|  |
| --- |
| Rückwärtssalto  Erstellen eines EER-Diagramm und RM einer RDBMS Datenbank mittels JDBC  08.01.2014  4BHIT  Nenad Gligorevic, Melanie Göbel |

Inhaltsverzeichnis

[Aufgabenstellung 2](#_Toc406148992)

[Aufwand und Zeit 3](#_Toc406148993)

[Schätzung und Realität 3](#_Toc406148994)

[Designüberlegung 3](#_Toc406148995)

[Arbeitsdurchführung 3](#_Toc406148996)

[Testbericht 3](#_Toc406148997)

# Aufgabenstellung

Erstelle ein Java-Programm, dass Connection-Parameter und einen Datenbanknamen auf der Kommandozeile entgegennimmt und die Struktur der Datenbank als EER-Diagramm und Relationenmodell ausgibt (in Dateien geeigneten Formats, also z.B. PNG für das EER und TXT für das RM)

Verwende dazu u.A. das ResultSetMetaData-Interface, das Methoden zur Bestimmung von Metadaten zur Verfügung stellt.

Zum Zeichnen des EER-Diagramms kann eine beliebige Technik eingesetzt werden für die Java-Bibliotheken zur Verfügung stehen: Swing, HTML5, eine WebAPI, ... . Externe Programme dürfen nur soweit verwendet werden, als sich diese plattformunabhängig auf gleiche Weise ohne Aufwand (sowohl technisch als auch lizenzrechtlich!) einfach nutzen lassen. (also z.B. ein Visio-File generieren ist nicht ok, SVG ist ok, da für alle Plattformen geeignete Werkzeuge zur Verfügung stehen)

Recherchiere dafür im Internet nach geeigneten Werkzeugen.

Die Extraktion der Metadaten aus der DB muss mit Java und JDBC erfolgen.

Im EER müssen zumindest vorhanden sein:

* korrekte Syntax nach Chen, MinMax oder IDEFIX
* alle Tabellen der Datenbank als Entitäten
* alle Datenfelder der Tabellen als Attribute
* Primärschlüssel der Datenbanken entsprechend gekennzeichnet
* Beziehungen zwischen den Tabellen inklusive Kardinalitäten soweit durch Fremdschlüssel nachvollziehbar. Sind mehrere Interpretationen möglich, so ist nur ein (beliebiger) Fall umzusetzen: 1:n, 1:n schwach, 1:1
* Kardinalitäten

Fortgeschritten (auch einzelne Punkte davon für Bonuspunkte umsetzbar)

* Zusatzattribute wie UNIQUE oder NOT NULL werden beim Attributnamen dazugeschrieben, sofern diese nicht schon durch eine andere Darstellung ableitbar sind (1:1 resultiert ja in einem UNIQUE)
* optimierte Beziehungen z.B. zwei schwache Beziehungen zu einer m:n zusammenfassen (ev. mit Attributen)
* Erkennung von Sub/Supertyp-Beziehungen

# Aufwand und Zeit

Die Aufgabe scheint nicht sehr einfach zu sein aber machbar. Wichtig ist ein Tool zu erstellen von den EER-Diagrammen zu finden, die Verbindung mit der Datenbank via JDBC scheint einfacher zu sein

## Schätzung und Realität

|  |  |  |  |
| --- | --- | --- | --- |
| Arbeit | Geschätzte Zeit (in min) | Zuständig | Benötigte Zeit (in min) |
| Verbindung zur Datenbank | 80 | Melanie | 30 |
| Struktur zum Speichern der Daten | 100 | Nenad | 120 |
| Auslesen und Erstellen des RM | 120 | Melanie | 100 |
| Internetrecherche zu EER-Diagramm Tool | 50 | Melanie | 30 |
| Erstellen eines EER-Diagramm | 200 | Nenad | 220 |
| Beziehungen und Kardinalitäten im EERD | 160 | Nenad |  |
| Dokumentation | 100 | Melanie | 100 |
| Zusätzliche Arbeit (zB. Bug fixinig) | 0 | Melanie, Nenad | 60 |
| Insgesamt | 13 Stunden 30 Min |  |  |

## Arbeitsaufzeichung

Melanie

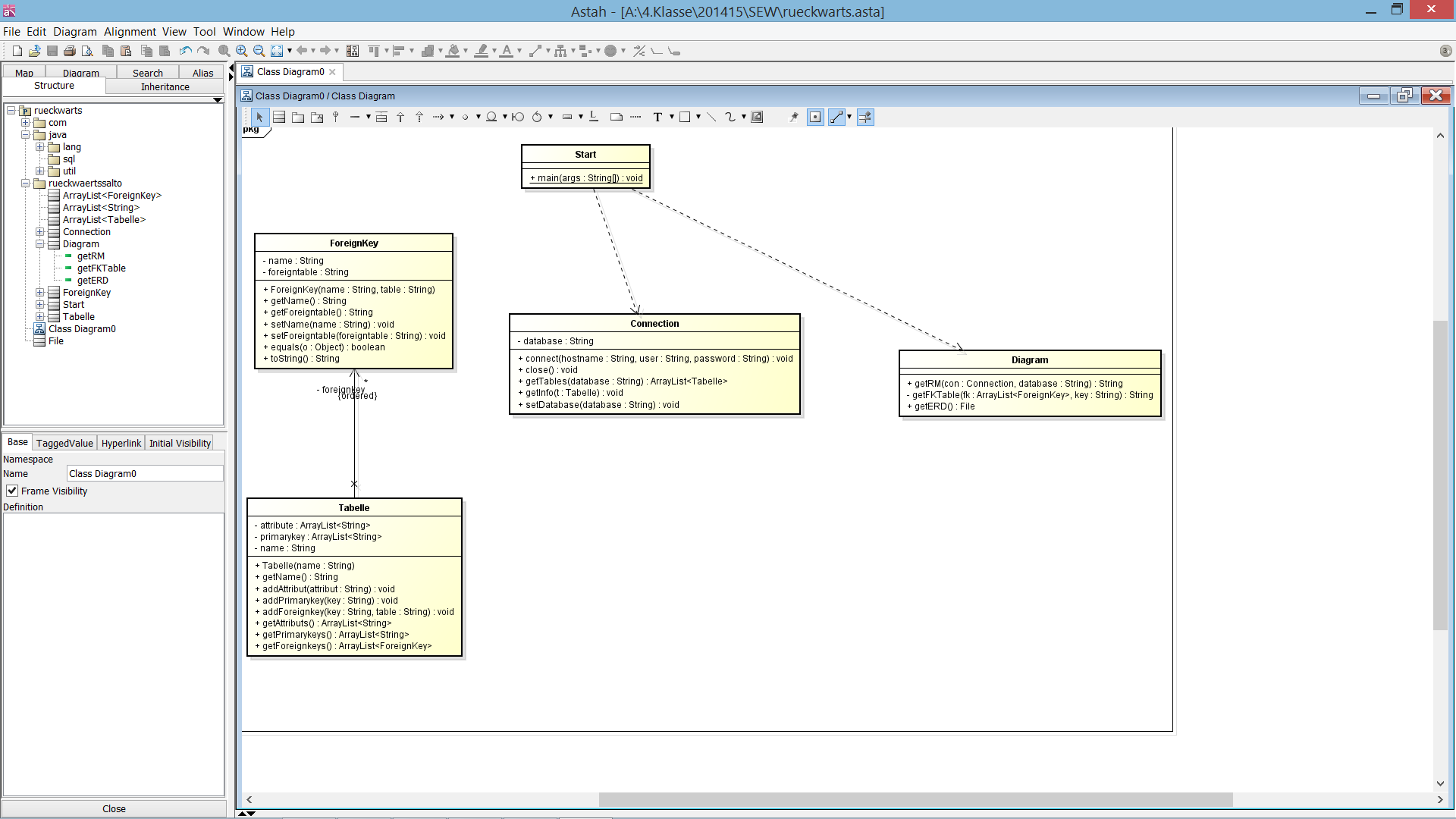
|  |  |  |
| --- | --- | --- |
| Arbeit | Datum, Uhrzeit | Benötigte Zeit in min |
| Verbindung zur Datenbank | 14.1.2015 9:50-11:30 | 100 |
| Auslesen und Erstellen des RM | 20.1.2015 17:00-18:00 | 60 |
| Designänderung (Zusätzliche Arbeit) | 30.1.2015 22:00 – 22:40 | 40 |
| Designänderung, RM | 06.1.2015 23:00-00:00 | 60 |
| ERD | 07.02.2015 13:00-14:20 | 80 |
| ERD | 10.02.2015 18:00-20:20 | 140 |

Nenad

|  |  |  |
| --- | --- | --- |
| Arbeit | Datum, Uhrzeit | Benötigte Zeit |
| Struktur zum Speichern der Daten | 14.1.2015 9:50-11:30 | 100 |
|  |  |  |

# Designüberlegung

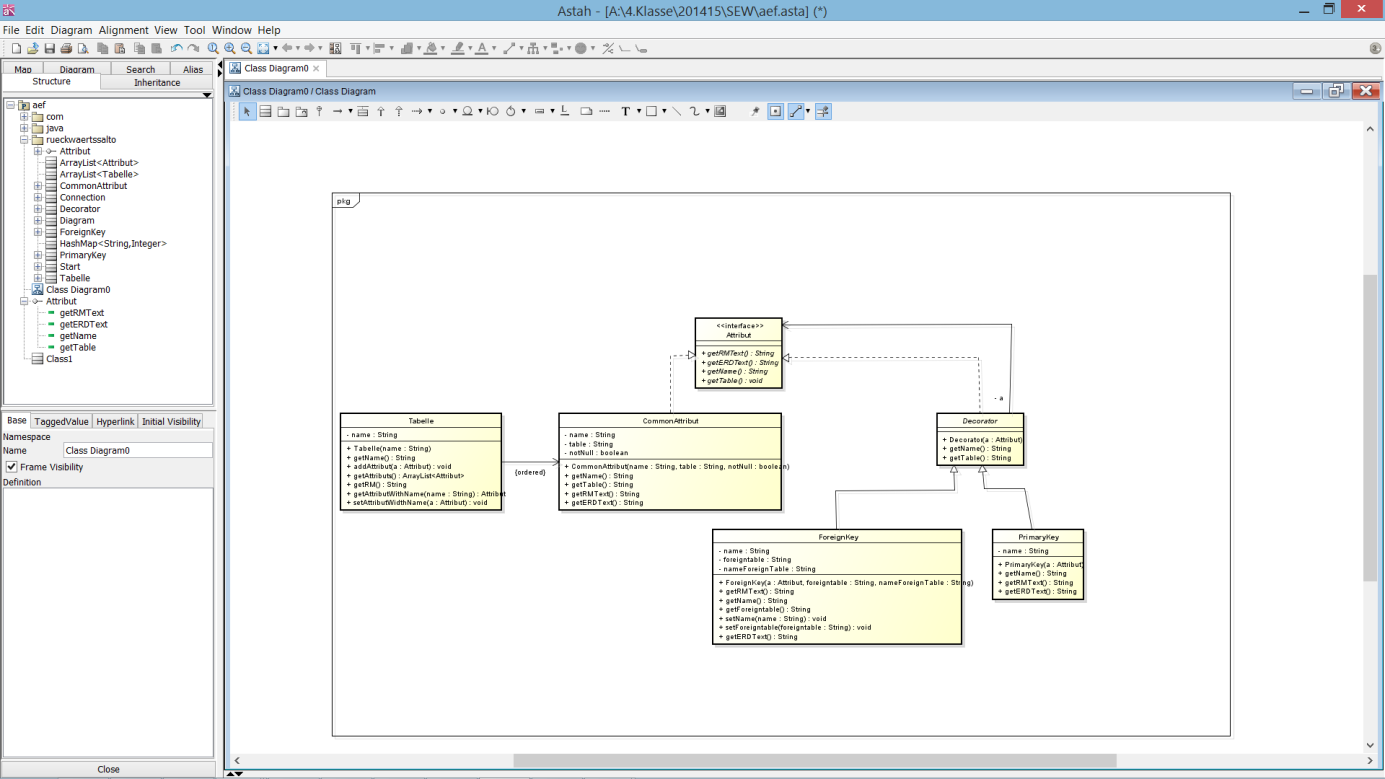
Erste Überlegung war, das Speichern der Information in die Klasse Tabelle auszulagern sowie die Verbindung zur Datenbank und auslesen der Datenbank in Connection. Diagram ist eine Klasse die sowohl ein Relationenmodel sowie ein Entity-Relationship-Diagram erstellen kann. In Start findet das Argumentparsing (Parameterüberprüfung) statt.



Probleme bei dem vorläufigen Design:

* + Änderungen wie zB zusätzliche Infos bei einen Attribut in einer Tabelle (Datentyp) ist schwer zu implementieren (Änderungen am Code)
  + Verwendung von mehreren Listen in Tabelle (kann zu doppelten Attributen führen)

Um Änderungen willkommen zu heißen, wird das Design-Pattern Decorator-Pattern angewendet. Dies hilft zur Abschaffung von mehreren Listen in Tabelle.



# 

# Arbeitsdurchführung

## Connection

Erstellen der Verbindung zur RDBMS mit JDBC.

ds = **new** MysqlDataSource();

ds.setServerName(hostname);

ds.setUser(user);

ds.setPassword(password);

**this**.con = ds.getConnection();

Auslesen der Daten: Tabelle und deren Information (Attribute, PrimaryKeys, ForeignKeys und Tabellen von denen die Foreignkeys sind)

st = con.createStatement();

ResultSet rs1 = st.executeQuery("use "+database+";");

ResultSet rs = st.executeQuery("show tables;");

**while**(rs.next()){

tables.add(**new** Tabelle(rs.getString(1)));

}

Auslesen der Attribute und PrimaryKeys mit Desc:

ResultSet rs = st.executeQuery("desc "+t.getName()+";");

**while**(rs.next()){

t.addAttribut(rs.getString(1));

**if**(rs.getString(4).equals("PRI"))

t.addPrimarykey(rs.getString(1));

}

Auslesen der ForeignKeys mit DatabaseMetaData:

DatabaseMetaData meta = con.getMetaData();

ResultSet rsK = meta.getImportedKeys(database, **null**, t.getName());

**while**(rsK.next()){ t.addForeignkey(rsK.getString("FKCOLUMN\_NAME"),rsK.getString("PKTABLE\_NAME"));

}

## Tabelle & ForeignKey

Speichern der Daten mit Klassen.

Bei ForeignKey überschreiben der equal Methode (**public** **boolean** equals(Object o) {}), damit man sie mit Wörtern vergleichen kann.

## Diagram

Das RM wird mit **public** String getRM(Connection con, String database){} erstellt.

# Testbericht

# Quellen