Az UML kiterjesztési lehetőségei

Kiterjesztések: megszorítások, név-érték párok, sztereotípusok, profilok

- A szabványhoz képest eltéréseket fogalmaznak meg➔óvatosan a kódgenerálással
- Metamodell kiterjesztés/módosítás
- Profilokban tároljuk a kiterjesztéseket: egyes szakterületeken különálló metamodell kiterjesztések elérhetők. Pl. SPEM: Software Process Engineering Metamodel

Object Constraint Language (OCL)

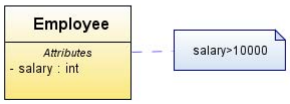
- Tisztán **kifejezésközpontú** nyelv: nincsenek mellékhatások
- **Modellezési** nyelv: nincsenek megvalósítási kérdések
- **Nem programozási** nyelv: nem végrehajtható, nincsen vezérlés
- **Predikatív** nyelv: minden kifejezés kiértékelhető, logikai eredménnyel
- **Erősen típusos** nyelv: minden jól formált OCL kifejezésnek típusa van, a típusegyeztetés kötelező
- **Objektumorientált** nyelv: ezért használható az UML rendszerekben
- **Megszorító** nyelv: valamilyen példányhalmazhoz kapcsoljuk, megszorítja a halmazt olyan egyedekre, amelyekre a kifejezés igazra értékelődik ki.

Az OCL használati környezetei

- Osztályok, típusok és kapcsolatok invariánsai
- Sztereotípusok típusinvariánsai
- Műveletek elő- és utófeltételei
- Tevékenység- és állapotdiagramok őrfeltételei
- Navigációt leíró nyelv
- Műveletek megszorításai
- A grafikus nyelv is megfogalmazható OCL megszorításként.

```
xor(x:Boolean):Boolean=or(x).not();
```
- Műveletleíró megszorítás

```
addInterest(sum:Real,int:Real):Real
pre: sum>=0 and int>0
post: addInterest>sum
```
- Művelet elő és utófeltétele



Az OCL felépítése

- Egy OCL feltétel egy Boolean típusú kifejezés
- Egy OCL kifejezésből navigációval egy másik kifejezés képezhető. Az új kifejezés típusát a navigáció határozza meg.
  - Navigálhatunk az alapkifejezés típusmeghatározásában megadott **tulajdonság**gal. Az eredmény típusa a tulajdonság típusa lesz. (`Employee.salary`)
  - Navigálhatunk az alapkifejezés típusához rendelt **kapcsolat**tal. Az eredmény típusa a navigációban használt kapcsolatvég típusa lesz (`Person.child`)
  - Navigálhatunk **műveletkifejezés**sel. Az eredmény típusa a művelet eredményének típusa lesz. Egyes műveletek operátorok, vagyis nem kötelező a zárójeles kifejezés használata. (`Person.doMarry`)
  - **Gyűjtemény-művelet** navigációhoz a **„->”** operátort használjuk (`hétVezér->size=7`)

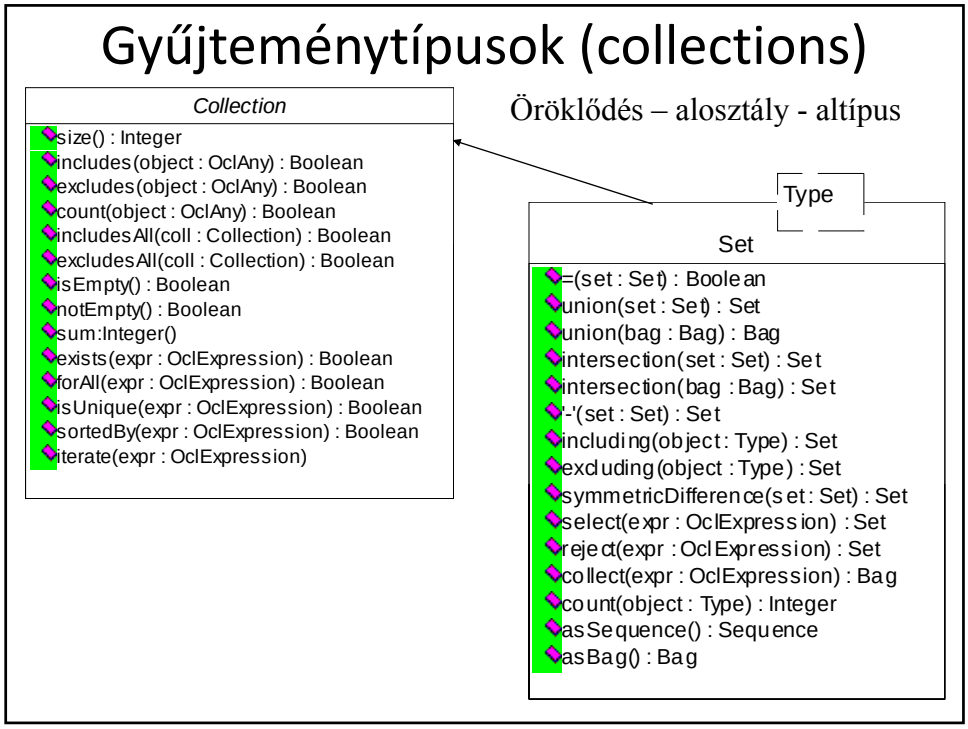
Az OCL típusai

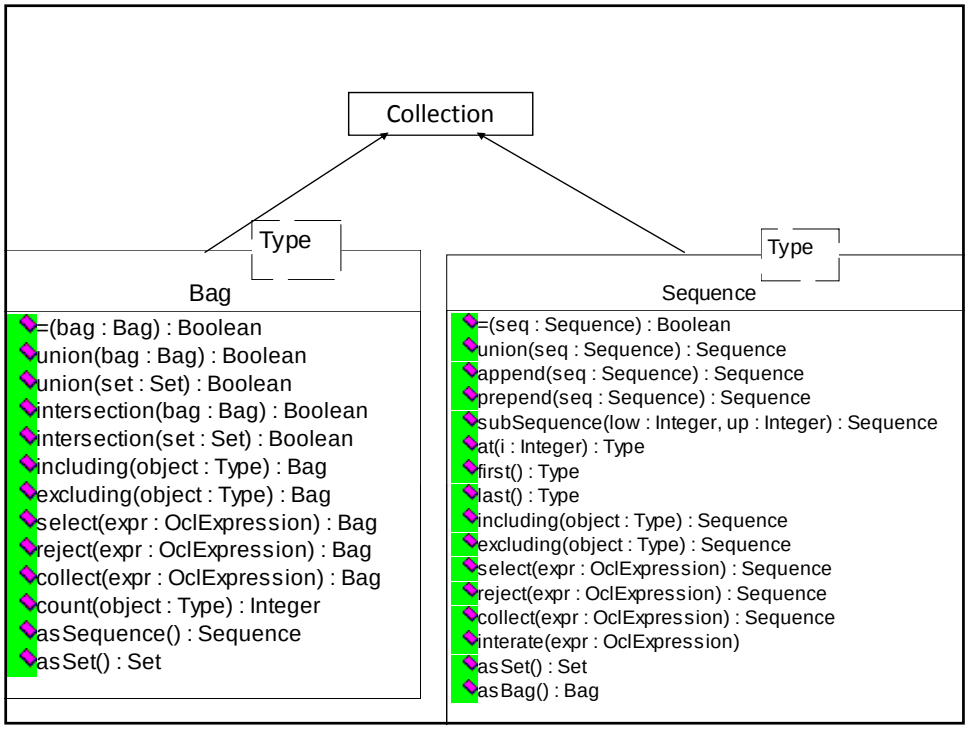
- Alaptípusok: `Boolean, Integer, Real, String`
- Típusmegfelelés: egy típus megfelel az őstípusainak tranzitíven is
- `OclAny`: minden típus őstípusa
- `OclType`: amelynek a típusok a példányai
- `Collection`
  - `Set, Bag, Sequence`
  - `Set(T), Bag(T), Sequence(T)`

Literális konstansok OCL-ben

- `Set(alma, körte, barack)`
- `Bag(alma, alma, körte)`
- `Sequence(0, 1, 1, 2, 3, 5, 8)`

Gyűjteménytípusok (collections)



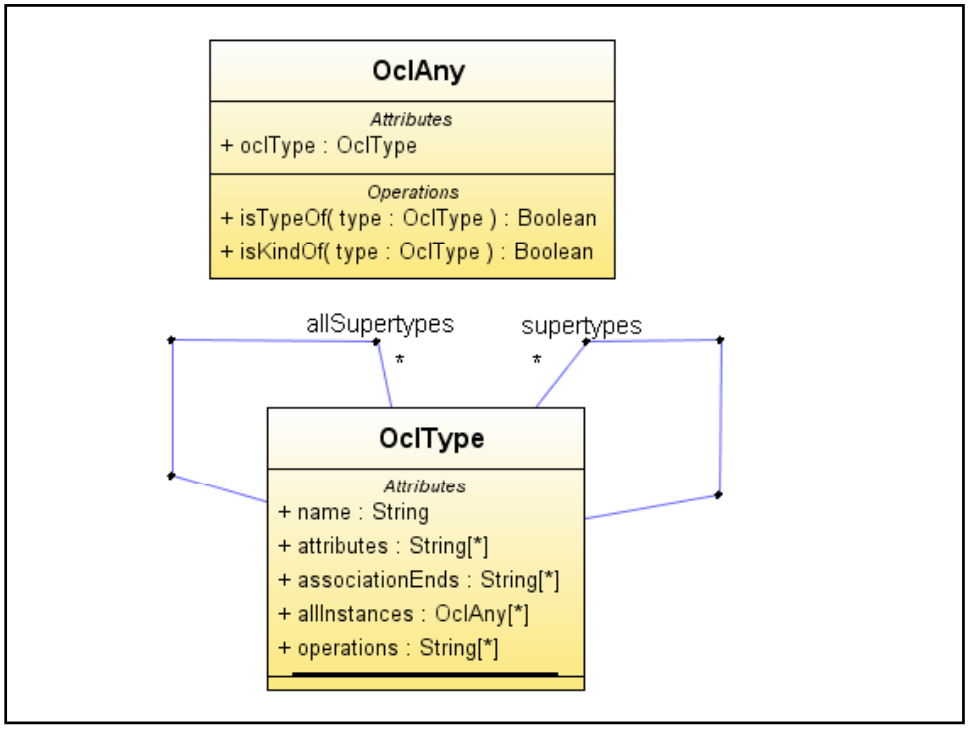


Felsorolástípusok

- Osztályok `<<enumeration>>` , `<<enum>>` sztereotípussal
- Értékeik: attribútumnevek
- OCL nyelvtan: `sex{male,female}`
- Hivatkozás felsorolásértékre:
- `sex=#female`

Metaműveletek

- `OclAny::oclType: OclType`
- `OclAny::oclIsTypeOf(t:OclType):Boolean`
- `OclAny::oclIsKindOf(t:OclType):Boolean`
- `OclType::name:String`
- `OclType::attributes:Set(String)`
- `OclType::associationEnds:Set(String)`
- `OclType::allInstances:Set(OclAny)`
- `OclType::operations:Set(String)`
- `OclType::supertypes:Set(OclType)`
- `OclType::allSupertypes:Set(OclType)`


OclAny
Attributes
+ oclType : OclType
Operations
+ isTypeOf(type : OclType) : Boolean
+ isKindOf(type : OclType) : Boolean
allSupertypes
*
supertypes
*
OclType
Attributes
+ name : String
+ attributes : String[*]
+ associationEnds : String[*]
+ allInstances : OclAny[*]
+ operations : String[*]