Klassediagrammer (III)

Tabeller og qualified associations

originally by Michael R. Hansen
modified/extended by Anne E. Haxthausen

Informatics and Mathematical Modelling
Technical University of Denmark

Oversigt

- Tabelbegrebet
- Modellering med qualified associations og tabeller
- Implementering af tabeller (Java Maps, HashMaps)

Tabelbegrebet

En *tabel* fra en mængde A til en mængde B, er

- en endelig delmængde A_1 af A og
- en funktion $t\colon A_1 \to B$

En tabel vises ofte på tabular form:

a_1	b_1
a_2	b_2
$\vdots$	$\vdots$
a_n	b_n

- $A_1 = \{a_1, \ldots, a_n\}$
- $t(a_i) = b_i$, for $1 \leq i \leq n$

- A_1 kaldes t's *domæne*
- $a_i \in A_1$ kaldes en *nøgle*
- $b_i = t(a_i)$ kaldes en *værdi*
- (a_i, b_i) kaldes en *indgang*

Anvendelse af Tabeller

For mange anvendelser findes naturlige nøgler:

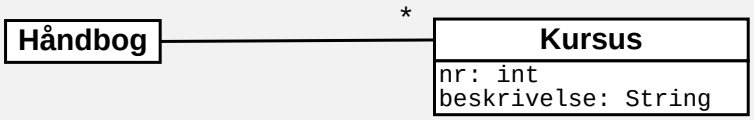
- Studienummer
- Kursusnummer
- CPR nummer
- Varekoder
- Medlemsnumre

En kursushåndbog kan, for eksempel, modelleres som en tabel fra kursusnumre til kursusbeskrivelser.

- Tabeller er nyttige som modelleringsredskab.
- Tabeller understøttes af gode programbiblioteker.

Studiehåndbog: Modellering (I)

En studiehåndbog beskriver en samling kurser, hvor hvert kursus beskrives ved et nummer, skemaplacering, formål, forudsætningskurser, osv.



- ikke synligt at et kursus er entydigt bestemt ved et kursusnummer

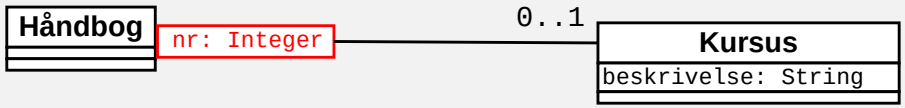
utilfredsstillende modellering

Studiehåndbog: Modellering (II)

En studiehåndbog beskriver en samling kurser, hvor hvert kursus er entydigt bestemt ved et nummer, og . . . .

Modelleres vha en *kvalificeret* (eng. *qualified*) association

- en håndbog har 0-1 kursus for hvert nummer



Den kvalificerede association kan realiseres ved at `Håndbog` kan indeholde en tabel, hvor

- numre er nøgler, og
- kurser er værdier

Java: Tabel-Interfacet `Map<K, V>`

`int size()` : $|A_1|$

`boolean containsKey(K k)` : $k \in A_1$

`V get(K k)` : $t(k)$, hvis $k \in A_1$, og `null` ellers

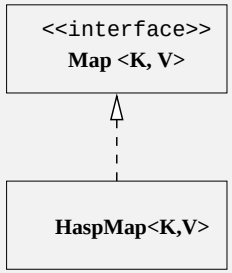
`V put(K k, V v)` :

$$A_1' = A_1 \cup \{k\}, \quad t'(k_1) = \begin{cases} v & \text{hvis } k = k_1 \\ t(k_1) & \text{ellers} \end{cases}$$

returnerer $t(k)$, hvis $k \in A_1$, og ellers `null`

hvor $t : A \to B$ og $A_1 \subseteq A$ er tabellen før operationen
og $t' : A \to B$ og $A_1' \subseteq A$ er tabellen efter, hvis der er sidevirkning.

Java: Tabel Implementering `HashMap<K, V>`



I klassebiblioteket for Java findes der klasser, der implementer `Map<K,V>` interfacet: bl.a. klassen `HashMap<K,V>`, der som datastruktur bruger såkaldte *hashtabeller*.

I algoritmer og datastrukturer vil I lære om hashtabeller.

Eksempel:

```
Map<Integer,Kursus> katalog = new HashMap<Integer,Kursus>();

katalog.put(2161, new Kursus(...));
```

Klasse: Haandbog

- operationer på håndbogen implementeres direkte ved HashMap-operationer

```
import java.util.*;
public class Haandbog {

    private Map<Integer,Kursus> katalog;

    public Haandbog(){ katalog = new HashMap<Integer,Kursus>();}

    public void opret(int nr, Kursus k){ katalog.put(nr, k); }

    public Kursus find(int nr){ return katalog.get(nr); }

    public String toString(){   return "" + katalog; }
}
```


Klasse: Kurser

```
public class Kursus
{
    private String navn;
    private String indhold;

    public Kursus(String n, String i)
    {
        navn = n;
        indhold = i;
    }

    public String toString()
    {
        return "Navn: " + navn + "\n" + indhold + "\n";
    }
}
```


Hovedprogram

```
Kursus k3333 = new Kursus("Programmering", "Java, C++, UML");
Kursus k4444 = new Kursus("Matematik", "Funktioner, relationer");
Kursus k5555 = new Kursus("Fysik", "Newtons love");
Kursus k6666 = new Kursus("Kemi", "Gasser");

Haandbog hb = new Haandbog();

hb.opret(3333,k3333);
hb.opret(4444,k4444);
hb.opret(5555,k5555);
hb.opret(6666,k6666);

System.out.println( hb );

System.out.println( hb.find(4444));
```


Udskrift

```
{3333=Navn: Programmering
Java, C++, UML
, 4444=Navn: Matematik
Funktioner, relationer
, 6666=Navn: Kemi
Gasser
, 5555=Navn: Fysik
Newtons love
}
Navn: Matematik
Funktioner, relationer
```

Opsummering

- Modellering vha tabeller
- Implementering vha Maps
- mange operationer i en model kan ofte direkte implementeres ved operationer i tabelbiblioteker
- der findes mange effektive implementeringer af tabeller

kurser i Algoritmer og Datastrukturer