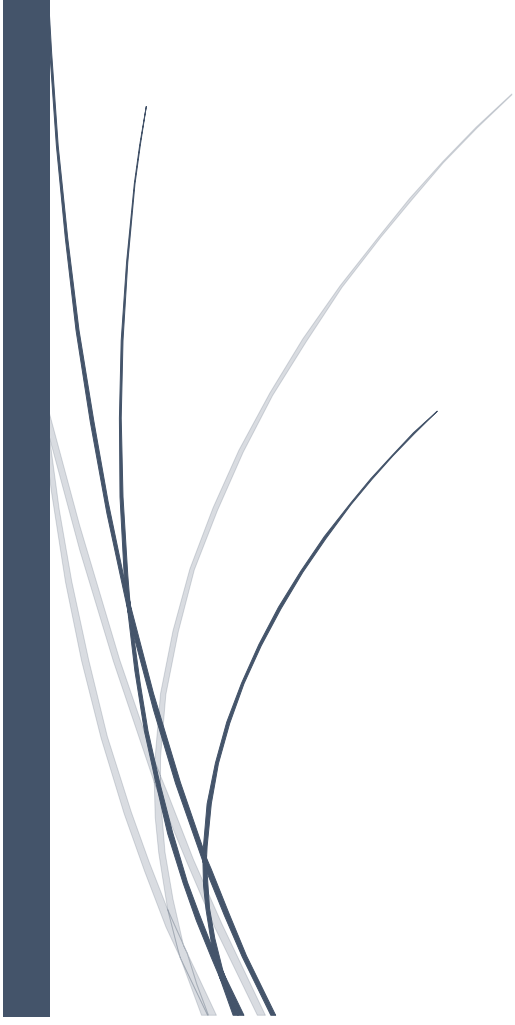
7.1.2015

Rückwärtssalto

A05 - Metadata



Hampl & Kritzl

Inhalt

1. Angabe

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

2. detaillierte Arbeitsaufteilung mit Aufwandsabschätzung

2.1 Funktionale Anforderungen

Arbeitspaket	Person	Schätzung(min)	Tatsächlich(min)	Erledigt	Testen
Usereingabe	**Hampl**	**0**	**0**	**x**	**x**
Parsen	Hampl	10	5	x	x
Überprüfen	Hampl	10	5	x	x
Hilfe	Hampl	20	20	x	x
Datenbankverbindung	**Kritzl**	**0**	**0**	**x**	
MySQL	Kritzl	30	60	x	
Auslesen Metadaten	**Kritzl**	**0**	**0**	**x**	**x**
Datenbank	Kritzl	60	30	x	x
Tabelle	Kritzl	60	60	x	x
Attribut	Kritzl	60	120	x	x
Referenz	Kritzl	90	120	x	x
Export in EER	**Hampl**	**0**	**0**	**x**	
Syntaktisch	Hampl	120	420	x	x
optisch	Hampl	120	10	x	
Export in RM	**Hampl**	**0**	**0**	**x**	**x**
Syntaktisch	Hampl	30	120	x	x
Gesamt		**610**	**970**	**x**	**x**

2.2 Nicht funktionale Anforderungen

Arbeitspaket	Person	Schätzung(min)	Tatsächlich(min)	Erledigt
Exception-handling	**Kritzl**	**40**	**20**	**x**
Build-Automation	**Hampl**	**45**	**120**	**x**
Testen	**Hampl/Kritzl**	**150**	**600**	**x**
Gesamt		**235**	**740**	**x**

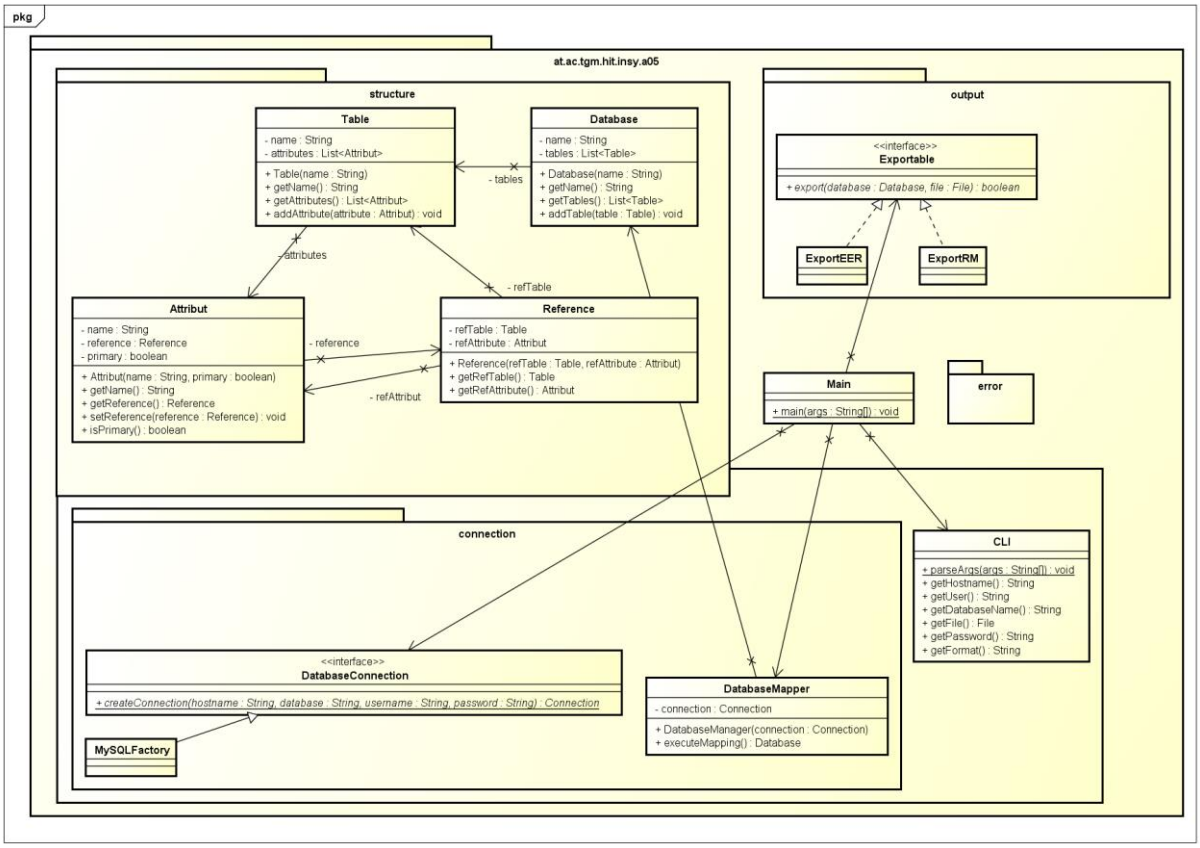
2.3 Organisatorische Anforderungen

Arbeitspaket	Person	Schätzung(min)	Tatsächlich(min)	Erledigt
Dokumentation	**Kritzl/Hampl**	**0**	**0**	**x**
JavaDoc	Kritzl/Hampl	40	20	x
Protokoll	Kritzl/Hampl	120	180	x
Gesamt		**160**	**200**	**x**

3. Designüberlegung

3.1 Abbildung

Das UML-Diagramm wurde mit dem Programm „Astah“ erstellt.



3.2 Überlegung

3.2.1 Structure

- Das Package `structure` stellt eine Datenbank und deren Inhalte in einer objektorientierten Form dar. Damit ist die Datenbank, deren Tabellen, deren Attribute und deren Eigenschaften wie Primary, Unique und Not Null genauso wie die Foreign Keys der Attribute gespeichert. Dadurch kann sehr einfach auf die einzelnen Eigenschaften zugegriffen werden und nicht immer umständlich direkt mit der Datenbank kommunizieren zu müssen.

3.2.2 Input

3.2.2.1 Source

- Das Package `source` ist für die Verbindung zu den verschiedenen Datenbanken zuständig. Dabei wurde eine Abstract-Factory angewendet, um einfach neue Datenbankverbindungen oder allgemein Verbindungen die eine `java.sql.Connection` haben, hinzufügen zu können.
- Die Methoden der Abstract-Factory wurden statisch implementiert da keine Informationen über die Verbindung zusätzlich gespeichert werden müssen.
- Die Klasse `DatabaseMapper` ist für die Umsetzung der Daten der Datenbank in die objektorientierte Form des Packages `source`.

3.2.2.2 CLI

- Ist für die Überprüfung der Parameter des Aufrufes zuständig.

3.2.3 Output

- Ist für die Ausgabe des bereits in objektorientierten Form der Datenbank zuständig. Hirbei wurde das Strategy-Pattern angewendet, um ohne weiteres neue Ausgabemöglichkeiten zu implementieren. Dabei gibt es momentan zwei verschiedene Varianten die Ausgabe der ermittelten Daten darzustellen:
    - In Form eines RM als `html`
    - In Form eines EER als `dot`
- Die Klasse `Export-Factory` gibt aufgrund des gewünschten Formats das richtige `Exportable` zurück.

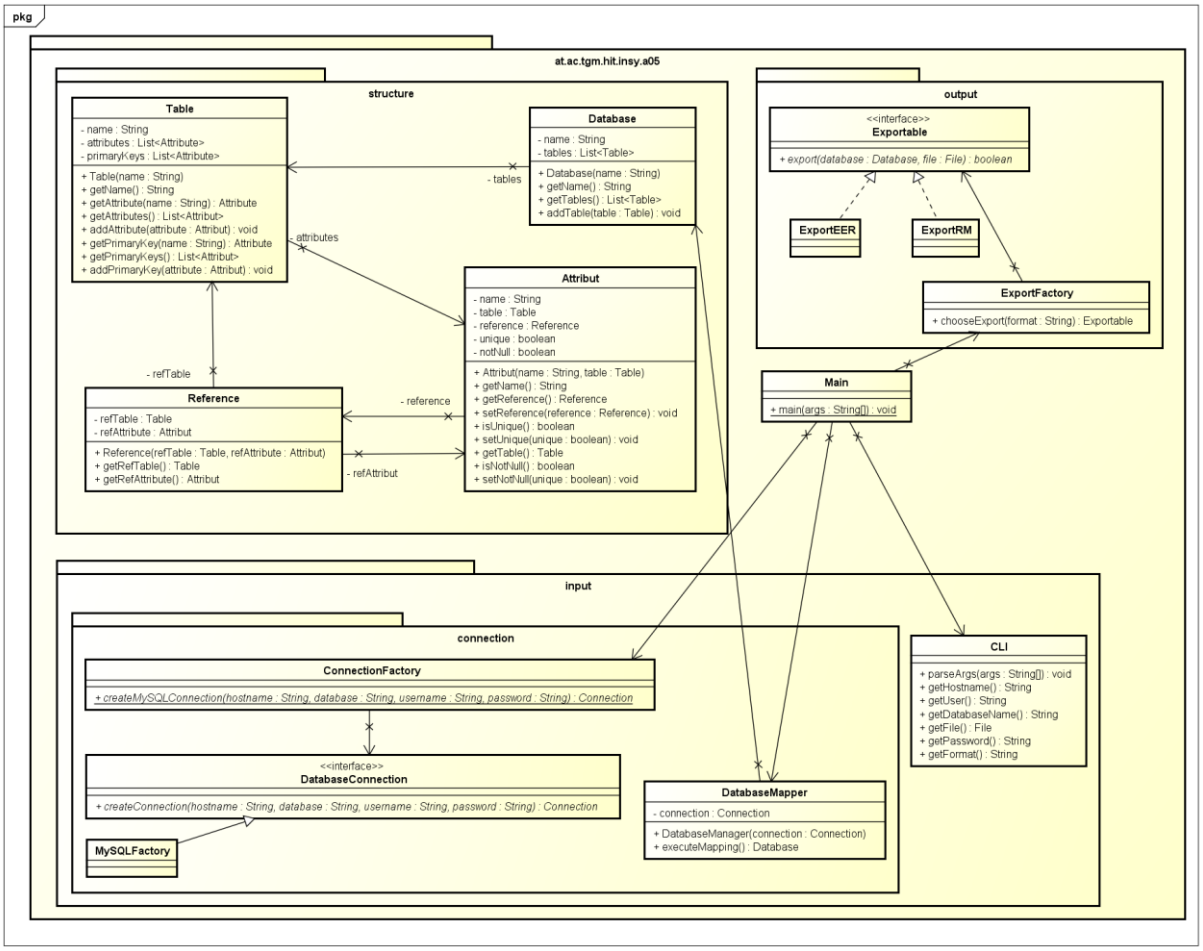
3.2.4 Main

- Ist Startpunkt der Applikation und betreibt das Exception-Handling um dies so spät wie möglich verwalten zu können.

4. Arbeitsdurchführung

4.1 Abbildung

Da wir während der Implementierung auf einige Verbesserungen bezüglich der Struktur gekommen sind, sieht unser finales UML-Diagramm folgendermaßen aus:



4.2 Änderungen zur Designüberlegung

- Dem Attribut wurden die Methoden zum Verwenden von „`Not Null`“ und „`unique`“ hinzugefügt.
- Die Festlegung der `PrimaryKeys` wurde in die Tabelle gehoben und durch eine weitere Liste verwirklicht.
- Dem Attribut wurde seine Tabelle hinzugefügt.
- Die Herstellung der Connection wurde durch die `ConnectionFactory` verbessert.
- Durch die `ExportFactory` wurde die Ermittlung des richtigen Formats verbessert und dadurch leichter erweiterbar zu machen

5. Lessons learned

- Der bessere Umgang mit einer jdbc-Verbindung.
- Die Einbindung eines externen Programms in das eigene.
- Die Verwendung von Design-Patterns selbst in der Implementierungszeit hilfreich ist.
- Die Testung mit Mock-Objekten ist sehr aufwendig, da jeder einzelne Schritt vorgegeben werden muss.
- Die die Verwendung des Build-Tools Gradle.
- Das Umsetzen von Daten der Datenbank in eine objektorientierte Form.
- Weitere Erfahrungen mit args4j für die Benutzereingaben-Verwaltung.
- Wiederholung der genauen Definitionen des EER.
- Testen von `System.exit()` mit System-Rules in den Test-Cases.

6. Quellenangaben

Gradle:
http://gradle.org/docs/current/userguide/userguide_single.html (Autor: Hans Dockter, Adam Murdoch, zuletzt abgerufen am 04.02.2015)

Graphviz:
http://www.graphviz.org/Gallery/undirected/ER.html (Autor: AT&T, zuletzt abgerufen am 04.02.2015

System-Rules:
http://stefanbirkner.github.io/system-rules/index.html (Autor: stefanbirkner zuletzt abgerufen am 04.02.2015)