Metamodellek a szoftverfejlesztésben

Object Constraint Language

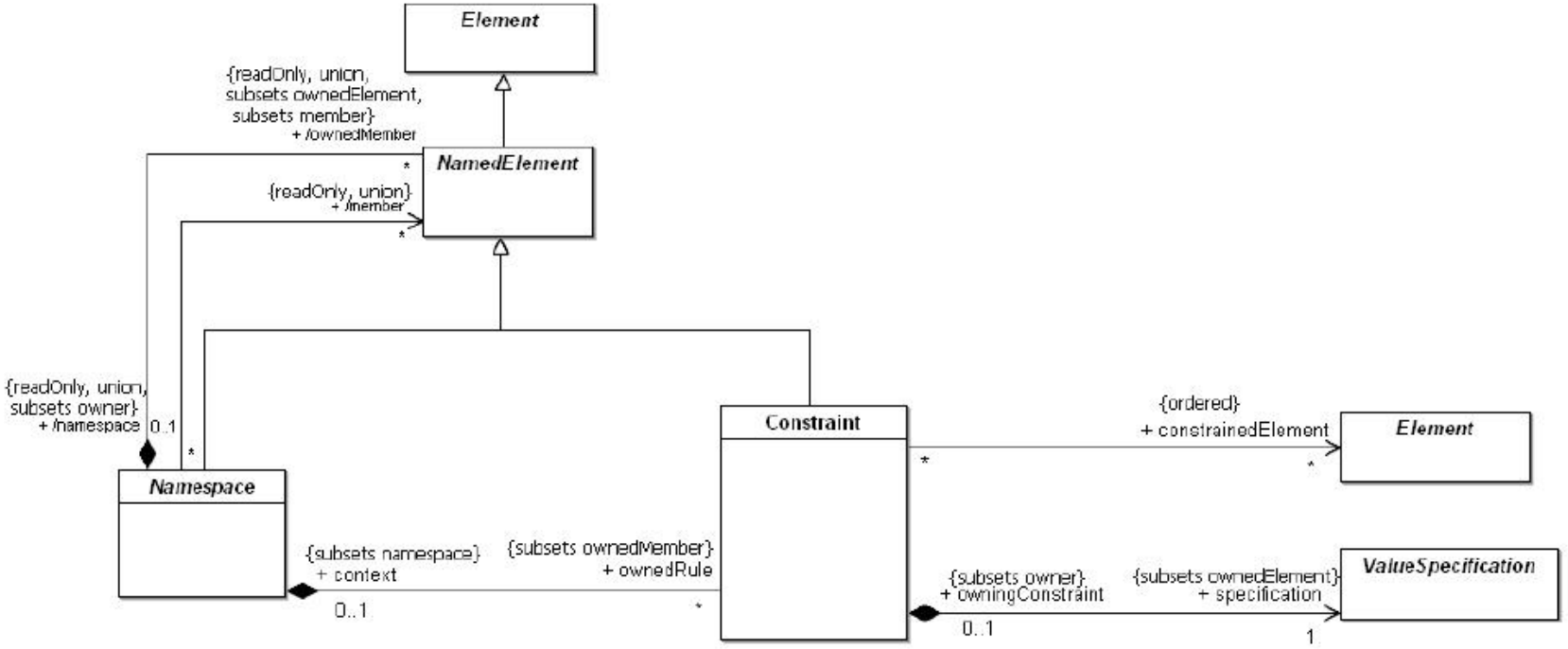
Constraint

- "represents additional semantic information attached to the constrained elements."
- "is an assertion that indicates a restriction that must be satisfied by a correct design of the system."

  UML Infrastructure

Constraint

- UML Infrastruktúra:



OCL ?

- formál nyelv – omg standard

  `http://www.omg.org/spec/OCL/2.0/`

- OCL kifejezések
  - mindig UML modelhez kapcsolódnak
  - előírások – invariáns, előfeltétel
  - lekérdezések
  - kiértékelésnek nincs mellékhatása
  - operációkat specifikálnak - utófeltétel

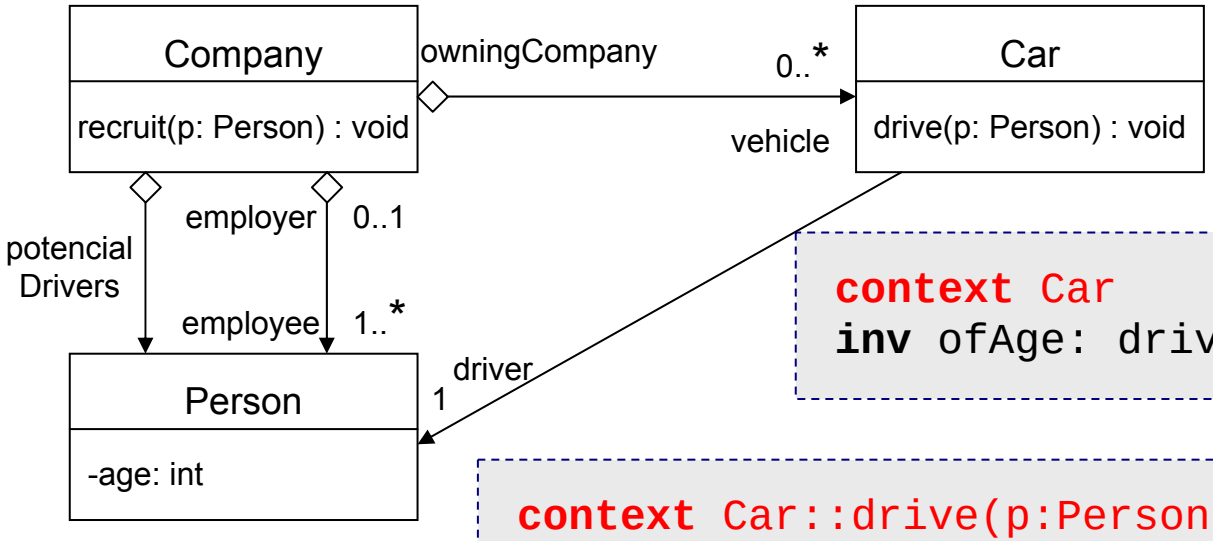
Miért OCL ?

- Alapja: IBM és Syntropy
- Matematikailag megalapozott
  - nem matematikai jelölés – civileknek is
- Deklaratív nyelv
  - MIT és nem HOGYAN
- Szigorúan típusos
  - könnyű ellenőrizni

OCL és UML kapcsolata

- Context = azon UML elem, amelyhez az OCL kifejezés kapcsolódik.
- Lehet:
  - classifier
  - operáció
  - property

OCL és UML kapcsolata




```
context Car
inv ofAge: driver.age >= 18
```

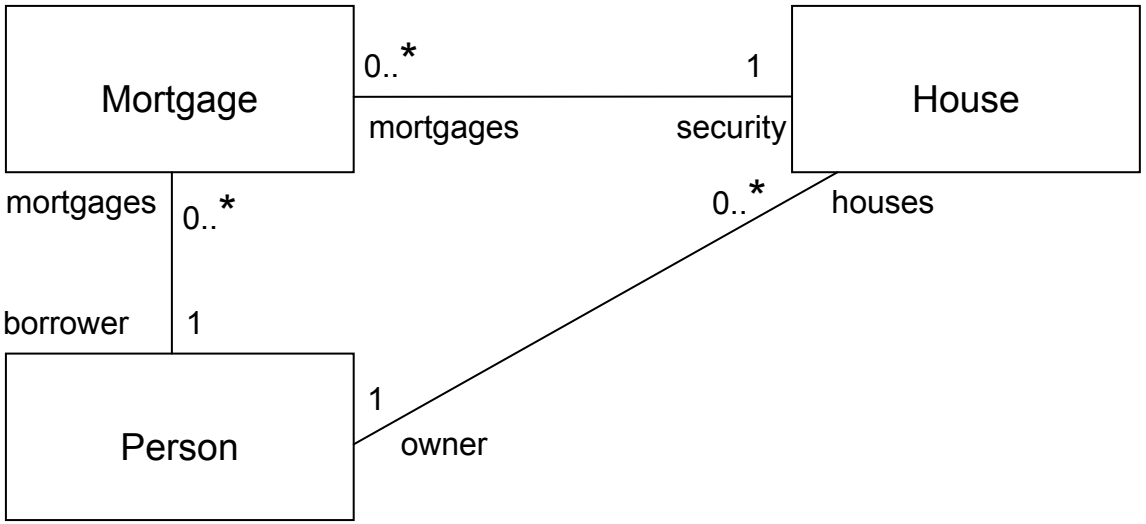
```
context Car::drive(p:Person)
pre:   (self.driver == null) && (p.age >= 18)
post:  self.driver = p
```

```
context Person::age : Integer
init: 16
```

OCL és UML kapcsolata

- Contextual type:
  - a context vagy az őt tartalmazó classifier típusa
- Contextual instance:
  - A contextual type éppen használt példánya
  - `self`

Self szerepe




```
context Person
inv: self.mortgages.security.owner
          ->forAll(owner: Person | owner = self)
```

OCL típusok

- Predefinit típusok (value type)
  - Alaptípusok
  - Kollekciók
  - Speciális
- Felhasználó által definiált típusok
  - Az UML model példányosítható elemeinek típusai (value vagy object type)

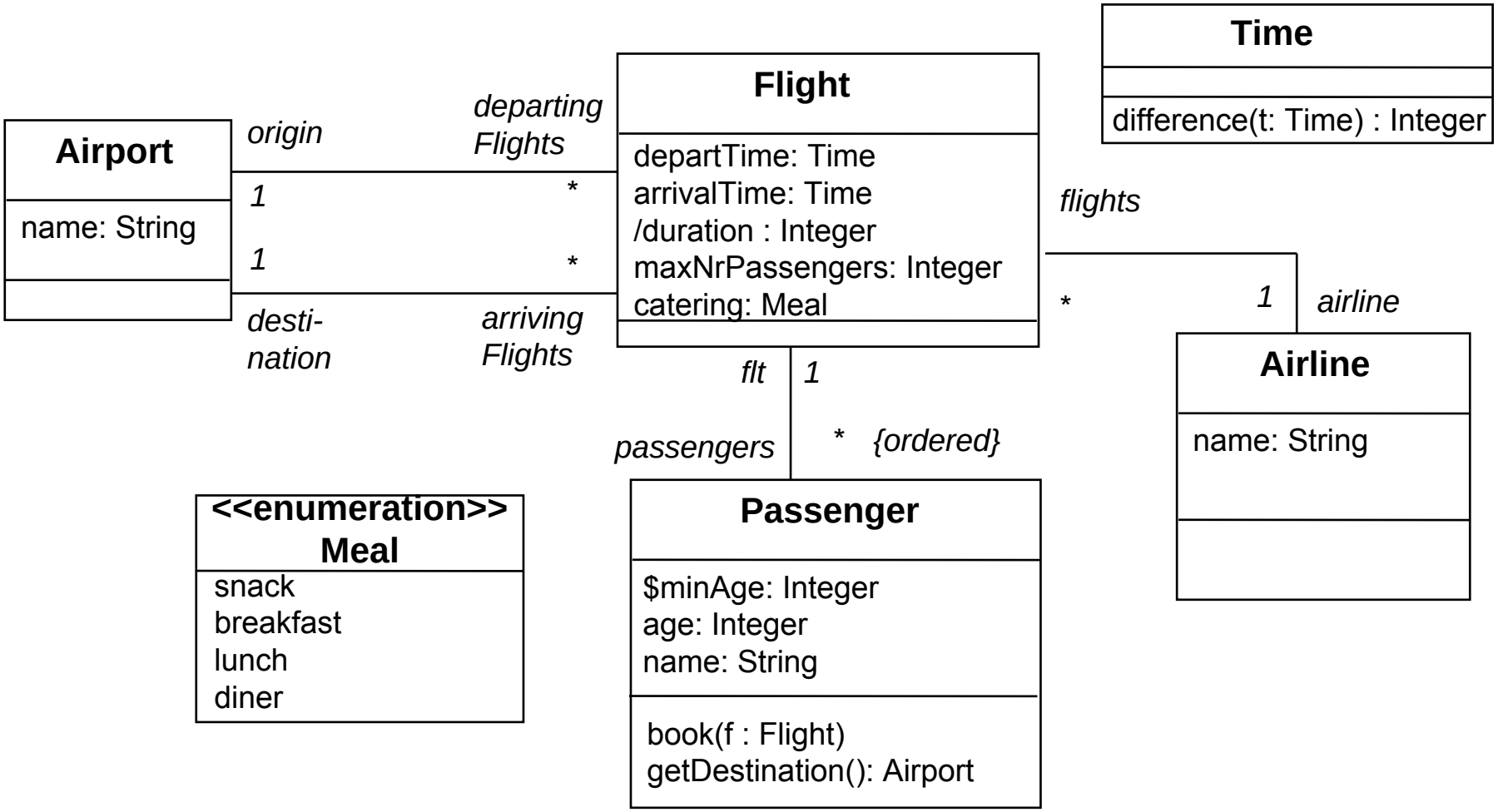
OCL kifejezések elemei

- Alaptípusok
  - String, Boolean, Integer, Real
- UML model classifier és annak jellemzői
  - (osztály) attribútum
  - (osztály) lekérdező művelet
  - láthatóságot figyelmen kívül hagyjuk
- UML enumerációk
- UML asszociációk

Alaptípusok

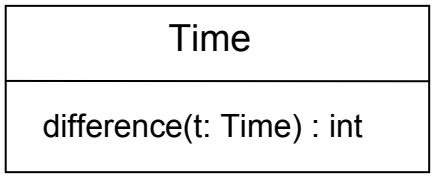
- Boolean
  - `true, false`
  - `and, or, ... implies, if-then-else`
- Integer, Real
- String
  - 'ez a string'
  - `concat(String), size(), toLower()`
  - `substring(int, int), =, <>`

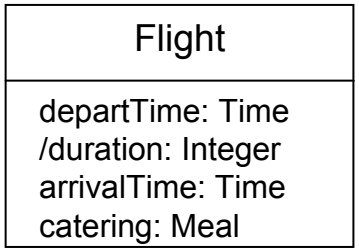
UML model elemek - példa

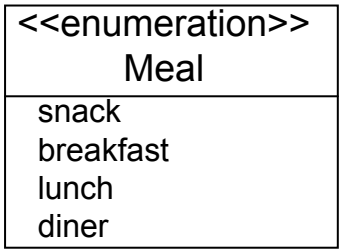


UML model elemek

- Lekérdező művelet







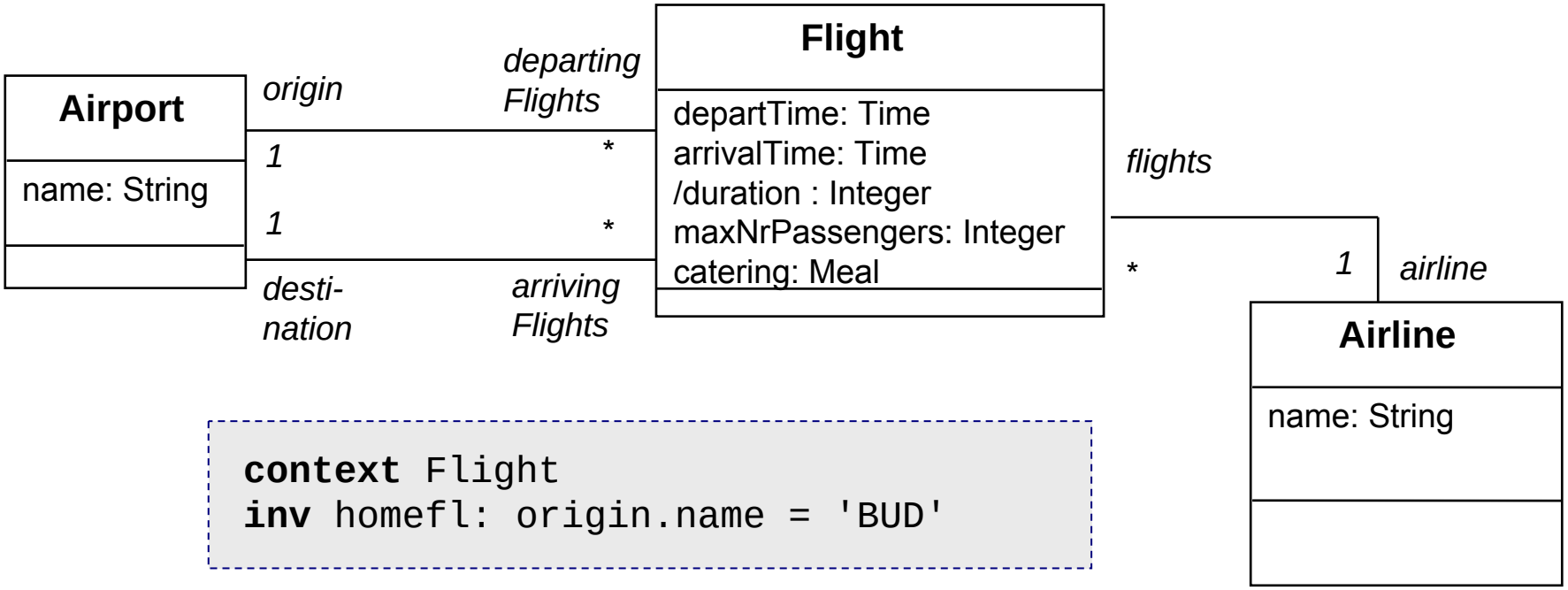
- Enumeráció

```
context Flight
inv: self.departTime.difference(self.arrivalTime)
                                          = self.duration
inv: self.duration < 60 implies self.catering = Meal::snack
```

UML asszociációk

- Minden asszociáció egy navigációs útvonalat definiál
- A `context` a kiinduló pont
- A role név azonosítja a kapcsolódó classifiert
- "." jelölést használunk

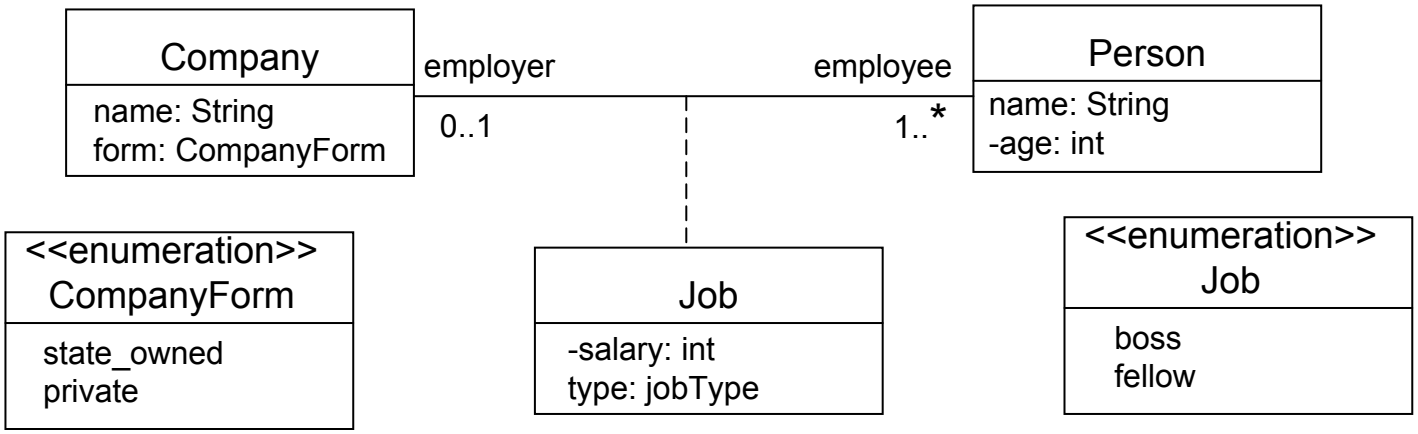
UML asszociációk




```
context Flight
inv homefl: origin.name = 'BUD'
```

```
context Airport
inv: if departingFlights->isEmpty() then .. else ..
```

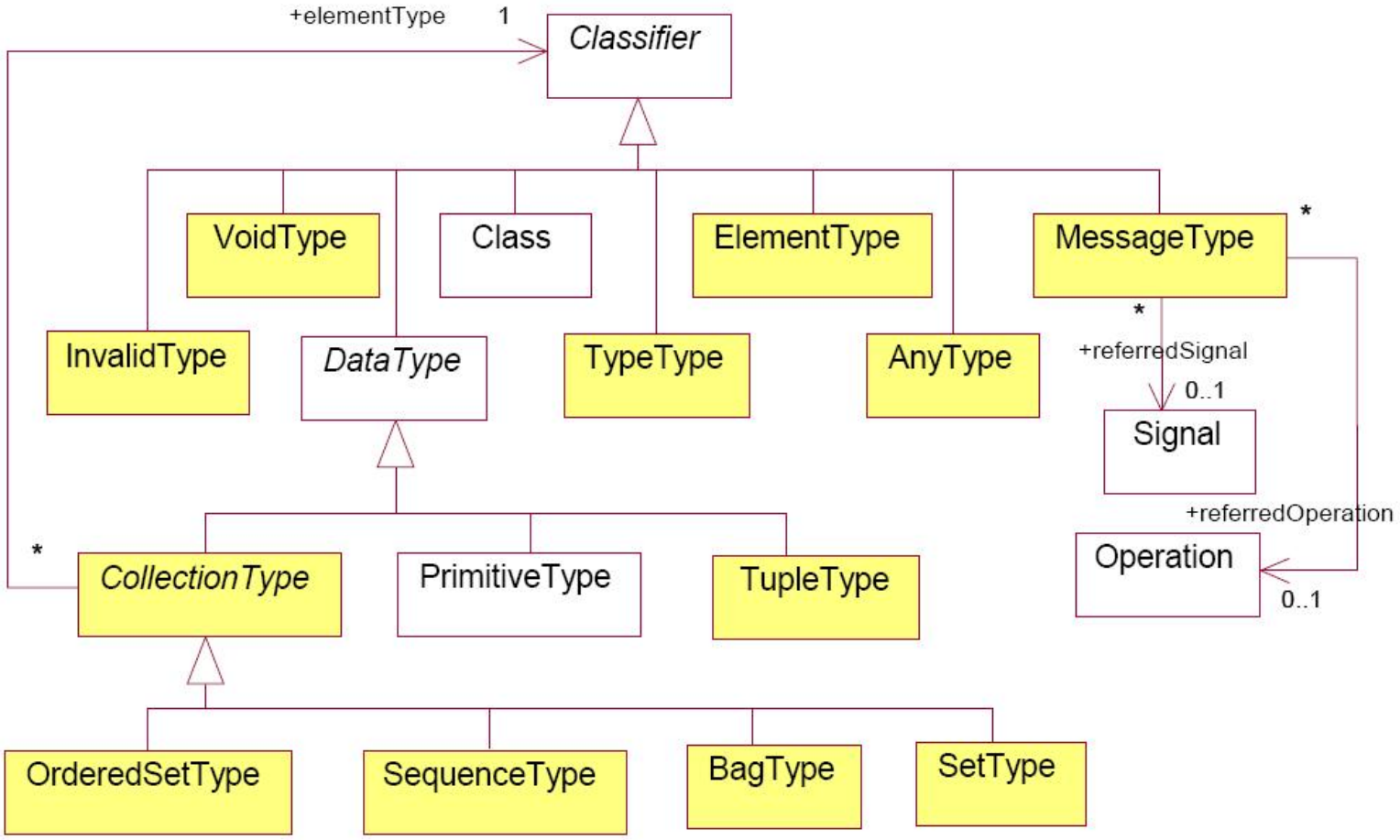
UML asszociációs osztály




```
context Person
inv: if employer.form = CompanyForm::state_owned
        then Job.salary >= 10000
     else Job.salary = 5000
     endif

context Company
inv: Job->select(salary > 10000)->size() > 5
```

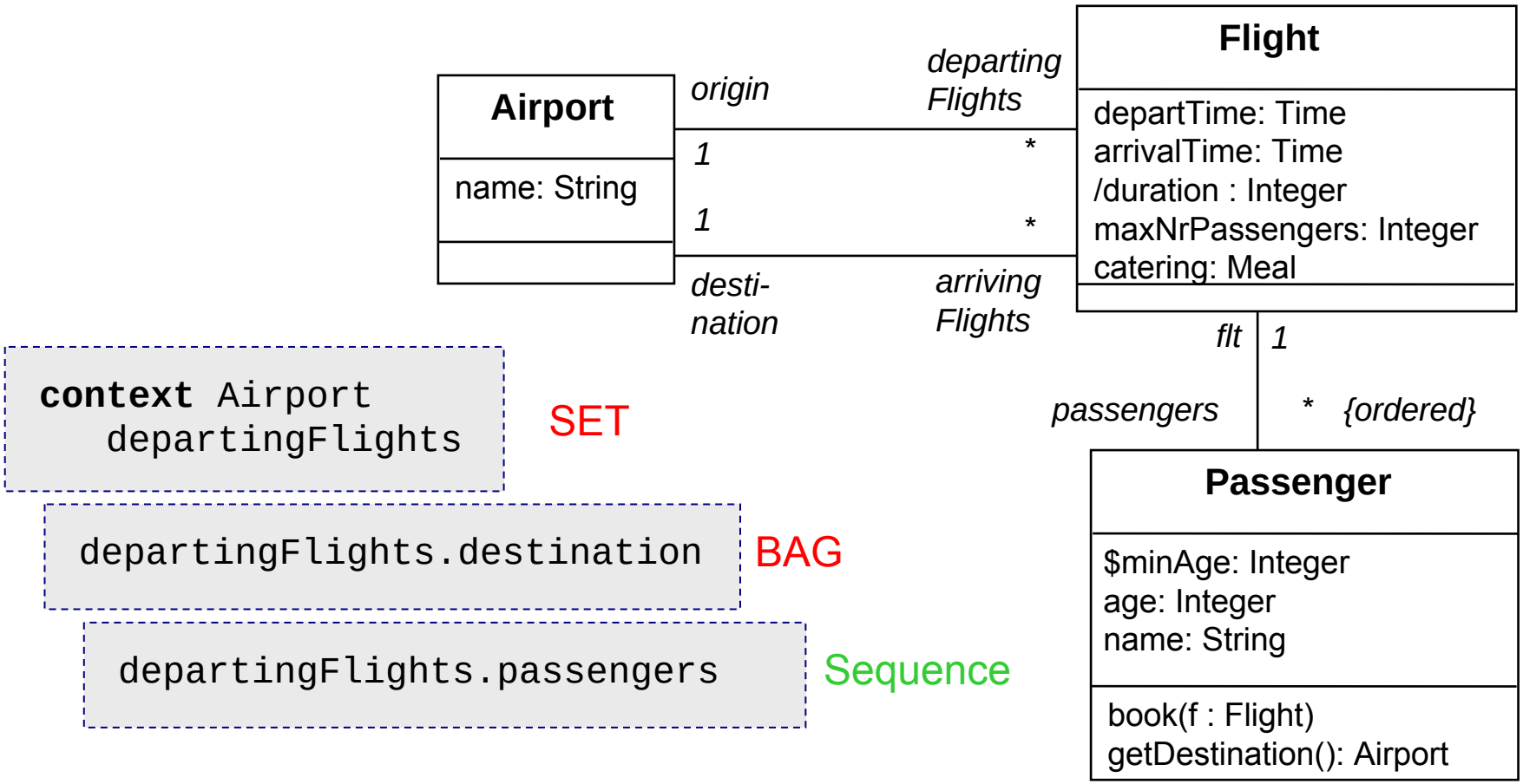
OCL kollekciók



OCL kollekciók

	non-ordered	ordered
non-unique	BAG	SEQUENCE
unique	SET	ORDERED SET

OCL kollekciók



OCL kollekciók

- Konstansok
  - `Set {'apple', 'orange', 'pear'}`
  - `Sequence {1, 4, -2, 4, 3, 7}`
  - `Bag {}`
- Elemi kifejezés
  - `OrderedSet {Student}`
- Műveletek – szintaxis:
  - `collection->operation`

OCL kollekciók

- Közös műveletek

count(object) : Integer	előfordulások száma
excludes(object): Boolean	igaz, ha az object nincs a kollekcióban
excludesAll(collection): Boolean	igaz, ha a pm kollekció nincs a cél-kollekcióban
includes(object): Boolean	igaz, ha az object benne van a kollekcióban
includesAll(collection): Boolean	igaz, ha a pm kollekció benne van a cél-kollekcióban
isEmpty(): Boolean	igaz, ha a kollekció üres
notEmpty(): Boolean	igaz, ha a kollekció nem üres
size(): Integer	elemek száma
sum(): Integer v Real	elemek összege

OCL kollekciók

- Közös műveletek – változó jelentés

Művelet	Set	OrderedSet	Bag	Sequence
=	X	X	X	X
<>	X	X	X	X
-	X	X		
append(object)		X		X
asBag()	X	X	X	X
asOrderedSet	X	X	X	X
asSequence()	X	X	X	X
asSet()	X	X	X	X

OCL kollekciók

- Közös műveletek – változó jelentés

Művelet	Set	OrderedSet	Bag	Sequence
at(index)		X		X
excluding(object)	X	X	X	X
first()		X		X
flatten()	X	X	X	X
including(object)	X	X	X	X
indexOf(object)		X		X
insertAt(index, object)		X		X

OCL kollekciók

- Közös műveletek – változó jelentés

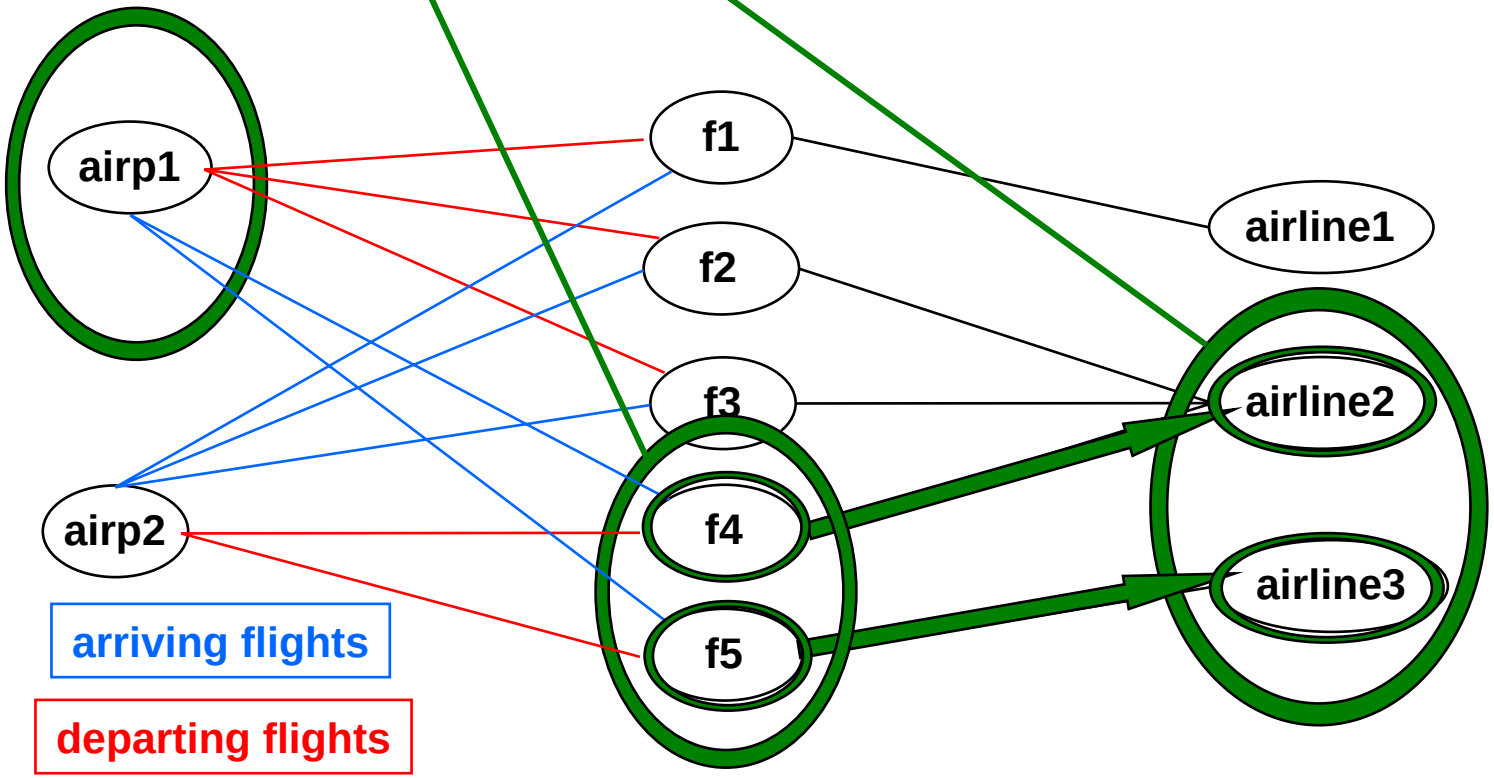
Művelet	Set	OrderedSet	Bag	Sequence
intersection(collection)	X		X	
last()		X		X
prepend(object)		X		X
subOrderedSet(lower, upper)		X		
subSequence(lower, upper)				X
symmetricDifference(collection)	X			
union(collection)	X	X	X	X

OCL kollekciók - iterációk

- collect
  - szintaxis
    - `collection->collect(elem : T | expr)`
    - `collection->collect(elem | expr)`
    - `collection->collect(expr)`
  - rövid forma
    - `collection.expr`
  - azon objektumok kollekciója (bag vagy sequence), amelyek a `collection`-ön kielégítik `expr`-t
  - lapos (flattened) kollekció

OCL kollekciók - collect

```
context Airport inv:
self.arrivingFlights -> collect(airLine) ->notEmpty
```



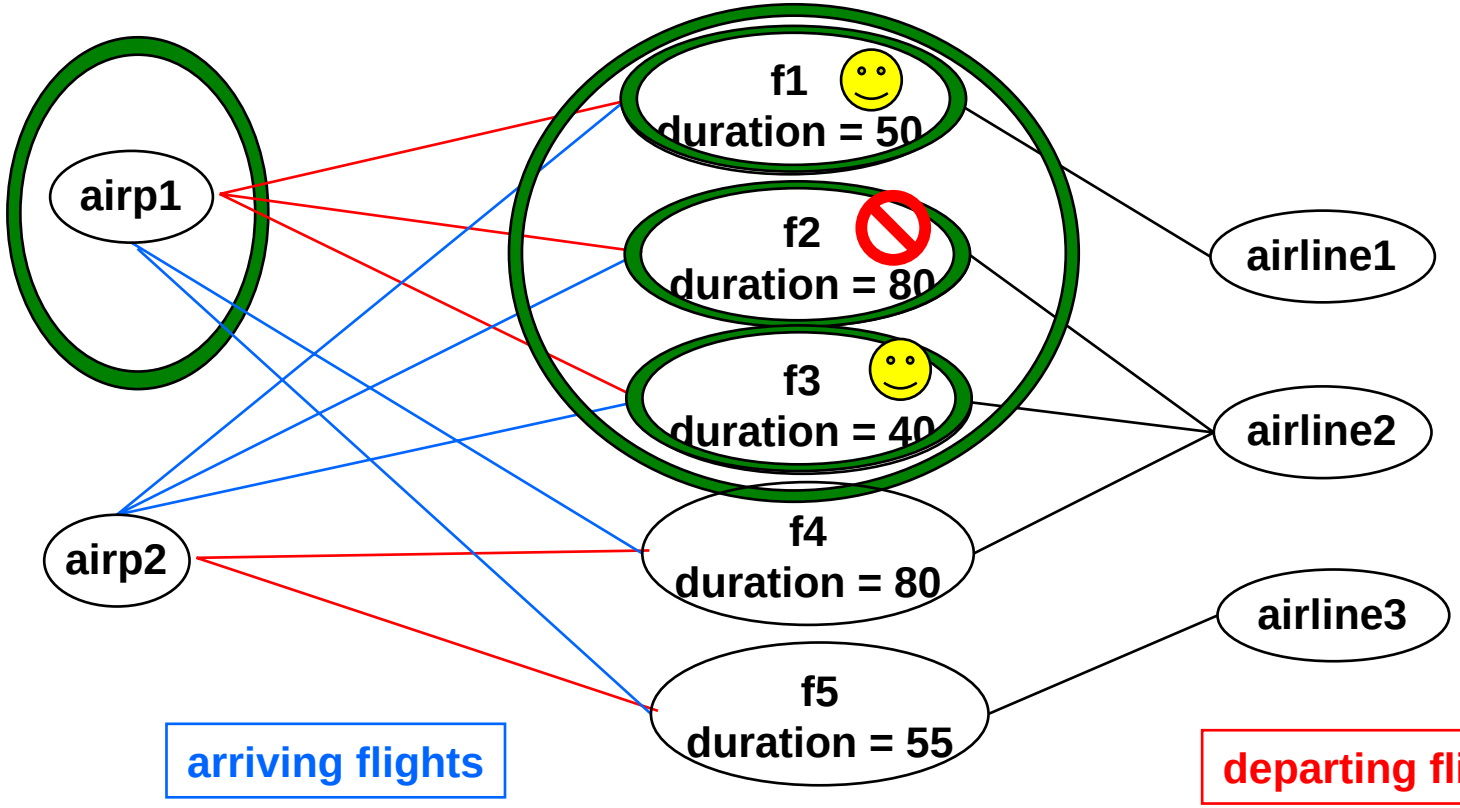
OCL kollekciók - iterációk

- select
  - szintaxis
    - `collection->select(elem : T | expr)`
    - `collection->select(elem | expr)`
    - `collection->select(expr)`
  - `collection` egy részhalmaza, amelynek elemei kielégítik `expr`-t

OCL kollekciók - select

context Airport inv:

self.departingFlights->select(duration<60)->notEmpty



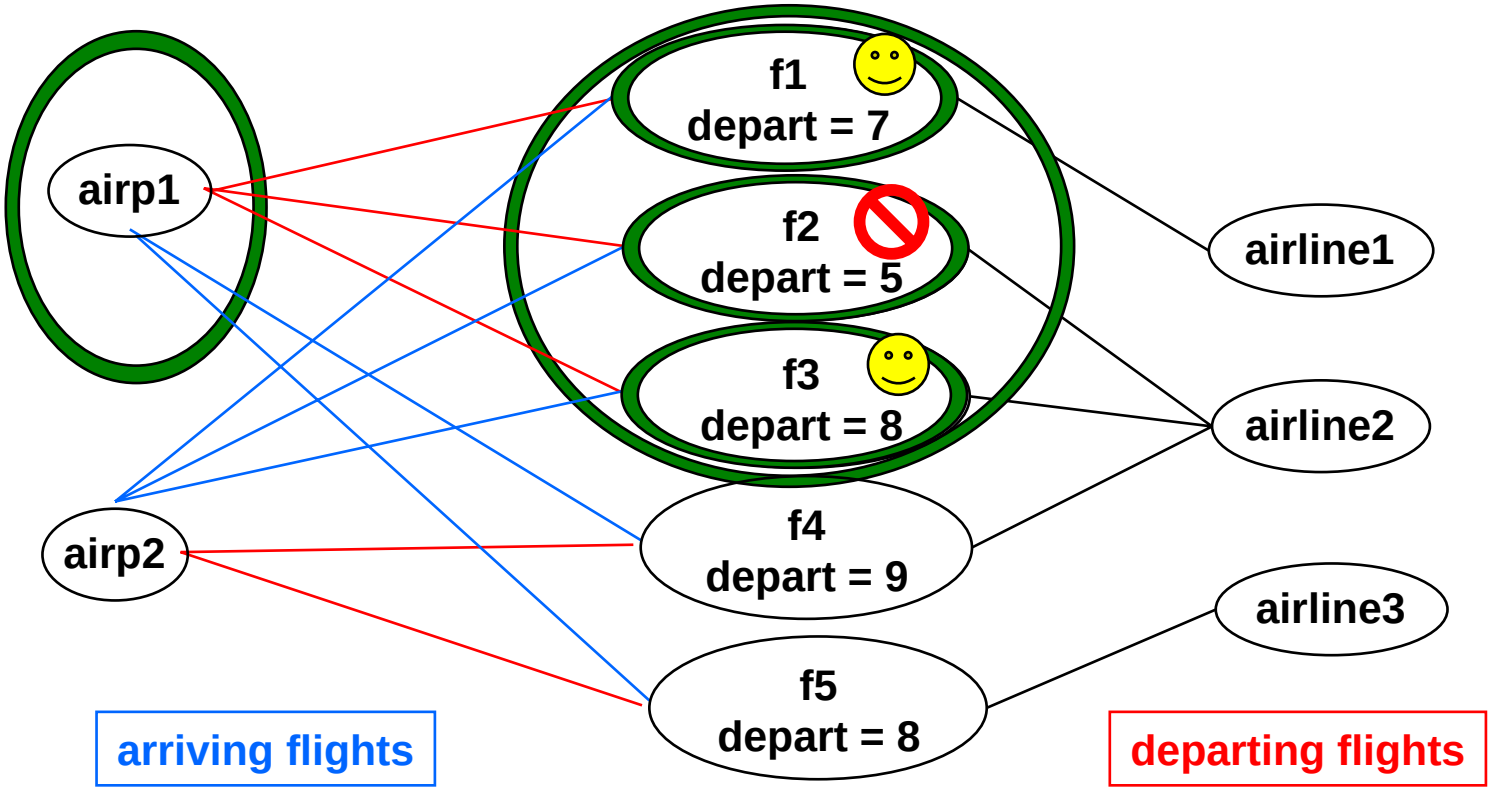
OCL kollekciók - iterációk

- forAll
  - szintaxis
    - `collection->forAll(elem : T | expr)`
    - `collection->forAll(elem | expr)`
    - `collection->forAll(expr)`
  - igaz, ha `expr` igaz a `collection` valamennyi elemére

OCL kollekciók - forAll

context Airport inv:

self.departingFlights->forAll(departTime.hour>6)



OCL kollekciók - iterációk

- iterate
- általános iteráció, valamennyi specifikus iteráció leírható
  - szintaxis

```
collection->iterate(elem : Type1;
           result : Type2 = <expression> |
          <expression-with-elem-and-result>)
```

OCL kollekciók - iterate

- példa

```
context Airline inv:
flights->select(maxNrPassengers > 150)->notEmpty
```

```
context Airline inv:
flights->iterate (f : Flight;
      answer : Set(Flight) = Set{ } |
            if f.maxNrPassengers > 150 then
                  answer->including(f)
            else
                  answer endif )->notEmpty
```

OCL kollekciók - iterációk

- Egyéb műveletek

any(expr)	az expr-et kielégítő véletlen elemet ad
collectNested(expr)	a collect speciális esete, amely megtartja a kollekció belső szerezetét (nem lapít)
exists(expr)	igaz, ha a kollekció legalább egy elemére expr fennáll
isUnique(expr)	igaz, ha expr a kollekció minden elemére egyedi
one(expr)	igaz, ha a kollekció pontosan egy elemére áll fenn expr
reject(expr)	select(not expr)
sortedBy(expr)	rendezést csinál expr alapján ("< " kötelező)

Context-ek

- Classifier
  - `inv` – invariáns
    - konstruktor és operáció után igaz
  - `def` – attribútum és operáció definíció

```
context Passenger
def: initial: String = name.substring(1, 1)

context Airport
def: nrOfDest() : Integer =
        departingFlights.destination->asSet()->size()
```

Context-ek

- Property
  - `derive` – származtatott attrib ("/")
    - invariánsként viselkedik
  - `init` – kezdőérték
    - a létrehozás pillanatában érvényes

```
context Flight::duration
derive: self.departTime.difference(self.arrivalTime)

context Airport::departingFlights : Set(Flight)
init: Set{}
```

Context-ek

- Operation
  - `pre, post` – elő- és utófeltétel
    - `@pre` postfix
    - `result=` eredményt ad

```
context Passenger::getDestination(): Airport
post: result = self.flt.destination
```

  - body – lekérdező fv. eredménye

```
context Passenger::getDestination(): Airport
body: self.flt.destination
```

Egyéb

- `hasSent` operátor
  - utófeltételben, üzenetet küldünk
  - az eredmény lényegtelen

```
context Subject::hasChanged()
post: observer^update(5, 24)

post: observer^update(?: Integer, ?:Integer)
```

Egyéb

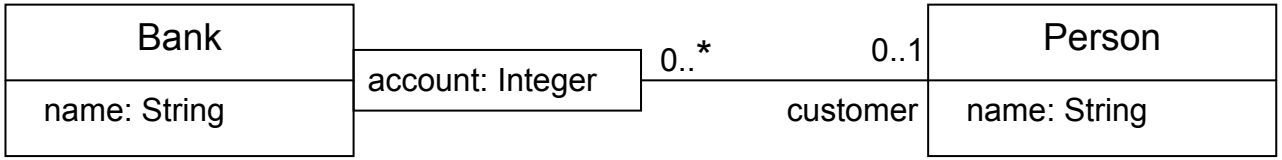
- Lokális változók
  - tetszőleges OCL kifejezésben

```
context Person
inv: if employer.form = CompanyForm::state_owned
        then Job.salary >= 10000
     else Job.salary = 5000
     endif
```

```
context Person
inv: let income : Integer = Job.salary
     in
        if employer.form = CompanyForm::state_owned then
            income >= 10000
        else income = 5000
        endif
```

Qualified association

- opcionális




```
context Bank
inv: self.customer[14527894].name = 'Macska Jancsi'
```

Precedenciák

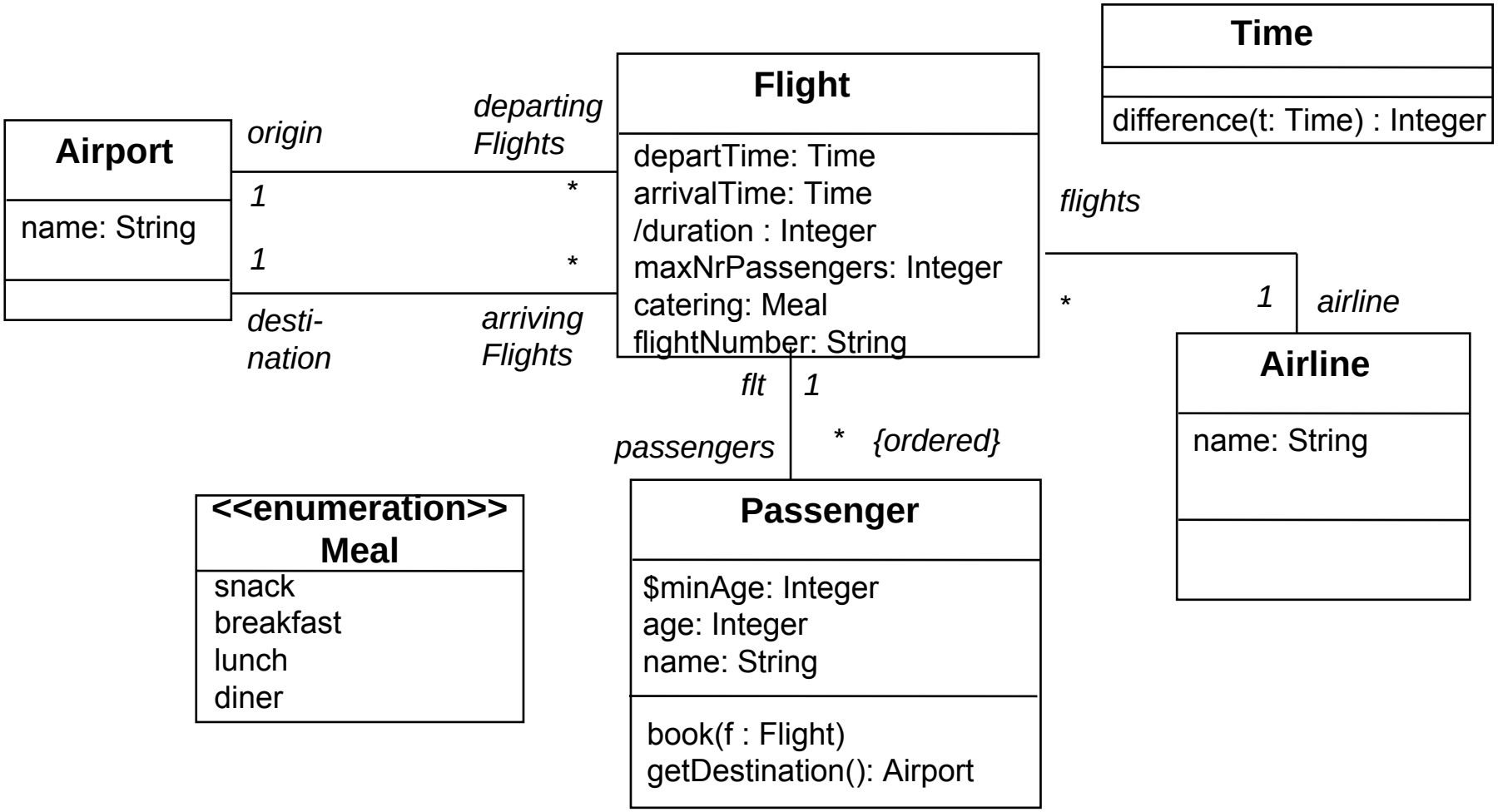
útvonal	::
időbeliség	@pre
referencia	., ->, ^, ^^
unáris	-, not
multiplikatív	*, /
additív	+, -
relációs	<, >, <=, >=, <>, =
logikai	and, or, xor
implikáció	implies

Egyéb

- Kommentek

```
context Passenger -- a sor végéig komment
/*  ez meg a zárásig
       több soron át */
```

UML – példa környezet



UML példa

- Max annyi utas lehet, amennyi a maxNrPassenger
- A járatszám az útvonalat jelöli.
- Ugyanazon útvonalon több járatszámú járat is van.
- Nincs csak induló és cél reptér