UML-kielen formalisointi Object-Z:lla

Kalvot ja seminaarityö WWW:ssä:
http://users.jyu.fi/~minurmin/opiskelu/form/

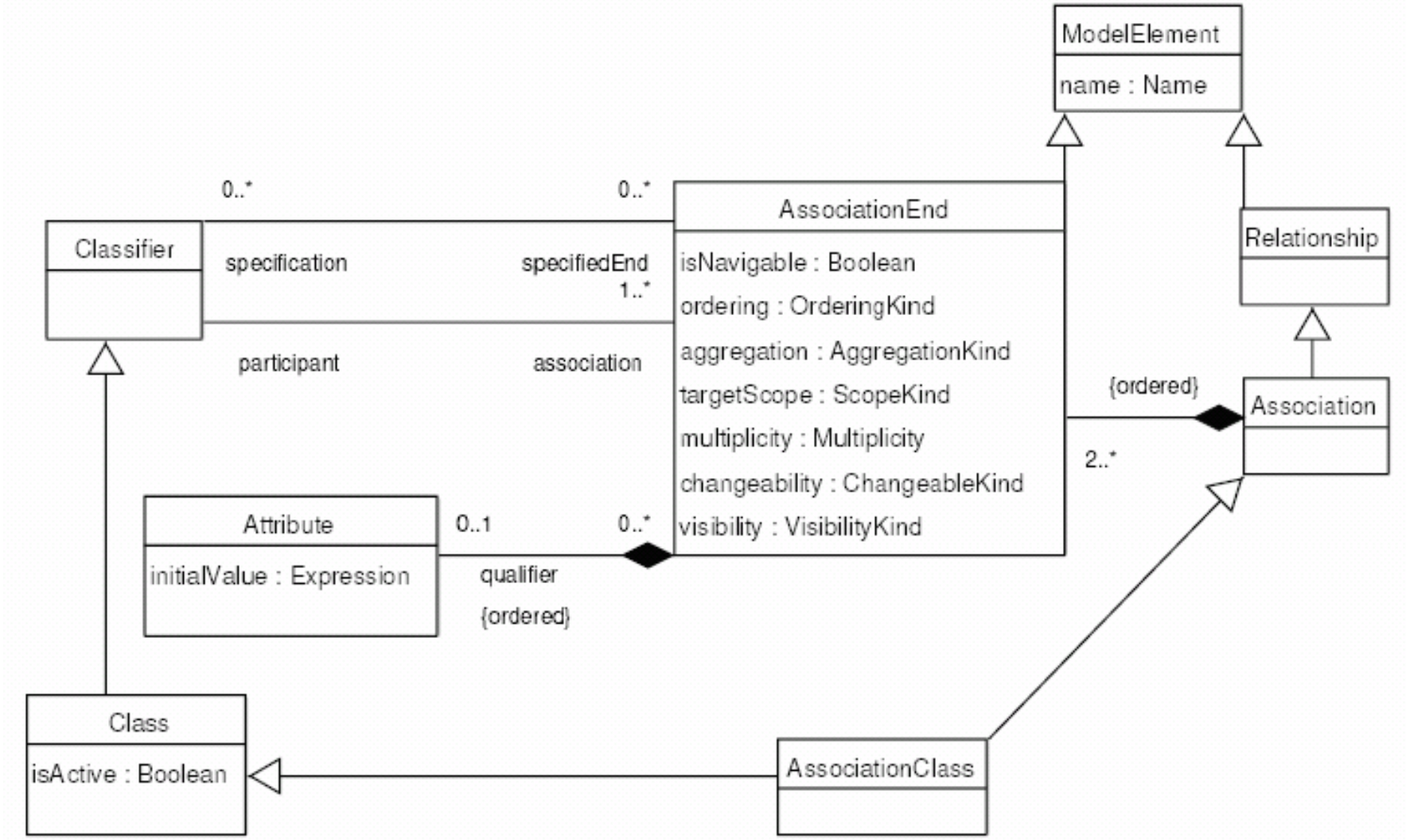
UML

- UML == *Unified Modelling Language.*
- OMG:n standardoima kieli ohjelmistojärjestelmien, liiketoimintamallien ja muiden järjestelmien spesifiointiin, visualisointiin, muodostamiseen ja dokumentointiin.
- 8 kaaviota: käyttötapauskaavio, luokkakaavio, tilakaavio, aktiviteettikaavio, sekvenssikaavio, yhteistoimintakaavio, komponenttikaavio ja sijoituskaavio.
- Laajennusmekanismit: stereotyypit, rajoitteet ja kiinnitetyt arvot.

UML-kielen määrittely

- 4-kerroksinen metamalli (metametamalli, metamalli, malli ja käyttäjän oliot).
- Abstrakti syntaksi määritellään metamallissa luokkakaavioilla.
- Syntaktiset rajoitteet määritellään OCL-kielellä (*Object Constraint language*).
- Semantiikka kuvataan luonnollisella kielellä.
  - > 3 eri näkymää kielen määrittelyyn.
  - > Näkymien välillä ei ole täsmällistä kuvausta.
  - > Määrittely sisältää moniselitteisyyksiä ja epäselvyyksiä.

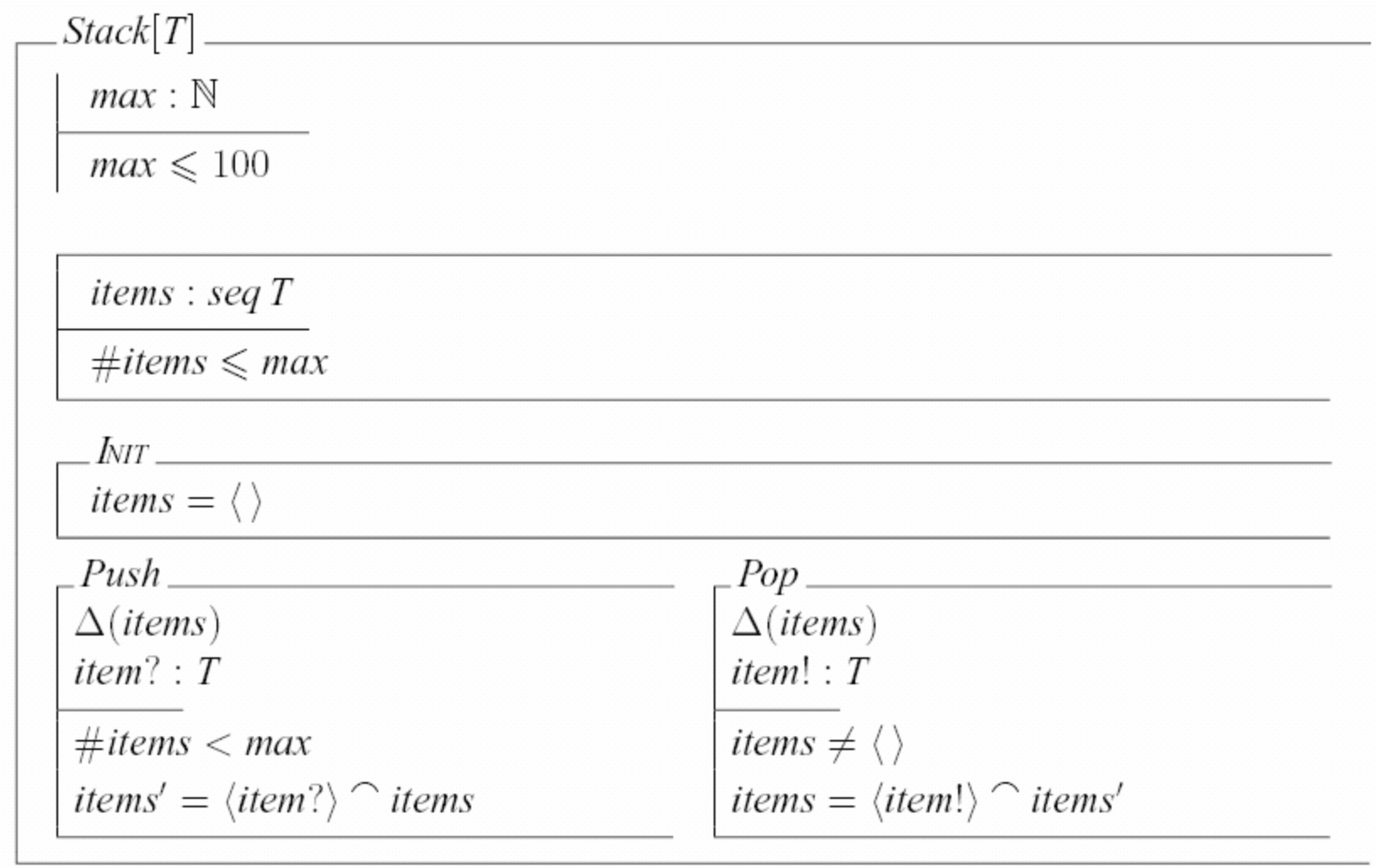
Esimerkki: assosiaatio



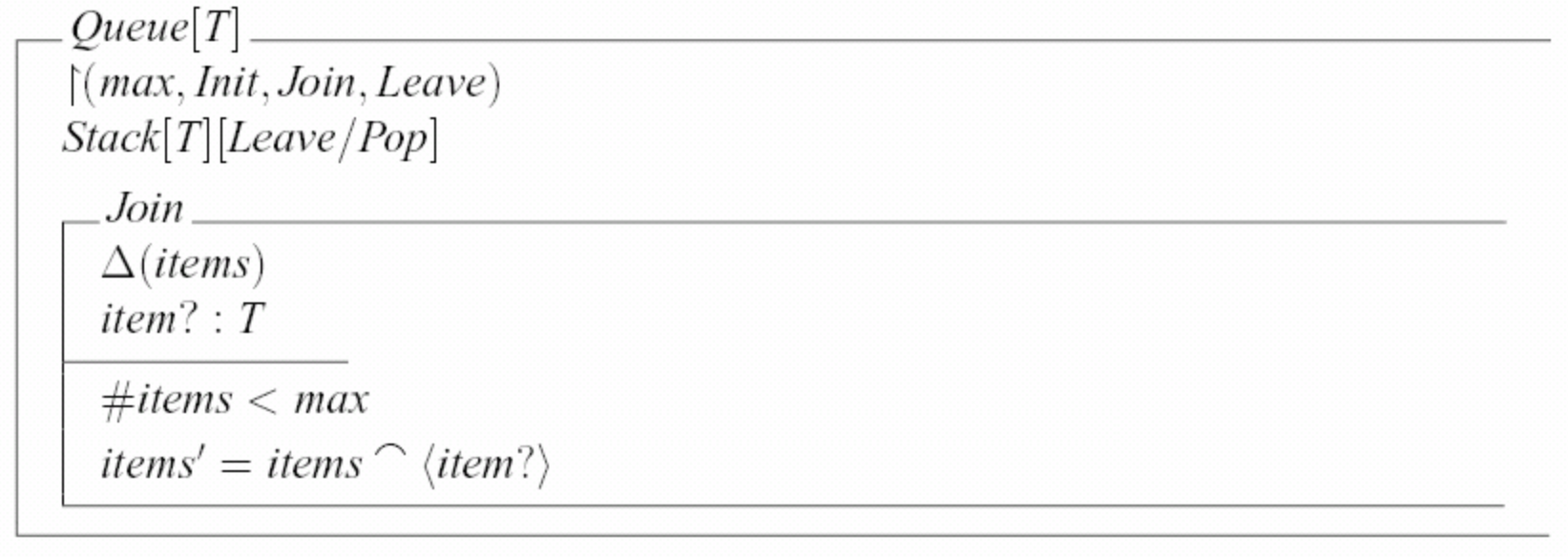
Object-Z

- Oliopohjainen laajennus Z-kuvauskieleen.
- Uusi mallinnuselementti: luokkaskeema.
- Z:n tila- ja operaatioskeemat käytössä luokkaskeeman sisällä. -> luokkainvariantti, metodit.
- Tuettuja oliokäsitteitä: olion identiteetti, (moni)perintä, polymorfismi, tiedon suojaus, viitteet, koosteet.
- Ei metodien kuormitusta -> olion alkutila ilmaistaan yhdellä *init*-operaatiolla.
- Formaali kieli. UML:ää ilmaisuvoimaisempi, ei välttämättä selkeämpi.

Esimerkki: Pino



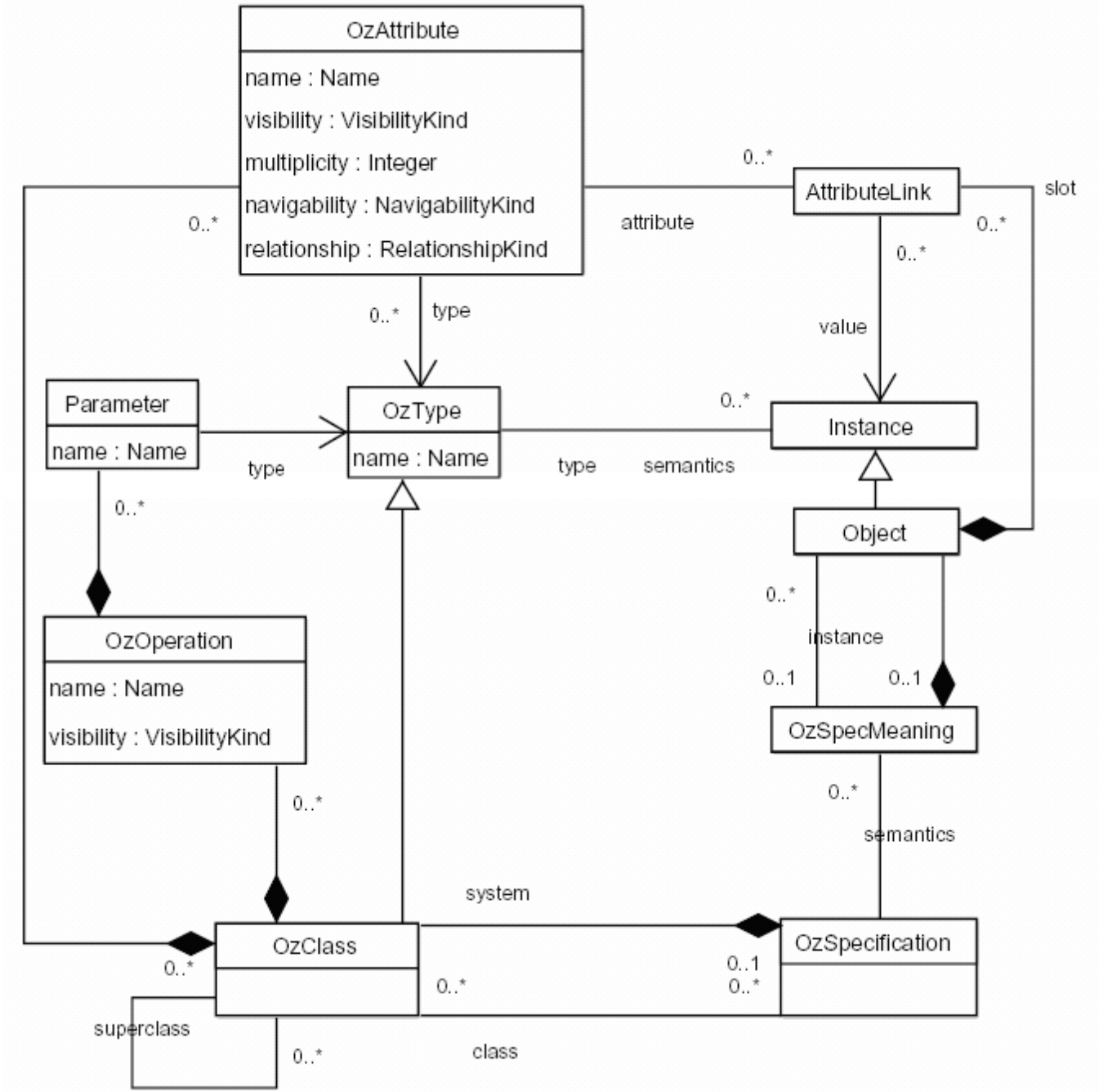
Esimerkki: Jono



- *max*, *Init*, *Join* ja *Leave* ovat luokan julkiset jäsenet (erityisesti *Push* ei ole jonon julkinen metodi).
- *Pop*-metodi nimetty uudelleen nimelle *Leave*.
- Uudelleennimeäminen mahdollistaa polymorfismin.

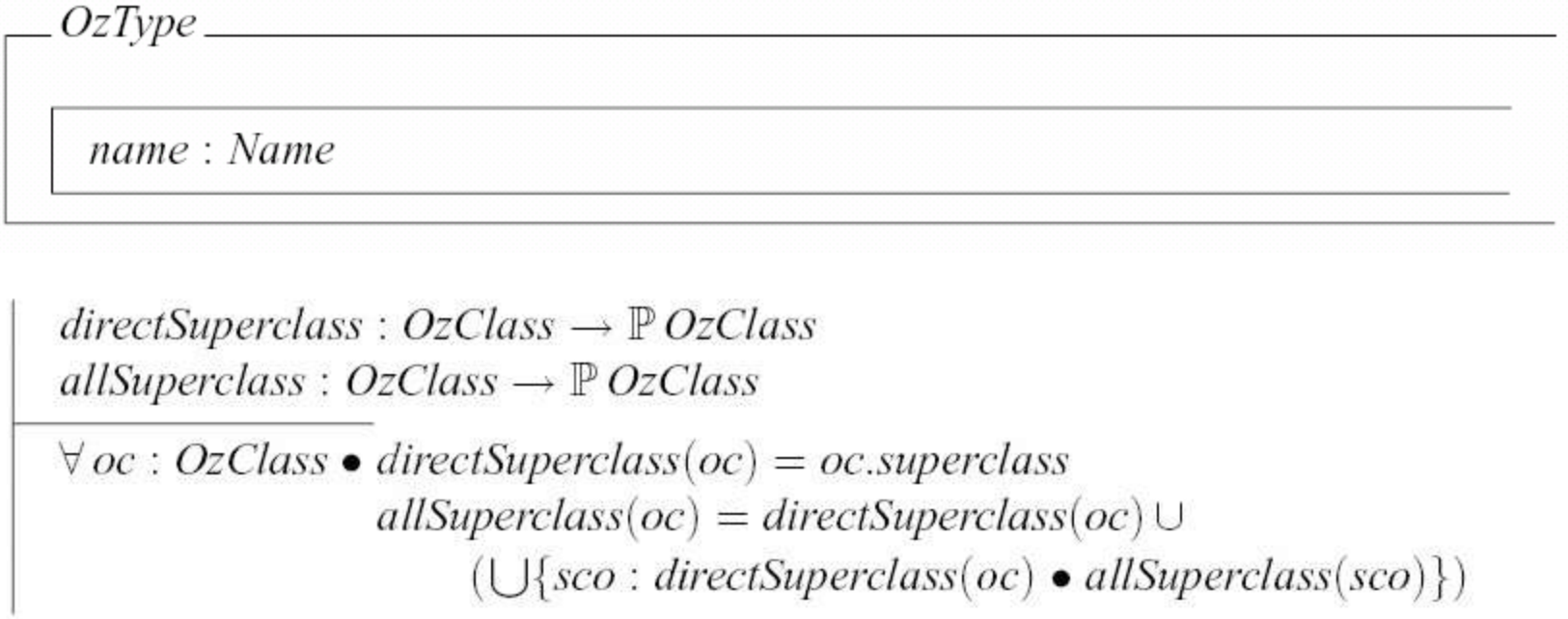
Object-Z:n semantiikasta

- Object-Z:n semantiikka on määritelty Z:lla (hankalaa).
- Vaihtoehtoinen ratkaisu: UML-kielen määrittelyn kaltainen metamalli.
- Metamalli yksinkertaistaisi muunnoksia muihin oliopohjaisiin kieliin.
- Esim. muunnos UML-kielestä Object-Z:aan:
  - > Määritellään kielten metamallit Object-Z -luokkina.
  - > Määritellään muunnos UML-metamallista Object-Z -metamalliin.
  - > Sovelletaan muunnosta malliin.
  - > Tarkennetaan Object-Z -spesifikaatiota niiden piirteiden osalta, joita ei voi kuvata UML:llä.


OzAttribute
name : Name
visibility : VisibilityKind
multiplicity : Integer
navigability : NavigabilityKind
relationship : RelationshipKind
0..*
attribute
AttributeLink
slot
0..*
0..*
value
0..*
type
Parameter
name : Name
type
OzType
name : Name
type
semantics
0..*
Instance
Object
0..*
instance
0..1
0..1
OzSpecMeaning
0..*
semantics
0..*
OzOperation
name : Name
visibility : VisibilityKind
0..*
system
OzClass
0..*
0..*
0..1
0..*
OzSpecification
superclass
0..*
class

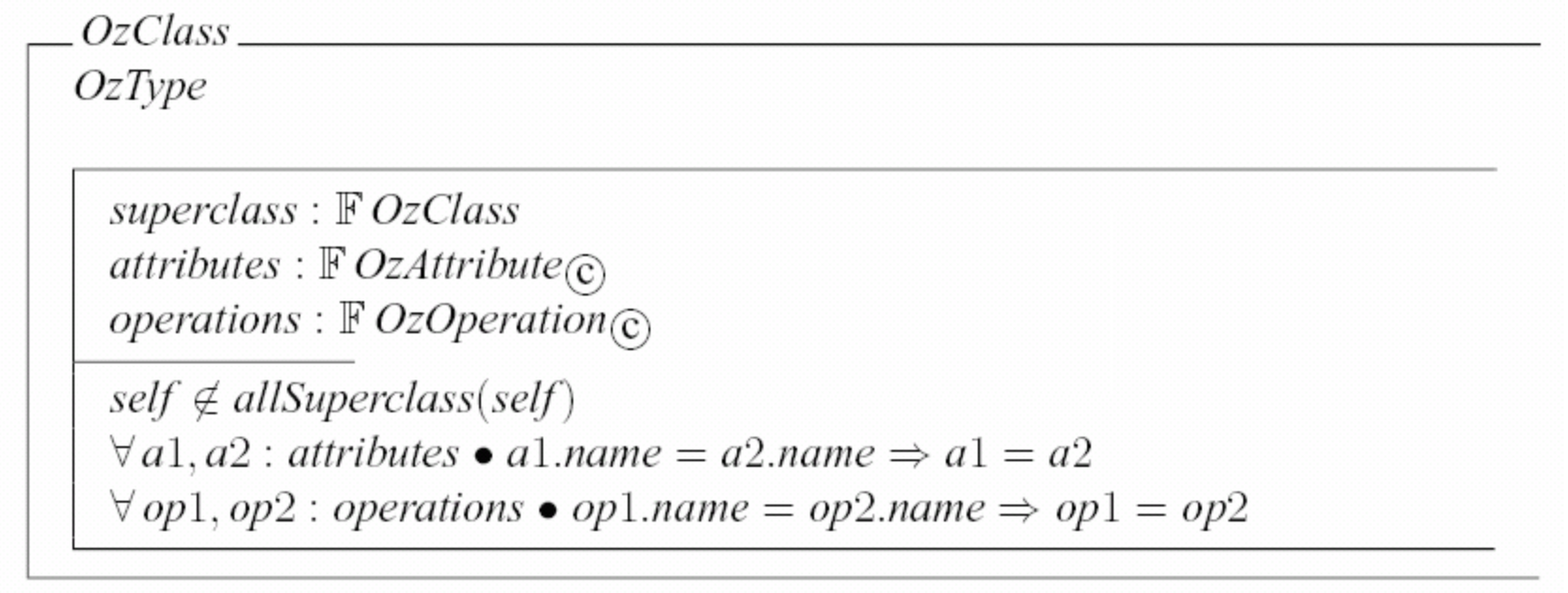
Object-Z -metamalli...

- Määritellään Object-Z:n luokan käsite metamallilla.



- *allSuperClass* palauttaa rekursiivisen sulkeuman kaikista luokan yliluokista.

Object-Z -metamalli...



- Luokalla on äärellinen määrä välittömiä yliluokkia.
- Luokka ei voi olla itsensä yliluokka.
- Luokka koostuu äärellisestä määrästä yksikäsitteisiä attribuutteja ja operaatioita.

UML-metamalli formaalisti

- Kaikkien UML-kaaviotyyppien abstrakti syntaksi määritellään metamallissa luokkakaavioilla.
  > UML-kielen formalisointi aloitettava luokkakaavioista.
- Kielen laajuudesta johtuen formalisointi on ositettava (esim. UML:n pakettirakenne: ydinelementit, käyttäytymiseen liittyvät, mallien hallintaan liittyvät).
- Käsitellään luokkakaavioiden formalisointia, esimerkkinä UML-luokan käsite.
- Muistakin kaavioista esitetty Object-Z -malleja (esim. käyttötapaus- yhteistoiminta- ja tilakaaviot).

UML-metamalli...

UmlClassifier

$$
\begin{array}{l}
name : Name \\
attributes : \mathbb{F}\, UmlAttribute© \\
operations : \mathbb{F}\, UmlOperation©
\end{array}
$$

UmlClass

UmlClassifier

$$
\begin{array}{l}
\forall\, a1, a2 : attributes \bullet a1.name = a2.name \Rightarrow a1 = a2 \\
\forall\, op1, op2 : operations \bullet \\
\quad (op1.name = op2.name \land \#op1.parameters = \#op2.parameters \land \\
\quad \forall\, i : 1..\#op1.parameters \bullet \\
\qquad op1.parameters(i).name = op2.parameters(i).name \land \\
\qquad op2.parameters(i).type = op2.parameters(i).type) \Rightarrow op1 = op2
\end{array}
$$

Muunnosesimerkki

- Määritellään funktio, joka kuvaa UML-metamallin luokan Object-Z-metamallin luokaksi.
- Funktio käyttää abstraktia funktiota *convType*, joka vastaa yksittäisen UML-tyypin muuntamisesta Object-Z -tyypiksi.

$$convType : \downarrow UmlClassifier \rightarrowtail\!\!\!\!\!\rightarrow \downarrow OzType$$

- *convType* on määriteltävä jokaiselle tyypille erikseen.

Muunnosesimerkki

$$
\begin{array}{l}
mapUmlClassToOz : UmlClass \rightarrow \mathbb{P}\, OzClass \\
\hline
\forall\, uc : UmlClass \bullet \\
\quad mapUmlClassToOz(uc) = \{oc : OzClass \mid uc.name = oc.name \wedge \\
\qquad \forall\, ua : uc.attributes \bullet \\
\qquad\quad \exists\, oa : oc.attributes \bullet \\
\qquad\qquad oa.name = ua.name \wedge oa.type = convType(ua.type) \wedge \\
\qquad\qquad oa.visibility = ua.visibility \wedge oa.multiplicity = ua.multiplicity \wedge \\
\qquad\qquad oa.relationship = relNone \wedge oa.navigability = navNone \\
\qquad \forall\, uo : uc.operations \bullet \\
\qquad\quad \exists\, oo : oc.operations \bullet \\
\qquad\qquad oo.name = uo.name \wedge oo.visibility = uo.visibility \\
\qquad\qquad \forall\, up : ran\, uo.parameters \bullet \\
\qquad\qquad\quad \exists\, op : ran\, oo.parameters \bullet \\
\qquad\qquad\qquad op.name = up.name \wedge op.type = convType(up.type)\}
\end{array}
$$

- kaikille UML-luokan attribuuteille ja operaatioille on oltava vastaava Object-Z:n luokan attribuutti ja operaatio tyypinmuunnoksen jälkeen.

Muunnos Object-Z -> UML

- Object-Z -kuvausten rakennetta voi havainnollistaa esittämällä sen UML-luokkakaaviona.
- Yksinkertainen toteutus: määritellään XML-esitys Object-Z -kuvauksille, muunnetaan XSLT:llä.
- Muunnos HTML-muotoon mahdollistaa skeemojen katselun WWW:ssä.
- Muunnos XMI (XML Metadata Interchange)-formaattiin mahdollistaa mallin siirtämiseen nykyisiin CASE-ohjelmiin ja UML-kaavioiden generoinnin mallin pohjalta.
  > Edistää Object-Z:n käyttöä osana nykyisiä välineitä ja menetelmiä.

XML-esimerkki

```
<objectZnotation>
<classdef>
  <name>Stack</name>
<!-- ... -->
  <state>
    <decl>
    <name>items</name>
    <dtype>&seq; <type>T</type></dtype>
    </decl>
    <st/>
    <predicate>#items &leq; max</predicate>
  </state>
  <init>
    <predicate>items = &emptyseq;</predicate>
  </init>
  <op>
    <name>Push</name>
    <delta>items</delta>
    <decl>
    <name>item?</name>
    <dtype><type>T</type></dtype>
    </decl>
    <st/>
    <predicate>#items? &lt; max</predicate>
    <predicate>items′ = &lseq; item? &rseq; &cat; items</predicate>
  </op>
<!-- ... -->
</classdef>
</objectZnotation>
```

Yhteenveto

- UML-kielen formalisointi poistaisi kielen epäselvyydet ja lisäisi formaalien menetelmien käyttöä ohjelmistokehityksessä.
- Muunnos UML:stä Object-Z:aan tukee prosessia, jossa karkea suunnittelu tehdään UML:llä ja tarkennukset formaalisti (jos on tarvetta).
- Muunnoksella Object-Z:sta UML:ään voisi havainnollistaa formaalia kuvausta (tosin kaikkia Object-Z:n piirteitä ei pysty esittämään UML:llä) ja hyödyntää kuvausta nykyisissä työkaluissa.