Prof. Dr. Ingo Claßen
Prof. Dr. Martin Kempa
Patrick Dohmeier

2. Übungsblatt: JDBC

Voraussetzungen:

Für die folgende Aufgabe muss das Datenbankschema der **Miniwelt *Mauterhebung*** in Ihrer Datenbank vorhanden sein.

Sämtliche Vorgaben befinden sich in der Datei ***dbtech_jdbc.zip***, die ein vorbereitetes Eclipse-Projekt enthält und das Sie von der Webseite dieser Lehrveranstaltung herunterladen müssen. Dieses Projekt benötigt einige Java-Bibliotheken, die sich ebenfalls als Eclipse-Projekt in der Datei ***dbtech_lib.zip*** befinden.

Beide Projekte können Sie über den Menüpunkt **File > Import > Existing Projects into Workspace** (Zip-Datei unter Punkt select archive file: auswählen, Finish aktivieren) Ihrem Eclipse-Arbeitsbereich hinzufügen. Das Projekt ***dbtech_jdbc*** enthält bereits Klassen des Anwendungssystems. Diese Klassen dürfen Sie nur dann verändern, wenn es in der Aufgabe angegeben ist.

Während der Entwicklung des Programmcodes können Sie jederzeit Ihren aktuellen Zustand testen. Dafür gibt es eine Testklasse mit Namen ***MautVerwaltungTest.java*** (im Paket ***de.htwberlin.maut.test***). Die Testklasse basiert auf den Testframeworks JUnit und DBUnit, mit denen automatisierte Tests unterstützt werden. In die Eclipse Entwicklungsumgebung ist eine Testausführungskomponente integriert, mit der Sie die Tests durchführen können. Dazu aktivieren Sie aus dem Kontextmenü der Testklasse, das Sie über die rechte Maustaste erreichen, den Menüpunkt Run as > JUnit Test.

Für das Test- und Anwendungssystem ist eine Datenbankverbindung notwendig. Diese konfigurieren Sie über die Datei ***DbCred.java*** (im Paket ***de.htwberlin.utils***).

Bedenken Sie bitte, dass eine erfolgreiche Ausführung aller Tests nicht automatisch die Korrektheit Ihrer Lösung sicherstellt. Tests können immer nur die Anwesenheit von Fehlern zeigen, nicht aber deren Abwesenheit. Das liegt daran, dass Tests im Allgemeinen nicht vollständig alle Fehlersituationen abdecken. In der Bewertung Ihrer Lösung ist daher der erfolgreiche Durchlauf aller Tests eine notwendige Bedingung zum Erreichen der vollen Punktzahl. Es kann aber trotzdem Punktabzug geben, falls Ihre Lösung Fehler enthält, die durch die Tests nicht aufgedeckt werden.

1. Aufgabe (10 Punkte)

In diesem Übungsblatt sollen Sie eine Mautverwaltung implementieren, die mittels JDBC auf ein Datenbanksystem zugreift. Dabei handelt es sich um die fünf Methoden, die in der Schnittstelle MautVerwaltung beschrieben sind:

Prof. Dr. Ingo Claßen
Prof. Dr. Martin Kempa
Patrick Dohmeier

```java
package de.htwberlin.mautverwaltung;

/**
 * Die Schnittstelle enthält die Methoden für eine Mautverwaltung.
 *
 * @author Patrick Dohmeier
 */
public interface IMautVerwaltung {

	/**
	 * Liefert den Status eines Fahrzeugerätes zurück.
	 *
	 * @param fzg_id
	 *            - die ID des Fahrzeuggerätes
	 * @return status - den Status des Fahrzeuggerätes
	 **/
	String getStatusForOnBoardUnit(long fzg_id);

	/**
	 * Liefert die Nutzernummer für eine Mauterhebung, die durch ein Fahrzeug im
	 * Automatischen Verfahren ausgelöst worden ist.
	 *
	 * @param maut_id
	 *            - die ID aus der Mauterhebung
	 * @return nutzer_id - die Nutzernummer des Fahrzeughalters
	 **/
	int getUsernumber(int maut_id);

	/**
	 * Registriert ein Fahrzeug in der Datenbank für einen bestimmten Nutzer.
	 *
	 * @param fz_id
	 *            - die eindeutige ID des Fahrzeug
	 * @param sskl_id
	 *            - die ID der Schadstoffklasse mit dem das Fahrzeug angemeldet
	 *            wird
	 * @param nutzer_id
	 *            - der Nutzer auf dem das Fahrzeug angemeldet wird
	 * @param kennzeichen
	 *            - das amtliche Kennzeichen des Fahrzeugs
	 * @param fin
	 *            - die eindeutige Fahrzeugindentifikationsnummer
	 * @param achsen
	 *            - die Anzahl der Achsen, die das Fahrzeug hat
	 * @param gewicht
	 *            - das zulässige Gesamtgewicht des Fahrzeugs
	 * @param zulassungsland
	 *            - die Landesbezeichnung für das Fahrzeug in dem es offiziell
	 *            angemeldet ist
	 *
	 * **/
	void registerVehicle(long fz_id, int sskl_id, int nutzer_id,
			String kennzeichen, String fin, int achsen, int gewicht,
			String zulassungsland);

	/**
	 * Aktualisiert den Status eines Fahrzeuggerätes in der Datenbank.
	 *
	 * @param fzg_id
	 *            - die ID des Fahrzeuggerätes
	 * @param status
	 *            - der Status auf dem das Fahrzeuggerät aktualisiert werden
	 *            soll
	 */
```

```
    void updateStatusForOnBoardUnit(long fzg_id, String status);

    /**
     * Löscht ein Fahrzeug in der Datenbank.
     *
     * @param fz_id
     *            - die eindeutige ID des Fahrzeugs
     */
    void deleteVehicle(long fz_id);

    /**
     * liefert eine Liste von Mautabschnitten eines bestimmten Abschnittstypen
     * zurück. z.B. alle Mautabschnitte der Autobahn A10
     *
     * @param abschnittsTyp
     *             - der AbschnittsTyp kann bspw. eine bestimmte Autobahn (A10)
     *             oder Bundesstrasse (B1) sein
     * @return List<Mautabschnitt> - eine Liste des Abschnittstypen, bspw. alle
     *         Abschnitte der Autobahn A10
     **/
    List<Mautabschnitt> getMautabschnitte(String abschnittstyp);

}
```

Ihre Aufgabe besteht darin, die Klasse ***MautVerwaltungImpl.java*** zu realisieren, deren Funktionalität als JavaDoc-Kommentare in der Schnittstelle ***IMautVerwaltung.java*** beschrieben wird. Dabei dürfen Sie sämtliche Klassen der Vorgabe verwenden, aber nicht verändern. Es empfiehlt sich, zunächst die Abfragen mit dem SQL-Developer zu entwickeln und zu testen. In der Aufgabe werden Sie die grundlegenden Datenbankoperationen behandeln müssen. Diese umfassen die sogenannten CRUD-Operationen:

- **C**reate - anlegen eines Datensatzes mit dem insert-Befehl
- **R**ead – lesen eines Datensatzes mit dem select-Befehl
- **U**pdate – aktualisieren eines Datensatzes mit dem update-Befehl
- **D**elete – löschen eines Datensatzes mit dem delete-Befehl

Zusatzaufgabe (2 Punkte)

Präsentieren und erklären Sie Ihre erarbeitete Lösung in der nächsten Übung.

Abgabe

Termin und Modus wird durch den Lehrenden der jeweiligen Übung festgelegt.