

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
DE  
EDUCACIÓN  
SUPERIOR  
AUTÓNOMA  
DE  
MÉXICO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.v

## 5. OBJS. Y PROPIEDADES



- Las expresiones OCL pueden hacer referencia a las *clases, interfaces, asociaciones y tipos de datos del modelo UML* que hace de contexto.
- Además, también se puede hacer referencia a las *propiedades* que define la *clase, en concreto a los atributos, a los métodos sin efectos laterales y a los extremos remotos de las asociaciones*.
- *Para acceder a las propiedades se usa la notación punto, con el contexto* (referido por self o el nombre que se le haya dado), *seguido de un punto y el nombre de la propiedad.*

Prof. Ana M<sup>a</sup> Roldán

36

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
DE  
EDUCACIÓN  
SUPERIOR  
AUTÓNOMA  
DE  
MÉXICO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.v

## 5. OBJS. Y PROPIEDADES

### Atributos

**EJEMPLO:** para acceder a la edad de una persona se usa el atributo edad de la clase Persona:



- En este caso, en el que **la multiplicidad** del atributo es **1**, la expresión **self.edad** es del tipo del que se ha definido el atributo, en este caso **Integer**.
- **Si la multiplicidad del atributo es mayor que 1**, el resultado devuelto es una colección de valores del tipo del que se ha declarado el atributo.

```
context Persona inv:  
    self.edad > 0
```

Prof. Ana M<sup>a</sup> Roldán

37

DEPARTAMENTO DE  
Tecnologías de la  
Información  ETSI  
Ingeniería de Sistemas  
y Computadoras

[1. Objetivos.](#) [2. Introducción.](#) [3 .OCL](#)  
[4. Tipos y Operaciones](#) . [5. Obj y Propiedades.](#) [6. Colecciones.](#)  
[Referencias.v](#)

## 5. OBJS. Y PROPIEDADES

### Operaciones

- El acceso a las operaciones tiene una sintaxis similar al acceso a los atributos pero se añade entre paréntesis la lista de parámetros de entrada.

Ejemplo1:

```
unaPersona.sueldo(unaFecha)
```

Prof. Ana M<sup>a</sup> Roldán

38

DEPARTAMENTO DE  
Tecnologías de la  
Información  ETSI  
Ingeniería de Sistemas  
y Computadoras

[1. Objetivos.](#) [2. Introducción.](#) [3 .OCL](#)  
[4. Tipos y Operaciones](#) . [5. Obj y Propiedades.](#) [6. Colecciones.](#)  
[Referencias.v](#)

## 5. OBJS. Y PROPIEDADES

### Operaciones

- El acceso a las operaciones tiene una sintaxis similar al acceso a los atributos pero se añade entre paréntesis la lista de parámetros de entrada

Ejemplo2: Se puede definir una operación mediante la definición de su postcondición:

```
context Persona::sueldo(f: Fecha) : Integer
post : result = age * 1000
```



es del mismo tipo que el que devuelve la operación

Prof. Ana M<sup>a</sup> Roldán

39

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
DE  
EDIMBURGO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. **Objes y  
Propiedades.** 6. Colecciones.  
Referencias.

## 5. OBJS. Y PROPIEDADES

### Extremos de asociación y navegación

- Se puede navegar desde un objeto específico a través de las asociaciones definidas en el diagrama de clases:



- La sintaxis** es similar a la del acceso a un atributo (el nombre del objeto, un punto y el nombre del extremo remoto de la asociación por la que navegamos)
- El resultado de la navegación** es el conjunto de elementos que se encuentran en el extremo remoto de la asociación:

```
context Empresa
inv: self.manager.estaEnParo = falso
inv: self.empleados->notEmpty()
```

Prof. Ana M<sup>a</sup> Roldán

(40)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSIDAD  
NACIONAL  
DE  
EDIMBURGO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. **Objes y  
Propiedades.** 6. Colecciones.  
Referencias.

## 5. OBJS. Y PROPIEDADES

### Extremos de asociación y navegación

- Para multiplicidades:



- Mayores que 1:** el tipo del resultado de la navegación es un **Set**. Pero si la asociación es ordenada, el resultado es un **OrderedSet**.
- 0...1 ó 1:** el tipo del resultado de la navegación es el tipo del objeto (lo habitual) ó bien un Set con un único elemento\* → ese objeto se podrá usar como conjunto aplicando una transformación implícita añadiéndole un operador de conjunto.

```
context Empresa:
self.manager->size() = 1
```

En este caso también sirve para comprobar si existe algún objeto en el otro extremo

```
context Persona:
self.esposa->notEmpty() implies self.esposa.sexo
= Sexo.mujer
```

Prof. Ana M<sup>a</sup> Roldán

(41)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

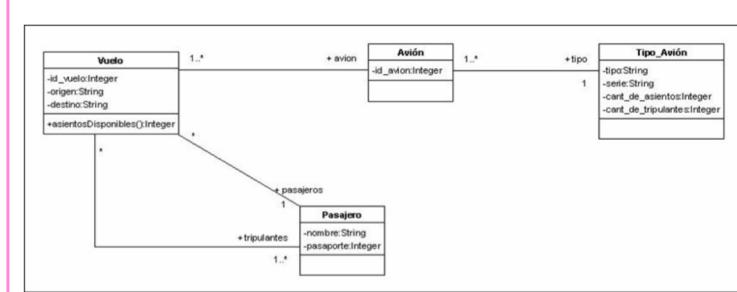
ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 5. OBJS. Y PROPIEDADES

### Extremos de asociación y navegación

EJEMPLO:



¿Cómo expresamos que un vuelo tendrá un número de pasajeros no superior al número de asientos del tipo de avión que tiene asignado?

42

Prof. Ana M. Roldán

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

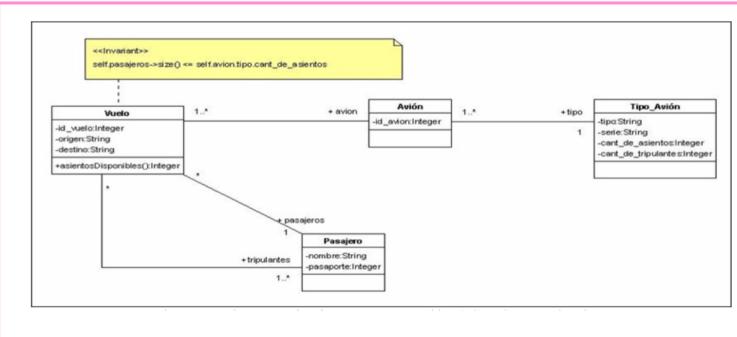
ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 5. OBJS. Y PROPIEDADES

### Extremos de asociación y navegación

EJEMPLO:



43

Prof. Ana M. Roldán

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

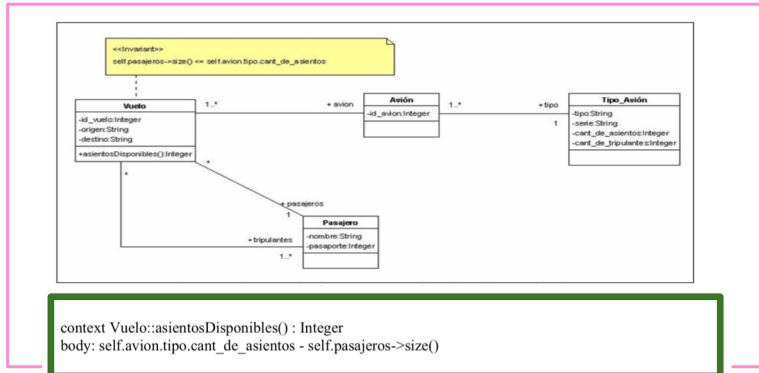
ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objets y  
Propiedades. 6. Colecciones.  
Referencias.

## 5. OBJETS. Y PROPIEDADES

### Extremos de asociación y navegación

EJEMPLO:



44

Prof. Ana M<sup>a</sup> Roldán

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objets y  
Propiedades. 6. Colecciones.  
Referencias.

## 5. OBJETS. Y PROPIEDADES

### Operaciones predefinidas sobre objetos

Aplicables a todos los objetos:

- **oclIsTypeOf (t: Classifier): Boolean**
- **oclIsKindOf (t: Classifier): Boolean** *-buscar-*
- **oclIsInState(t: Classifier): Boolean**
- **oclIsNew (): Boolean** *-buscar-*
- **oclAsType (t: Classifier): instance of Classifier** *-buscar-*

**EJEMPLO:** **oclIsTypeOf** devuelve true si el tipo de self y el tipo parámetro son el mismo:

```

context Person
inv: self.oclIsTypeOf( Persona ) — es true
inv: self.oclIsTypeOf( Empresa ) — es false

```

Prof. Ana M<sup>a</sup> Roldán

45

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN **ETSI**  
INSTITUTO POLITÉCNICO  
NACIONAL

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. **Objs y Propiedades.** 6. Colecciones.  
Referencias.

## 5. OBJS. Y PROPIEDADES

### Operaciones predefinidas sobre objetos

**Ejemplo:**

```
context Libro
inv: self.oclIsTypeOf(Libro) --is true
inv: self.oclIsKindOf(Libro) --is true
inv: self.oclIsTypeOf(Producto) --is false
inv: self.oclIsKindOf(Producto) --is false
inv: self.oclIsTypeOf(Libreria) --is false
inv: self.oclIsKindOf(Libreria) --is false
```

Prof. Ana M<sup>a</sup> Roldán

46

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN **ETSI**  
INSTITUTO POLITÉCNICO  
NACIONAL

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. **Objs y Propiedades.** 6. Colecciones.  
Referencias.

## 5. OBJS. Y PROPIEDADES

### Operaciones sobre clases

**⚠** Se puede acceder a las propiedades estáticas de las clases usando la notación de los dobles puntos, *donde id es un atributo estático de la clase Empleado:*

```
context Empleado inv:
Empleado:::id
```

- La operación **allInstances** permite obtener todas las instancias de una clase que hay en el modelo → *Se accede a esta operación con la notación punto y no con la de doble dos puntos pq no es una propiedad de la clase.*

**EJEMPLO:**

```
context Persona inv:
Persona .allInstances ()->forAll(p1, p2| p1 <>p2
implies p1.nombre <> p2.nombre)
```

Prof. Ana M<sup>a</sup> Roldán

47

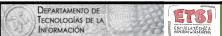
	<a href="#">1. Objetivos.</a> <a href="#">2. Introducción.</a> <a href="#">3 .OCL</a> <a href="#">4. Tipos y Operaciones</a> . <a href="#">5. Objs y Propiedades</a> . <b><a href="#">6. Colecciones</a></b> . <a href="#">Referencias</a> .
---	---

## 6. COLECCIONES

- El **tercer conjunto de tipos disponibles** en las restricciones OCL son las colecciones (*los otros dos son los tipos básicos predefinidos y los correspondientes a las clases del modelo UML que actúa de contexto*).
- OCL define un tipo abstracto, **Collection**, que se refina en tres tipos concretos: **Set**, **OrderedSet**, **Bag** y **Sequence**:
  - **Set** representa el concepto matemático de conjunto: *una colección de elementos sin repeticiones ni orden*.
  - **OrderedSet** es un conjunto de elementos ordenado.
  - **Bag** se diferencia de los conjuntos en que se permiten repeticiones, pero tampoco tiene orden y,
  - **Sequence** representa un **Bag** donde los elementos están ordenados.

Prof. Ana M<sup>a</sup> Roldán

48

	<a href="#">1. Objetivos.</a> <a href="#">2. Introducción.</a> <a href="#">3 .OCL</a> <a href="#">4. Tipos y Operaciones</a> . <a href="#">5. Objs y Propiedades</a> . <b><a href="#">6. Colecciones</a></b> . <a href="#">Referencias</a> .
---	---

## 6. COLECCIONES

### LITERALES EN LAS COLECCIONES

- Se pueden definir elementos de un tipo colección mediante literales, con el nombre del tipo seguido de los elementos de la colección entre llaves y separados por comas:

```
Set{1, 4, 2 }, Set{'manzana', 'naranja'}
Bag{1, 3, 4, 3},
Sequence{'lunes', 'martes'}, Sequence{2,5,76,2}
```

Prof. Ana M<sup>a</sup> Roldán

49

DEPARTAMENTO DE  
Tecnologías de la  
Información

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 6. COLECCIONES

### LITERALES EN LAS COLECCIONES

- Set, OrderedSet, Bag y Sequence son subtipos de Collection, pero no son subtipos entre sí en ningún caso:

**Un conjunto de elementos de un tipo T1 será subtipo de un conjunto de elementos de otro tipo T2 cuando:**

- T1 sea el mismo que T2 o
- T1 sea subtipo de T2

■ **Por ejemplo:**

- ✓ Set(Bicycle) es subtipo de Set(Transport),
- ✓ Set(Bicycle) es subtipo de Collection(Bicycle),
- ✓ Set(Bicycle) es subtipo de Collection(Transport) **pero** no es cierto que lo sea de Bag(Bicycle).

Prof. Ana M<sup>a</sup> Roldán

(50)

DEPARTAMENTO DE  
Tecnologías de la  
Información

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

#### SELECT

- La operación **select** devuelve un subconjunto de la colección sobre la que se aplica → a ese subconjunto pertenecen los elementos para los que se cumple una condición lógica que se incluye al final.

```
context Empresa inv:  
    self.empleado() -> select(edad>50) ->notEmpty()
```

Esquema pág.11

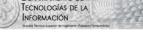
donde:

- La expresión **self.empleado()** es del tipo Set{Person}; → La operación select selecciona aquellas personas cuya edad es superior a los 50 años.
- En el ejemplo anterior es imposible acceder a las personas en concreto, sólo a sus propiedades. Para acceder a los elementos se incluye un iterador en la expresión:

```
context Empresa inv:  
    self.empleado() -> select(p | p.edad>50) ->notEmpty()  
( )
```

Prof. Ana M<sup>a</sup> Roldán

(51)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN  ETSI 

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

#### **SELECT**

- También se puede indicar explicitamente el tipo del iterador en la expresión.

```
context Empresa inv:  
    self.empleado() -> select(p : Persona | p.edad>50)  
        ->notEmpty()
```

#### **REJECT**

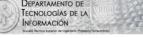
- La operación **reject** es análoga a **select**, pero *selecciona el conjunto complementario*, es decir, *aquellos elementos para los que la condición es falsa*.

Ejemplo:

```
context Empresa inv:  
    self.empleado() -> reject (estaCasado) -> isEmpty()
```

Prof. Ana M<sup>a</sup> Roldán

52

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN  ETSI 

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

#### **forAll**

- **Es el cuantificador universal** sobre una colección y permite especificar una condición booleana que satisfacen todos los elementos de una colección.

```
Collection -> forAll (v:Type | boolean-expresión-with-v)
Collection -> forAll (v | boolean-expresión-with-v))
Collection -> forAll (boolean-expresión-with-v))
```

Prof. Ana M<sup>a</sup> Roldán

53

DEPARTAMENTO DE  
Tecnologías de la  
información

**ETSI**  
INVESTIGACIÓN  
y ESTUDIOS  
INTERNAZIONALI

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

- Ejemplo 1:

```
context Empresa
inv: self.empleado->forAll( edad <= 65 )
inv: self.empleado->forAll( p | p.edad <= 65 )
inv: self.empleado->forAll( p : Persona | p.edad
<= 65 )
```

- Ejemplo 2:

```
context Empresa inv:
self.empleado->forAll( p1, p2 : Persona | p1 <>p2
implies p1.apellido <> p2.apellido)
```

Prof. Ana M<sup>a</sup> Roldán

54

DEPARTAMENTO DE  
Tecnologías de la  
información

**ETSI**  
INVESTIGACIÓN  
y ESTUDIOS  
INTERNAZIONALI

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

- Ejemplo 3:

Context Factura inv:

self.items → forAll(i1,i2: Item | i1<>i2 implies i1.producto<>i2.producto)

Prof. Ana M<sup>a</sup> Roldán

55

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN ETSI INGENIERÍA UNED

1. Objetivos. 2. Introducción. 3 .OCL 4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones. Referencias.

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

#### exists

- **Es el cuantificador existencial** sobre una colección que sirve para saber si al menos un elemento de la colección satisface una condición → Es decir, *especifica un resultado lógico, que será verdadero cuando la condición asociada a la operación sea cierta para al menos uno de los elementos de la colección.*

```
Collection -> exist (v:Type | boolean-expresión-with-v)
Collection -> exist (v | boolean-expresión-with-v)
Collection -> exist (boolean-expresión-with-v)
```

- Ejemplo 1:
 

```
context Empresa inv:
    self.empleado->exists( p : Persona | p.apellido =
        'Moreno')
```

Prof. Ana M<sup>a</sup> Roldán

56

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN ETSI INGENIERÍA UNED

1. Objetivos. 2. Introducción. 3 .OCL 4. Tipos y Operaciones . 5. Objs y Propiedades. 6. Colecciones. Referencias.

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

#### exists

- Ejemplo 2: Todos los ítems de una factura deben tener una cantidad>0

```
context Factura inv:
    not self.items->exists(i: Item | i.cantidad=0)
```

```

classDiagram
    class Libro {
        <<verde>> libro : EString
        <<azul>> verdesClases : EString, In EString : EDuple
        <<azul>> deudasClases : EString, In EString : EDuple
        <<azul>> clientesPrestacionEstring : EString, In EString : Eclase
        <<azul>> clientesDeudas : EString, In EString : Eclase
        <<azul>> personas : EString, In EString : Lclase
        <<azul>> tiposPrestacionEstring : EString, In EString : Tpsta
        <<azul>> tiposDeudas : EString, In EString : Tpsta
        <<azul>> tiposPersonas : EString, In EString : Tpsta
        <<azul>> tiposTipos : EString, In EString : Tpsta
    }
    class Factura {
        <<negro>> numero : Edt = 0
        <<negro>> fecha : EString
        <<negro>> total : EDuple = 0.0
        <<negro>> subtotal : EDuple = 0.0
        <<negro>> impuestos : EDuple = 0.0
        <<negro>> descuento : EDuple = 0.0
        <<negro>> neto : EDuple = 0.0
    }
    class Item {
        <<negro>> orden : Edt = 1
        <<negro>> cantidad : Elec = 0
        <<negro>> precio : EDouble = 0.0
        <<negro>> descuento : EDouble = 0.0
        <<negro>> Neto : EDouble = 0.0
        <<negro>> stock : Elec = 0
    }
    class Producto {
        <<negro>> precio : EDuple = 0.0
        <<negro>> nombre : EString
        <<negro>> stock : Elec = 0.0
        <<negro>> stockMin : Elec = 0
        <<negro>> stockMax : Elec = 100
        <<negro>> vendidos : EString, In EString : Ethng
    }
    Libro "1..1" --> "1..1" Factura : <<negro>> libro
    Factura "1..1" --> "1..1" Item : <<negro>> factura
    Item "1..1" --> "1..1" Producto : <<negro>> producto
  
```

Prof. Ana M<sup>a</sup> Roldán

57

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN 

**1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.**

# 6. COLECCIONES

## OPERACIONES SOBRE COLECCIONES

**sum**

El **objetivo** de esta operación es sumar una colección de números (Real, Integer)  
`collection →sum()`

- Ejemplo Calcular el total de las ventas

```
Context Librería:: totalaVentas(): Integer
body: self.ventas.total → sum()
```



The diagram illustrates the following associations:

- Librería** has a **[1..1] ventas** association with **Factura**.
- Factura** has a **[1..1] ítems** association with **Item**.
- Item** has a **[1..1] producto** association with **Producto**.

Prof. Ana M<sup>a</sup> Roldán 

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN 

**1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.**

# 6. COLECCIONES

## OPERACIONES SOBRE COLECCIONES

**collect**

Especifica una colección derivada de otra colección → no es una sub-colección

Collection -> collect (v:Type| boolean-expresión-with-v)  
 Collection -> collect (v| boolean-expresión-with-v)  
 Collection -> collect (expresión)

Prof. Ana M<sup>a</sup> Roldán 

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

**1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.**

## 6. COLECCIONES

### OPERACIONES SOBRE COLECCIONES

**collect**

**Ejemplos:**

```
context Libreria::idFiscalClientes():Sequence(String)
body: self.clientes→collect(c: Cliente | c.idFiscal)
```

```
context Libreria::idFiscalClientes():Sequence(String)
body: self.clientes→collect(c | c.idFiscal)
```

```
context Libreria::idFiscalClientes():Sequence(String)
body: self.clientes→collect(idFiscal)
```

```
context Libreria::idFiscalClientes():Sequence(String)
body: self.clientes.idFiscal
```

Prof. Ana M<sup>a</sup> Roldán

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

**1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.**

## Extensiones OCL

### a) Valores iniciales

- Una expresión OCL puede ser utilizada para asignarle el valor inicial a una propiedad.
- El contexto en este caso es la propiedad (asociación o atributo) que se quiere definir.

```
context <tipo:: <nombref/rol propiedad>: <tipo propiedad>
init <expression calculo>
```

donde:

- <tipo> es el nombre de la entidad que contiene la operación (p.e. una clase)
- <nombref/rol propiedad> define el nombre de la operación dentro del tipo
- <tipo propiedad> define el tipo del valor de la propiedad
- <expression calculo> define la expresión precondición (pueden referenciarse los parámetros)

Prof. Ana M<sup>a</sup> Roldán

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

# Extensiones OCL

**Ejemplo 1:**

- context Meeting::isConfirmed : Boolean  
init: false
- context TeamMember:meetings : Set(Meetings)  
init: Set $\emptyset$

**Ejemplo 2:**

- context Librería::nombre: String  
init: 'DefalutName'

Prof. Ana M<sup>a</sup> Roldán

(62)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

# Extensiones OCL

**b) Atributos y Asociaciones derivadas**

- **Atributos derivados**  
context Team::size:Integer  
derive: members->size()
- **Asociaciones derivadas**  
context Meeting::conflict:Set(Meeting)  
derive: select(m | m <> self and self.inConflict(m))

Prof. Ana M<sup>a</sup> Roldán

(63)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2.Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

# Extensiones OCL

**c) Expresión body**

- **Se utiliza para expresar el resultado de una operación de consulta:**

```
context NombreDelTipo::NombreOperacion(param1:Type1,...):TipoDelResultado
    pre:expresiónOcl:OclExpresion
        body:expresiónOcl:OclExpresion
```

- **En el contexto de una operación puede aparecer en expresiones de pre y postcondición:**

```
context Persona::hijosMayorDeEdad( ): Set (Persona)
    pre: self.tieneHijo=true
        body: self.hijo -> select (h|h.edad>=18)

    ó

    context Persona::hijosMayorDeEdad( ): Set (Persona)
        pre: self.tieneHijo=true
            post: result =sel.hijo -> select (h|h.edad>=18)
```

Prof. Ana M<sup>a</sup> Roldán

64

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2.Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

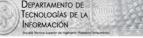
# Extensiones OCL

- **Consultas:** Operaciones que no cambian el estado del sistema

```
context Teammember::getMeetingTitles(): Bag(String)
    body: meetings->collect(title)
```

Prof. Ana M<sup>a</sup> Roldán

65

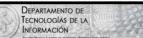
DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN  ETSI  
INVESTIGACIONES  
y DESARROLLO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## LIMITACIONES DE OCL

- No proporciona soporte para detección de inconsistencias
- Problema marco: supone que el resto del sistema permanece estático
- Recursión limitada
- allInstances() de tipos infinitos no permitido

Prof. Ana M<sup>a</sup> Roldán 

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN  ETSI  
INVESTIGACIONES  
y DESARROLLO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## LIMITACIONES DE OCL

- Diagramas de transición de estados, interacción, actividad, componentes, casos de uso

**OCL in Statecharts – Example (oclInState())**

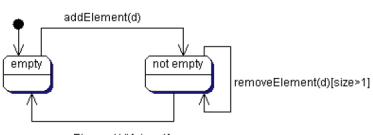
operation on all objects (Typ OclAny)

```

oclInState(s: OclState) : Boolean
  
```

**Vector**

-size:Integer
+removeElement(d:Data):Boolean
+addElement(d:Data):void



```

context Vector::removeElement(d>Data)
  pre: oclInState(notEmpty)
  post: size@pre = 1 implies oclInState(empty)
  
```

Prof. Ana M<sup>a</sup> Roldán 

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## LIMITACIONES OCL

- Cualquier expresión OCL puede evaluarse a undefined (OclVoid)
- Cuando una subexpresión se evalúa a undefined la expresión completa se evalúa a undefined
- Excepciones:
  - True or undefined = True
  - False and undefined = False
  - False implies undefined = True
- *oclIsUndefined(): Boolean*  
 -- true if the object is undefined  
 -- otherwise false

Prof. Ana M<sup>a</sup> Roldán

(68)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSI  
INGENIERÍA  
UNIVERSITARIO

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

## + Ejemplos OCL

```

classDiagram
    class Trip {
        +tripnr:Int
        +trips
    }
    class Coach {
        -id:Int
        -numberOfSeats:Int
    }
    class PrivateTrip {
        <|-- Trip
    }
    class RegularTrip {
        -availableSeats:Int
        +trips
    }
    class Person {
        -name:EString
    }

    Trip "*" -- "*" Coach : +coach
    Trip "*" -- "*" Person : +passengers
    PrivateTrip <|-- Trip
    RegularTrip --> Trip : +trips
    RegularTrip --> Person : +trips
  
```

Prof. Ana M<sup>a</sup> Roldán

(69)

DEPARTAMENTO DE  
TÉCNICAS DE LA  
INFORMACIÓN

ETSII  
Escuela Técnica Superior de Ingenieros Industriales

**Bibliografía**

1. Objetivos. 2. Introducción. 3 .OCL  
4. Tipos y Operaciones . 5. Objs y  
Propiedades. 6. Colecciones.  
Referencias.

- Object Management Group. Ocl Specification 2.4 (<http://www.omg.org/spec/OCL>)
- The object constraint language: precise modelling with UML. Warmer J., Kleeppe A.
- Universidad de Málaga
- Especificación de Contratos de Software usando OCL. Andrés Vignag.
- Como reforzar Diagramas de Clases UML aplicando OCL y Object-Z: un caso práctico. Elizabeth Vidal-Duarte y Cristian Vidal Silva
- Universidad de Sevilla
- Universidad de Castilla-La Mancha

Prof. Ana Mª Roldán

70