Aufgabenblatt 4
JDBC

- Abgabetermin: **Dienstag, 25.06.2018, 23:59**
- Zur Prüfungszulassung muss ein Aufgabenblatt mit mind. 25% der Punkte bewertet werden und alle weiteren Aufgabenblätter mit mindestens 50% der Punkte.
- Die Aufgaben sollen in Zweiergruppen bearbeitet werden.
- Abgabesytem unter
  `https://www.dcl.hpi.uni-potsdam.de/submit/`

- **Die Abgabe besteht diesmal aus folgenden drei Elementen:**
  - PDF-Dateien *im A4-Format*
  - eine PDF-Datei **Aufgabe-1.pdf** (Lösung zu Aufgabe 1)
  - eine PDF-Datei **Aufgabe-2.pdf** (Ausgabe der zwei gefragten Keywords)
  - eine Java-Datei mit folgendem Dateinamen und entsprechendem Klassennamen:
    ImdbSuche.java

Aufgabe 1: Sichten

a) Erstelle auf Basis der IMDb-Relationen eine View mit den folgenden Informationen:

- die Namen aller Schauspieler und Schauspielerinnen,
- deren Geschlecht ('m'/'f'),
- deren Filme (movie_id) und
- deren Rolle

Das Schema soll also wie folgt aussehen: `ActorMW(name, sex, movie_id, role)`. Gib das Statement in deiner Abgabe an. **2 P**

b) Für welche beiden Anfragen aus Aufgaben 1 und 2 des letzten Aufgabenblatts (SQL) kann diese View *sinnvoll* verwendet werden? Gib die entsprechenden Anfragen an (Umformulieren ist nicht nötig). **2 P**

Aufgabe 2: Keyword-Suche für IMDb

Vorbemerkung:
Um euch mittels JDBC mit der Datenbank zu verbinden benötigt ihr einen User (inklusive Passwort), der die Berechtigung hat auf die relevanten Tabellen zuzugreifen. Diesen könnt ihr in der PSQL Shell mithilfe des folgenden Kommandos erzeugen:

CREATE ROLE <username> WITH LOGIN SUPERUSER PASSWORD '<password>';

In dieser Aufgabe bauen wir eine IMDb-Keyword-Suche. Es soll ein Java-Programm geschrieben werden, das ein Keyword entgegennimmt und alle Filme und Schauspieler(innen) ausgibt, in deren Namen dieses Keyword enthalten ist. Der Zugriff auf die Datenbank soll über JDBC erfolgen. Den JDBC Treiber für PostgreSQL findet ihr zum Download unter `https://jdbc.postgresql.org/download.html` oder unter `https://mvnrepository.com/artifact/org.postgresql/postgresql` als Maven-Dependency. Hinweise zum Bauen der URL findet ihr unter `https://jdbc.postgresql.org/documentation/80/connect.html`

Im Internet findet ihr diverse Beispielprogramme, welche JDBC verwenden, z.B. unter `https://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm`

Hier zwei Beispiele, die die gewünschte Ausgabe des Programms darstellen:

Beispiel 1: Keyword = `Krug`

```
java -jar ImdbSuche.jar -d imdb -s 127.0.0.1 -p 5432 -u alice
   -pw secret -k Krug
```

Erwartete Ausgabe:

```
MOVIES

ACTORS
Krug, Manfred
        PLAYED IN
                Tatort - Leiche im Keller
                Tatort - Lockvögel
                Tatort - Rattenlinie
        CO-STARS
                Brauer, Charles (3)
                Hart, Kurt (2)
                Hoppe, Edgar (2)
                Martens, Dirk (I) (2)
                Becker, Rolf (1)
```

Beispiel 2: Keyword = `Blacks`

```
java -jar ImdbSuche.jar -d imdb -s 127.0.0.1 -p 5432 -u alice
   -pw secret -k Blacks
```

Erwartete Ausgabe:

```
MOVIES
Blacksmith, The, 1922, Comedy, Short
        Fox, Virginia
        Keaton, Buster
        Roberts, Joe (I)

ACTORS
Blackshaw, Anthony
        PLAYED IN
                Fiction Makers, The
        CO-STARS
```

```
Armitage, Graham (1)
Ashmore, Peter (1)
Ball, Ralph (1)
Boyd, Roy (1)
Clegg, Tom (1)
```

20 P

Programmablauf

Der Benutzer soll das Programm mit folgenden Parametern starten können:

- -d <Datenbankname>
- -s <IP-Adresse des Datenbank-Servers>
- -p <Port der Datenbank-Instanz>
- -u <Username>
- -pw <Passwort>
- -k <Keyword>

Das Programm stellt mittels JDBC eine Verbindung zur Datenbank her (**3 P**) und ermittelt alle notwendingen Informationen, um folgende Ausgabe zu ermöglichen:

a) Die Ausgabe beginnt mit dem Teil `MOVIES`. Hier sollen alle Filme gelistet werden, in denen das Keyword im Titel vorkommt. Die Ausgabe ist aufsteigend sortiert nach dem Titel. **1 P**

b) Für jeden Film wird ausgegeben:

   1) Der Titel, das Jahr, und alle Genres des Films (jeweils durch Komma getrennt). **2 P**

   2) Außerdem umfasst die Ausgabe für jeden Film die beteiligten Schauspieler(innen), sortiert nach dem Namen. Es werden jedoch nur die ersten fünf ausgegeben. **3 P**

c) Im zweiten Teil `ACTORS` werden alle Schauspieler(innen) ausgeben, in deren Namen das gegebene Keyword vorkommt. Das Ergebnis soll dabei aufsteigend nach den gefundenen Namen sortiert werden. **2 P**

d) Für jede(n) Schauspieler(in) wird ausgegeben:

   1) Alle Filme, in denen der Schauspieler mitgewirkt hat (`PLAYED IN`). **3 P**

   2) Die Schauspieler(innen), die in denselben Filmen mitgewirkt haben (`CO-STARS`). Hinter jedem/-r Schauspieler(in) steht in Klammern die Anzahl der Filme, in denen beide mitgespielt haben. Es sollen die Top 5 ausgegeben werden, sortiert nach Anzahl gemeinsamer Filme, dann nach Name. **6 P**

Weitere Hinweise:

- Alle Ausgaben sollen grundsätzlich nur einmal erfolgen (z.B. sollen keine Filme doppelt ausgegeben werden).
- Der Name eines Schauspielers kennzeichnet ihn eindeutig.
- Die Nutzung von `PreparedStatement`s wird empfohlen. Vergleicht testweise die Performance mit normalen `Statement`s für häufig aufgerufene Anfragen.

Abgabeform

- Es soll *eine* Java-Klasse abgegeben werden.
- Es soll darüberhinaus als PDF die Ausgabe des Programms für die folgenden beiden Anfragen abgegeben werden:
  - Keyword = `Prior`
  - Keyword = `Schwarz`