

Übungsblatt 5

Aufgabe 1 Datenbank-Connect-ORACLE

Stellen Sie einen Datenbank-Connect zur Datenbank mit Hilfe des Java-Thin-Drivers her:

- a) Herstellen des Connects:
- Passen Sie die `connect.properties`-Datei, die der `ConnectionManager` nutzt, an Ihren Oracle-Account an (inkl. Passwort).
 - Der Oracle-Thin-Driver ist in `ojdbc14.jar` enthalten (zu finden im Verzeichnis `\\Samba\skripte\allgemeines\software\java\jdbc`). Nehmen Sie ihn in den CLASSPATH auf. (In Eclipse geht das unter `Project->ProjectProperties->Java BuildPath`)

```
#legt die Parameter für die Datenbankverbindung fest
#uid = Benutzername
#pwd = Passwort
#driver = Name der JDBC-Treiber-Klasse (inkl. Paketstruktur)

# von meinem Notebook, da in lokaler Domäne

uid = aheinze
pwd = password
driver = oracle.jdbc.driver.OracleDriver
#dburl = jdbc:oracle:thin:@localhost:1521:ofbi
dburl = jdbc:oracle:thin@wspoolserv.inform.fh-hannover.de:1521:ofbi
#dburl = jdbc:oracle:thin@wspoolserv.inform.fh-hannover.de:1522:ora9i
#für Access-Datenbank:
#driver = sun.jdbc.odbc.JdbcOdbcDriver
#dburl = jdbc:odbc:<Name der Datenquelle>
```

- b) Erzeugen Sie mit den Beispiel-Sourcen die Tabellen `COFFEES` und `SUPPLIERS`. Prüfen Sie in `SQLPlus`, ob die Tabellen erzeugt wurden.
- c) Fügen Sie Datensätze in die beiden Tabellen ein.
- d) Führen Sie ein `SELECT` auf die Tabelle `SUPPLIERS` durch und geben den Inhalt aus.

```
select * from SUPPLIERS;
```

Aufgabe 2 OID-Generator

Implementieren Sie eine Klasse `OidGenerator` mit einer statischen Methode `int getOid()`, die eine `Oid` (eindeutiger Objekt-Identifikator) zurückliefert. Dabei soll eine Oracle-Sequence verwendet werden, die man vor Programmausführung in Oracle erzeugen muss.

`OidGenerator.java`:

```
import java.sql.*;

public class OidGenerator {
    static {
    }
}
```

`Oid.java`:

```
import java.sql.*;
public class Oid {

    /* Sequenz erstellen (Oracle) */
    public static void init() {
        Connection con = ConnectionManager.newConnection();
        System.out.println("OidGenerator> Erfolgreich verbunden");

        Connection con = ConnectionManager.newConnection();
        System.out.println("OidGenerator> Erfolgreich verbunden");

        String createPersonSequence="CREATE SEQUENCE personSeq START WITH 1";

        try {
```

```

        Statement stmt = con.createStatement();
        stmt.executeUpdate(createPersonSequence);
        System.out.println("OidGenerator> Sequenz angelegt");
        stmt.close();
        con.close();
    } catch (SQLException ex) {
        System.err.println("SQLException: " + ex.getMessage());
    }
}

public static int getOid() {
    int retOid = 0;
    Connection con = DriverManager.getConnection();
    System.out.println("OidGenerator> Erfolgreich verbunden");
    String query = "SELECT personSeq.nextval FROM DUAL";

    try {
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        if (rs.next())
            retOid = rs.getInt(1);
        stmt.close();
        con.close();
    } catch (SQLException e) {
        System.err.println("SQLException: " + e.getMessage());
    }

    return retOid;
}
}

```

Aufgabe 3 Klasse Person

a) Schreiben Sie eine Klasse, die Ihnen eine DB-Tabelle „Person“ mit den Attributen aus dem UML-Diagramm in Oracle erzeugt.

b) Implementieren Sie eine Klasse Person mit den angegebenen Methoden.

1) Es soll drei Konstruktoren geben, die ein Person-Objekt erzeugen.

- Person(int oid, String name, String vorname, Date gTag) :

setzt alle Attribute

- Person(String name, String vorname, Date gTag) :

Hinweis: Oid soll mit einer entsprechender DB-Funktion gesetzt werden.

- Der 3. Konstruktor soll als Parameter eine Oid haben: Person(int oid)

Hinweis: das Person-Objekt soll aus der Datenbank erzeugt werden.

2) Schreiben Sie eine Methode insert(), die ein neues Objekt in die Datenbank einfügt.

3) Schreiben Sie eine Methode update(), die die Attribute eines bereits in der Datenbank vorhandenen Objekts ändert.

4) Schreiben Sie eine Methode allPersons(likeName), die alle Personen ausgibt, deren Namen mit einem als Parameter übergebenen String anfangen.

```

import java.sql.Connection;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Vector;

```

```

/*
 * Created on 21.04.2006
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */

```

```

/**
 * @author      ajaworsk aheinze
 *
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates
 */

```

```

public class Person {
    private int oid;

```

```

private String vorname;
private String name;
private Date gTag;

public Person(int oid, String name, String vorname, Date gTag) {
    this.oid = oid;
    this.name = name;
    this.vorname = vorname;
    this.gTag = gTag;
}

public Person(String name, String vorname, Date gTag) {
    createNewTable();
    this.oid=Oid.getOid();
    this.name = name;
    this.vorname = vorname;
    this.gTag = gTag;
    this.insert();
}

public Person(int oid){
    String query = "SELECT * FROM person WHERE oid = " + oid ;
    Connection con = null;
    Statement statement = null;
    ResultSet result = null;

    try {
        con = ConnectionManager.newConnection();
        statement = con.createStatement();
        result = statement.executeQuery(query);
        if(result.next()){
            this.oid = oid;
            this.name = result.getString(2);
            this.vorname = result.getString(3);
            this.gTag = result.getDate(4);
        }
    }catch (SQLException e) {
        e.getMessage();
        e.printStackTrace();
    }catch (Exception e){

    }

    finally{
        try {
            result.close();
            statement.close();
            con.close();
        }catch (Exception e) {
            e.getMessage();
            e.printStackTrace();
        }
    }
}

public void createNewTable() {
    Connection con = ConnectionManager.newConnection();

    String createString = "CREATE TABLE person (oid INTEGER, name VARCHAR(30), vorname VARCHAR(30),
gTag DATE) ";

    Statement stmt = null;

    try {
        stmt = con.createStatement();
        stmt.executeUpdate(createString);
        Oid.init();
        System.out.println("Tabelle angelegt");
    }catch (SQLException ex) {

```

```

        ex.printStackTrace();
    }finally {
        try {
            stmt.close();
            con.close();
            System.out.println("aufgeraeumt");
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public void insert() {
    String newInsert = "INSERT INTO person VALUES(" + this.oid + "," + this.name + "," + this.vorname + "" + ", {d '" + this.gTag + "'})";

    Statement stmt = null;
    Connection con = null;

    try {
        con = DriverManager.getConnection();
        stmt = con.createStatement();
        stmt.executeUpdate(newInsert);

        System.out.println("Datensaetze in Tabelle PERSON eingefuegt");

    }catch (SQLException ex) {
        ex.printStackTrace();
    }

    finally {
        try {
            stmt.close();
            con.close();
            System.out.println("aufgeraeumt");
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public void update() {
    String updateElement = "UPDATE person SET name = " + this.name +
        ", vorname = " + this.vorname +
        ", gTag = " + "{d '" + this.gTag + "'}" +
        " WHERE oid = " + this.oid;

    Statement stmt = null;
    Connection con = null;

    try {
        con = DriverManager.getConnection();
        stmt = con.createStatement();
        stmt.executeUpdate(updateElement);

        System.out.println("Datensaetze in Tabelle PERSON eingefuegt");

    }catch (SQLException ex) {
        ex.printStackTrace();
    }

    finally {
        try {
            stmt.close();
            con.close();
            System.out.println("aufgeraeumt");
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
}
}

```

```

public static Vector allPersons(String likeName){
    Connection con = null;
    Statement stmt = null;
    ResultSet result = null;
    Vector personen = null;

    try {
        con = DriverManager.getConnection();
        stmt = con.createStatement();
        result = stmt.executeQuery("SELECT * FROM Person WHERE Name = " + likeName + "");
        while(result.next()){
            personen.add(new Person(result.getInt(1),
                                     result.getString(2),
                                     result.getString(3),
                                     result.getDate(4)));
        }

    }catch (SQLException e) {
        e.getMessage();
        e.printStackTrace();
    }catch (Exception e){
    }
    finally{
        try {
            result.close();
            stmt.close();
            con.close();

        }catch (Exception e) {
            e.getMessage();
            e.printStackTrace();
        }
    }
    return personen;
}

public String toString(){
    String person = this.oid + "\t" + this.name + "\t" + this.vorname + "\t" + this.gTag.toString();
    return person;
}

public static Vector allPersons(){
    Connection con = null;
    Statement stmt = null;
    ResultSet result = null;
    Vector personen = new Vector();

    try {
        con = DriverManager.getConnection();
        stmt = con.createStatement();
        result = stmt.executeQuery("SELECT *" +
                                     " FROM Person");

        while(result.next()){
            personen.add(new Person(result.getInt(1),
                                     result.getString(2),
                                     result.getString(3),
                                     result.getDate(4)));
        }

    }catch (SQLException e) {
        e.getMessage();
        e.printStackTrace();
    }catch (Exception e){
    }
    finally{
        try {

```

```

        result.close();
        stmt.close();
        con.close();

    }catch (Exception e) {
        e.getMessage();
        e.printStackTrace();
    }
}
return personen;
}

public Date getGTag() {
    return gTag;
}

public void setGTag(Date tag) {
    gTag = tag;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public int getOid() {
    return oid;
}

public void setOid(int oid) {
    this.oid = oid;
}

public String getVorname() {
    return vorname;
}

public void setVorname(String vorname) {
    this.vorname = vorname;
}
}

```

Schreiben Sie ein Testprogramm, das sich alle Person-Objekte aus der Datenbank holt, ein Objekt ändert und wieder in die Datenbank zurückschreibt.

```

import java.sql.Date;
import java.util.Vector;

```

```

/*
 * Created on 25.04.2006
 *
 * TODO To change the template for this generated file go to
 * Window - Preferences - Java - Code Style - Code Templates
 */

```

```

/**
 * @author ajaworsk aheinze
 *
 * TODO To change the template for this generated type comment go to Window -
 * Preferences - Java - Code Style - Code Templates

```

```
*/  
  
public class Test {  
  
    public static void main(String []args){  
  
        Person hexe = new Person("Blocksberg","Bibi",Date.valueOf("1989-3-7"));  
        Person elefant = new Person("Bluemchen","Benjamin",Date.valueOf("1984-5-12"));  
        Person schlumpf = new Person("Gei","Papa",Date.valueOf("1971-6-3"));  
        Person regina = new Person("Gei", "Mama",Date.valueOf("1986-11-9"));  
  
        Vector personen = Person.allPersons();  
        for(int i=0; i<personen.size();i++){  
            System.out.println(((Person)personen.get(i)).toString());  
        }  
  
        ((Person)personen.get(3)).setVorname("Schlumpfine");  
        ((Person)personen.get(3)).update();  
  
    }  
  
}
```

