JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

INSY – Übung

Prepared Statements

Informationssysteme
4CHIT 2015/16

Johannes Ucel,
Michael Stöger

Version 1.0

Note:

Begonnen am 18. Februar 2016

Betreuer: Michael Borko, Christoph Roschger

Beendet am 25. Februar 2016

Inhaltsverzeichnis

1 Einführung

1.1 Einleitung

„PreparedStatements sind in JDBC eine Möglichkeit SQL-Befehle vorzubereiten um SQL-Injections zu vermeiden. Die Typüberprüfung kann somit schon bei der Hochsprache abgehandelt werden und kann so das DBMS entlasten und Fehler in der Businesslogic behandelbar machen." [BOR16]

1.2 Ziele

„Es ist erwünscht Konfigurationen nicht direkt im Sourcecode zu speichern, daher sollen Property-Files [3] zur Anwendung kommen bzw. CLI-Argumente (Library verwenden) [1,4] verwendet werden. Dabei können natürlich Default-Werte im Code abgelegt werden.
Das Hauptaugenmerk in diesem Beispiel liegt auf der Verwendung von PreparedStatements [2]. Dabei sollen alle CRUD-Aktionen durchgeführt werden." [BOR16]

1.3 Voraussetzungen

Funktionierender Java Database Connector und das Verstehen von Prepared Statements.

1.4 Aufgabenstellung

" *Verwenden Sie Ihren Code aus der Aufgabenstellung "Simple JDBC Connection" um Zugriff auf die Postgresql Datenbank "Schokofabrik" zur Verfügung zu stellen. Dabei sollen die Befehle (CRUD) auf die Datenbank mittels PreparedStatements ausgeführt werden. Verwenden Sie mindestens 10000 Datensätze bei Ihren SQL-Befehlen.*
Diese können natürlich sinnfrei mittels geeigneten Methoden in Java erstellt werden.

Die Properties sollen dabei folgende Keys beinhalten: host, port, database, user, password

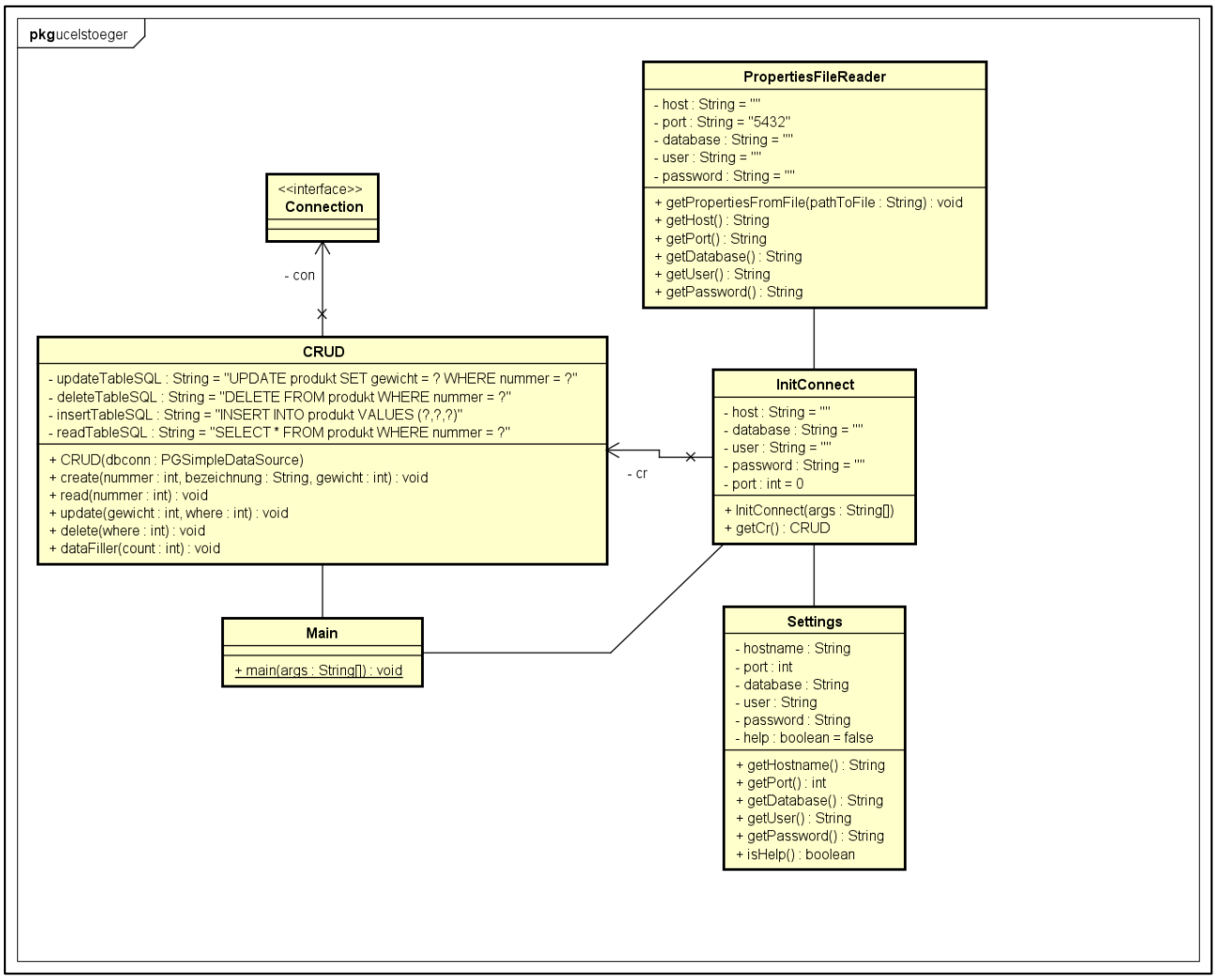
Vergessen Sie nicht auf die Meta-Regeln (Dokumentation, Jar-File, etc.)! Die Testfälle sind dabei zu ignorieren. Diese Aufgabe ist als Gruppenarbeit (2 Personen) zu lösen." [BOR16]

2 Ergebnisse

2.1 Database Connector

In dieser Übung wurde der bereits vorhandene Database Connector aus der Übung „Simple JDBC Connection“ übernommen. Dieser wurde nun so erweitert, dass die Parameter host, port, database, user und password entweder in einem Property – File oder in der CLI übermittelt werden können. Dadurch sind diese Werte nicht statisch eingetragen (hardcoded) und der Database Connector lässt sich dadurch flexibler verwenden.

2.2 UML – Entwurf



2.3 CLI (JCommander)

Um das Verwenden von CLI – Argumenten zu ermöglichen, wird die Library „JCommander" verwendet. [JCO16]

Um eine problemlose Erweiterbarkeit des Programms zu gewährleisten, werden die CLI – Einstellungen in einem separaten File festgelegt. Hierfür muss in Java die Library importiert werden (*import com.beust.jcommander.*;*).
Danach werden die gewünschten CLI – Argumente mithilfe der Annotation @Parameter bestimmt, wobei auch zusätzlich eine Beschreibung zu dem jeweiligen Argument hinzugefügt wird. Zusätzlich lässt sich och festlegen, ob es sich um ein Pflichtargument (required) handelt.

```
@Parameter(names = "-h", description = "Hostname", required = true)
private String hostname;
```

Abbildung 1 Beispiel eines CLI – Arguments

Um dann vom Connector aus auf die privaten Attribute zugreifen zu können, werden Getter – Methoden generiert.
Sollte der Benutzer eine Übersicht der CLI – Argumente wollen, erhält er diese mittels --help

```
@Parameter(names = "--help", help = true)
private boolean help = false;
```

Abbildung 2 Definition vom Help – Argument

Wird dieses Argument verwendet, wird die von der Library vorhandene Methode „usage" aufgerufen, welche die Befehle mit der davor hinzugefügten Beschriftung auflistet.

```
if (settings.isHelp()) {
    cmd.usage();
    return;
}
```

Abbildung 3 Auswertung von --help

Für die weitere Auswertung wird wie in Abbildung 3 zu sehen ist, ein Objekt der Settings – Klasse benötigt. Dieses Objekt wird dann dem JCommander Objekt als Parameter übergeben, da er sich um diese Argumente kümmern soll. Anschließend können dann mittels Getter – Methoden die gewünschten Werte aus der CLI ausgelesen werden.

2.4 Properties File

Anstatt dem Eingeben der Daten über die CLI, können die Einstellungen auch über ein properties file ausgelesen werden.
Dieses properties file kann folgende Optionen haben:

- host
- port(optional)
- database
- user
- password

Beim Ausführen des Programms muss es mit „-f PfadZurDatei" angegeben werden.

2.5 CRUD

In diesem Beispiel werden auf die Datenbank CRUD (Create Read Update Delete) Befehle ausgeführt.

CREATE

Zu Beginn wird die Datenbank mit 10000 Zufallsdaten gefüllt, welche mithilfe eines erstellten Datafiller generiert werden.

```
/**
 * Generiert Zufallsdaten in der "produkt" Tabelle
 *
 * @param count
 *              <- Wie viele Zeilen
 */
public void dataFiller(int count) {
    int maxnummer = 0;
    try {
        Statement tmp = con.createStatement();
        // Holt sich die aktuelle Maxnummer, um zukünftige Fehler mit der
        // falschen Produktnummer zu verhindern
        ResultSet tmp2 = tmp.executeQuery("SELECT max(nummer) FROM produkt;");
        while (tmp2.next()) {
            maxnummer = (tmp2.getInt(1));
        }
    } catch (SQLException e) {
        System.out.println("Fehler beim Lesen der Maxnummer!");
    }
    int begin = maxnummer + 1;
    for (int i = 0; i < count; i++, begin++)
        create(begin, "TEST" + begin, begin + 1);

}
```

Abbildung 4 Datafiller

Dieser übernimmt die maximale Produktnummer +1, um eine richtige Weiternummerierung zu ermöglichen

Das Prepared Statement für das Inserten schaut folgendermaßen aus:
private final String insertTableSQL = "INSERT INTO produkt VALUES (?,?,?)";

Mittels setInt bzw. setString können diese Fragezeichen mit sinnvollen Werten ersetzt werden.

READ
Das Prepared Statement für den Read – Befehl wurde so definiert:
*private final String readTableSQL = "SELECT * FROM produkt WHERE nummer = ?";*

Hier kann mittels setInt die gewünschte Produktnummer ausgewählt werden. Anschließend wird die Ergebnismenge (ResultSet) in der Konsole ausgegeben.

UPDATE
Um das Gewicht eines Produktes in der Tabelle zu verändern, wurde folgendes Prepared Statement definiert:
private final String updateTableSQL = "UPDATE produkt SET gewicht = ? WHERE nummer = ?";

Hier kann das gewünschte Gewicht bei einem Produkt anhand der Produktnummer geändert werden.

DELETE
Eine Zeile kann anhand der Produktnummer durch folgendes Prepared Statement gelöscht werden:
private final String deleteTableSQL = "DELETE FROM produkt WHERE nummer = ?";

ExecuteUpdate() wird bei INSERT, UPDATE oder DELETE Statements verwendet und gibt **KEIN** ResultSet zurück

3 Teamarbeit

Die Arbeit wird so aufgeteilt, dass Johannes Ucel für die CLI – Umsetzung und Michael Stöger für das Auslesen aus dem Property-File zuständig ist.

Weiteres werden die Prepared Statements so aufgeteilt, dass Michael Stöger die ersten zwei CRUD – Teile (CREATE, READ) und Johannes Ucel die restlichen zwei Teile (UPDATE, DELETE) übernimmt.

Korrekturarbeiten werden nicht explizit geteilt, sondern durchgeführt, sobald diese notwendig sind.

Der Aufwand wird wie folgt geschätzt:

Arbeitsteil	**Durchführender**	**geschätzter Aufwand**
CLI + UPDATE/DELTE	Johannes Ucel	2,5 h
Property + CREATE/UPDATE	Michael Stöger	2,5 h
Korrekturarbeiten	M. Stöger & J. Ucel	1 h/Person
Gesamt	**M. Stöger & J. Ucel**	**6 h**

Der tatsächliche Aufwand ist wie folgt aufgeschlüsselt:

Michael Stöger:

Durchgeführte Arbeit	**Datum**	**tatsächlicher Aufwand**
Properties File	18.02.2016	1 h
CRUD – Durchführung	18.02.2016 – 25.02.2016	2,5 h
Gesamt	**18.02.2016 – 25.02.2016**	**3,5 h**

Johannes Ucel:

Durchgeführte Arbeit	**Datum**	**tatsächlicher Aufwand**
CLI (JCommander)	18.02.2016	1,5 h
CRUD – Durchführung	18.02.2016 – 25.02.2016	2,5 h
Gesamt	**18.02.2016 – 25.02.2016**	**4 h**

Gesamt:

Durchführender	**tatsächlicher Aufwand**
Michael Stöger	3,5 h
Johannes Ucel	4 h
Gesamt	**7,5 h**

Git – Repository: https://github.com/mstoeger-tgm/INSY_Prepared_Statements

4 Literaturverzeichnis

[BOR16] Michael Borko. (2016, Februar). Prepared Statements [Online]. Available at: https://elearning.tgm.ac.at/mod/assign/view.php?id=47181 [abgerufen am 23.02.2016]

[OPS16] Oracle. Using Prepared Statements [Online]. Available at: https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html [abgerufen am 18.02.2016]

[JCO16] JCommander. CLI Library [Online]. Available at: http://jcommander.org/ [abgerufen am 18.02.2016]