OCL
Object Constraint Language

Maestría en Ingeniería de Software

Agenda

- Model Driven Architecture (MDA)
- Unified Model Language (UML)
- **Object Constraint Language (OCL)**
- Patrones
- Conclusiones

Contenido

Object Constraint Language (OCL)

- Introducción a OCL
  - Motivaciones
  - Historia
  - Objetivos
- Conceptos básicos
  - Propiedades
  - Invariantes
  - Pre y post condiciones
  - Operaciones

Contenido

Object Constraint Language (OCL)

- Tipos
  - Tipos básicos
  - Tipos especiales
- Ejemplos
- OCL y metamodelo
- OCL y Perfiles UML
- OCL en MDA

Introducción a OCL

Niveles de madurez

N0 Sin especificación
N1 Especificación textual
N2 Texto con modelos
N3 Modelos con texto
N4 Modelos precisos
N5 Solamente modelos

Niveles de madurez

N0 No UML
N1 No UML
N2 Poco UML
N3 Mucho UML
N4 Mucho UML + OCL
N5 ----------

[Fuente: Kleppe and Warmer]

Introducción a OCL

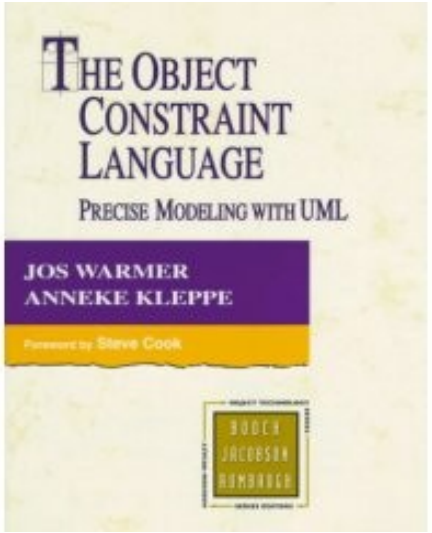
OCL

- Lenguaje de especificación formal que combina cálculo de predicados y teoría de conjuntos.
- El contexto sintáctico está determinado gráficamente.
- Su sintaxis es simple.
- Permite describir expresiones y restricciones en modelos y artefactos orientados a objetos
- Por ser un lenguaje de especificación no tiene efectos colaterales.

Introducción a OCL

Historia

- OCL fue desarrollado por IBM como lenguaje de modelado de negocio
- En UML 1.1 planteado como lenguaje de constraints
- En UML 2.0 puede ser usado para escribir cualquier expresión sobre los elementos en el diagrama.



Introducción a OCL

Por qué OCL?

- Para lograr especificaciones precisas y no ambiguas en modelos UML.
- Necesidad de describir restricciones adicionales sobre los objetos del modelo sin usar lenguaje natural.
- Lenguajes formales tradicionales son difíciles para modelador de negocios o de sistemas promedio. El OCL ha sido desarrollado para cubrir esa brecha.

Introducción a OCL

OCL puede ser usado en modelos UML y metamodelos

con diferentes propósitos:

- Para especificar invariantes sobre clases y tipos en modelos de clase
- Para especificar guardas en diagramas de estado
- Para describir precondiciones y postcondiciones sobre operaciones y métodos
- Para especificar invariantes en estereotipos
- …

OCL ha sido usado para definir la semántica de UML

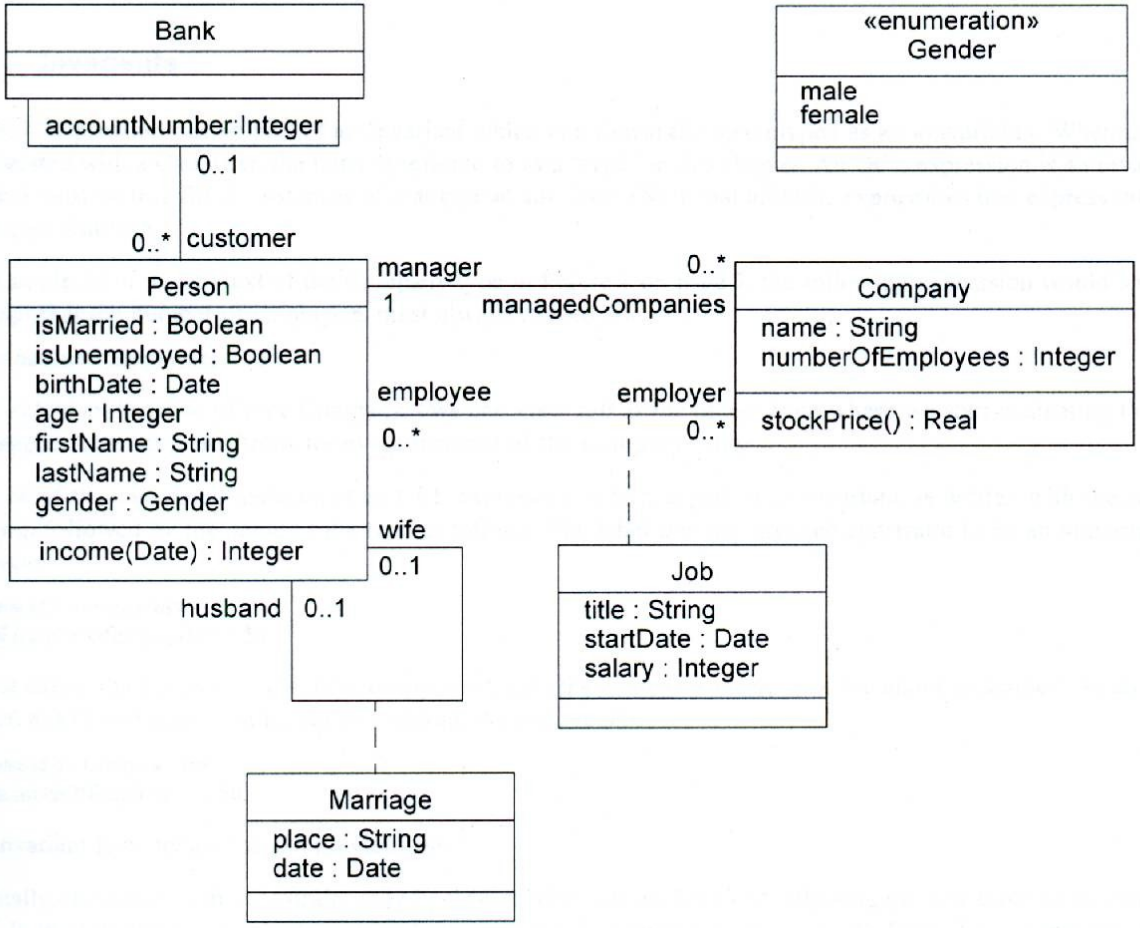
- Introducción a OCL
- **Conceptos básicos**
- Tipos
- Ejemplos
- OCL y metamodelo
- OCL y Perfiles UML
- OCL en MDA

Conceptos básicos

Gramática

```
<ocl-formula> -- truth values
```

Conceptos básicos



Contexto

Las expresiones OCL se escriben en el contexto de una instancia de un tipo específico

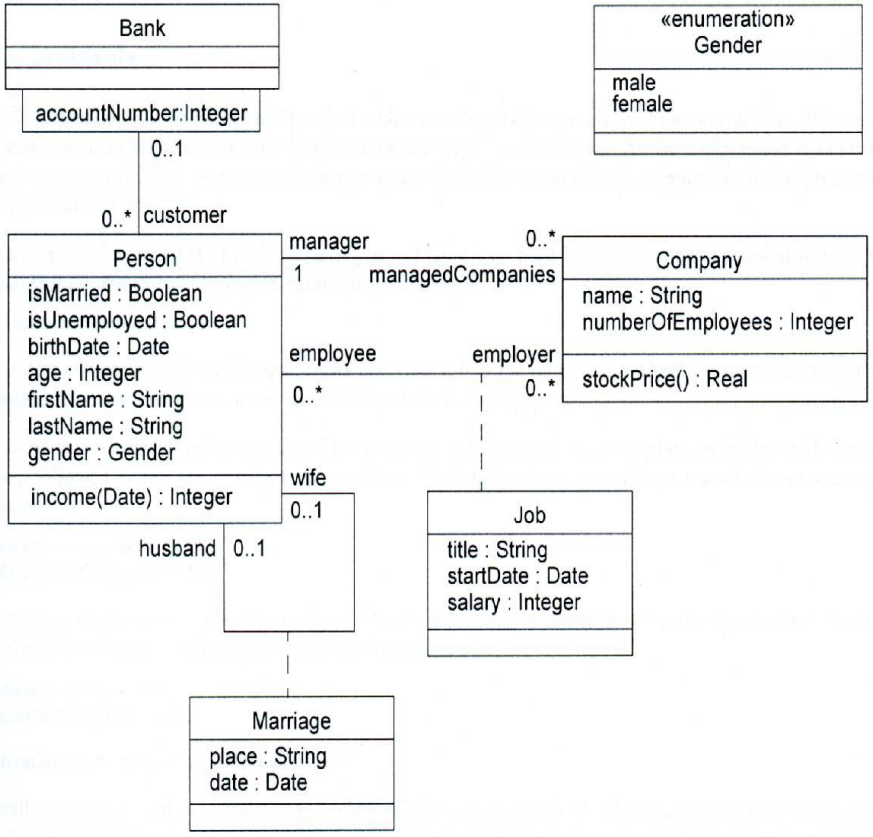
context Company

Self

La palabra reservada "self" se refiere a la instancia contextual

Conceptos básicos

Propiedades



Tipos válidos

Clasificadores de diagramas UML (interfaces, tipos de datos, componentes, nodos, casos de uso y subsistemas)

Propiedades

Tipos válidos tienen asociadas propiedades. (atributos, asociaciones finales, métodos y operaciones sin efectos colaterales)

```
context Company
        .........

        self.numberOfEmployees
```

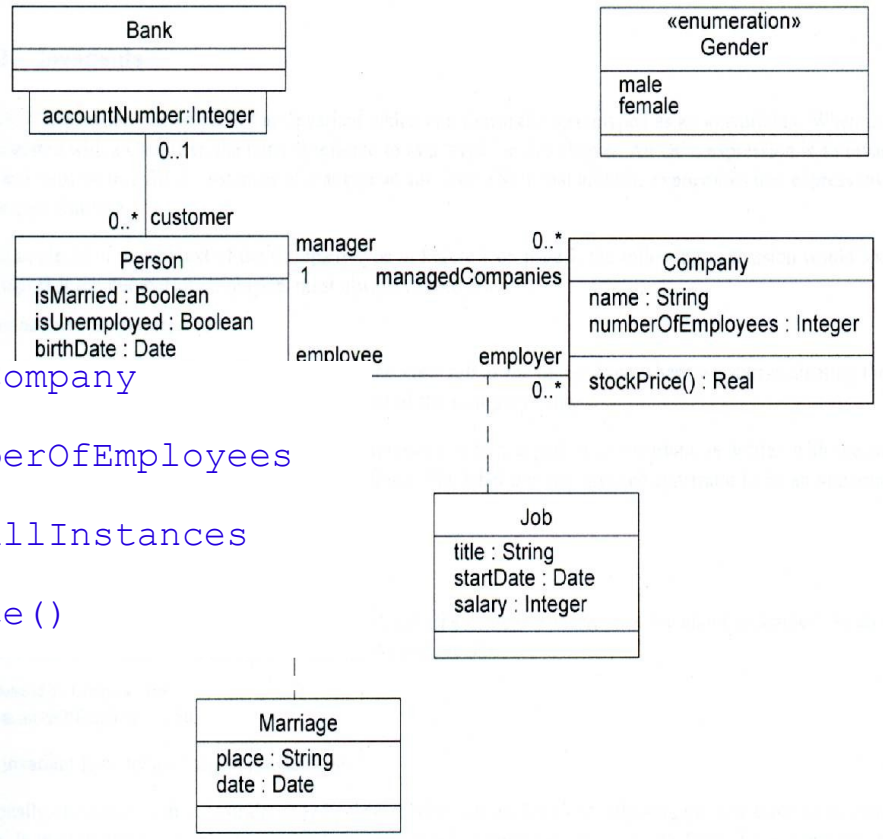
Conceptos básicos

- Expresiones formales básicas (atómicas):

  `<type>.<attribute>`

  Dependiendo del contexto, `<type>` puede ser:

  - objeto
  - clase
  - implícito




```
context Company
.........
self.numberOfEmployees
......
Company.allInstances
......
stockPrice()
```

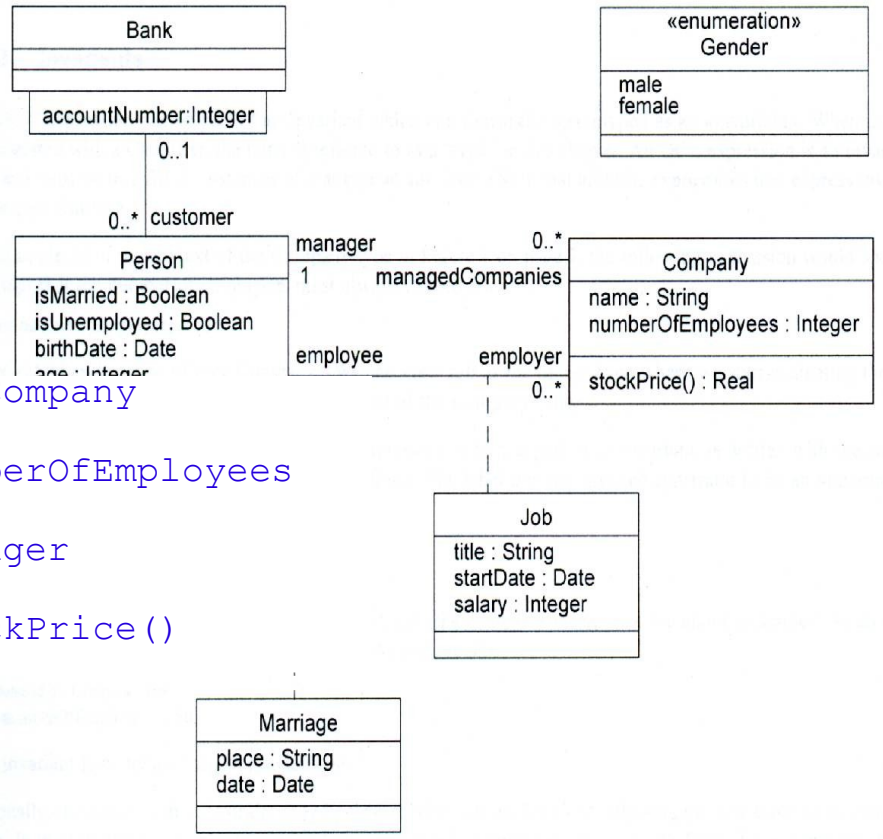
Conceptos básicos

- Expresiones formales básicas (atómicas):

  `<type>.<attribute>`

  Dependiendo del contexto, `<attribute>` puede ser:

  - atributo de clase
  - asociaciones finales (navegación)
  - operacion de clase




```
context Company
.........
self.numberOfEmployees
......
self.manager
......
self.stockPrice()
```

Conceptos básicos

- **Invariante**

  Expresión booleana asociada a un clasificador (relación, clase, interfaz, caso de uso) que es cierta en cada instante de ejecución del programa
- **Precondición**

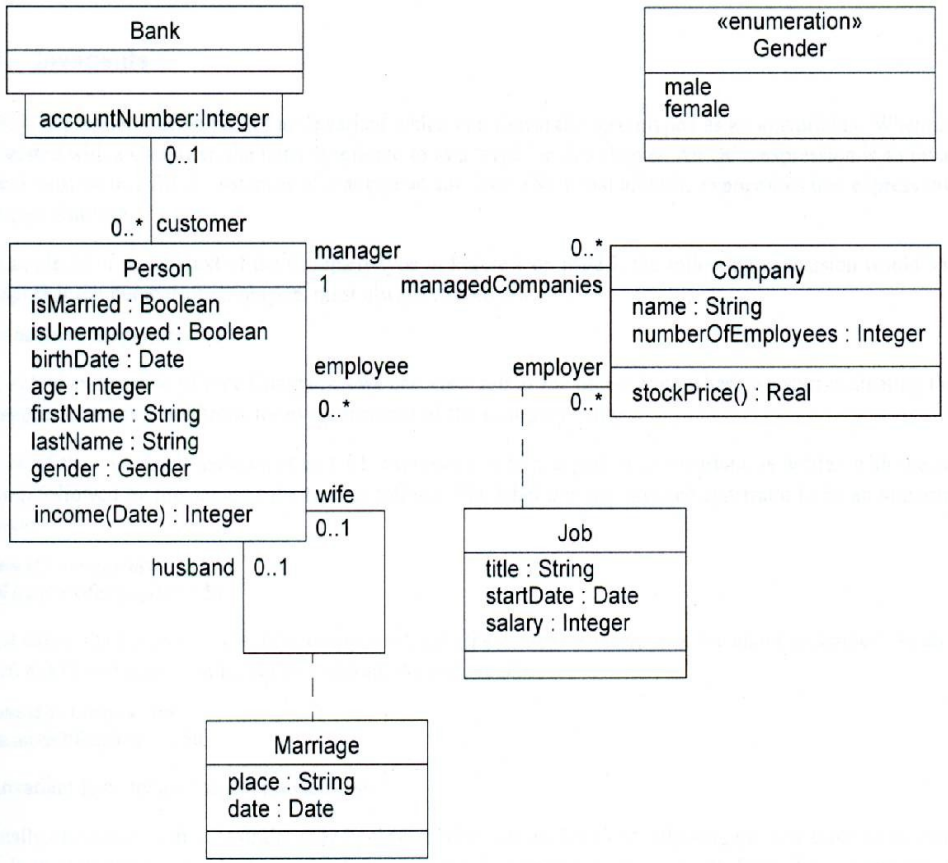
  Expresión booleana asociada a un método que es cierta en el instante en que se comienza a ejecutar
- **Postcondición**

  Expresión booleana asociada a un método que es cierta en el instante en que se termina de ejecutar

Conceptos básicos

Invariantes

```
context Company

inv:
self.numberOfEmployees>100
```



Conceptos básicos

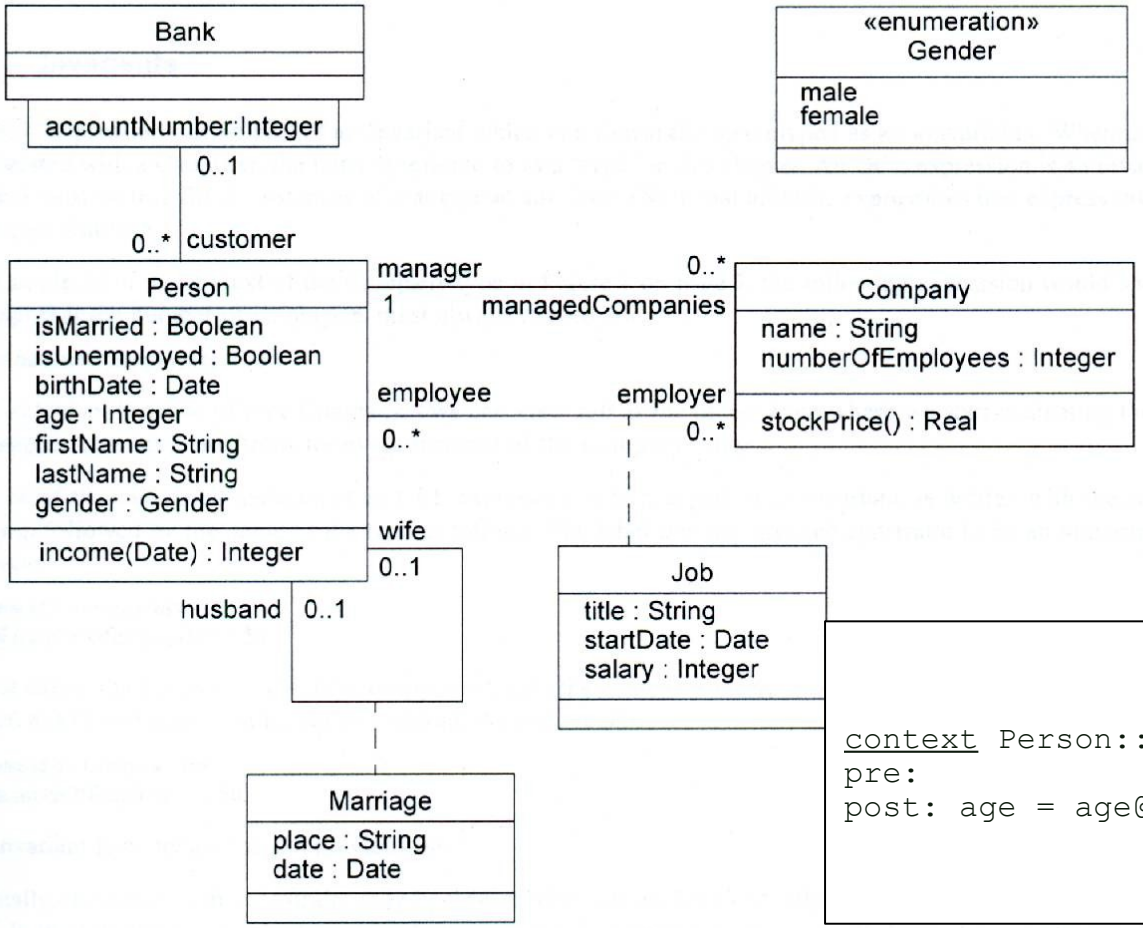
Pre y Post condiciones

Las pre-condiciones o post-condiciones se aplican tanto a Métodos como a Operaciones.

```
nombreDeTipo::nombreDeOperación(parametro1 : Tipo1, ... ): TipoDevolución

    pre :  parametero1 > …

    post:  resultado = ...
```

Conceptos básicos

Pre y Post condiciones




```
context Person::birthdayHappens()
pre:
post: age = age@pre+1
```

- Introducción a OCL
- Conceptos básicos
- **Tipos**
- Ejemplos
- OCL y metamodelo
- OCL y perfiles UML
- OCL en MDA

Tipos OCL

- Tipos predefinidos
  - Básicos (*Integer, Real, String, Boolean* y *Enumerate*)
  - Colecciones *(Collection, Set, Ordered-Set, Bag, Sequence)*
- Tipos definidos por el usuario
- Jerarquía de tipos

Un tipo *t* conforma a otro *t1* cuando una instancia de tipo *t* puede ser sustituida por una de tipo *t1:*

-Cada tipo conforma a sus supertipos.

-Si *t1* conforma a *t2*, y *t2* conforma a *t3*, luego *t1* conforma a *t3*.

Tipos básicos

Operaciones

Boolean

- b = (b2:Boolean) : Boolean
- b or (b2 : Boolean) : Boolean
- b xor (b2: Boolean) : Boolean
- b and (b2: Boolean) : Boolean
- not b : Boolean
- b implies (b2: Boolean) : Boolean

```
<ocl-formula> -- truth values
    ( <ocl-formula> and <ocl-formula> )
    ( <ocl-formula> or <ocl-formula> )
    ( <ocl-formula> implies <ocl-formula> )
    ( <ocl-formula> xor <ocl-formula> )
    ( not <ocl-formula> )
```

Tipos básicos

Operaciones

Real

- r = (r2 :Real) : Boolean
- r < > (r2 :Real) : Boolean
- r + (r1: Real) : Real
- r – (r1 :Real) : Real
- r * (r1 :Real) :Real
- r / (r1:Real) : Real
- r. abs : Real

Tipos básicos

Operaciones

Integer

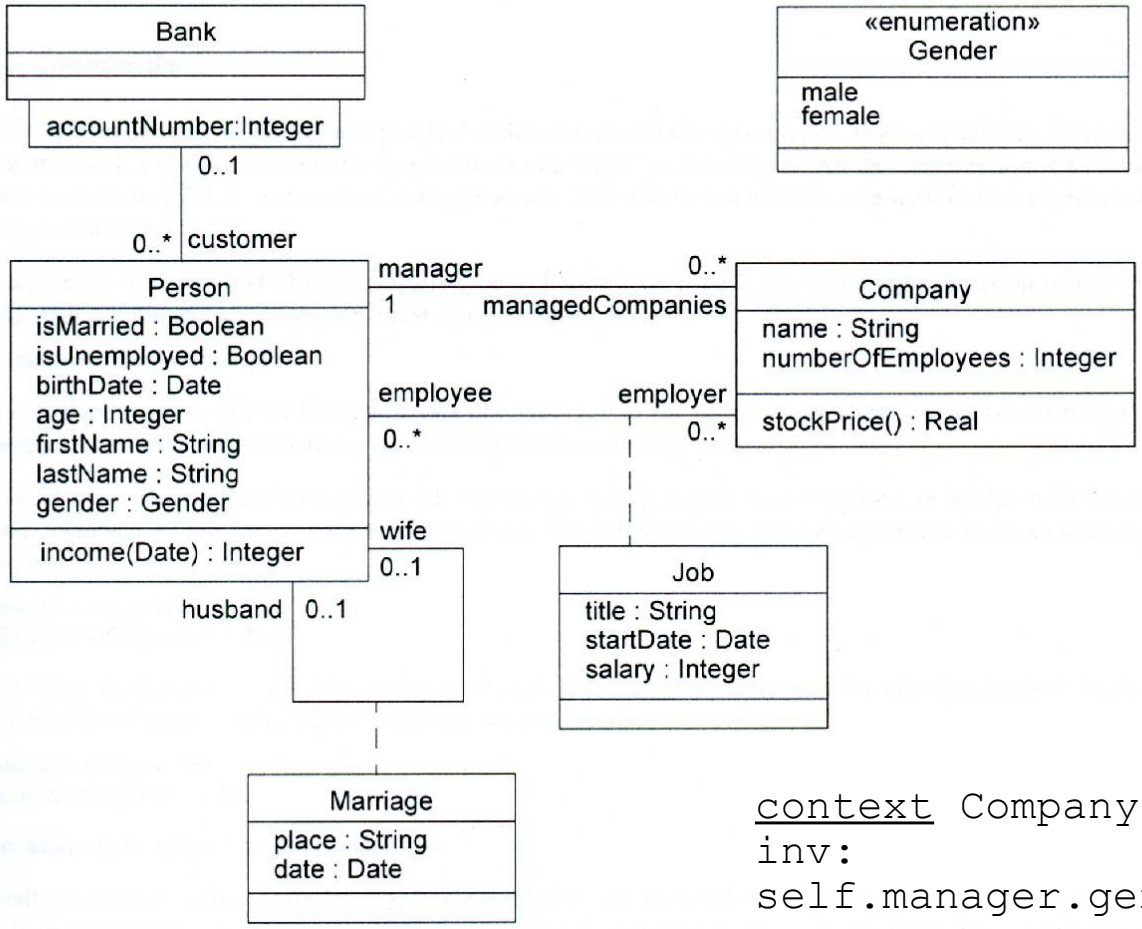
- i = (i2 : Integer) : Boolean
- i + (i2: Integer) : Integer
- i – (i2: Integer) : Integer
- i * (i2: Integer) : Integer
- i / (i2:Integer) : Real
- i.abs : Integer
- i.div (i2: Integer) : Integer

String

- string = (string2 : String) : Boolean
- string.size : Integer
- string.concat (string2: String): String
- string.toUpper : String
- string.toLower : String

Tipos básicos

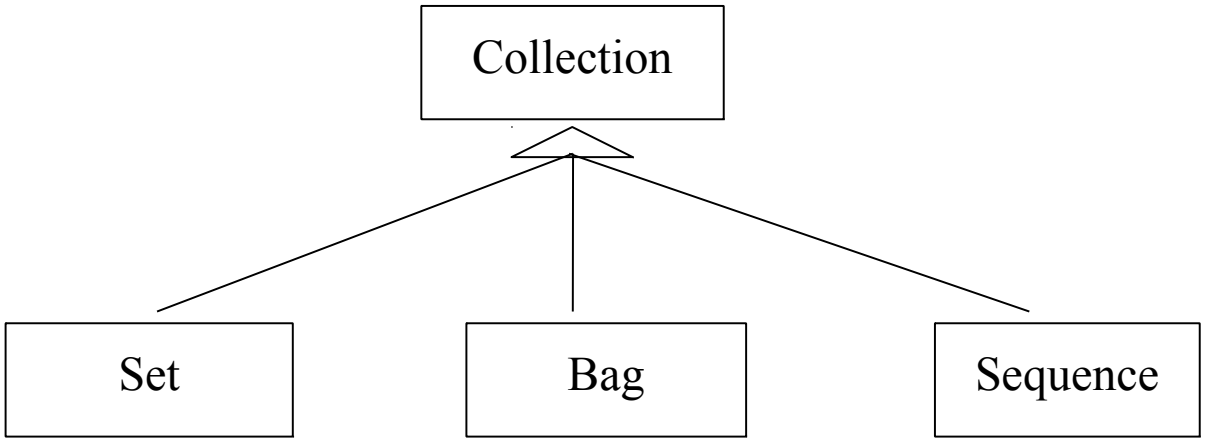
Enumeration




```
context Company
inv:
self.manager.gender=#female
```

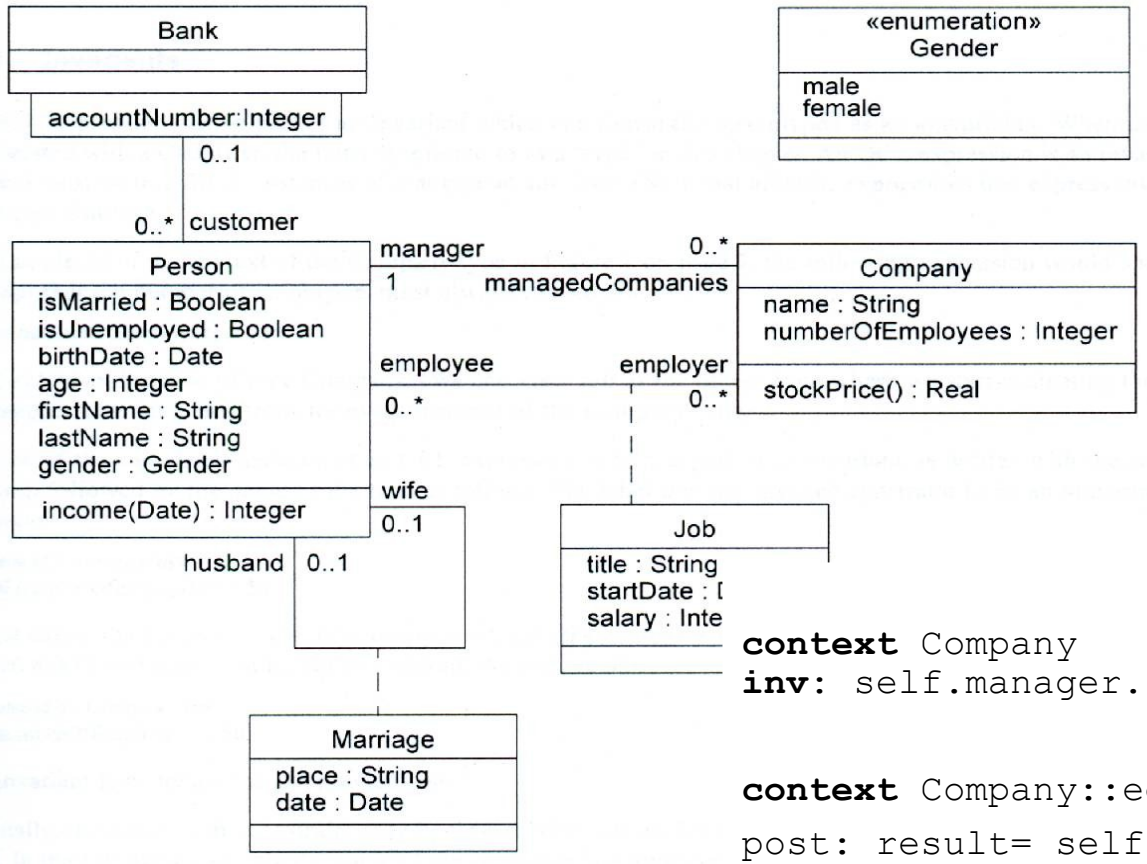
Tipos Especiales

Colecciones



Tipos especiales

Combinación de propiedades




```
context Company
inv: self.manager.isUnemployed = false

context Company::edades():Set(Integer)
post: result= self.employee.age
```

Tipos Especiales

Tipos predefinidos: Colecciones

Notación:

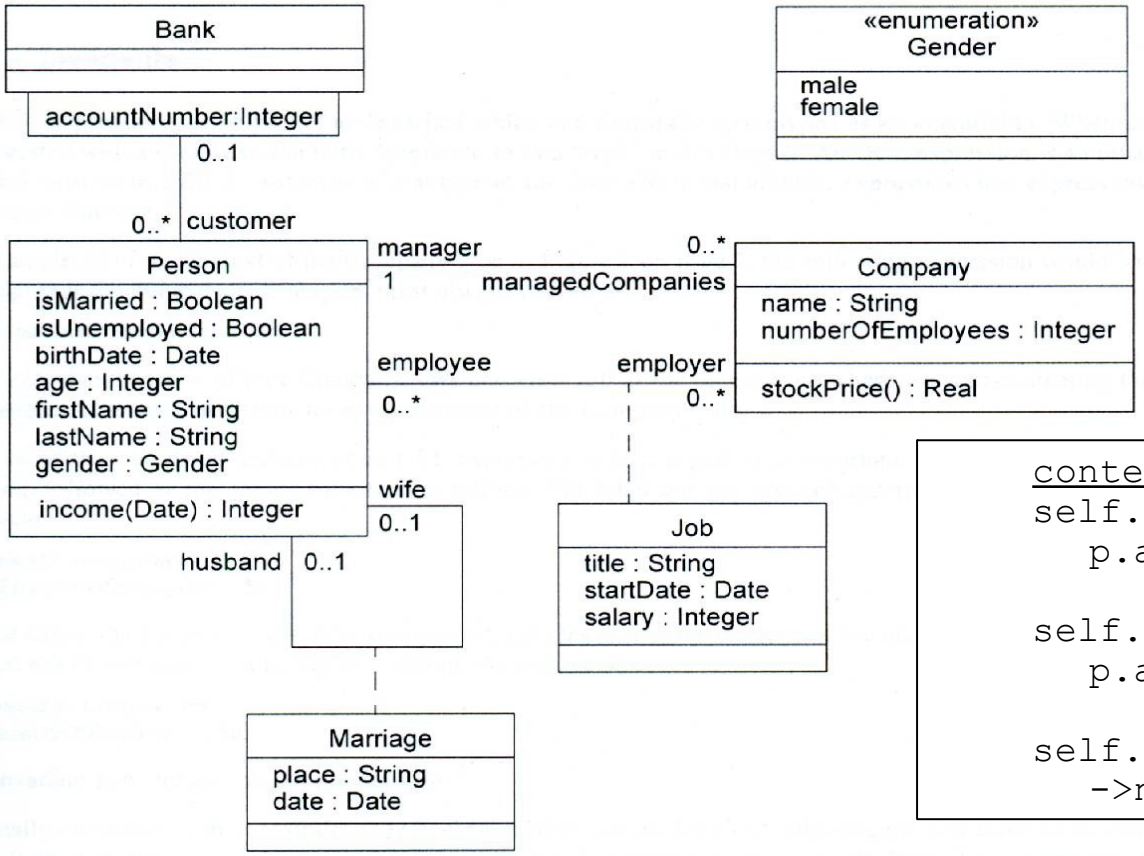
<collection>→ <operation>

Operaciones:

select, collect, forAll, exists, etc

Tipos Especiales

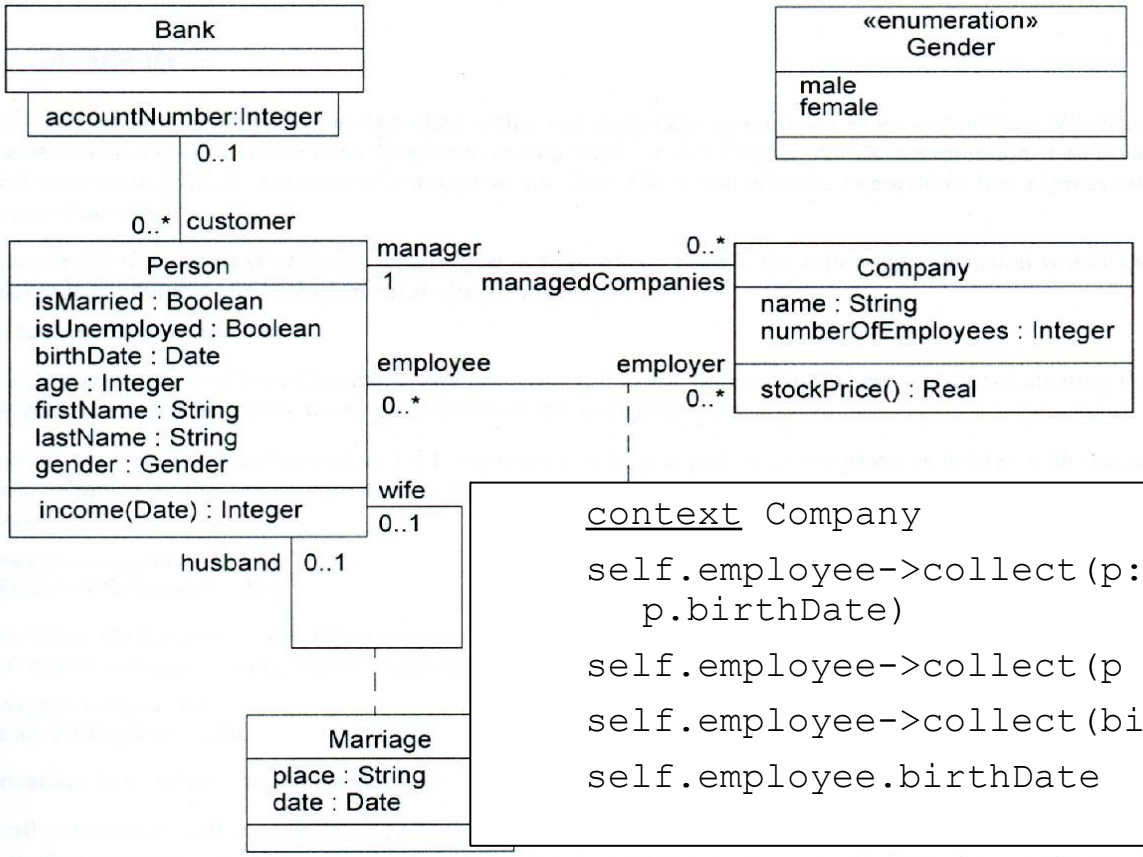
Select




```
context Company
self.employee->select(p:Person|
   p.age>50)->notEmpty()

self.employee->select(p |
   p.age>50)->notEmpty()

self.employee->select(age>50)
   ->notEmpty()
```

Tipos especiales

Collect




```
context Company
self.employee->collect(p:Person|
   p.birthDate)
self.employee->collect(p | p.birthDate)
self.employee->collect(birthDate)
self.employee.birthDate
```

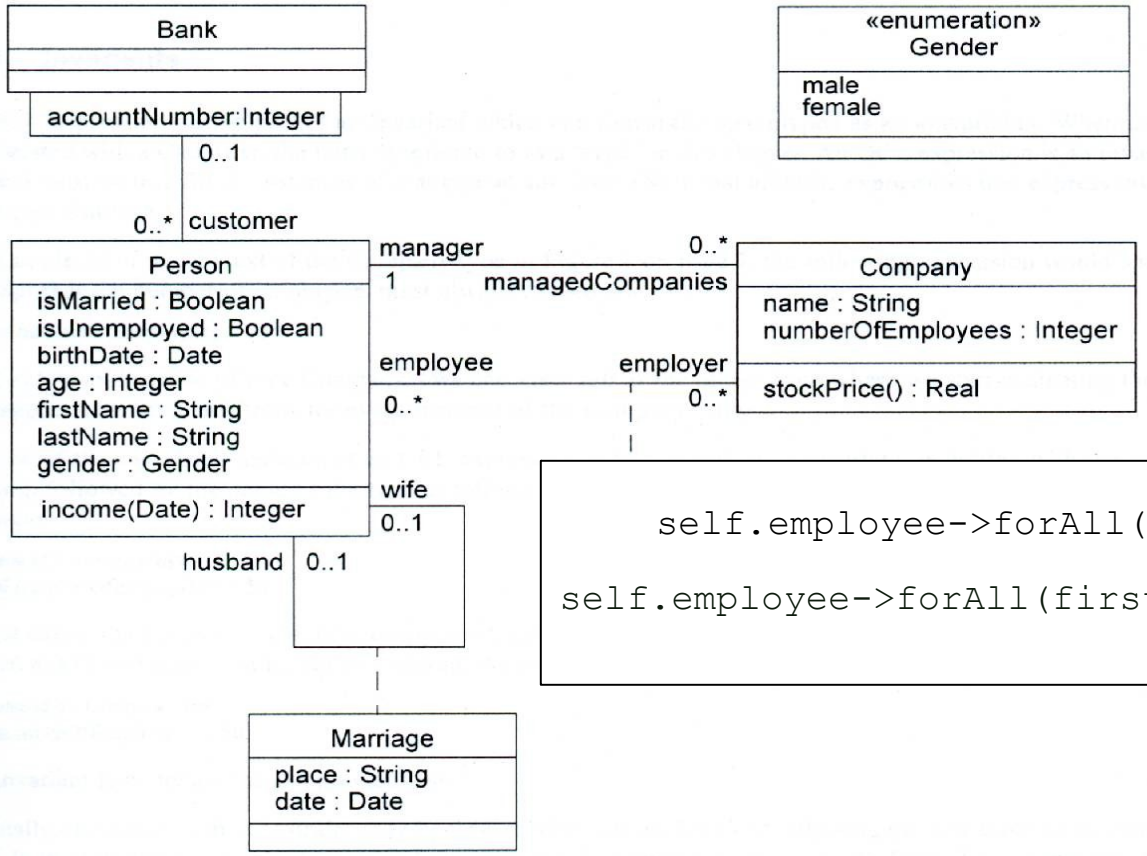
Tipos especiales

Cuantificadores

```
<ocl-term>->forAll( <variable | <ocl-formula> )
<ocl-term>->exists( <variable | <ocl-formula> )
```

Tipos especiales

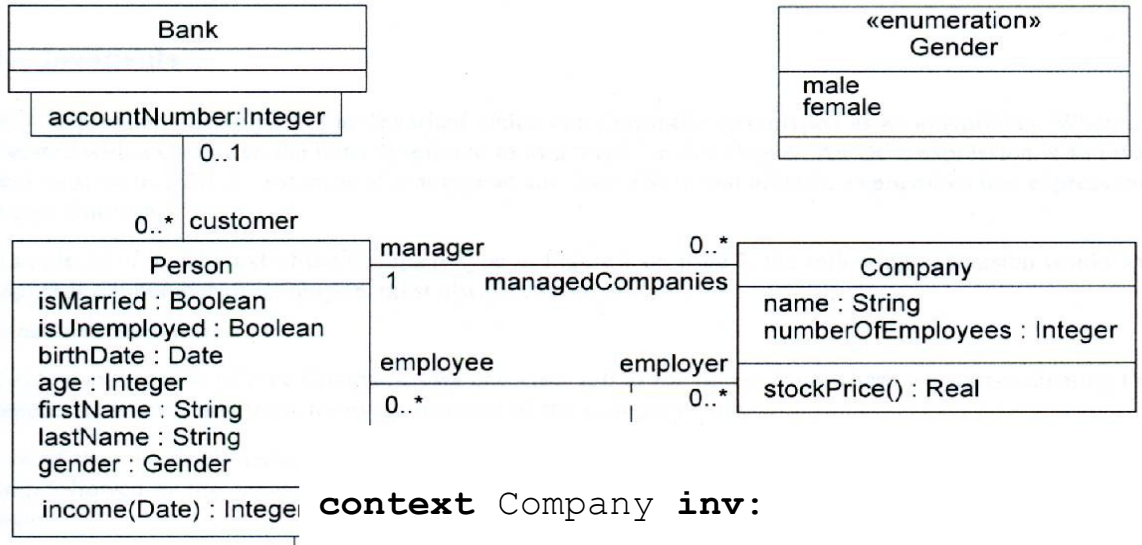
Cuantificadores




```
self.employee->forAll(p | p.age > 18)

self.employee->forAll(firstName='Jack')
```

Tipos especiales

Cuantificadores




```
context Company inv:
self.employee->exists( forename = 'Jack' )
context Company inv:
self.employee->exists( p | p.forename = 'Jack' )
context Company inv:
self.employee->exists( p : Person | p.forename = 'Jack' )
```

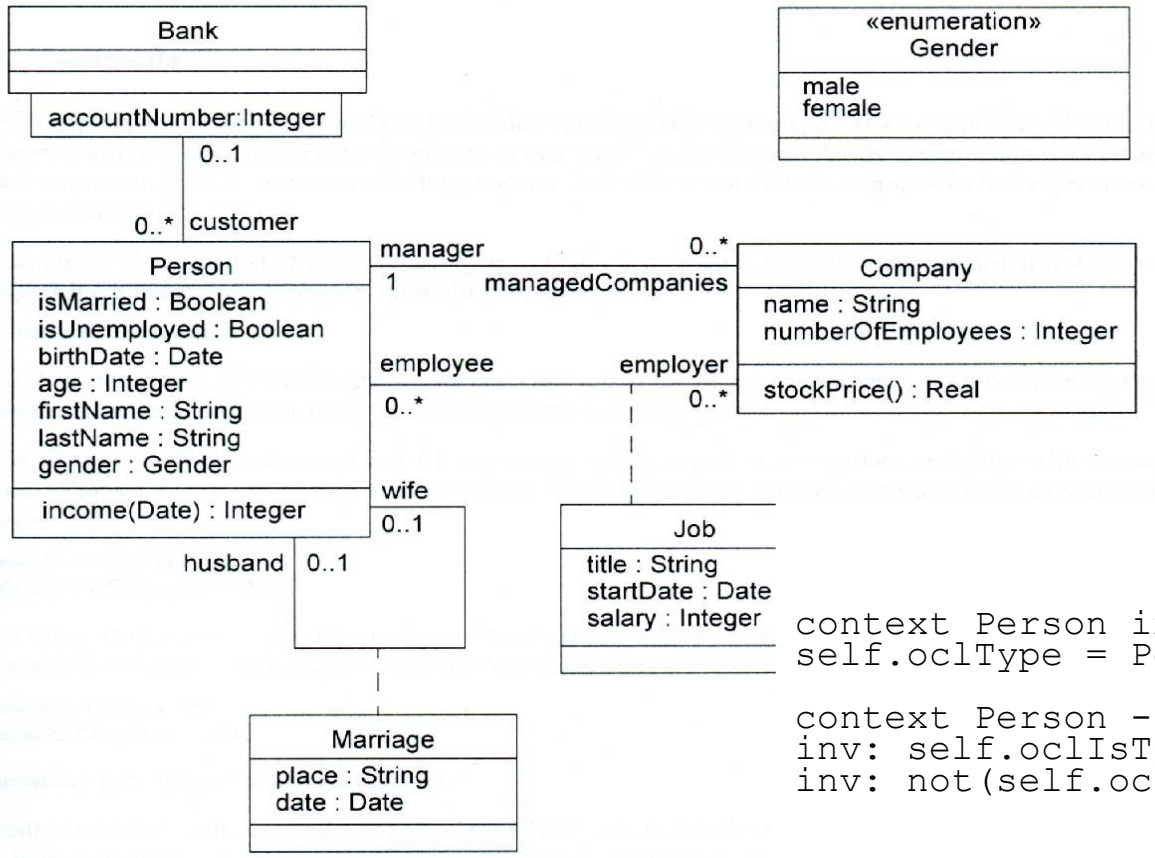
Tipos especiales

Operaciones

```
collection->size : Integer

collection->includes(objet : OclAny) : Boolean

collection->count(objet : OclAny) : Integer

collection->includesAll(c : Collection(T)) : Boolean

collection->isEmpty : Boolean

collection->notEmpty : Boolean
```

Tipos especiales

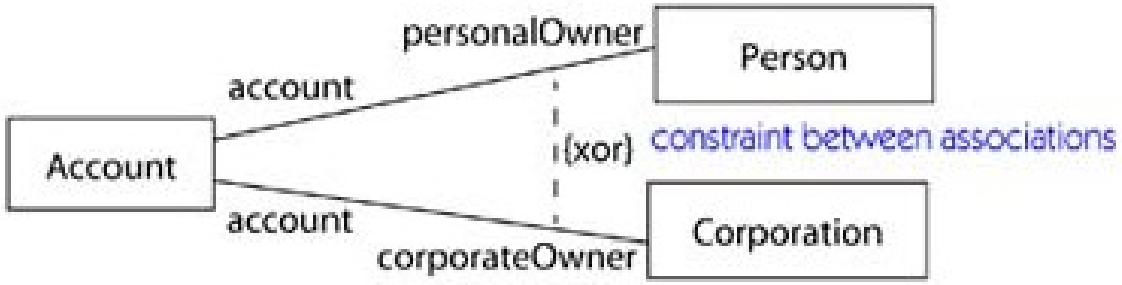
“oclType” y “oclIsTypeOf”

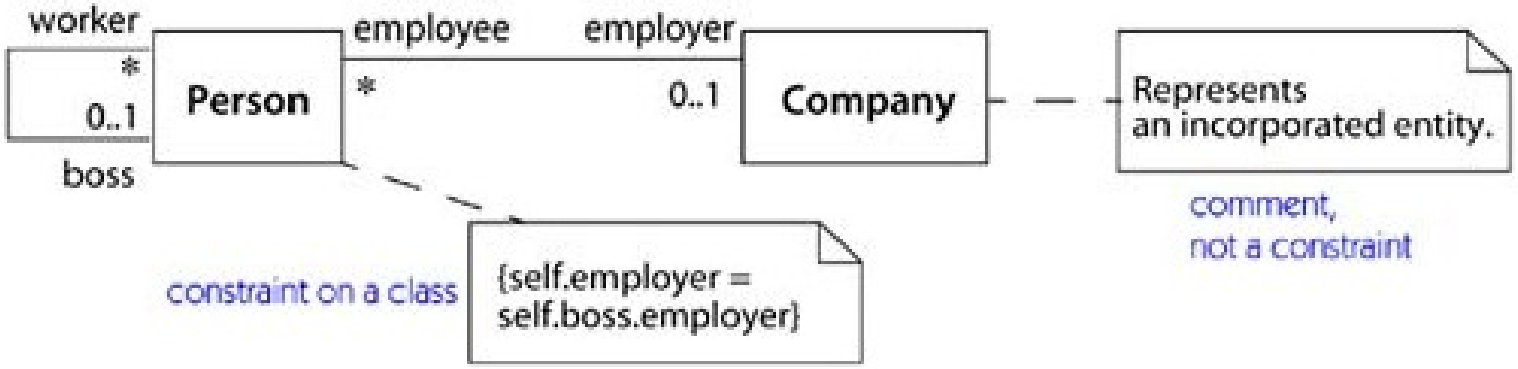



```
context Person inv: -- oclType
self.oclType = Person

context Person -- oclIsTypeOf
inv: self.oclIsTypeOf(Person)
inv: not(self.oclIsTypeOf(Company))
```

- Introducción a OCL
- Conceptos básicos
- Tipos
- **Ejemplos**
- OCL y metamodelo
- OCL y perfiles UML
- OCL en MDA

Ejemplos


personalOwner
Person
account
Account
{xor}
constraint between associations
account
corporateOwner
Corporation


worker
employee
employer
*
0..1
Person
*
0..1
Company
Represents
an incorporated entity.
boss
comment,
not a constraint
constraint on a class
{self.employer =
self.boss.employer}

Ejemplos

[1] Married people are of age >= 18

```
context person inv:
self.wife->notEmpty implies
self.wife.age >= 18 and
self.husband->notEmpty
implies self.husband.age >=
18
```

[2] a company has at most 50 employees

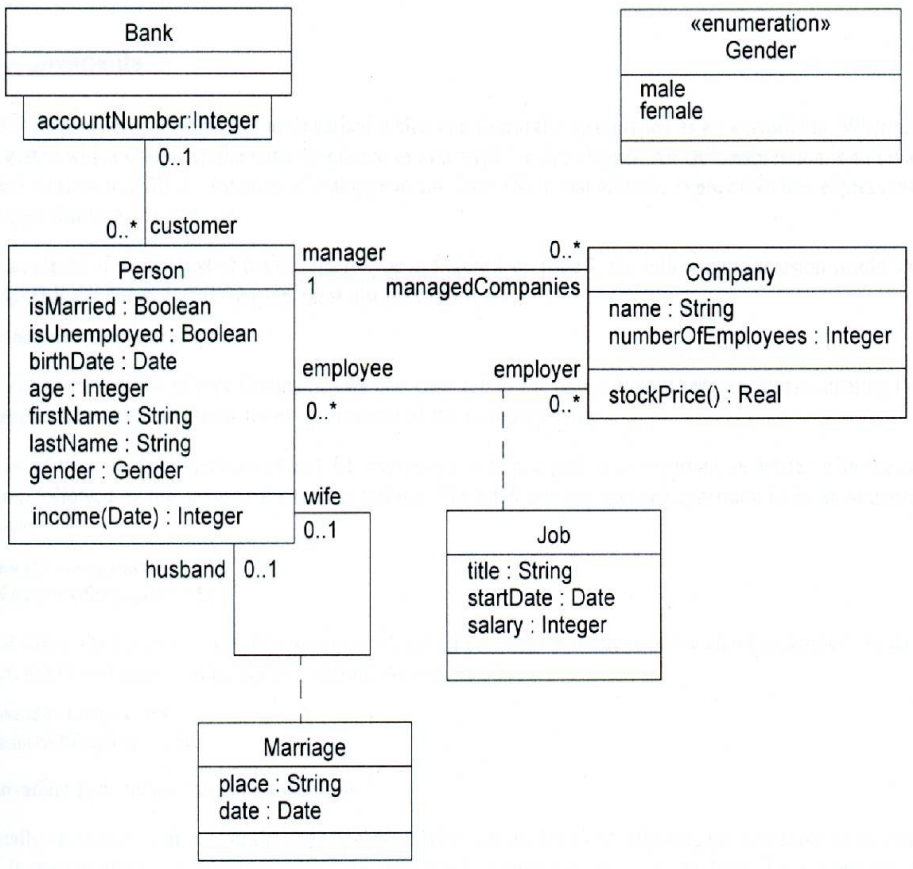
```
context Company inv:
self.employee->size <= 50
```

[3] A marriage is between a female (wife) and male (husband)

```
context marriage inv:
self.wife.gender = #female
and
self.husband.gender = #male
```

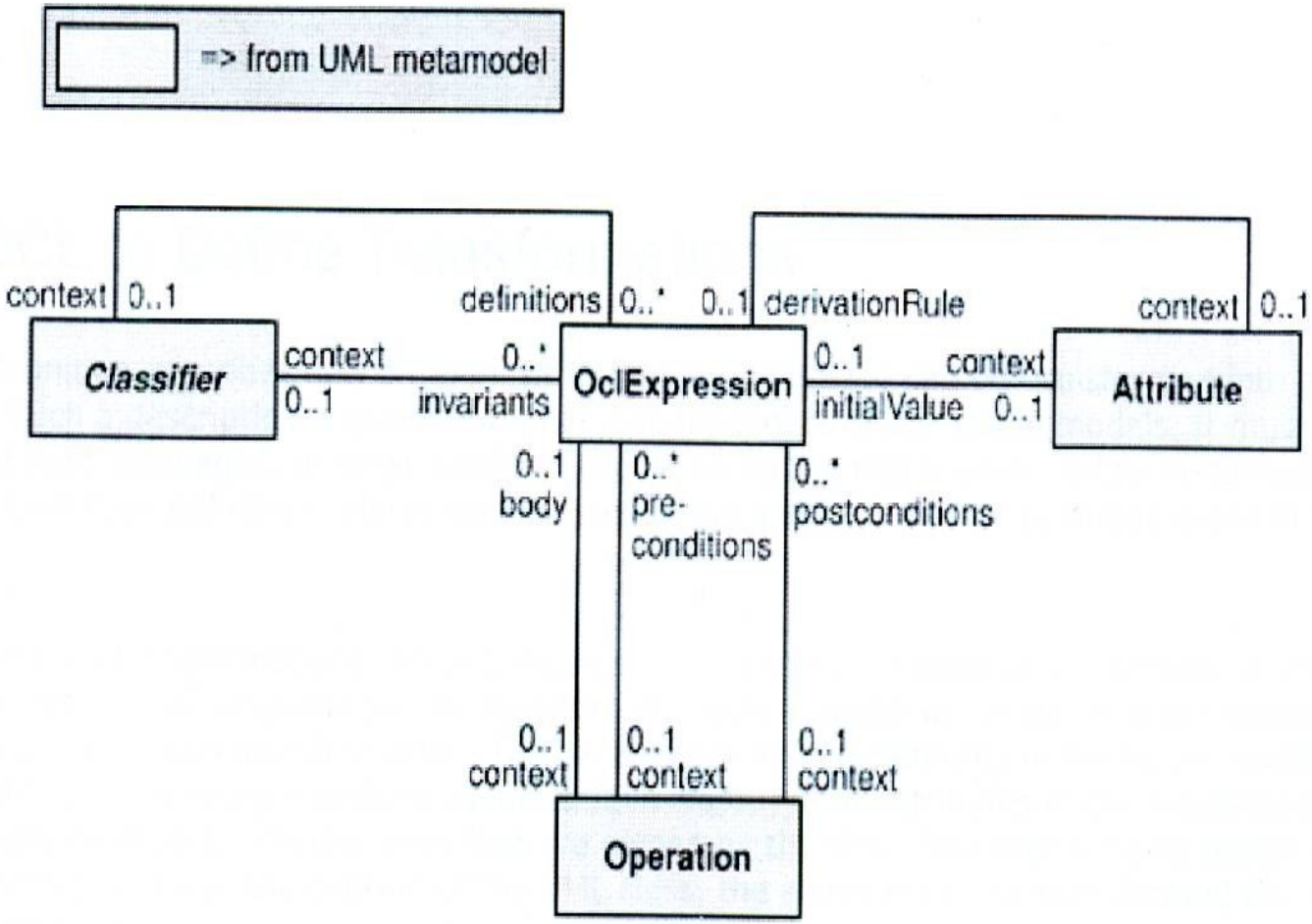
[4] A person can not both have a wife and a husband

```
context person inv:
not ((self.wife->size=1) and
(self.husband->size=1))
```



- Introducción a OCL
- Conceptos básicos
- Tipos
- Ejemplos
- **OCL y metamodelo**
- OCL y perfiles UML
- OCL en MDA

OCL y metamodelo



OCL y metamodelo

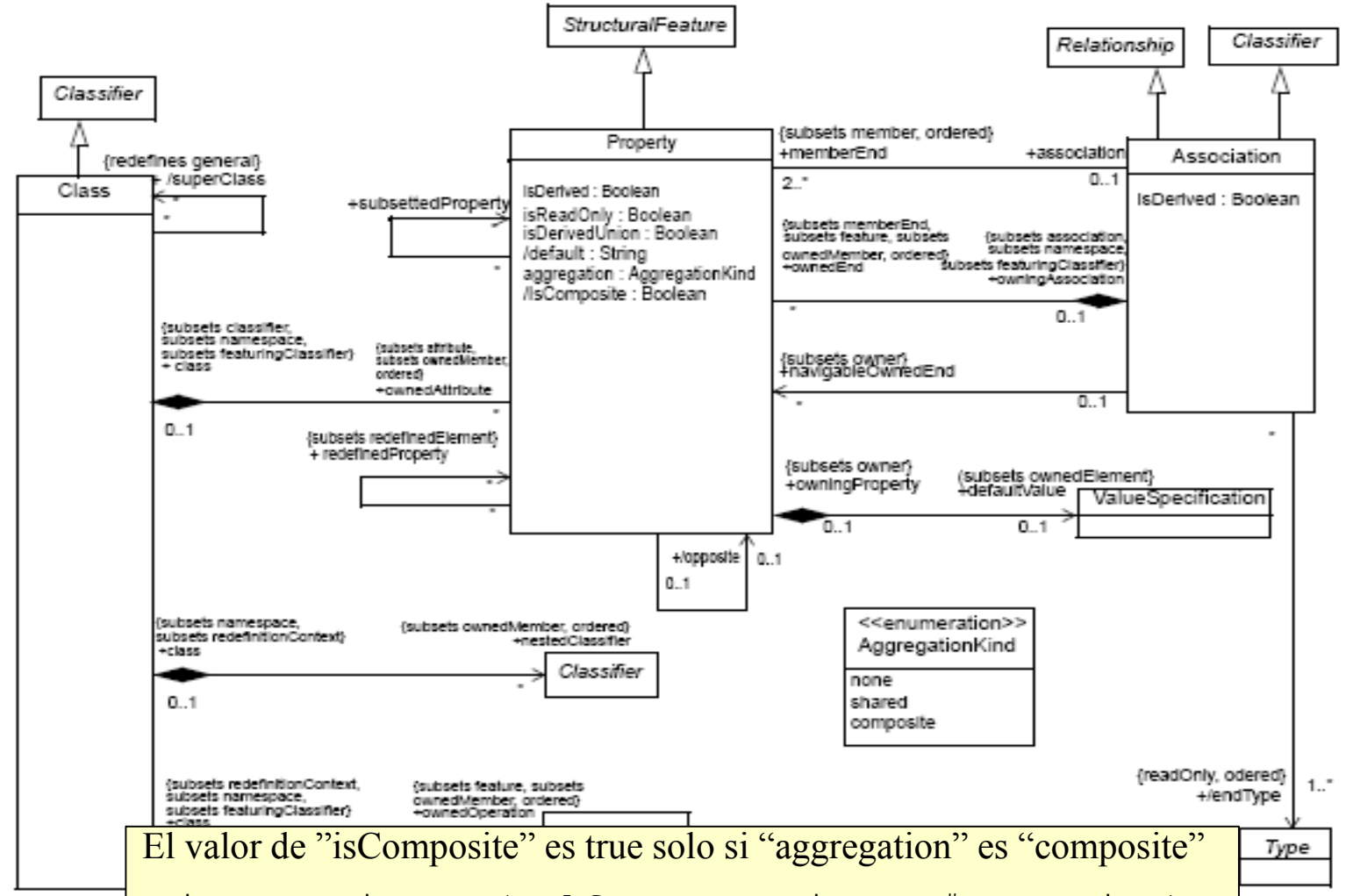
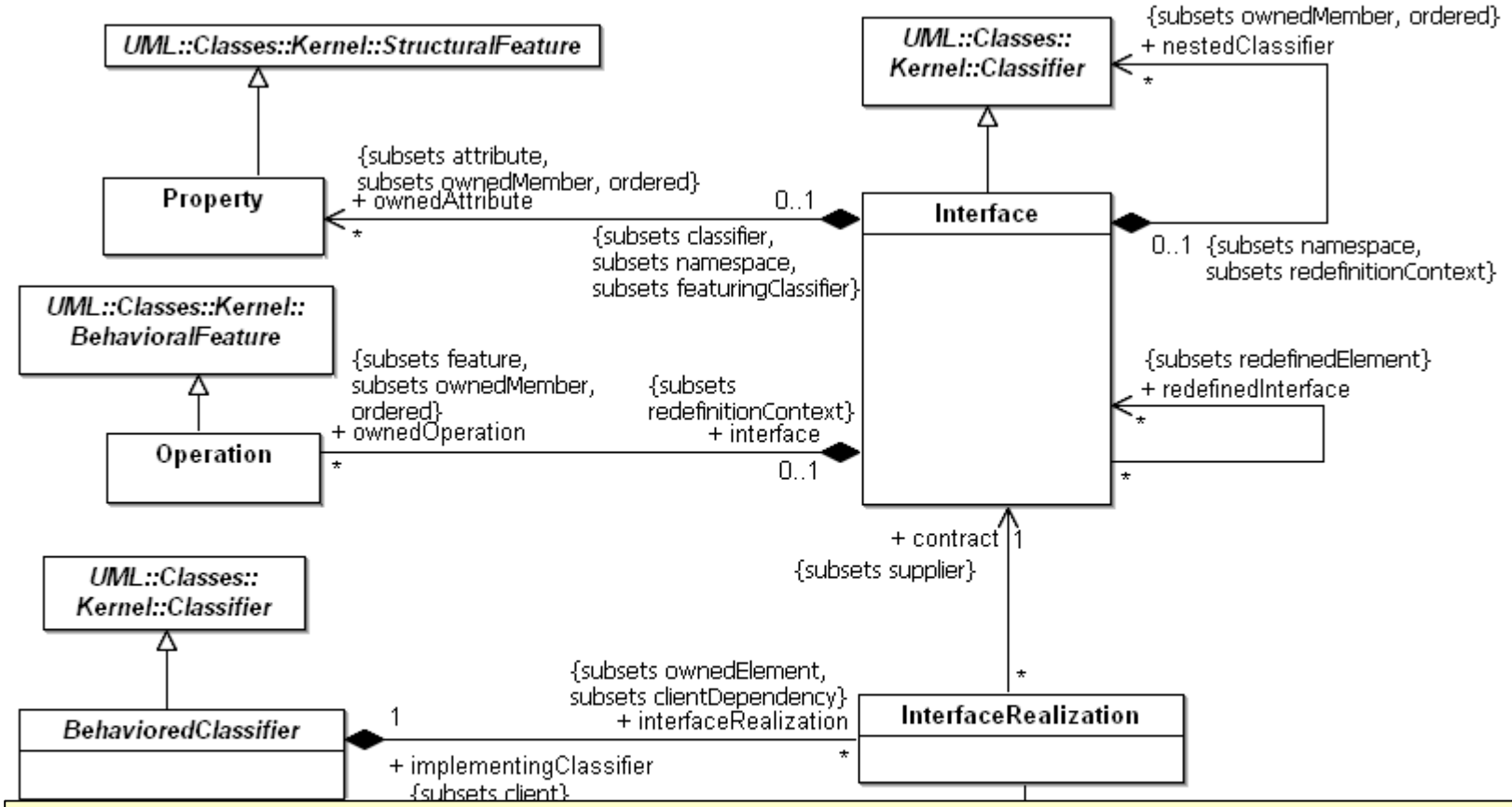


Figure 7.12

El valor de ”isComposite” es true solo si “aggregation” es “composite”

```
isComposite = (self.aggregation = #composite)
```

OCL y metamodelo

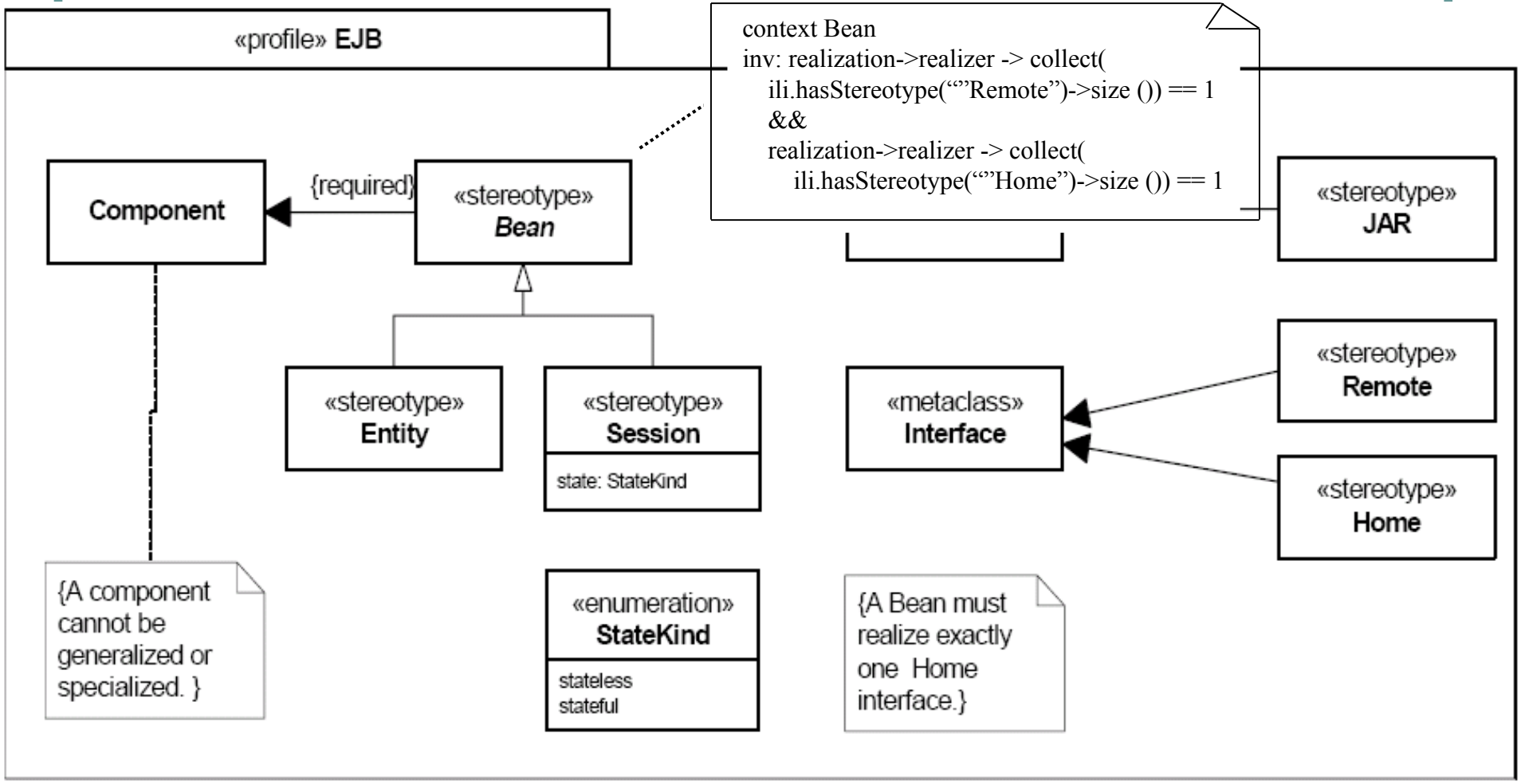


Una interface solo puede contener operaciones

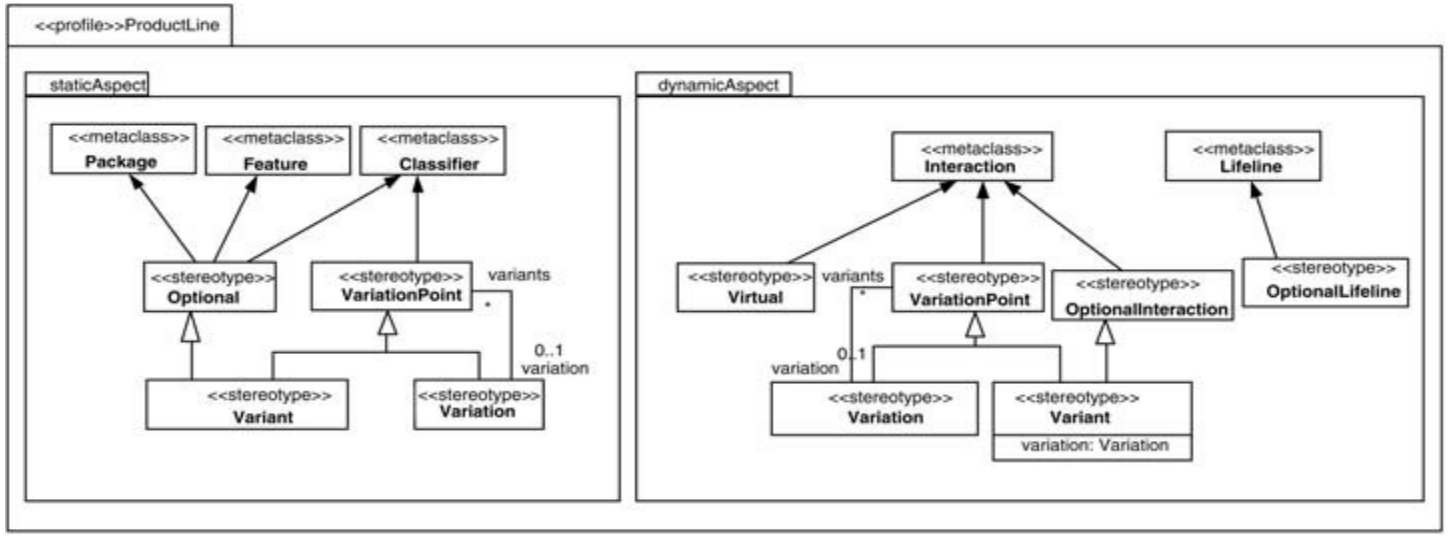
```
self.allFeatures->forAll(f | f.oclIsKindOf(Operation) or
          f.oclIsKindOf(Reception))
```

- Introducción a OCL
- Conceptos básicos
- Tipos
- Ejemplos
- OCL y metamodelo
- **OCL y perfiles UML**
- OCL en MDA

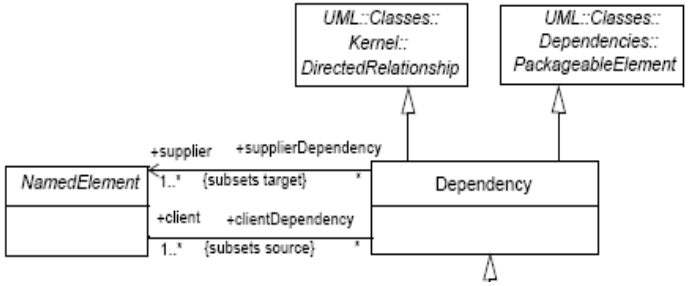
OCL y Perfiles UML



OCL y Perfiles UML




```
context Dependency
inv:
self.supplier->exists(S:ModelElement
| S.isStereotyped ('optional'))
implies
self.client->forAll( C:ModelElement |
C.isStereotyped('optional') )
```



- Introducción a OCL
- Conceptos básicos
- Tipos
- Ejemplos
- OCL y metamodelo
- OCL y perfiles UML
- **OCL en MDA**

OCL en MDA

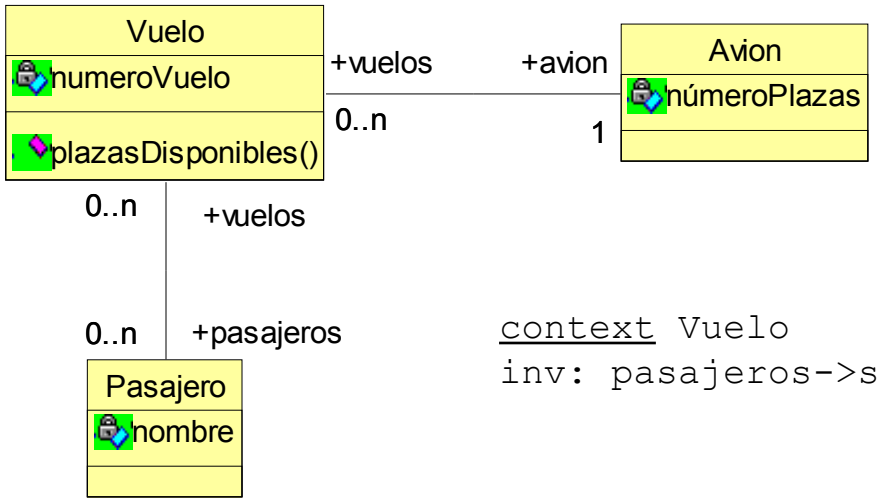
OCL es utilizado en MDA de 3 formas:

- Elaborar modelos mas precisos
- Definir lenguajes de modelado
- Especificación de semántica de Perfiles UML
- Definir transformaciones

OCL en MDA

Modelos mas precisos

- Existen limitaciones para crear modelos precisos y completos.



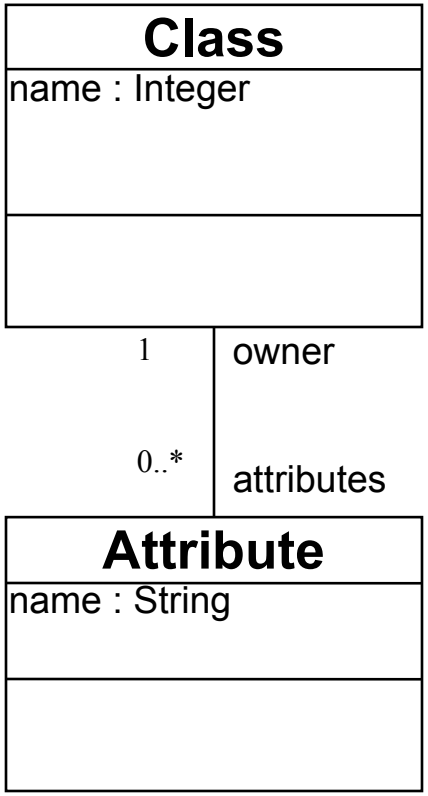

```
context Vuelo
inv: pasajeros->size()<=avion.númeroPlazas
```

OCL en MDA

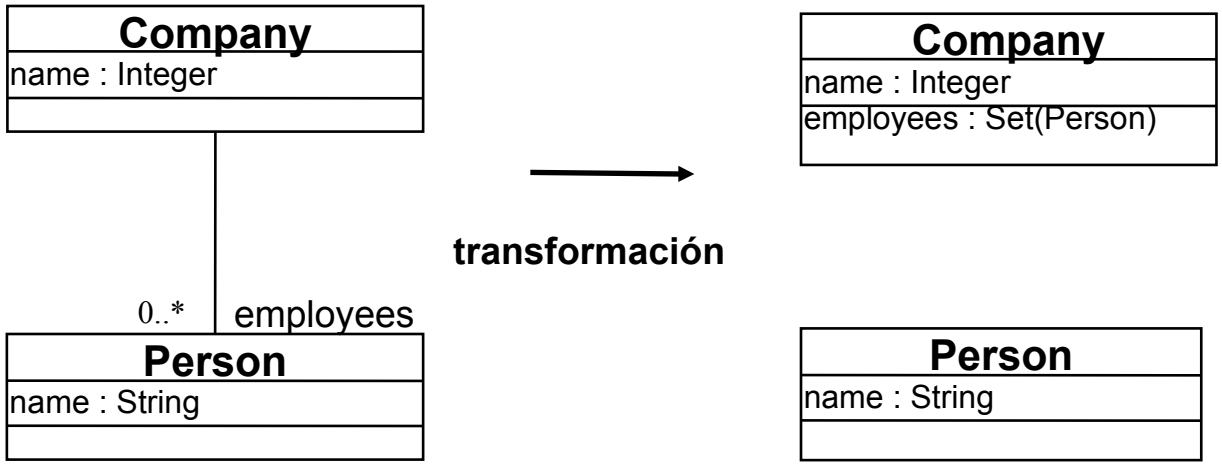
Definición de lenguajes

UML metamodel

```
Context Class inv:
attributes->isUnique(name)
```



OCL en MDA




```
Transformation ManyAssociationToAttribute  (UML, UML) {
source ae  : UML::AssociationEnd ;
target   att  : UML::Attribute;
source condition
        ae.multiplicity = MultiplicityKind::many;
target condition
        att.visibility = VisibilityKind::public  and  att.type.isTypeOf(Set);
mapping
        ae.name <~> att.name;
        ae.type   <~> att.type.elementType;}
```

OCL en MDA

OCL en MOF-QTV

Parte esencial en el lenguaje de transformación QTV

OCL en MDA

Semántica en Perfiles UML

Reglas en un perfil UML de java:

```
context Class inv:
generalizations->size() <= 1
```

.