Praktikum aus Softwareentwicklung 2

WS 2006/2007, LVA 365.009

Abgabetermin: Mo. 11. 12. 2006, 17:00 **Erreichbare Punkte: 24**

Übung 3 – Datenbankzugriff mit JDBC

Aufgabe 1: Schema 6 Punkte

In einer relationalen Datenbank sind Daten mit folgendem Schema gespeichert:

```
CREATE TABLE User (
        userId VARCHAR(20) PRIMARY KEY,
        surname VARCHAR(30),
        lastname VARCHAR(30) NOT NULL,
        password VARCHAR(20) NOT NULL,
        createDate DATETIME  NOT NULL
)

CREATE TABLE FaxPhoneContact (
        userId VARCHAR(20),
        phoneNumber VARCHAR(30) NOT NULL,
        isFax BIT DEFAULT '0',
        UNIQUE (userId, phoneNumber),
        CONSTRAINT fr_userIdFaxPhone FOREIGN KEY (userId) REFERENCES
User (userId)
)

CREATE TABLE EmailContact (
        userId VARCHAR(20),
        email VARCHAR(256) NOT NULL,
        isPrivate BIT DEFAULT '0',
        UNIQUE (userId, email),
        CONSTRAINT fr_userIdEmail FOREIGN KEY (userId) REFERENCES User
(userId)
)

CREATE TABLE MobileContact (
        userId VARCHAR(20),
        mobileNumber VARCHAR(256) NOT NULL,
        UNIQUE (userId, mobileNumber),
        CONSTRAINT fr_userIdEmail FOREIGN KEY (userId) REFERENCES User
(userId)
)
```

Implementieren sie dazu Java-Klassen (`User`, `FaxPhoneContact`, `EmailContact`, `MobileContact`) mit den entsprechenden Attributen sowie get/set-Methoden. Verwenden sie eine abstracte Basisklasse für die 3 Kontaktmöglichkeiten. Für die folgenden Aufgaben müssen obige Tabellen in einer relationalen Datenbank erzeugt werden. Verwenden Sie dazu die in der Übung vorgestellt HSQLDB Datenbank. Für diese Klasse sind noch keine Datenbank Zugriffe nötig.

Aufgabe 2: User-Kontakt-Datenbank 12 Punkte

Implementieren Sie Java-Klassen zum Zugriff auf die Userdaten und Kontaktdaten, welche folgende Schnittstellen implementierten:

```
public interface UserDB {
  public User getUser(String userId) throws NoSuchEntityException;
  public Benutzer getBenutzer(String benutzerName)
                  throws NoSuchEntityException;
  public List getUserListOrderedByName();
  public void addUser(User newUser)
              throws DuplicateEntityException;
  public void removeUser(String userId)
              throws NoSuchEntityException;
  public void updateUser(String userId, User userData)
              throws NoSuchEntityException, DuplicateEntityException;
  public List getUsersCreatedAfter(java.util.Date createDate);
}

public interface ContactDB {
  public List getEmailContact(User user) throws
              NoSuchEntityException;
  public List getFaxPhoneContact(User user) throws
              NoSuchEntityException;
  public List getMobileContact(User user) throws
              NoSuchEntityException;
  public List getAllContacts(User user) ) throws
              NoSuchEntityException;
  public List getContactsByState(boolean active);
  public void addContact(User user, Contact contact) throws
              NoSuchEntityException;
// type of Contacts: 0 → FaxPhone, 1 → Mobile, → 3 Email
  public Contact getContact(User user, String contactValue, int type)
                 throws NoSuchEntityException;
  public void removeContact(User user, String contactValue, int type)
              throws NoSuchEntityException;
```

Testen Sie Ihre Klasse ***ausführlich*** mit JUnit Tests, welches unter Verwendung obiger Interfaces User und Kontakte anlegt.

Aufgabe 3: Konfiguration 6 Punkte

Entwickeln sie eine Klasse **`DB`**, welche das Registrieren des JDBC-Treibers und das Anfordern einer Datenbankverbindung kapselt. Die Daten für den JDBC-Treiber sollen aus einem Property File geladen werden (siehe dazu: `java.util.Properties`) und diese Daten sollen beim Laden der Klasse durch den Klassenlader gelesen werden. Außerdem soll diese Klasse 2 Methoden zur Verfügung stellen, um eine `Connection` anzufordern und wieder freizugeben. Benutzen Sie diese Klasse in den Klassen für die User und Kontaktverwaltung. Vergessen Sie nicht bei einem Fehler die Datenbankverbindung wieder zu schliessen.
Für Debug-Zwecke entwickeln Sie noch die 2 Methoden **`getMetaDataDB()`** und **`getMetaDataTable(String tableName).`** Die Methode `getMetaDataDB()` soll zumindest den Namen und die Version der Datenbank, so wie den Namen und die Version des JDBC-Treibers ausgeben. Die Methode `getMetaDataTable(String tableName)` soll die Attributenamen einer Tabelle ausgeben.