Prepared Statements

|  |  |
| --- | --- |
| **Autor/en:** | **Özsoy Ahmet**  **Tiryaki Seyyid** |
| **Klasse:** | 4AHITM |
| **Letzte Änderung:** | 04.05.2016 |
| **Version:** | 1.0 |

**Inhaltsverzeichnis**

[Aufgabenstellung 2](#_Toc445285967)

[Arbeitsaufteilung mit Aufwandschätzung 3](#_Toc445285972)

[Endzeitaufteilung 4](#_Toc445285973)

[Technologiebeschreibung 5](#_Toc445285974)

[Arbeitsdurchführung 6](#_Toc445285975)

[Quellenangaben 7](#_Toc445285976)

# Aufgabenstellung

### **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.

### **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.

### **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.

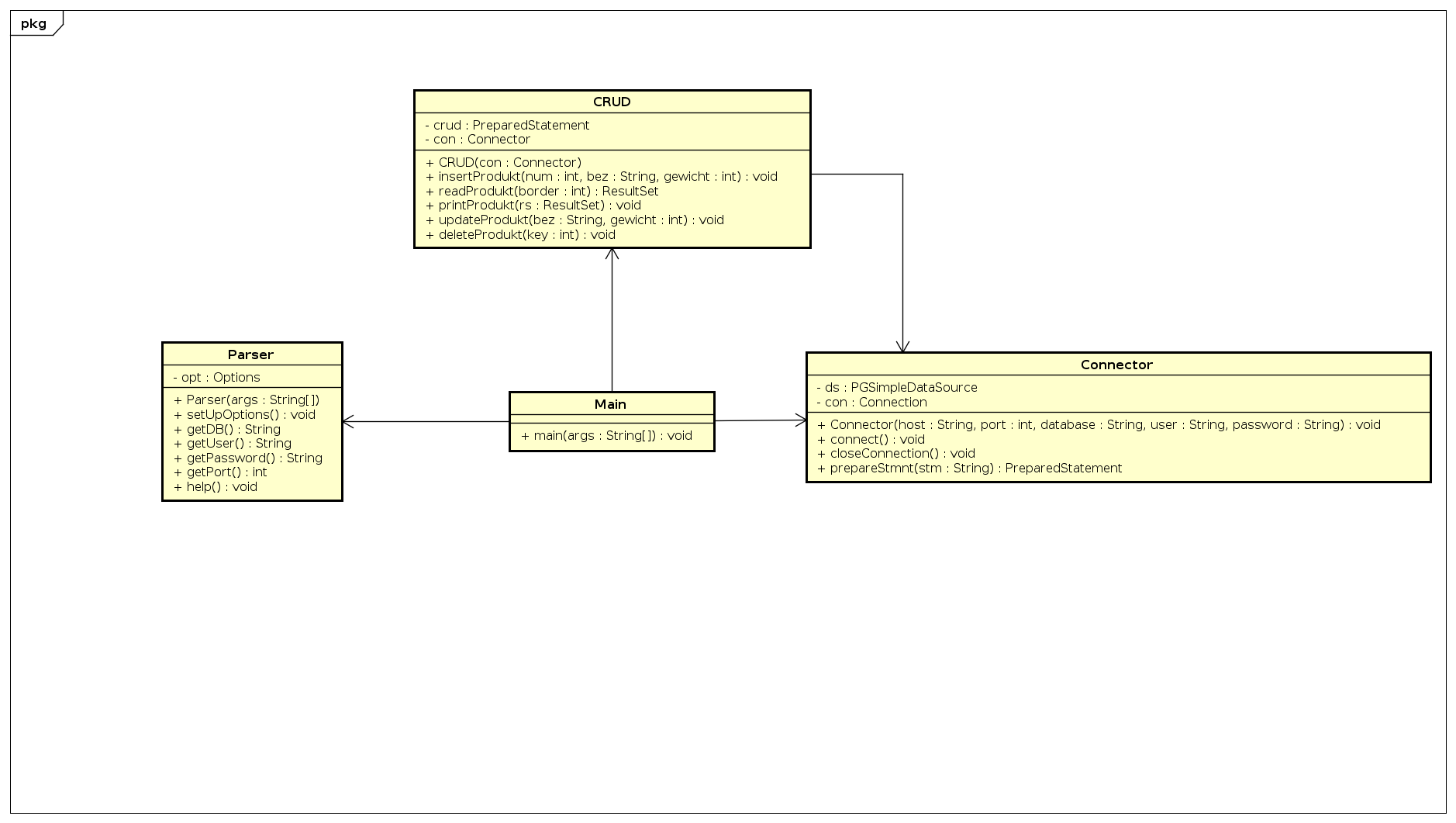
Die Properties sollen dabei folgende Keys beinhalten: host, port, database, user, password

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.

### **Quelle**

[1] Apache Commons CLI; Online: http://commons.apache.org/proper/commons-cli/  
[2] Java Tutorial JDBC "Prepared Statements"; Online: https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html  
[3] Java Tutorial Properties; Online: https://docs.oracle.com/javase/tutorial/essential/environment/properties.html  
[4] Overview of Java CLI Libraries; Online: http://stackoverflow.com/questions/1200054/java-library-for-parsing-command-line-parameters

# Designüberlegung



Die Applikation wird für diese Aufgabenstellung in mehrere Klassen unterteilt:

* **Connector:** um die Verbindung zuhaben
* **Parser:** um Kommandozeilenargumente zu verwalten
* **Main:** um Teilaufgaben zusammenzufügen
* **CRUD:** um die Prepared Statements durchzuführen

# Arbeitsaufteilung mit Aufwandschätzung

|  |  |  |  |
| --- | --- | --- | --- |
| Zuständige  Person(en) | Task | Beschreibung | Geschätzte Zeit in h |
| Özsoy Ahmet | CRUD | PreparedStatements Aufrufe | 1 h |
| Tiryaki Seyyid | Parser | CLI Argumente parsen | 1 h |
| Tiryaki Seyyid, Özsoy Ahmet | Design | UML und Designüberlegung | 3 h |

Geschätzter Gesamtzeitaufwand

|  |  |
| --- | --- |
| Person | Zeitaufwand in h |
| Özsoy Ahmet | 2,5 h |
| Tiryaki Seyyid | 2,5 h |
| Summe: | 5 h |

# Endzeitaufteilung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Zuständige  Person(en) | Task | Geschätzte Zeit in h | Tatsächliche Zeit in h | Kommentar |
| Özsoy Ahmet | CRUD | 1 | 1 |  |
| Tiryaki Seyyid | Parser | 1 | 2 |  |
| Tiryaki Seyyid, Özsoy Ahmet | Designüberlegung | 3 | 2 |  |

Tatsächlicher Gesamtzeitaufwand

|  |  |
| --- | --- |
| Person | Zeitaufwand in h |
| Özsoy Ahmet | 2 |
| Tiryaki Seyyid | 3 |
| Summe: | 5 |

# Technologiebeschreibung

PreparedStatements - Postgres

Apache CLI

# Arbeitsdurchführung

**Apache Commons CLI**

Apache Commons CLI ist eine Bibliothek um Kommandozeilenargumente auszulesen und zu verarbeiten. Dies geschieht in 3 Schritten. Zuerst müssen die erlaubten Optionen definiert werden. Dazu muss ein Options Objekt erstellt werden, welchem später die Optionen hinzugefügt werden.

Beispiel:

**opt** = **new** Options();

**opt**.addOption(Option.*builder*(**"H"**).argName(**"hostname"**).desc(**"Adresse des Servers"**).hasArg().longOpt(**"host"**)

.numberOfArgs(1).build());

Beispiel für eine getter-Methode:

**public** String getDB() {

**if** (**cli**.hasOption(**"d"**))

**return cli**.getOptionValue(**'d'**);

**else** {

System.***out***.println(**"Datenbankname fehlt!"**);

printHelper();

System.*exit*(-1);

**return ""**;

}

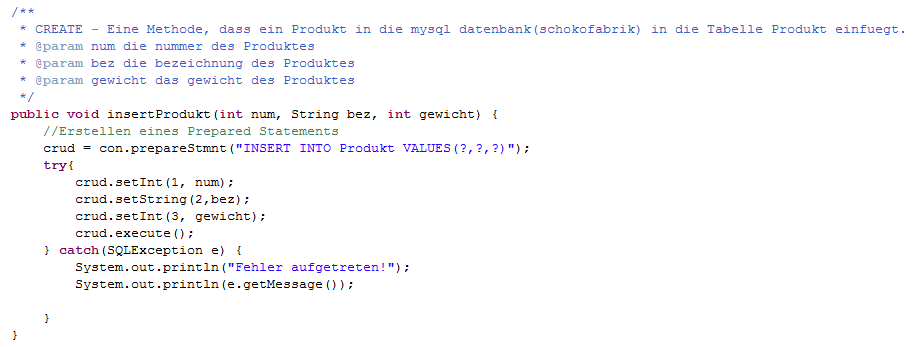
}

Als Ergebnis erhält man ein CommandLine Objekt, welches die Informationen aus der CLI Eingabe enthält. Zuletzt muss man dann noch im eigenen Code auf die Eingabe reagieren.

**Prepared Statements (CRUD)**

**CREATE**

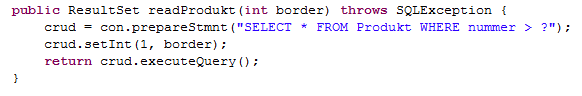
Beispiel:



Hier wird ein Produkt eingefügt. Die einzelnen Werte im Prepared Statement werden gefüllt und dann wird die Query ausgeführt.

**READ**

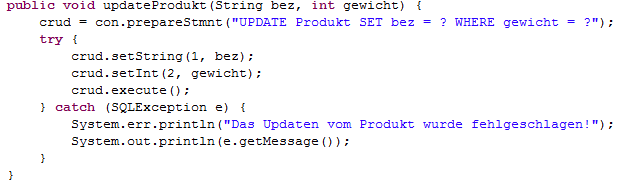
Beispiel:



Auslesen der Datenbank mithilfe des Prepared Statements.

**UPDATE**

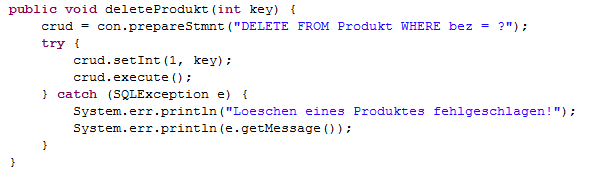
Beispiel:



Das Update Prepared Statement. Update auf ein Produkt.

**DELETE**

Beispiel:



Delete Statement

**GitHub Repository:** <https://github.com/stiryaki-tgm/PreparedStatements.git>

# Quellenangaben

|  |  |
| --- | --- |
| [BOR] | Titel: Michael Borko. (2016, März). Prepared Statements  Autor: -  Online/Quelle: http://commons.apache.org/proper/commons-cli/  geändert am: -  entnommen am: 09.03.2016 |
| [BOR] | Titel: *Michael Borko. (2016, März).* Prepared Statements  Autor: -  Online/Quelle: *https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html*  geändert am: -  entnommen am: 09.03.2016 |
| [BOR] | Titel: *Michael Borko. (2016, März).* Prepared Statements  Autor: -  Online/Quelle: *https://docs.oracle.com/javase/tutorial/essential/environment/properties.html*  geändert am: -  entnommen am: 09.03.2016 |
| [BOR] | Titel: *Michael Borko. (2016, März).* Prepared Statements  Autor: -  Online/Quelle: *http://stackoverflow.com/questions/1200054/java-library-for-parsing-command-line-parameters*  geändert am: -  entnommen am: 09.03.2016 |