# Data & Information Example test 1

Test 1 (2 May 2014) will contain questions that are comparable with the ones in this example test.
Solutions are given in Appendix 4.
Grade = #points/10

*Please note: you may not bring electronic devices, books, notes, etc. to the test. Relevant resources are included in the test as an appendix.*

## 1 – class diagram (40 points)

Make a class diagram based on the information below.
As the case is situated in the Netherlands, the case descrption is given in Dutch.
*Hint: Please note that some characteristics of a module are invariable (i.e. remain the same each time a module is taught) while others may differ across years. What would be a good way to model that?*

Invoering van het Twents Onderwijsmodel (TOM) heeft tot gevolg dat de gegevens over docenten, vakken, en toetsen op een andere manier moeten worden opgeslagen.

De Universiteit Twente biedt verschillende bachelorprogramma's aan. Ieder programma heeft een naam en een (unieke) afkorting. Bijvoorbeeld de bachelor Civiele Techniek wordt afgekort tot "CiT"
Een module is een onderwijseenheid van 15 EC die in één kwartiel wordt aangeboden. Een module wordt gekarakteriseerd door een uniekevakcode (9 cijfers) en een naam. (*N.B. een vakcode "201300180" betekent dat de module in het academisch jaar 2013/14 voor het eerst gegeven is. Als de module herhaald wordt in 2014/15, 2015/16, enz., blijft de vakcode 201300180.)*
Een programma bestaat uit een aantal verplichte modules die verder aangevuld kunnen worden met keuzemodules. Per programma is bekend welke modules verplicht zijn. (*Welke keuzemodules bij een programma mogelijk zijn, en onder welke voorwaarden, moet nog nader worden uitgezocht en wordt voorlopig niet opgeslagen.*) Het is mogelijk dat een module verplicht is voor meerdere programma's. Maar er zijn ook (keuze)modules die voor geen enkel programma verplicht zijn.
Een module wordt minstens één keer per studiejaar aangeboden. Het is mogelijk (maar komt niet vaak voor) dat een module ook twee keer wordt aangeboden, bijvoorbeeld als deze voor meerdere programma's verplicht is.
Uiteraard is bekend in welk kwartiel (evt. kwartielen) een module in ieder studiejaar wordt aangeboden.
De uitvoering van een module is de verantwoordelijkheid van een coördinator (één van de docenten, zie hieronder). De coördinator kan van jaar tot jaar verschillen
Bij de uitvoering van een module zijn naast de coördinator mogelijk ook nog andere docenten betrokken. Van een docent is bekend: medewerkersnummer, voornaam, achternaam, e-mailadres, telefoonnummer, en kantoor.

Soms zijn ook *externe* docenten bij een module betrokken. Dat zijn personen die niet aan de UT werkzaam zijn, maar bijv. aan een andere universiteit of in een bedrijf. Van externe docenten is bekend: docentnummer (unieke identificatie, nodig om toegang tot Blackboard te krijgen), voornaam, achternaam, e-mailadres, telefoonnummer, bedrijf, en bijzonderheden (gegevens van verschillende aard kunnen hier als tekst worden ingevuld). De coördinator van een module is nooit een externe docent, altijd een interne docent.

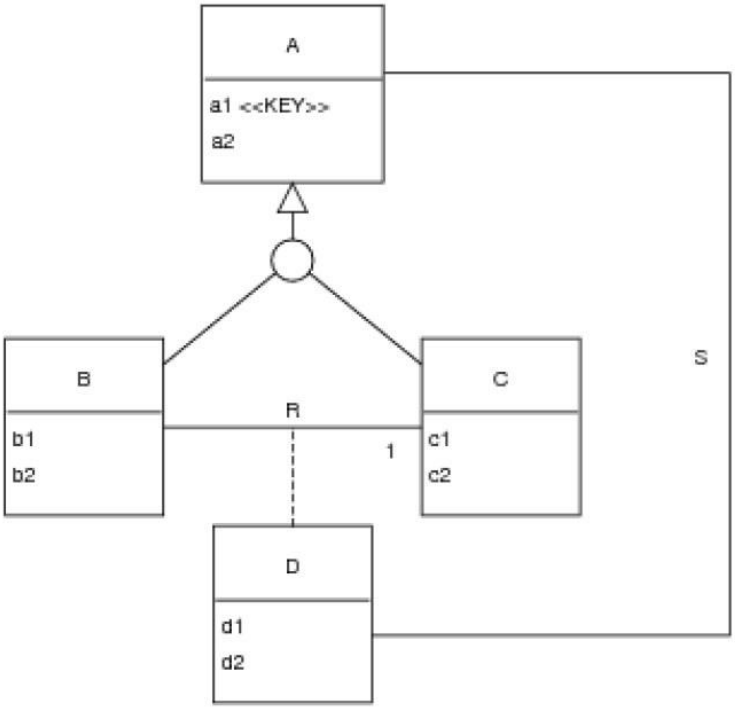
*Gegevens van studenten en cijfers voor toetsen en modules worden natuurlijk ook in het bovenstaande systeem opgenomen. Deze zijn hier weggelaten om de opgave beperkt te houden*

## 2 – Quality requirements (30 points)

- State three quality characteristics that you consider most important for a system which stores course (module) information, teacher and student information, as well as student's grades, ordered from 1 to 3. See Appendix 2 for an overview of quality characteristics. (Choose from the 31 detailed characteristics, not the 8 top categories)
- For each quality characteristic, explain why you consider it to be a top priority
- Formulate a (possible) requirement for each of the three characteristics mentioned

## 3 – databaseschema (30 points)

Consider the following UML class diagram.
Where no multiplicities are given, this should be interpreted as "*" (zero or more).
The generalization is not covering and not disjoint.




Translate the diagram to a database schema that stores exactly the information in the diagram and satisfies the following characteristics:

- the translation does not require the use of NULLs
- the transtion does not introduce redundancy

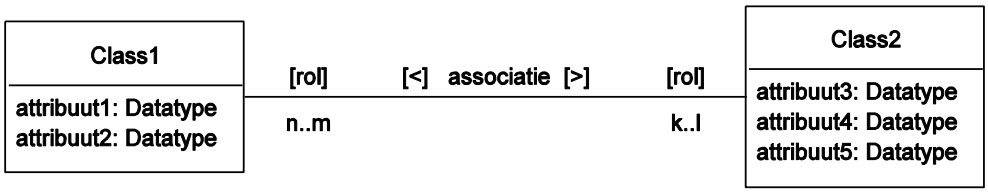
- all attributes have atomic values and also, taking into account the above requirements
- there are as few tables as possible.

Give the tables in SQL syntax. Assume that the domain of all attributes is integer. An (incomplete) definition of the SQL92 syntax for a database schema is given in Appendix 3.

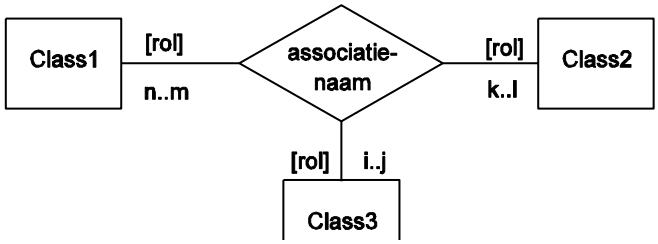
# Appendix 1: Notations for class diagrams

*meta-notation:*
[ ... ] Optional (can be deleted)
.. | .. Choice: one of the given alternatives

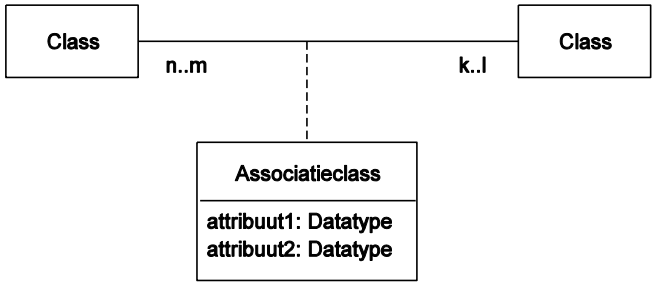
## Class and Association



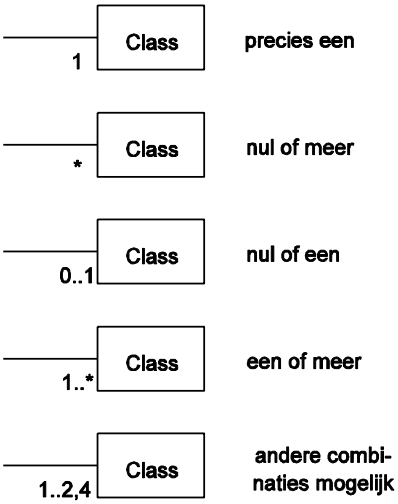

## Ternary (or *n*-ary) association



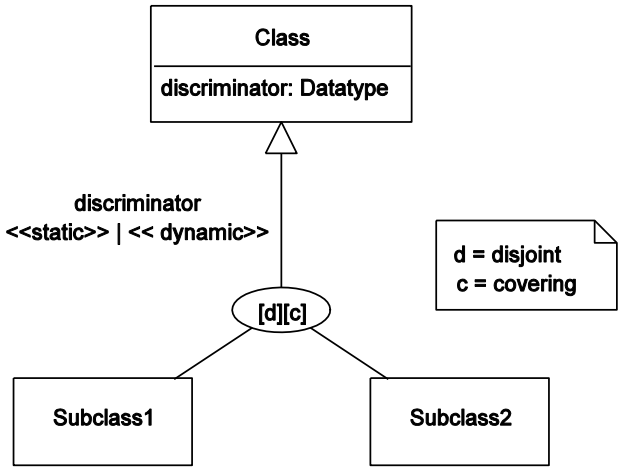

## Association class



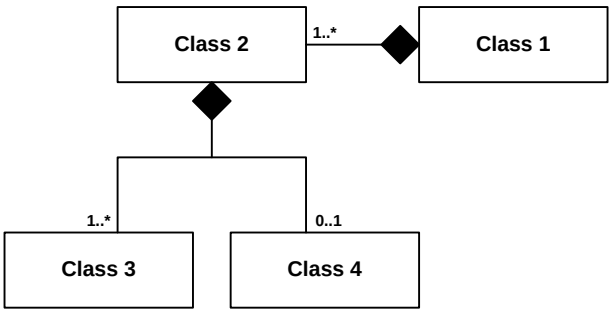

## Multiplicity




## Generalization




## Composition



## Appendix 2: Quality characteristics (ISO/IEC 25010:2011)

| | |
|---|---|
| **Functional suitability**<br>Functional completeness<br>Functional correctness<br>Functional appropriateness | **Reliability**<br>Maturity<br>Availability<br>Fault tolerance<br>Recoverability |
| **Performance efficiency**<br>Time behavior<br>Resource utilization<br>Capacity | **Security**<br>Confidentiality<br>Integrity<br>Non-repudiation<br>Accountability<br>Authenticity |
| **Compatibility**<br>Co-existence<br>Interoperability | **Maintainability**<br>Modularity<br>Reusability<br>Analysability<br>Modifyability<br>Testability |
| **Usability**<br>Appropriateness recognizability<br>Learnability<br>Operability<br>User error protection<br>User interface aestethics<br>Accessability | **Portability**<br>Adaptability<br>Installability<br>Replaceability |

## Appendix 3: SQL92 syntax for database schema (incomplete)

SQL92 informal syntax ('|' for choice and '[ ]' for optional):

```
CREATE TABLE table (
   column type [NULL|NOT NULL] [UNIQUE] [DEFAULT value] [PRIMARY KEY]
   [, column type ... ]
   [, PRIMARY KEY ( column, ... ) ]
   [, CHECK ( condition ) ]
   [, FOREIGN KEY (column, ...) REFERENCES table(column, ...) ]
);

Examples of type:
   CHAR(n) | VARCHAR(n) | INT | DATE | TIME | BOOLEAN | BLOB |
   ...

Examples of condition:
   column = value [ (OR | AND) [NOT] column <> value ] |
   column IS [NOT] NULL |
   column [NOT] IN (value, ...) |
   ...
```

# Appendix 4: Solutions

## 1 – Class diagram

See assignment 1 of the lab session.

## 2 – Quality requirements

Security is obviously very important. The answers must include something like

> Data integrity, because it is of vital importance that the grades cannot by modified by unauthorized persons.
> Possible requirement: The system shall be protected against unauthorized access
> (note: there are many different (partial) solutions to decrease the risk of unauthorized access, e.g.: write access requires authentication by means of a secure device of from a specific location; specific measures to prevent hacking, …)

Other requirements can depend on people's personal preferences. Obvious candidates are:

> Confidentiality, because grades are privacy-sensitive information
> Possible requirement: Grades shall only be accessible for authorized staff and the student him/herself
> Recoverability, because it is important that information in the system is complete and correct
> Possible requirement: Grade information shall not be lost or corrupted due to system crashes

There could other answers that most people would not think of – but if the motivation is sound, the anwer is OK. For example:

> Modifyability, because the way TOM is defined seems to be somewhat volatile. If the CvB decides to change the rules (perhaps following a demand from the UR), that should not result in an administrative nightmare.
> Possible requirement: rules about composition of modules and computation of results should be modifiable a posteriori

## 3 – Database schema

```
CREATE TABLE A(
  a1 INT,
  a2 INT,
  PRIMARY KEY (a1)
);

CREATE TABLE B(
  Aa1 INT,
  b1 INT,
  b2 INT,
  Ca1 INT NOT NULL,
  d1 INT,
  d2 INT,  -- om R (= D) te representeren
  PRIMARY KEY (Aa1),
  FOREIGN KEY (Aa1) REFERENCES A(a1),
  FOREIGN KEY (Ca1) REFERENCES C(Aa1)
);

CREATE TABLE C(
  Aa1 INT,
  c1 INT,
  c2 INT,
```

```
  PRIMARY KEY (Aa1),
  FOREIGN KEY (Aa1) REFERENCES A(a1)
);

CREATE TABLE S(
  Aa1 INT,
  Ba1 INT,
  Ca1 INT,
  PRIMARY KEY (Aa1, Ba1, Ca1),
  FOREIGN KEY (Aa1) REFERENCES A(a1),
  FOREIGN KEY (Ba1, Ca1) REFERENCES B(Aa1, Ca1)
);
```

Toelichting (NB de toelichting is niet vereist op het tentamen):

Afwezigheid van de 'not null' constraint wordt niet fout gerekend. De eigenschap not null bij precies iedere foreign key is nodig om te voorkomen dat er volgens het database schema andere informatie opgeslagen kan worden dan volgens het ER-diagram. (Ieder attribuut in een primary key is per definitie al not null.)

Merk op dat R gelijk is aan D (desgewenst met minder attributen). De 1 multipliciteit bij R is gemodelleerd door R in B te representeren. Relatie R (=D) kan als volgt uit B verkregen worden:

```
select Aa1, Ca1, d1, d2 from B; (met attributen)
select Aa1, Ca1 from B;  (zonder attributen)
```

Tenslotte: het is toegestaan om bij het tentamen een kortere schrijfwijze te gebruiken, d.w.z.:

- “CREATE TABLE” en “INT” weg te laten
- “PRIMARY KEY”, “FOREIGN KEY” en “REFERENCES” af te korten als “PK”, “FK”, en “REF”.