Übungsblatt zur Anwendung

Stand: Leistungskurs 2024

*Zu beachten ist, dass die Kontexte **nicht unbedingt** Aufgabenübergreifend gelten.*
Die Aufgabenstellungen sind ab hier in Kursiv dargestellt.

Aufgabe 1:

Schreiben Sie eine Java-Anwendung, die mithilfe der bereitgestellten Klassen DatabaseConnector und QueryResult Daten aus einer Datenbank abruft und verarbeitet. Verwenden Sie die folgenden Schritte:

1. Erstellen Sie ein Objekt der Klasse DatabaseConnector mit den folgenden Verbindungsinformationen:
    - IP-Adresse: "localhost"
    - Port-Nummer: 3306
    - Datenbankname: "schuldatenbank"
    - Benutzername: "benutzer"
    - Passwort: "geheim"
2. Führen Sie einen SQL-Befehl aus, um alle Schüler aus der Tabelle "Schueler" abzurufen. Verwenden Sie die Methode executeStatement.
3. Überprüfen Sie, ob die Abfrage erfolgreich war und ob Ergebnisse vorhanden sind. Nutzen Sie dafür die Methode getCurrentQueryResult. Falls keine Ergebnisse vorliegen, geben Sie eine entsprechende Meldung aus.
4. Falls Ergebnisse vorhanden sind, geben Sie die Daten der Schüler aus. Verwenden Sie dazu die Methoden der Klasse QueryResult (getData, getColumnNames, getColumnTypes, getRowCount, getColumnCount).
5. Schließen Sie die Verbindung zur Datenbank, indem Sie die Methode close der Klasse DatabaseConnector aufrufen.

Anhang **ZA-Klassen:**

Dokumentation der Klasse `DatabaseConnector`

Konstruktor **`DatabaseConnector(String pIP, String pPort, String pDatabase, String pUsername, String pPassword)`**

Ein Objekt vom Typ `DatabaseConnector` wird erstellt, und eine Verbindung zur Datenbank wird aufgebaut. Mit den Parametern `pIP` und `pPort` werden die IP-Adresse und die Port-Nummer übergeben, unter denen die Datenbank mit Namen `pDatabase` zu erreichen ist. Mit den Parametern `pUsername` und `pPassword` werden Benutzername und Passwort für die Datenbank übergeben.

Auftrag **`void executeStatement(String pSQLStatement)`**

Der Auftrag schickt den im Parameter `pSQLStatement` enthaltenen SQL-Befehl an die Datenbank ab.
Handelt es sich bei `pSQLStatement` um einen SQL-Befehl, der eine Ergebnismenge liefert, so kann dieses Ergebnis anschließend mit der Methode `getCurrentQueryResult` abgerufen werden.

Anfrage **`QueryResult getCurrentQueryResult()`**

Die Anfrage liefert das Ergebnis des letzten mit der Methode `executeStatement` an die Datenbank geschickten SQL-Befehls als Objekt vom Typ `QueryResult` zurück.
Wurde bisher kein SQL-Befehl abgeschickt oder ergab der letzte Aufruf von `executeStatement` keine Ergebnismenge (z.B. bei einem INSERT-Befehl oder einem Syntaxfehler), so wird `null` geliefert.

Anfrage **`String getErrorMessage()`**

Die Anfrage liefert `null` oder eine Fehlermeldung, die sich jeweils auf die letzte zuvor ausgeführte Datenbankoperation bezieht.

Auftrag **`void close()`**

Die Datenbankverbindung wird geschlossen.

Dokumentation der Klasse `QueryResult`

Anfrage `String[][] getData()`

Die Anfrage liefert die Einträge der Ergebnistabelle als zweidimensionales Feld vom Typ `String`. Der erste Index des Feldes stellt die Zeile und der zweite die Spalte dar (d.h. `String[zeile][spalte]`).

Anfrage `String[] getColumnNames()`

Die Anfrage liefert die Bezeichner der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück.

Anfrage `String[] getColumnTypes()`

Die Anfrage liefert die Typenbezeichnung der Spalten der Ergebnistabelle als Feld vom Typ `String` zurück. Die Bezeichnungen entsprechen den Angaben in der MySQL-Datenbank.

Anfrage `int getRowCount()`

Die Anfrage liefert die Anzahl der Zeilen der Ergebnistabelle als `int`.

Anfrage `int getColumnCount()`

Die Anfrage liefert die Anzahl der Spalten der Ergebnistabelle als `int`.