-Anweisungen zur Verfügung. Intern wurden die Methoden der Klasse *DBManagerProcSQLite* so angepasst, dass die Methoden `sqlAnfrageAusfuehren`, `datensatzAendern`, `datenSatzEinfuegen` und `datenSatzLoeschen` genauso verwendet werden können, wie die Methoden der Klasse *DBManagerSQLite*. Der einzige Unterschied ist, dass die Methoden `datensatzAendern`, `datenSatzEinfuegen` und `datenSatzLoeschen` keinen Rückgabewert haben, da die entsprechenden Methoden der Klasse *SQLite* aus der Bibliothek *BezierSQLib* keinen Rückgabewert zur Verfügung stellen. Die entsprechende Übersicht über die Konstruktoren und Methoden ist in der Anleitung zum Erstellen einer Datenbankanwendung in Processing enthalten.

Wenn ein Processing-Projekt eine SQLite-Datenbank verwenden soll, muss die entsprechende Datei, welche die Datenbank enthält, im Unterordner *data* gespeichert sein.

Überblick über die Materialien

Für Schüler

Im Ordner *Schueler* werden Anleitungen und Beispiele für die Schülerinnen und Schüler bereitgestellt, die ihnen die Entwicklung eigener Datenbankanwendungen in Java oder Processing ermöglichen sollen.

Um eine Datenbankanwendung in Java implementieren zu können, muss, wie oben bereits erläutert, der entsprechende Datenbanktreiber heruntergeladen und in die Entwicklungsumgebung eingebunden werden. Die Vorgehensweise wird für die den *Java-Editor* (Röhner, 2023) in der Datei *Vorbereitung_JavaEditor_SQLite* schrittweise erklärt. Nachdem die Vorbereitungen durchgeführt wurden, wird in der Datei *Anleitung_DBAnwendung_JavaEditor* anhand eines einfachen Beispiels Schritt für Schritt die Implementierung einer Datenbankanwendung mithilfe des Java-Editors erklärt. Das grundsätzliche Vorgehen ist aber auch auf andere Entwicklungsumgebungen übertragbar. Das Einführungsbeispiel ist dabei bewusst einfach gehalten, um die Schülerinnen und Schüler nicht zu überfordern. Leistungsstärkere Schülerinnen und Schüler, die ihre Anwendung z. B. benutzerfreundlicher gestalten möchten, finden im Anhang der Anleitung Erläuterungen zur Verwendung weiterer Dialogelemente für die Benutzereingabe und die Ausgabe des Ergebnisses.

Arbeiten die Schülerinnen und Schüler mit Processing (Reas & Fry, 2024), werden sowohl die Vorbereitung als auch ein Beispiel Schritt für Schritt in der Datei *Anleitung_DBAnwendung_Processing* vorgestellt.

Im jeweiligen Anhang befindet sich auch eine Übersicht über die Konstruktoren und Methoden der Klassen *DBManagerSQLite bzw. DBManagerProcSQLite*. Alle in der Anleitung verwendeten Beispiele liegen auch als Quelltext in dem Unterordner *Beispiele* vor.

Sollen sich die Lernenden vor dem Erstellen eines eigenen Projektes aktiver mit dem Erstellen einer Datenbankanwendung auseinanderzusetzen, ist im Ordner Einstiegsaufgabe auch ein exemplarisches Arbeitsblatt enthalten, das die Erkundung und Erweiterung eines Beispiels anleitet.

Darauf aufbauend können die Schülerinnen und Schüler dann ihre eigenen, komplexeren Anwendungen in einem Projekt implementieren.

Für Lehrer

Im Ordner *Lehrer* finden Sie weitere Beispiele und Ideen für Projekte. Für einige Projekte sind auch exemplarische Umsetzungen in Java enthalten. Da die Schülerinnen und Schüler hierzu ihre eigenen Umsetzungen entwickeln sollen, sind diese Beispiele nicht im Ordner mit den Schülermaterialien enthalten. Weiterhin enthält der Ordner *Datenbanken* die SQLite-Datenbanken *schule* und *schule_erweitert*. Der Inhalt der beiden Datenbanken wird im Folgenden vorgestellt. Außerdem wird erläutert, wie sich die Daten an die Gegebenheiten des Kurses anpassen lassen. Dazu kann das Beispiel aus dem Ordner *Beispiel_DatenbankAendern* angepasst werden.

Die Datenbank *schule*

Die Datenbank *schule*[1] enthält eine Tabelle *schueler* mit schulbezogenen Daten der Schülerinnen und Schüler eines Oberstufenjahrgangs. Eine weitere Tabelle *kurse* enthält Daten zu den angebotenen Kursen. Die beiden Tabellen werden in einer dritten Tabelle *hatkurs* verknüpft. Hier ist abgelegt, welcher Schüler bzw. welche Schülerin an welchen Kursen teilgenommen hat und wie viele Punkte er oder sie in dem jeweiligen Kurs erzielt hat. Eine vollständige Übersicht über die Attribute der Tabellen wird hier in Kurzschreibweise gegeben:

schueler (ID_nummer, Name, Vorname, Geburtstag, Geburtsort, Staatsang, Geschlecht, Konfession, PLZ, Ort, Ortsteil, von_Schulform, Tutor_in_11, Tutor_in_12_13, WDH_11, WDH_12, WDH_13)

kurse (kursnummer, KHJ, kursthema, kurslehrerkurz, kursart)

hatkurs (↑ID_nummer, ↑kursnummer, note, punkte)

Die Attribute *PLZ*, *KHJ* und *Punkte* haben den Datentyp `int(11)`, alle anderen Attribute haben den Datentyp `char(…)`. Das Attribut Geburtstag hat das Format JJJJ-MM-TT.

Die Datenbank *schule_erweitert*

Die Datenbank *schule_erweitert* basiert auf den Daten der Datenbank *schule* und ist im Rahmen der Durchführung des Projektes Dating-Portal in einem Grundkurs Informatik[2] entstanden. Die Datenbank besteht ebenfalls aus den Tabellen *schueler*, *kurse* und *hatkurs*. Die Attribute der Tabelle *schueler* wurden jedoch aktualisiert und um Daten, die den Schülerinnen und Schüler für ein Dating-Portal relevant erschienen, ergänzt. Nicht benötigte Attribute wurden hingegen gelöscht. Es folgt ein Überblick über die Attribute in Kurzschreibweise.

schueler (ID_nummer, Name, Vorname, Geburtstag, Geburtsort, Staatsang, Geschlecht, Konfession, PLZ, Ortsteil, Groesse, Haarfarbe, Augenfarbe, Lieblingsfach, Sexualitaet, Gewicht, Musik)

kurse (kursnummer, KHJ, kursthema, kurslehrerkurz, kursart)

hatkurs (↑ID_nummer, ↑kursnummer, note, punkte)

[1] Die Daten der Datenbank *schule* wurden im Original ursprünglich von Georg Beckmann anonymisiert und bereitgestellt.
[2] Besonderer Dank gilt dabei dem Schüler Jonathan Malek, der die Ideen des Kurses maßgeblich in die Datenbank eingepflegt hat.

Die Attribute *PLZ*, *Gewicht*, *KHJ* und *Punkte* haben den Datentyp `int(11)`, alle anderen Attribute haben den Datentyp `char(…)` bzw. `varchar(…)`. Das Attribut Geburtstag hat das Format JJJJ-MM-TT.

Einen Überblick über den Inhalt der Datenbanken gewinnt man am einfachsten, indem man die jeweilige Datenbank in einem Datenbankbrowser öffnet.

Datenbanken ändern

Für die Motivation der Schülerinnen und Schüler kann es hilfreich sein, wenn sich die Daten der Datenbank möglichst eng an ihrem Lebensumfeld orientieren. So möchte man statt der Wohnorte Burgberg und Landsdorf vielleicht Orte aus der Umgebung der Schule wählen. Spielen die Geburtsjahrgänge der Schülerinnen und Schüler in der Anwendung eine Rolle, möchte man die Geburtstage in der Datenbank ggf. an die Geburtsjahrgänge des Kurses anpassen.

Das Ändern der Orte und Ortsteile kann mithilfe von SQL-UPDATE-Anweisungen direkt über den Datenbankbrowser erfolgen. Mithilfe einer geeigneten WHERE-Klausel können die Wohnorte durch andere ersetzt werden. Hier ein Beispiel:

```
1  UPDATE schueler SET Ort = 'Oberdorf' WHERE Ort = 'Burgberg';
```

Um mehr Variation bei den Wohnorten zu erhalten, kann die WHERE-Bedingung noch erweitert werden:

```
2  UPDATE schueler SET Ort = 'Oberdorf' WHERE Ort = 'Burgberg' and
   von_Schulform = 'IGS';
```

Hier sollten natürlich Attribute für die Auswahl verwendet werden, die für die spätere Anwendung keine Relevanz haben, da ansonsten unsachgemäße statistische Zusammenhänge erzeugt werden. Möchte man dies vermeiden, müsste man die Attributwerte zufällig zuordnen. Dies ist jedoch aufwändiger und erfordert ein Java-Programm, das beispielsweise für jeden Datensatz einen zufälligen Wert aus einem Pool von Attributwerten auswählt. Der Ordner *Datenbanken* → *Beispiel_DatenbankAendern* enthält die Java-Datei *DBZufallswerteEintragen,* die exemplarisch zeigt, wie die Attribute Ort und Ortsteil mit Zufallswerten belegt werden können.

Auch das Ändern des Jahres im Geburtstag lässt sich mithilfe einer SQL-UPDATE-Anweisung nicht so leicht bewerkstelligen. Der Ordner *Datenbanken* → *Beispiel_DatenbankAendern* enthält daher die Java-Datei *DBBearbeiten*, die exemplarisch zeigt, wie die Geburtstage zunächst aus der Datenbank ausgelesen, mithilfe von Funktionen zur Zeichenkettenverarbeitung in Java verändert und die veränderten Werte schließlich in die Datenbank eingetragen werden können.

Literaturverzeichnis

Kultusministerkonferenz. (2004). *Einheitliche Prüfungsanforderungen Informatik.* Von https://www.kmk.org/fileadmin/veroeffentlichungen_beschluesse/1989/1989_12_01_EPA_Informatik.pdf abgerufen

Mohtashim, M. (2024). *SQLite Tutorial*. Abgerufen von https://www.tutorialspoint.com/sqlite/index.htm

Niedersächsisches Kultusminiterium (Hrsg.). (2017). *Kerncurriculum für das Gymnasium – gymnasiale Oberstufe, die Gesamtschule – gymnasiale Oberstufe, das Kolleg Informatik.* Hannover: Unidruck.

Piacentini, M. u. (2023). *DB Browser for SQLite*. Von http://sqlitebrowser.org/ abgerufen

Reas, C., & Fry, B. (2024). *Processing*. Von https://processing.org/ abgerufen

Röhner, G. (2023). *Java-Editor*. Von http://javaeditor.org abgerufen

Saito, T. (2024). *sqlite-jdbc Downloads*. Von https://github.com/xerial/sqlite-jdbc#usage abgerufen

SQLite. (2024). Von sqlite.org abgerufen