Formale Anforderungen für Darstellungsformen, die im Informatikunterricht eingesetzt werden

Es gibt verschiedenen Varianten für Schreibweisen von Diagrammen und Quellcode. Damit zumindestens in unterschiedlichen Informatikkursen an einer Schule die Schreibweise einheitlich ist, ist eine Absprache nötig. Dieses Dokument stellt eine Möglichkeit dar, wie diese Absprache der Fachkonferenz für die Schüler transparent gemacht werden kann.

Inhaltsverzeichnis

1 Bezeichner

Bezeichner werden ohne Sonderzeichen[1] geschrieben. Sind Bezeichner aus mehreren Wörtern zusammengesetzt, so werden diese ohne Leerzeichen zusammengeschrieben und jedes folgende Wort mit einem Großbuchstaben begonnen wie z. B. gibName.

Klassenbezeichner beginnen mit einem Großbuchstaben. Alle anderen Bezeichner, wie für Objekte, Attribute, Methoden und Variablen, beginnen mit einem kleinen Buchstaben. Bei Attributwerten werden einzelne Zeichen und Zeichenketten in Anführungszeichen gesetzt. Zahlen werden als Zahlenwert dargestellt – Objekte werden durch Angabe des Objektbezeichners dargestellt. Für Wahrheitswerte stehen die beiden Möglichkeiten Wahr bzw. Falsch oder Ja bzw. Nein zur Verfügung. In der SQL werden die Tabellenbezeichner klein geschrieben. Dieses bezieht sich auch auf die Darstellung in den entsprechenden ER-Diagrammen.

[1] Sonderzeichen sind alle Umlaute (ä, ö, ü, ß, Ä, Ö, Ü), aber auch sämtliche Satzzeichen sowie spezielle Symbole, wie @, ², ³, ...

2 Diagramme und graphische Darstellungen

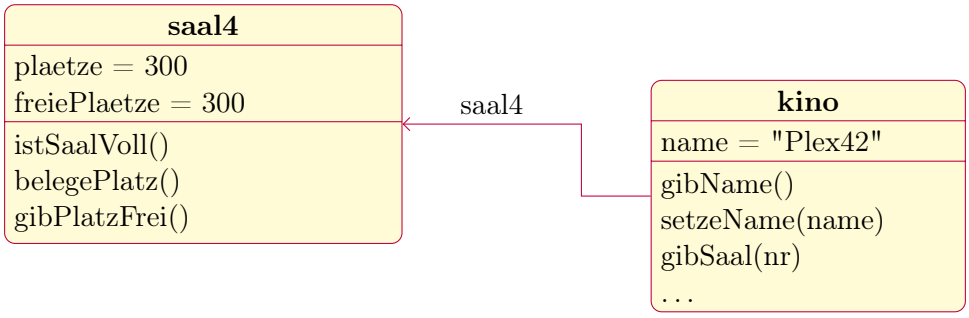
Als Grundlage für die UML-Diagramme dient die UML2 Spezifikation mit einigen Änderungen und Ergänzungen. Handschriftliche Diagramme sind mit Bleistift und Lineal anzufertigen.

2.1 Objektdiagramm

Im Unterschied zur UML Spezifikation wird eine Objektkarte mit runden Ecken dargestellt und der Objektbezeichner wird nicht unterstrichen. Außerdem ist es möglich, die Objektkarte um die Auflistung der Methoden zu ergänzen. Die zwei bzw. drei Bereiche der Objektkarte werden durch Linien voneinander getrennt.

Zwischen Objekten sind nur gerichtete und ungerichtete Assoziation als Beziehungen zugelassen. Die Bezeichnung für das Beziehungsattribut wird an das Ende des Beziehungspfeils geschrieben. Beziehungspfeile sollen – wenn möglich – im Manhattanstil angelegt werden.

Sollen in einer Objektkarte nicht alle benötigten Attribute und Methoden angegeben werden, so können diese durch drei waagerechte oder senkrechte Punkte ersetzt werden.



Werden Sichtbarkeiten angeben, so geschiht dies durch die Angabe von +,- oder o vor dem jeweiligen Attribut- oder Methodenbezeichner.

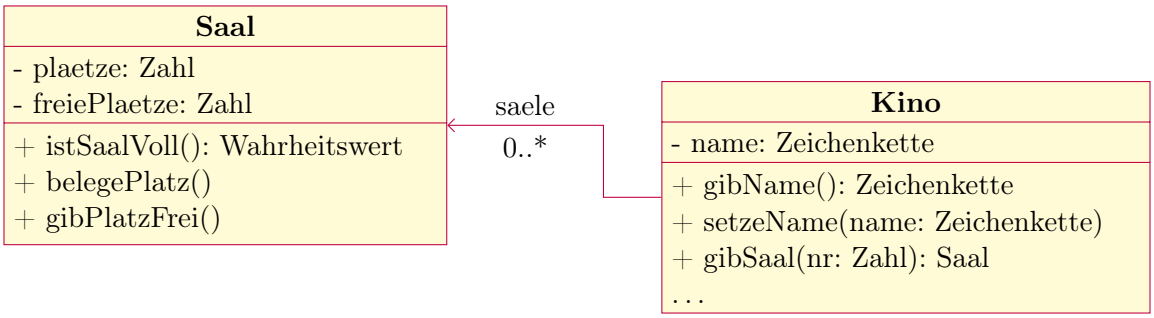
2.2 Klassendiagramm

Wie in den Vorgaben zum Zentralabitur für das Fach Informatik aufgeführt, wird bei Klassendiagrammen zwischen Entwurfsdiagrammen und Implementationsdiagrammen unterschieden. Entwurfsdiagramme sind unabhängig von einer Programmiersprache und Implementationsdiagramme beziehen sich auf eine Programmiersprache und die daraus resultierenden nötigen Modifikationen am Entwurf.

Alle Klassenkarten werden in ein Rechteck mit drei Teilbereichen gezeichnet. Bei (allen) Parametern und Attributen werden die Typen ausgewiesen. Dieses gilt auch für Methoden, die eine Rückgabe liefern.

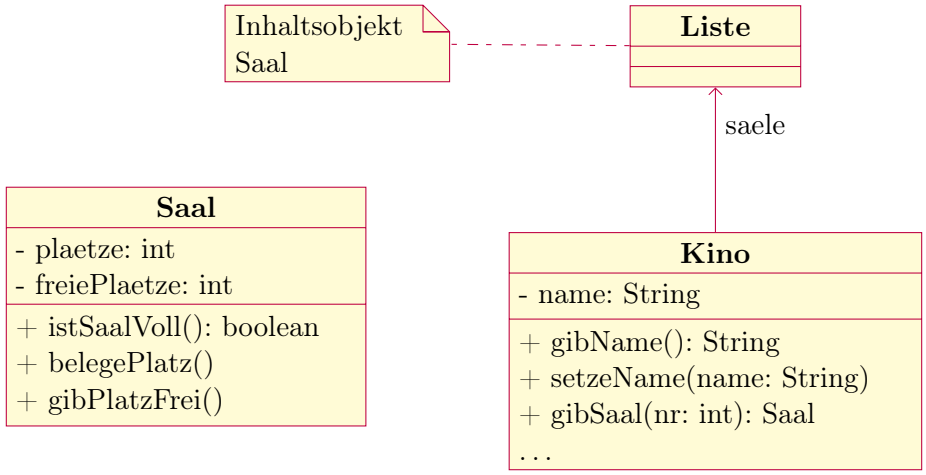
2.2.1 Entwurfsdiagramm

Da ein Entwurfsdiagramm unabhängig von der Programmiersprache ist, werden hier die elementaren Typen: Zahl, Zeichenkette und Wahrheitswert verwendet. Darüber hinaus können andere Klassen als Typen angegeben werden. Die Beziehungen werden analog zu dem Objektdiagramm gezeichnet. Hier können nach entsprechender Einführung im Unterricht auch die Kardinalzahlen ergänzt werden. Anstatt mehrerer Beziehungspfeile mit gleichem Start und Ziel wird ein einziger mit mehren Beziehungsattributen benutzt.



2.2.2 Implementationsdiagramm

Das Implementationsdiagramm bezieht sich auf eine konkrete Programmiersprache, so dass hier die in Java eingebauten Typen wie String, **int** und **boolean** zu verwenden sind. Die Beziehungspfeile werden ähnlich verwendet wie im Entwurfsdiagramm. Hier ist aber zu beachten, dass Kardinalitäten größer als eins über Datensammlungen realisiert werden müssen. So wird eine Beziehung mit höherer Kardinalität, bei der Umwandlung von Entwurfs- zu Implementationsdiagramm, zu einer Datensammlung umgebogen. An diese Datensammlung wird als Notiz der jeweilige Inhaltstyp angehängt.

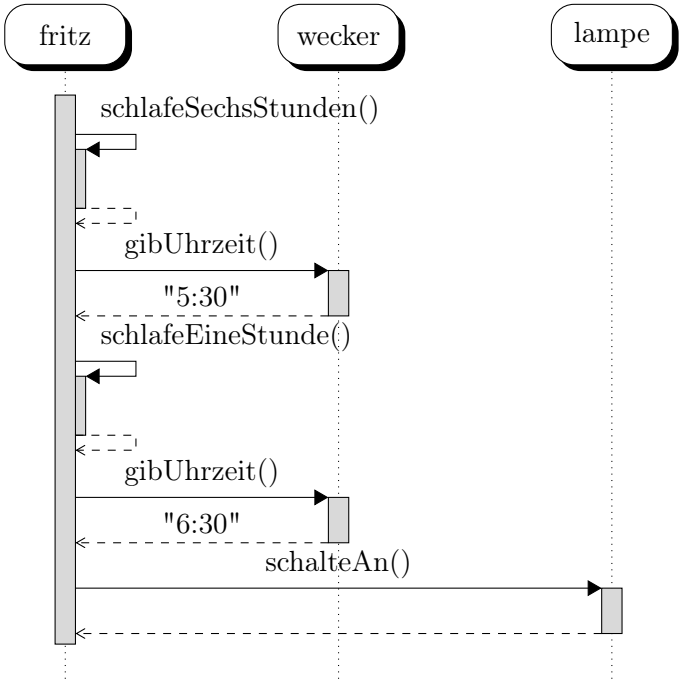


2.3 Sequenzdiagramm

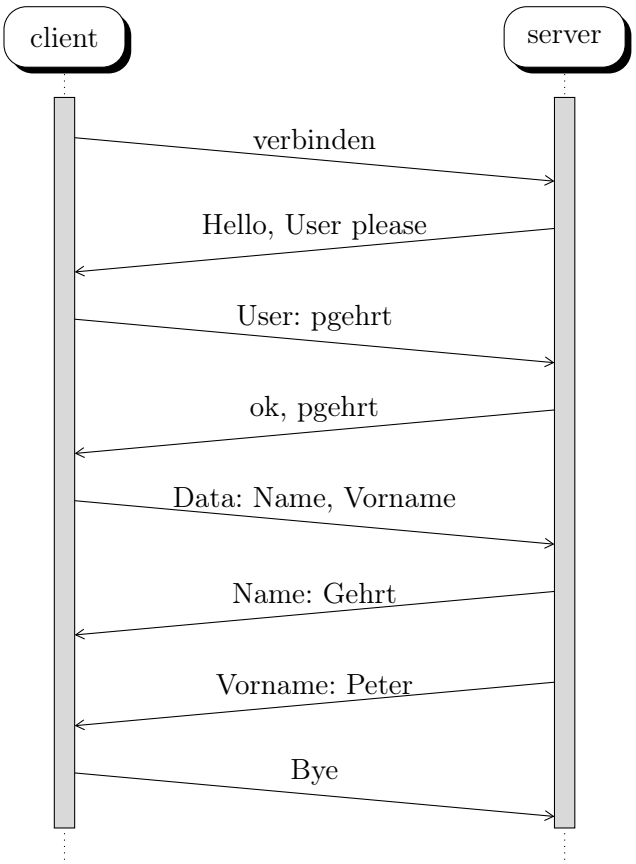
Sequenzdiagramme dienen im Unterricht dazu, um den linearen Ablauf von Aktionen zwischen verschiedenen Objekten oder Clients mit einem Server zu verdeutlichen. Allen ist gemeinsam, dass sie Aktionen bei den Partnern auslösen können.

Die Lebenslinien innerhalb von Sequenzdiagrammen verlaufen von oben nach unten. Je nach eingesetztem Werkzeug durchgängig oder gestrichelt.

Bei Sequenzdiagrammen zwischen Objekten werden die Objekte in verkleinerte Objektkarten, die nur den Bezeichner enthalten, über die jeweilige Lebenslinie geschrieben. Tritt ein Objekt in Aktion, so wird dieses durch einen Block auf der Lebensline verdeutlicht. Aktionen bei anderen Objekten werden mit einen durchgängigen waagerechten Pfeil gekennzeichnet, auf dem die aufgerufene Methode incl. der Parameter steht. In jedem Fall wird ein gestrichelter Pfeil am Ende der Aktion wieder zum Auslöser zurückgezeichnet, auf dem die mögliche Rückgabe steht.



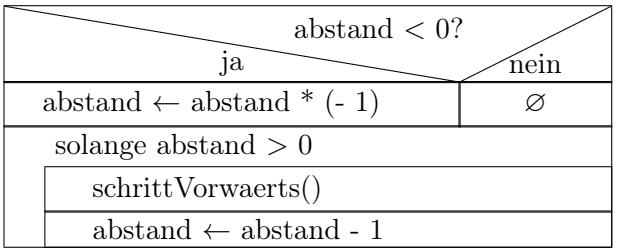
Der Ablauf der Aktionen zwischen Clients und Server wird in einem Sequenzdiagramm verdeutlicht, bei dem die Pfeile schräg nach unten verlaufen. Auch diese werden mit der entsprechenden Aktion versehen. Rückmeldungen können nur durch Aktionen in Gegenrichtung gegeben werden.



2.4 Struktogramm

Es ist darauf zu achten, dass ein einzelnes Struktogramm immer gleich breit ist. Fehlt der Alternativbereich bei einer Verzweigung, so kann dieser mit dem Zeichen für die leere Menge gekennzeichnet werden.

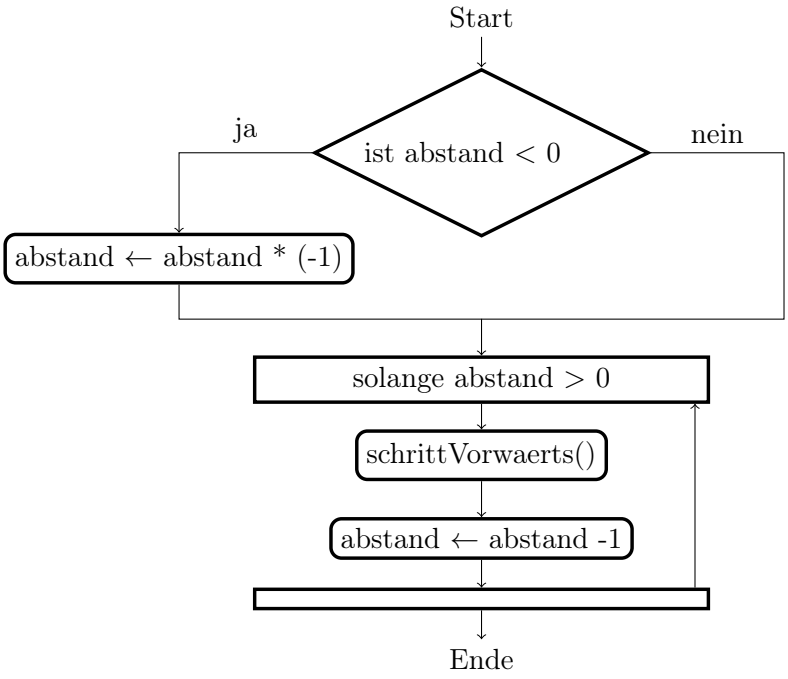
Bei Zuweisungen ist ein Pfeil von rechts nach links anzugeben. Damit wird betont, dass die rechte Seite zunächst ausgewertet und anschließend der linken Seite zugewiesen wird.



2.5 Ablaufdiagramm

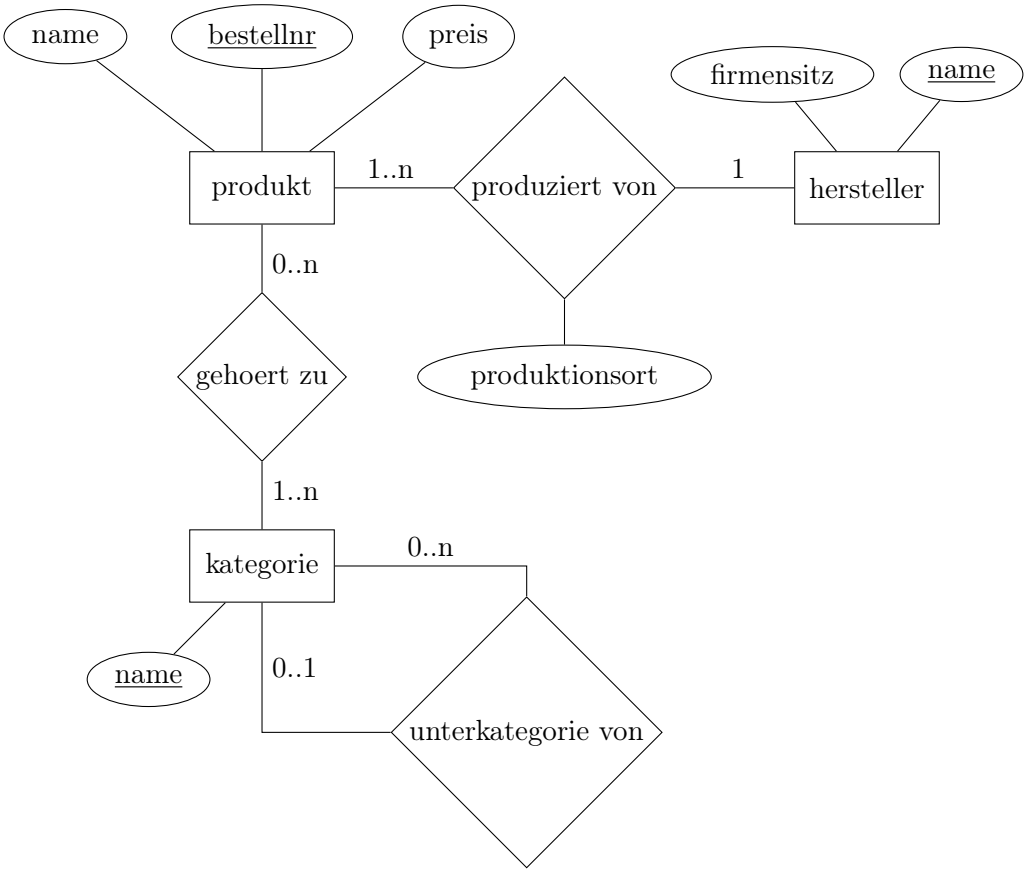
Alternativ zum Struktogramm kann auch ein Ablaufdiagramm eingesetzt werden. Zuweisungen werden hier wie im Struktogramm behandelt.

Ablaufdiagramme haben die generelle Richtung von oben nach unten. Für Wiederholungen müssen die Elemente speziell angeordnet werden, um diese Struktur zu verdeutlichen. Auf diese Weise kann im daraus erzeugten Programmcode auf Sprungmarken verzichtet werden.



2.6 ER-Diagramm

Zur graphischen Darstellung der Beziehungen zwischen Tabellen in einer relationalen Datenbank werden ER-Diagramme herangezogen. Entitäten werden dabei in Rechtecke geschrieben und mit den Attributen in Ovalen verbunden. Schlüsselattribute werden dabei durch unterstreichen deutlich gemacht. Die Beziehungen werden mit Rauten zwischen den Entitäten gezeichnet an deren Verbindungslinien die Kardinalitäten angegeben werden. Diese Beziehungen können selber auch zusätzliche Attribute haben.



2.7 Wertetabellen – Wertbelegungstabellen

Die Darstellung der Wertänderungen von Attributen und Variablen innerhalb des Ablaufs (z. B. in einer Methode) geschieht mit Hilfe von Wertetabellen. Dabei werden in den Zeilen die Werte einer Variablen oder eines Attributes angegeben. Die Spalten werden sinnvoll unterteilt z. B. durch die Angabe der Zeilen im Programm oder den Durchläufen einer Wiederholungsstruktur. Die Werte für die einzelnen Attribute oder Variablen werden immer dann neu hingeschrieben, wenn eine neue Zuweisung erfolgt. Diese erfolgt auch, wenn der gleiche Wert erneut zugewiesen wird. Kommt es zu zwei Zuweisungen für ein Attribut oder Variable innerhalb einer Spalte, so wird der vorherige Wert durchgestrichen und der neue dahinter geschrieben.

Bezeichner	Wert				
durchlauf	1	2	3	4	5
index	0	1	2	4	8
anfang	5		8		~~6~~ 4
ziel	3	3	4		

3 Programmcode

3.1 Java

Programmcode, der in der Sprache Java geschrieben wird, ist einzurücken. Dabei beginnt jede Einrückebene direkt nach dem Öffnen einer geschweiften Klammer. Die öffnende geschweifte Klammer wird dabei immer an das Ende der einleitenden Zeile geschrieben und die schließende Klammer immer alleine in einer Zeile in der gleichen Einrücktiefe wie die Zeile der öffnenden Klammer. Einzige Ausnahme von dieser Regel ist der „else“-Bereich einer bedingten Anweisung, hier steht } **else** { in einer Zeile in gleicher Einrücktiefe wie der Beginn der bedingten Anweisung.

Alle bedingten Anweisungen sind mit geschweiften Klammern zu schreiben, auch wenn der Anweisungsblock nur aus einer Anweisung besteht. Auf den „Ternary-Operator“, die Kurzform der bedingten Anweisung, ist zu verzichten.

Bei dem Zugriff auf die eigenen Attribute ist immer das Schlüsselwort **this** zu verwenden. Dafür ist auf die Verwendung von Präfixen für z. B. Parameter zu verzichten, wenn diese nicht durch andere Vorgaben benötigt werden. Innerhalb einer Klasse soll folgende Reihenfolge eingehalten werden: Attribute gefolgt von den Konstruktoren und danach die Methoden.

```
1  class Beispiel {
2    private String name;
3    private int zahl;
4
5    public Beispiel() {
6      this.name = "";
7    }
8
9    public Beispiel(String name) {
10     this.name = name;
11   }
12
13   public int gibAbsolutWert() {
14     if (zahl > 0) {
15       return this.zahl;
16     } else {
17       return this.zahl * (-1);
18     }
19   }
20
21   public void setzeZahl(int zahl) {
22     this.zahl = zahl;
23   }
24
25 }
```

3.2 Python

In unserem Quellcode werden grundsätzlich keine Präfixe verwendet. Bezeichner für Parameter werden mit dem ersten Buchstaben klein geschrieben. Die Bezeichner von Parametern sollten sich von Attributbezeichnern unterscheiden, um einer Verwechselung vorzubeugen.

Sollen Attribut- und/oder Methodenbezeichner nicht öffentlich (public) sein, so wird ihnen ein Unterstrich vorangestellt.

Python-Quellcode muss aus syntaktischen Gründen jeweils nach einem Doppelpunkt eingerückt werden. Dabei ist darauf zu achten, dass in dem Editor für eine Datei immer die Einrückungen gleich vorgenommen werden, d. h. insbesondere, dass empfohlen wird, immer um zwei Leerzeichen einzurücken und dies geeignet für den Editor festzulegen.

Damit selbstdokumentierender Quellcode erstellt werden kann, ist die Konstruktion """ ... """ jeweils eingerückt hinter die zu erläuternde Zeile zu setzen.

Jede Klasse wird mit einer Testroutine ausgestattet: Dazu wird am Ende der Datei die einleitende Konstruktion **if** __name__ == "__main__": verwendet – danach ist wie üblich einzurücken. Jede in der Datei befindlichen Klassen ist mindestens einmal zu instanziieren. Alle Methoden, die die Klasse anbietet, sind mindestens einmal auszuführen.

```
1  class Beispiel:
2    """Erläutert die Darstellung von Quellcode"""
3    def __init__(self,meinName=""):
4      """Konstruktor inkl. Polymorphie für 0 oder 1 Parameterwert. Es
       wird ein Wert gesetzt, wenn der Parameter leer ist (default)"""
5      self.name=meinName
6    def setzeZahl(self,neueZahl):
7      """setzt den Attributwert für zahl"""
8      self.zahl=neueZahl
9    def gibAbsolutWert(self):
10     """Dokumentation der Verzweigung, Standardmethode: __abs__() """
11     if self.zahl < 0:
12       return -self.zahl
13     else:
14       return self.zahl
15
16 if __name__ == "__main__":
17   print('Testroutine für die Klasse Beispiel')
18   print('meinTest=Beispiel("Ludger")')
19   meinTest=Beispiel("Ludger")
20   print('meinTest.setzeZahl(-1234567891011)')
21   meinTest.setzeZahl(-1234567891011)
22   print('meinTest.gibAbsolutWert()',meinTest.gibAbsolutWert())
```

3.3 SQL

Bei der Datenbankabfragesprache SQL sind alle Schlüsselworte in Großbuchstaben zu schreiben. Außerdem ist das abschließenden Semikolon mit anzugeben. Tabellennamen und Spaltennamen sollen klein geschrieben werden und keine Sonderzeichen enthalten. Daher müssen sie innerhalb von Befehlen auch nicht in Backticks eingefasst werden. Die Wahl zwischen einfachen oder doppelten Anführungszeichen bei dem Einfassen von Zeichenketten ist frei zu treffen. Dieses sollte, wie im folgenden Beispiel, innerhalb eines Befehls einheitlich sein.

```
SELECT name, preis FROM produkt WHERE hersteller = "Augusto" AND
 ursprungsland = "Deutschland";
```