

**Aufgabe 10 (1 Punkt)**

Geben seien die Menge  $Mat = \{42, 43, 44\}$  als Teilmenge von Integer und die Menge  $Studi = \{„Edna“, „Urs“\}$  als Teilmenge von String.

- Geben Sie die Potenzmenge von Mat und Studi an.
- Geben Sie das Kreuzprodukt  $Mat \times Studi$  und das Kreuzprodukt  $Studi \times Mat$  an.
- Definieren Sie was eine Multimenge ist und grenzen Sie den Begriff gegen den Mengen-Begriff ab.
- Gegeben sei die Tabelle T auf der rechten Seite, deren Zeilen jeweils ein Element aus der Menge  $Integer \times String \times Integer$  zeigen. Handelt es sich bei T um eine Menge oder eine Multimenge, warum?

Mat	Studi	Note
42	Edna	100
43	Urs	230
42	Edna	370
43	Urs	230

**Aufgabe 11 (3 Punkte)**

**Tabelle Vorfuehrung**

Kinoname	Ort	Fassung	Saal	Größe	Tag	Uhrzeit	Film	FSK	Laenge
Gloria	OS	300	S1	100	22.4.19	17:00	Die Wut	6	90
Gloria	OS	300	S2	80	22.4.19	20:00	Die Wut	6	90
Gloria	OS	300	S1	100	22.4.19	20:00	Das Glück	18	120
Gloria	OS	300	S1	100	23.4.19	17:00	Die Wut	6	90
Apollo	OS	200	S1	100	22.4.19	17:00	Die Wut	6	90
Apollo	OS	200	S2	50	22.4.19	22:00	T-Rex	6	100
Apollo	OS	200	S3	50	22.4.19	20:00	Das Glück	18	120
Gloria	HB	500	S1	200	22.4.19	17:00	Die Wut	6	90
Gloria	HB	500	S1	200	22.4.19	20:15	Die Wut	6	90
Gloria	HB	500	SX	40	22.4.19	20:00	Lisas Axt	18	60

- Ein Kino wird durch seinen Namen und seinen Ort eindeutig und hat ein genaues Fassung(svermögen)  
 $\{Kinoname, Ort\} \rightarrow \{Fassung\}$
  - Jeder Saal kommt pro Kino nur einmal vor und hat eine genaue Größe. (Der Begriff Kino wurde in 1) definiert)  
 $\{Kinoname, Ort, Saal\} \rightarrow \{Größe\}$
  - In jedem Saal kann zu einem bestimmten Tag und einer bestimmten Uhrzeit nur ein Film gezeigt werden.  
 $\{Kinoname, Ort, Saal, Tag, Uhrzeit\} \rightarrow \{Film\}$
  - Jeder Film hat einen eindeutigen Titel und hat eine FSK-Angabe sowie eine Länge.  
 $\{Film\} \rightarrow \{FSK, Laenge\}$
- Warum kann nicht  $\{Kinoname\} \rightarrow \{Film\}$  bei den gegebenen Beispieldaten gelten?
  - Nennen Sie alle Schlüsselkandidaten (in diesem Fall nur einer).
  - Nennen Sie die Menge der Schlüsselattribute und der Nichtschlüsselattribute.
  - Bringen Sie die Tabelle mit dem Standardverfahren in Tabellen in zweiter Normalform. Vermeiden Sie für die zweite Normalform überflüssige Umformungen.
  - Bringen Sie die Tabellen aus d) mit dem Standardverfahren in Tabellen in dritter Normalform. Vermeiden Sie für die dritte Normalform überflüssige Umformungen.

### Aufgabe 12 (4 Punkte)

- a) Aus der Vorlesung ist bekannt, dass aus Beispieldaten weder funktionale Abhängigkeiten noch Schlüsselkandidaten ablesbar sind. Es ist für Beispieldaten aber prüfbar, ob überhaupt ein solcher Zusammenhang vorliegen könnte, genauer: „So lange die Beispieldaten kein Gegenbeispiel liefern, könnte es gelten.“. Dieser Ansatz ist zu programmieren.

Laden Sie dazu das Eclipse-Projekt `dbAufgabePruefungFunktionaleAbhaengigkeit` von der Webseite der Veranstaltung und importieren Sie es in Ihren Workspace. Das Projekt enthält eine Klasse `Tabelle`, die eine Tabelle mit ihrem Namen, den Spaltenüberschriften (auch Attribute genannt) und dem Inhalt in einer Liste von Zeilen enthält. Jede Zeile ist dabei als Array mit Inhalten vom Typ `Object` modelliert. In den folgenden Beispielen werden alle Inhalte vereinfacht als Strings betrachtet. Die für die Aufgaben relevanten Methoden der Klasse `Tabelle` sind.

Type	Method and Description
<code>int</code>	<code>anzahlZeilen()</code> Gibt die Anzahl der vorhandenen Zeilen zurück.
<code>String[]</code>	<code>getSpaltenueberschriften()</code> Gibt als Array alle Spaltenüberschriften zurück.
<code>Object</code>	<code>wertInZeileVon(int zeile, String spalte)</code> Gibt das Objekt der Zeile <code>zeile</code> aus der Spalte mit der Überschrift <code>spalte</code> zurück.

Schreiben Sie eine Realisierung des Interfaces `tabelle.FunktionaleAbhaengigkeitPruefer`. Dabei sind folgende Methoden umzusetzen.

Gegeben sei die Tabelle `tabelle` mit ihren aktuellen Daten, es wird geprüft, ob eine funktionale Abhängigkeit zwischen den in „von“ genannten Attributen (=Spaltenüberschriften) und den in „abhaengig“ genannten Attributen bei den aktuellen Tabelleninhalten geben kann, also  $\{von\} \rightarrow \{abhaengig\}$  gelten kann.

```
boolean kannFunktionalAbhaengigSein(Tabelle tabelle  
    ,String[] von, String[] abhaengig)
```

Gegeben sei die Tabelle `tabelle` mit ihren aktuellen Daten, es wird geprüft, ob eine volle funktionale Abhängigkeit zwischen den in „von“ genannten Attributen (=Spaltenüberschriften) und den in „abhaengig“ genannten Attributen bei den aktuellen Tabelleninhalten geben kann, also  $\{von\} \rightarrow \{abhaengig\}$  voll gelten kann. Bei einer vollen funktionalen Abhängigkeit muss die Menge `von` minimal sein, es darf kein Attribut entfernt werden können und weiterhin eine funktionale Abhängigkeit gelten. Man beachte, dass sich anders als bei `kannFunktionalAbhaengigSein(.)`, das Ergebnis durch neue Tabelleninhalte auch von `false` auf `true` ändern kann.

```
boolean kannVollFunktionalAbhaengigSein(Tabelle tabelle  
    ,String[] von, String[] abhaengig)
```

Gegeben sei die Tabelle `tabelle` mit ihren aktuellen Daten, es wird geprüft, ob es sich bei der Attributmenge „kann“ bei den aktuell in der Tabelle befindlichen Daten um einen Schlüssel dieser Tabelle handeln kann.

```
boolean kannSchluesselSein(Tabelle tabelle, String[] kann)
```

Gegeben sei die Tabelle `tabelle` mit ihren aktuellen Daten, es wird geprüft, ob es sich bei der Attributmenge „kann“ um einen Schlüsselkandidaten dieser Tabelle handeln kann. Bei einem Schlüsselkandidaten muss die Menge von Attributen minimal sein, es darf kein Attribut entfernt werden können und es sich weiterhin um einen Schlüssel handeln. Man beachte, dass sich anders als bei `kannSchluesselSein(.)`, das Ergebnis durch weitere Inhalte auch von `false` auf `true` ändern kann. Es ist damit auch möglich, dass echte Schlüsselkandidaten auf Basis der Datenlage nicht als solche erkannt werden.

```
boolean kannSchluesselkandidatSein(Tabelle tabelle, String[] kann)
```

Zum Experimentieren enthält das Projekt einen Nutzungsdialog. Als Beispieltabellen stehen die Tabellen `Vorfuehrung.tabelle` aus der vorherigen Aufgabe und `Projektmitarbeit.tabelle` aus der Vorlesung, leicht ergänzt, zur Verfügung.

Nutzen Sie zum Testen des Programms die Klasse `SystemTest` (Run As -> JUnit Test), am Ende sollen alle Tests laufen.

Beachten Sie, dass Ihre Realisierung einen parameterlosen Konstruktor haben muss.

Im folgenden Beispieldialog sind Eingaben umrandet.

Was wollen Sie?

- (0) Programm beenden
- (1) Tabelle laden

1

aus welcher Datei? Vorfuehrung.tabelle

Tabelle Vorfuehrung geladen.

Was wollen Sie?

- (0) Programm beenden
- (1) Tabelle laden
- (2) Tabelle zeigen
- (3) funktionale Abhaengigkeit pruefen
- (4) voll funktionale Abhaengigkeit pruefen
- (5) Schluessel pruefen
- (6) Schluesselkandidaten pruefen:

2

#####

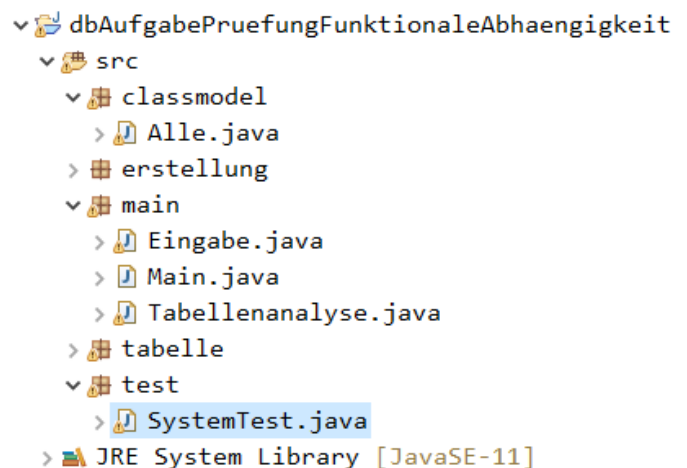
# Vorfuehrung #

#####

Kinoname	Ort	Fassung	Saal	Groesse	Tag	Uhrzeit	Film	FSK	Laenge
Gloria..	OS.	300....	S1..	100....	22.4.19	17:00..	Die Wut...	6..	90....
Gloria..	OS.	300....	S2..	80.....	22.4.19	20:00..	Die Wut...	6..	90....
Gloria..	OS.	300....	S1..	100....	22.4.19	20:00..	Das Glueck	18.	120...
Gloria..	OS.	300....	S1..	100....	23.4.19	17:00..	Die Wut...	6..	90....
Apollo..	OS.	200....	S1..	100....	22.4.19	17:00..	Die Wut...	6..	90....
Apollo..	OS.	200....	S2..	50.....	22.4.19	22:00..	T-Rex.....	6..	100...
Apollo..	OS.	200....	S3..	50.....	22.4.19	20:00..	Das Glueck	18.	120...
Gloria..	HB.	500....	S1..	200....	22.4.19	17:00..	Die Wut...	6..	90....
Gloria..	HB.	500....	S1..	200....	22.4.19	20:15..	Die Wut...	6..	90....
Gloria..	HB.	500....	SX..	40.....	22.4.19	20:00..	Lisas Axt.	18.	60....

Was wollen Sie?

- (0) Programm beenden
- (1) Tabelle laden
- (2) Tabelle zeigen
- (3) funktionale Abhaengigkeit pruefen



- (4) voll funktionale Abhaengigkeit pruefen
- (5) Schluessel pruefen
- (6) Schluesselkandidaten pruefen:

Geben Sie mit Leerzeichen getrennt die Nummern der Spalten der definierenden Attribute ein (z. B.: 1 3 )

1:Kinoname 2:Ort 3:Fassung 4:Saal 5:Groesse 6:Tag 7:Uhrzeit 8:Film 9:FSK  
10:Laenge

Geben Sie mit Leerzeichen getrennt die Nummern der Spalten der potenziell abhaengigen Attribute ein (z. B.: 2 4 )

1:Kinoname 2:Ort 3:Fassung 4:Saal 5:Groesse 6:Tag 7:Uhrzeit 8:Film 9:FSK  
10:Laenge

Pruefe ob moeglich: [Kinoname, Ort] --> [Fassung]  
Ergebnis: true

Was wollen Sie?

- (0) Programm beenden
- (1) Tabelle laden
- (2) Tabelle zeigen
- (3) funktionale Abhaengigkeit pruefen
- (4) voll funktionale Abhaengigkeit pruefen
- (5) Schluessel pruefen
- (6) Schluesselkandidaten pruefen:

Geben Sie mit Leerzeichen getrennt die Nummern der Spalten des potenziellen Schluesselkandidaten ein (z. B.: 1 3 )

1:Kinoname 2:Ort 3:Fassung 4:Saal 5:Groesse 6:Tag 7:Uhrzeit 8:Film 9:FSK  
10:Laenge

Pruefe ob moeglich: [Kinoname, Saal, Uhrzeit, Film] ist Schluesselkandidat von Vorfuehrung

Ergebnis: false

- b) Ihr Programm kann nur generell Hinweise geben, die sich auf den aktuellen Datenbestand beziehen. Welche Ergebnisse liefert Ihr Programm bei einer Tabelle mit mehreren Spalten, bei der nur eine Zeile eingetragen ist bei den verschiedenen Funktionen? Als Beispiel kann Analyse.tabelle genutzt werden.

Anmerkung: natürlich dürfen Sie statt „false“ auch das konkrete Gegenbeispiel ausgeben.

Hinweis: soll String[] von in List verwandelt werden, die veränderbar (remove) ist:

```
List<String> tmp = new ArrayList<>(Arrays.asList(von));
```

Arrays.asList(von) wandelt einen Array in eine nicht modifizierbare Liste um.