A.U.B. Iedere opgave op een apart vel maken.

201300180 Data & Information
Test 1 – 2 May 2014, 15:45-17:30

Please note: During the test you may not use electronic devices, books, notes, etc.
Relevant resources have been included in the attached sheet with appendices.

As the case is situated in the Netherlands, the case description is given in Dutch.

Betaalsystemen voor consumpties bij evenementen

Bij popfestivals en andere evenementen wordt vaak gebruik gemaakt van plastic muntjes – ook wel *tokens* genoemd – voor het betalen van eten en drinken. Als je een broodje of een drankje wilt kopen kan dat voor een klein aantal plastic muntjes. Deze muntjes kan je vooraf kopen bij een automaat of een loket. Overgebleven muntjes kan je ook weer inwisselen. Vaak zijn er meerdere aanbieders van versnaperingen, maar bij het gehele evenement worden dezelfde muntjes als betaalmiddel gebruikt.

Er zijn verschillende redenen waarom organisatoren van festivals hiervoor kiezen. Het lijkt onhandig voor de festivalbezoeker, maar het betalen van kleine consumpties wordt er wel makkelijker door. Veel omzet wordt gemaakt in de pauzes tussen de optredens, dan is er opeens veel vraag naar consumpties, en dan komt het goed uit dat iedereen gepast betaalt en er niet naar kleingeld hoeft te worden gezocht of wisselgeld terug hoeft te worden gegeven.

De verkoper kan daardoor sneller verkopen. Verkopers zijn meestal (werknemers van) zelfstandige ondernemers die bij een evenement het recht pachten om daar eten en drinken te mogen verkopen. Een voordeel voor de organisator van het evenement is dat hij met dit systeem de omzet van de verkopers kan controleren. De verkoper krijgt uiteraard geld voor de tokens die hij heeft ingenomen. Maar omdat hij zijn tokens bij (een vertegenwoordiger van) de organisatie in moet wisselen is precies bekend hoeveel hij omgezet heeft. Het contract legt meestal vast dat een bepaald percentage van de omzet aan de organisator moet worden afgedragen.

Degene die de tokens beheert maakt ook een zekere winst doordat niet alle tokens worden ingeleverd. Het komt voor dat bezoekers nog met tokens van een vorig evenement betalen, maar dat is altijd minder dan wat de bezoekers aan tokens verliezen en overhouden.

De firma LOC7000 is in Nederland een grote speler op deze markt. Bij veel festivals en evenementen regels LOC7000 het hele systeem van munten: verkoop aan de bezoekers, afrekening met de verkopers bij het festival, en het maken van een eindafrekening voor de organisator. *(Daarnaast verleent LOC7000 nog andere diensten, maar dat laten we hier buiten beschouwing.)*

De directie van LOC7000 ziet in de krant steeds vaker verhalen staan over nieuwe elektronische betaalmethoden en vraagt zich af of, en in welke vorm, ze daar zelf ook op in moeten zetten. De angst bestaat dat de plastic tokens wel eens hun langste tijd gehad zouden kunnen hebben.

Daarom is aan een afstudeerder van de Universiteit Twente gevraagd een onderzoek te doen naar mogelijkheden voor elektronisch betalen bij festivals.

Na enig participerend onderzoek is het de afstudeerder duidelijk geworden dat betalen op festivals wel iets anders is dan betalen in winkels. Het is vaak buiten, soms in de modder en in de stromende regen. Op piekmomenten willen veel mensen tegelijk iets kopen, terwijl een deel van die mensen al een behoorlijke hoeveelheid bier geconsumeerd heeft. Dit stelt eisen aan de verkopers en aan de systemen.

1 – class diagram (40 points)

Make a class diagram for the administration system that LOC7000 uses for their current (non-electronic) festival payment systems, according to case description above and the following information. For your convenience, the UML class diagram notations have been summarized in Appendix 1.

Voor de administratie van LOC7000 zijn de volgende gegevens van belang:

- Een evenement wordt altijd georganiseerd door een organisator. Van een organisator zijn bekend: firmanaam; adres; contactpersoon; telefoonnummer op kantoor; mobiel telefoonnummer; bankrekeningnummer.
- Van een evenement is de naam en de locatie bekend. Hetzelfde evenement (altijd met dezelfde organisator) kan meerdere keren plaatsvinden. Bijvoorbeeld de Zwarte Cross in Lichtenvoorde vindt jaarlijks plaats. Per keer dat het evenement plaatsvindt is de begintijd en eindtijd[1] bekend, evenals de prijs voor een token. (Omdat consumpties in gehele tokens worden afgerekend vinden prijsverhogingen meestal plaats door tokens duurder te maken.)
- Bij een evenement kunnen verschillende verkopers betrokken zijn (en als het evenement vaker plaatsvindt zijn het niet altijd dezelfde). De organisator kan zelf ook als verkoper optreden. Van een verkoper is bekend: firmanaam; adres; contactpersoon; mobiel telefoonnummer; bankrekeningnummer. Van iedere verkoper is (na afloop) bekend welke omzet hij op het betreffende evenement gerealiseerd heeft.
- LOC7000 richt zelf de verkooppunten voor tokens in. Dat kunnen automaten of (bemande) loketten zijn. Men wil later een analyse kunnen maken op welke plekken er veel en weinig tokens verkocht zijn. Daarom wordt bij ieder verkooppunt bij ieder festival de precieze locatie (GPS-coördinaten) bijgehouden, evenals het tijdstip dat het verkooppunt is neergezet en weggehaald. (Bij automaten is dit belangrijk; die gaan soms direct van het ene naar het andere evenement, daarom moet bij verkoop van tokens altijd duidelijk zijn voor welk evenement dat was).
- Automaten worden gekenmerkt door een serienummer. Ook is de datum bekend waarop deze automaat door LOC7000 in gebruik is genomen. Verder is er een tekstveld "bijzonderheden", waar bijv. storingen en reparaties in vermeld kunnen worden. Per automaat wordt bij iedere transactie (verkoop van tokens) bijgehouden: het tijdstip en het bedrag.
- Soms worden ook bemande loketten (kleine verplaatsbare huisjes) gebruikt voor tokenverkoop. Een loket wordt geïdentificeerd door een nummer. Men kan daar tegen contant geld tokens kopen. Uiteraard kan de verkoop van tokens niet per transactie worden bijgehouden. Wel worden er verschillende kasperiodes bijgehouden. Als de bezetting van een loket wisselt dan wordt in de regel ook de kas gewisseld. Per kasperiode is bekend: begintijd, eindtijd, beginsaldo en eindsaldo van de kas, evenals het aantal tokens aan het begin en het eind. Meestal is er een kasverschil (verlies of extra winst voor LOC7000).

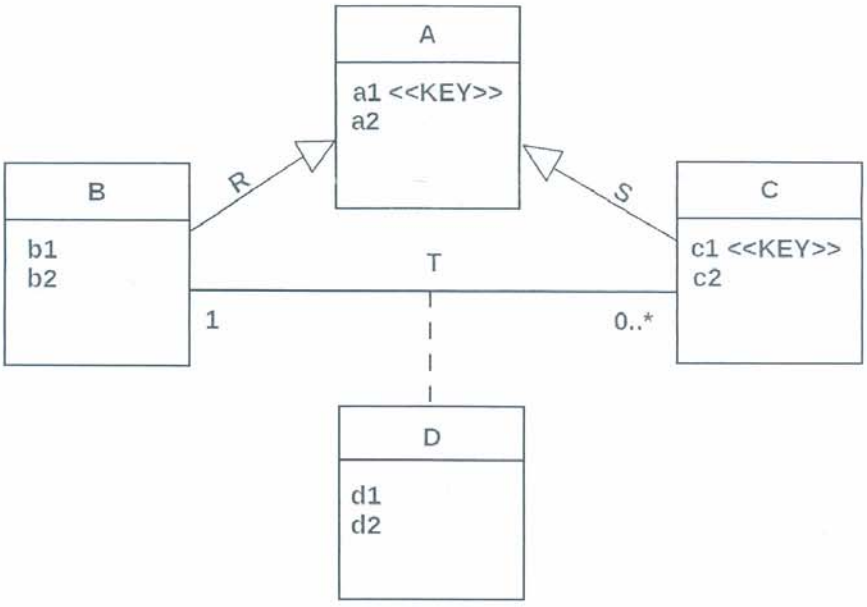
2 – Quality requirements (30 points)

- State three quality characteristics that you consider most important for an electronic payment system that could replace the plastic token system. 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.

[1] in feite begintijd en -datum en eindtijd en -datum. Voor het gemak gaan we ervan uit dat ook de datum is opgenomen in het opgeslagen tijdstip.

3 – Database schema (30 points)

Consider the following UML class diagram.



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 translation does not introduce redundancy
- all attributes have atomic values

and also, taking into account the above requirements

- there are as few tables as possible.

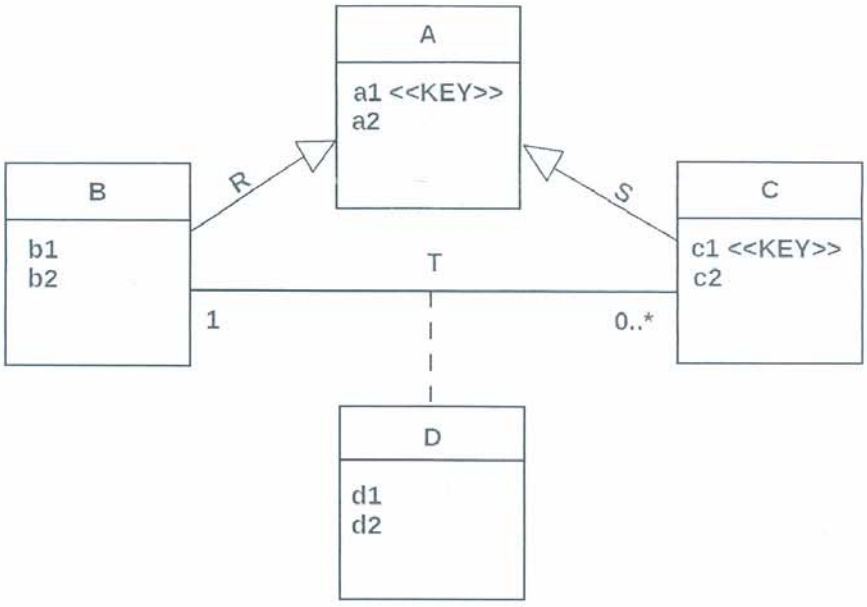
Give the tables in SQL syntax. The statement "CREATE TABLE" and the domain of attributes may be omitted. It is allowed to use abbreviations "PK" for "Primary Key" and "FK" for "Foreign Key". Use CHECKs if necessary.

Consider the following UML class diagram.
Where no multiplicities are given, this should be interpreted as "*" (zero or more).

Grade = #points/10

3 – Database schema (30 points)

Consider the following UML class diagram.



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 translation 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. The statement "CREATE TABLE" and the domain of attributes may be omitted. It is allowed to use abbreviations "PK" for "Primary Key" and "FK" for "Foreign Key". Use CHECKs if necessary.

Consider the following UML class diagram.
Where no multiplicities are given, this should be interpreted as "*" (zero or more).

Grade = #points/10

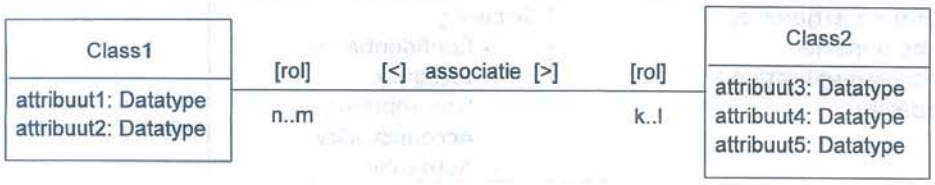
Appendix 1: Notations for class diagrams

meta-notation:

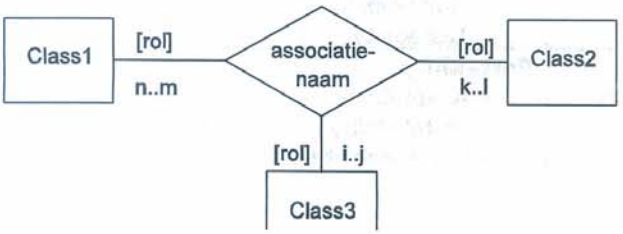
[...] Optional (can be deleted)

.. | .. Choice: one of the given alternatives

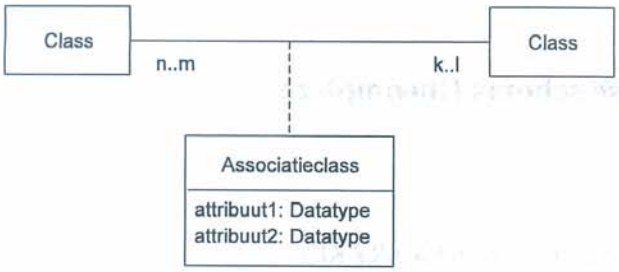
Class and Association



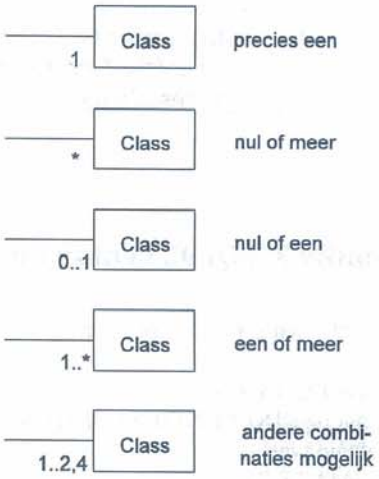
Ternary (or *n*-ary) association



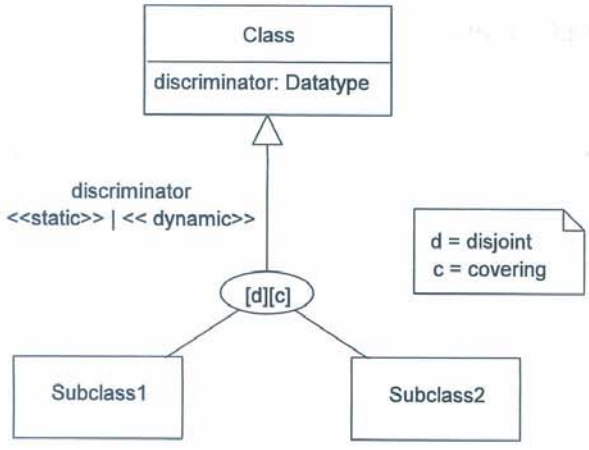
Association class



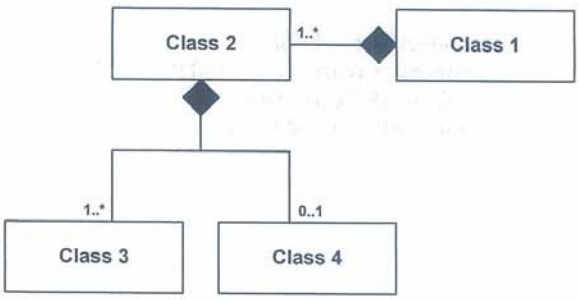
Multiplicity



Generalization



Composition



This page is intentionally left blank

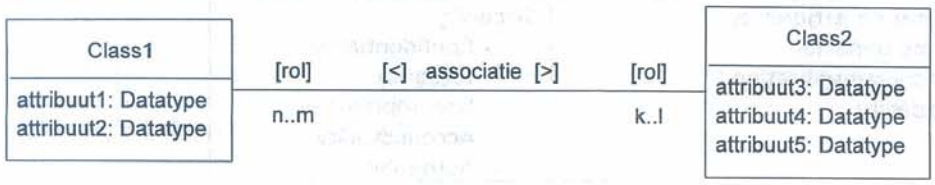
Appendix 1: Notations for class diagrams

meta-notation:

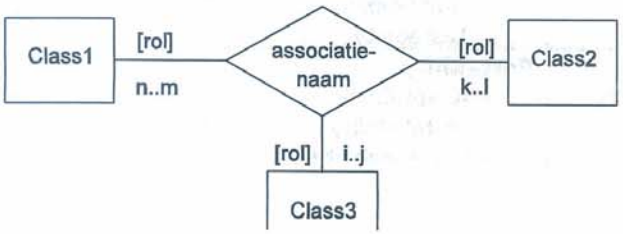
[...] Optional (can be deleted)

.. | .. Choice: one of the given alternatives

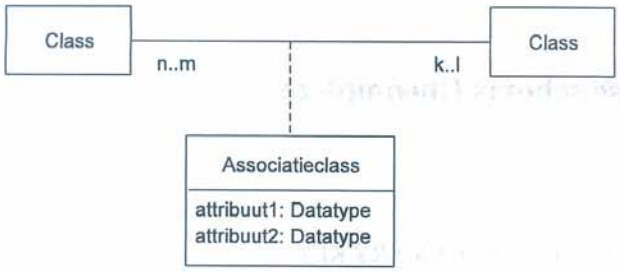
Class and Association



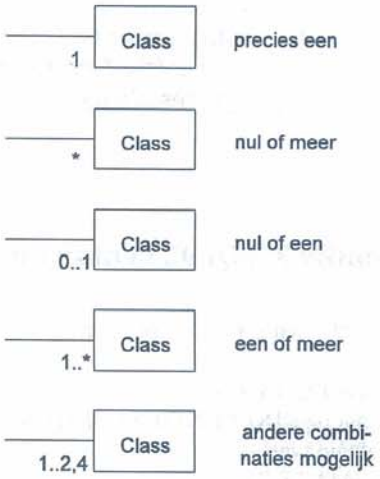
Ternary (or *n*-ary) association



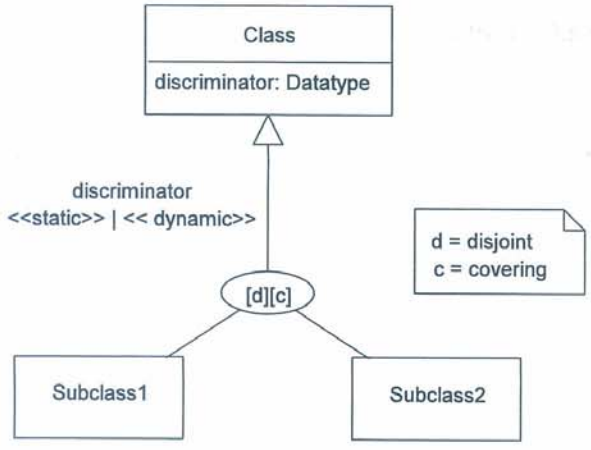
Association class



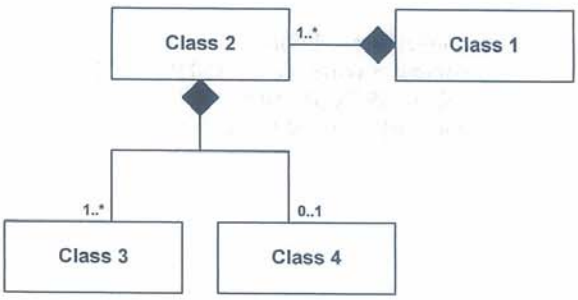
Multiplicity



Generalization



Composition



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

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