Zugriff auf eine Datenbank mit JDBC

Ziel: Kennenlernen elementarer Möglichkeiten eine Verbindung zu einem DBMS aufzubauen und Daten aus einer Datenbank abzurufen.

1.1. Vorbereitung: MySQL Server starten und Admin Oberfläche phpMyAdmin öffnen:

- In Moodle finden Sie das Dokument `32_MySQL_DBMS.pdf`; dort ist beschrieben, wie der MySQL Server gestartet wird und wie die Web-Schnittstelle zum DBMS geöffnet werden kann.
- Für die weitere Übung wird eine Übungsdatenbank benötigt. In Moodle liegt die Datei IT_Center_v2012.sql; diese muss importiert werden.
- Man kann sich in phpMyAdmin einen Überblick über die Tabellen der importierten DB verschaffen.

1.2. Vorbereitung: JDBC – MySQL Treiber Jar in Eclipse bereitstellen:

- Die .jar-Datei kann von Moodle geholt werden.
- Wie bereits andere .jar-Dateien, muss diese unter Project→Properties→Java Build Path eingebunden werden.

1.3. Erstellen einer Java Anwendung zur Ausführung von SQL SELECT Befehlen:

```
Select Anweisung (Abbruch mit <Enter>): SELECT ArtikelNr, Artikelname FROM Artikel WHERE Artikelgruppe = "SO"
80001                                   Kaspersky  Antivirus 2010 1. Liz. Update DVD [DE]
80002                                   pcANYWHERE 12.5 Host only dt.CD
80003                                   pcANYWHERE 12.5 Host und Remote dt.CD
80004                                   Kaspersky  Internet Security 2010  1Liz. BOX [DE]
80005                                   MS SB Win. 2008 SBS Pre.+5CAL FR. 1Pack+++
80006                                   MS SB Win. 2008 SBS 5 Dev.CAL Ste.FR.
80007                                   Kaspersky  Open Space Security -ExpansionPack (5W)
80008                                   Norton AntiVirus 2009  1 Usr dt.CD
Select Anweisung (Abbruch mit <Enter>):
```

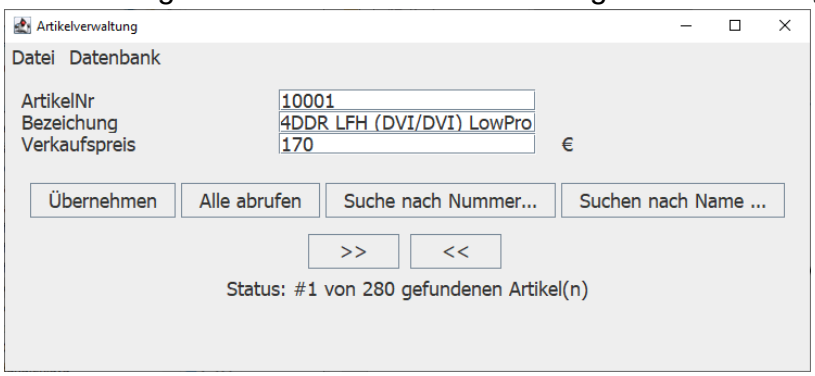
- Im Interface Daten werden die Informationen für Datenbank Verbindung bereitgestellt (siehe Bsp. im Skript; Achtung Datenbankname kann anders lauten)
- Die Methoden für den Verbindungsaufbau und DB Zugriff werden in einer eigenen Klasse bereitgestellt (siehe Bsp. im Skript)
- In Main wird versucht die DB Verbindung aufzubauen.
- Der Benutzer erhält in Main die Möglichkeit wiederholt einen SELECT Befehl einzugeben.
- Der SELECT Befehl wird in einer eigenen Methode ausgeführt und das Ergebnis dort angezeigt.
- Da die eingegebenen SELECT Befehle benutzerabhängig sind, muss die Anzeige des Ergebnisses aus dem ResultSet flexibel gestaltet werden:

  Die Anzahl der Spalten im ResultSet wird aus den Metadaten bestimmt (siehe Skript)

  Für das Abrufen der Daten aus dem ResultSet wird immer der Datentyp String verwendet: Methode `getString()`

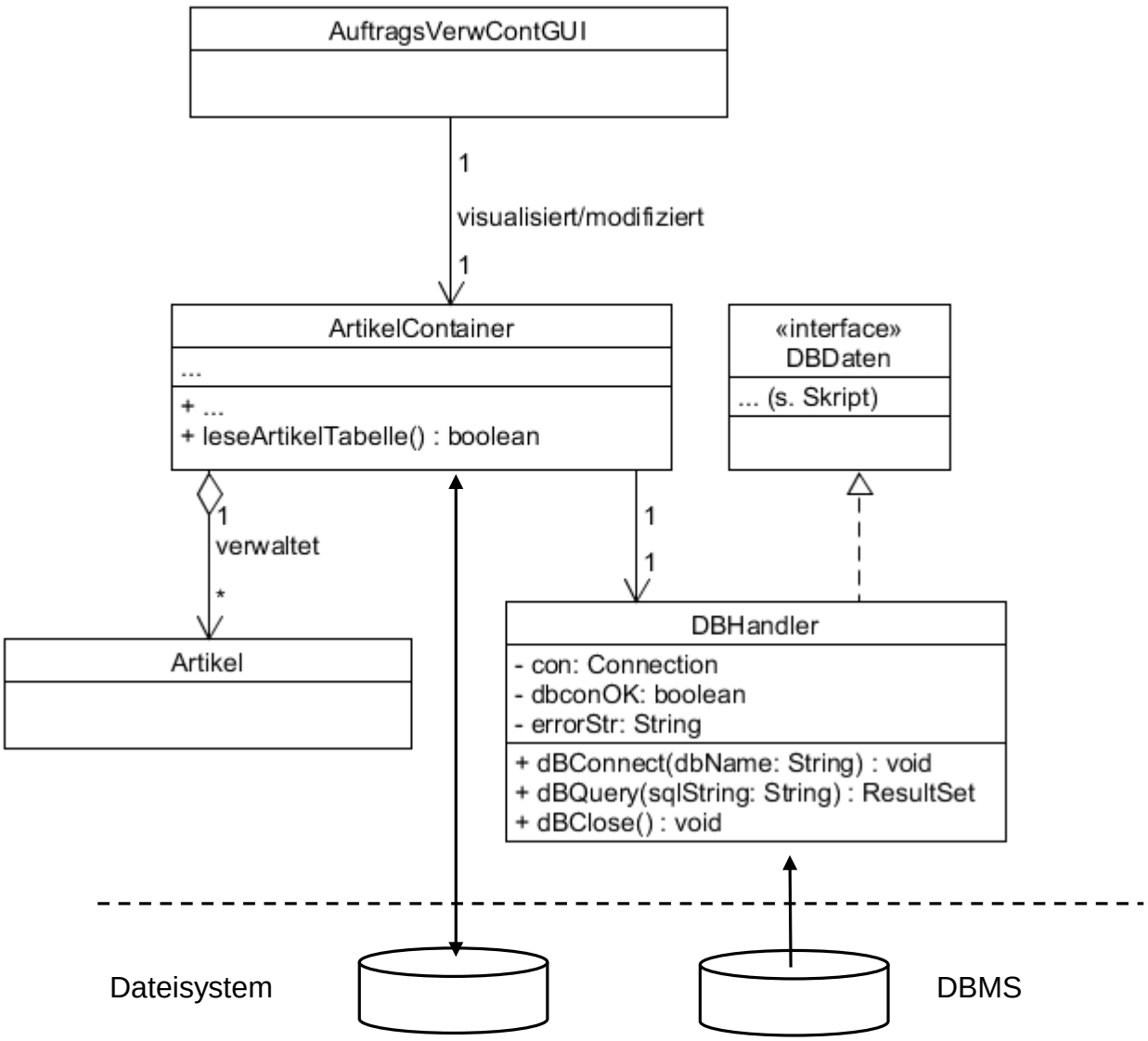
  Die Spaltenbreite für die Ausgabe am Bildschirm wird aus den Metadaten bestimmt.

2.1. Erweiterung der existierenden Anwendung zur Verwaltung von Artikeldaten:



In der bisherigen Variante konnten wir die Artikeldaten in einer serialisierten Datei speichern und daraus wieder lesen.

Das Programm soll jetzt so erweitert werden, dass mit Hilfe des Menüaufrufs `Datenbank → Artikeltabelle laden` die Artikeldaten aus einer Datenbank geladen werden.



Für eine bessere Strukturierung der Anwendung soll der Datenbankzugriff über eine zusätzliche Klasse `DBHandler` erfolgen:

Folgende Erweiterungen müssen an der bestehenden Anwendung durchgeführt werden:

1. Im Design-Fenster des Windowbuilder:
   - Der Menü-Bar ein zusätzliches Menü `Datenbank` hinzufügen.
   - Dem Menü ein zusätzliches Menü-Item `Artikeltabelle laden` hinzufügen.
2. Dem Menü-Item über das Kontextmenü Add `event handler` → `action` einen `ActionListener` hinzufügen.
   Die Klasse `DBHandler` implementieren (s. UML Diagramm):
   die Klasse dient zur Verwaltung der Datenbankverbindung (`Connection`) und der Ausführung von Queries. Falls eine Exception auftritt, wird diese in den Methoden der Klasse abgefangen, das Attribut `dbconOK` auf `false` gesetzt und der Exception-Fehlerstring in dem Attribut `errorStr` gespeichert.
   Für die Attribute `dbconOK` und `errorStr` werden get-Methoden bereitgestellt.
   - Der Konstruktor führt folgende Initialisierungen durch:
     `con` ← `null`
     `dbconOK` ← `false`
     `errorStr` ← "Keine DB Verbindung aufgebaut!"
   - Die Methode `dbConnect` hinzufügen:
     Die Methode versucht eine Datenbankverbindung aufzubauen und speichert diese in dem Attribut `con` und setzt im Erfolgsfall das Attribut `dbconOK` auf `true` (Fehlerverhalten s.o.) Die notwendigen Informationen für den Verbindungsaufbau werden dem Interface `DBDaten` und dem Parameter entnommen.
   - Die Methode `dbQuery` hinzufügen:
     Die Methode versucht den übergebenen SQL-String auszuführen und gibt das Ergebnis `ResultSet` im Erfolgsfall als Rückgabewert zurück (Fehlerverhalten s.o.)

3. Die Methode `leseArtikelTabelle` benutzt eine lokales Objekt der Klasse `DBHandler` um eine Datenbankverbindung aufzubauen und damit die notwendigen Daten mit Hilfe einer SQL-Anweisung zu lesen.
   Das ResultSet der Abfrage wird ausgewertet und die Artikel in der ArrayList `alleArtikel` gespeichert.
   Danach wird die DB Verbindung wieder geschlossen.
   Tritt ein Fehler auf, wird an den Aufrufer als Rückgabewert `false` zurückgegeben, sonst `true`.

4. Die Methode `leseArtikelTabelle` wird im ActionListener (s. 2.) aufgerufen. Im Erfolgsfall wird in einem MessageDialog eine Erfolgsmeldung ausgegeben und die Methode abrufenArtikel() aufgerufen. Im Fehlerfall wird eine Fehlermeldung angezeigt.