U.T.6: Elaboración de diagramas de clases con UML.

[Fuente: Entornos de Desarrollo, Alicia Ramos, Ed.Garceta]

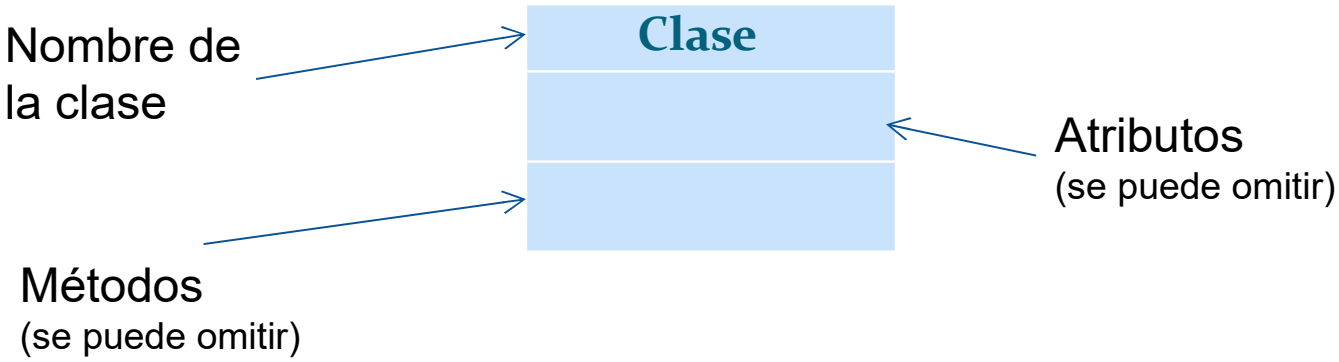
[Fuente: *EL LENGUAJE UNIFICADO DE MODELADO*, Grady Booch, James Rumbaugh, Ivar Jacobson, Ed. Addison Wesley]

[Fuente: *UML GOTA A GOTA,* Martin Fowler, Kendall Scott, Ed. Addison Wesley]

- Diagrama de clases
  - Clases, atributos, métodos
  - Relaciones
    - Asociación
    - Herencia
    - Composición
    - Agregación
    - Realización
    - Dependencia
- Generación de código a partir de diagramas de clases

Diagrama de clases: clases

- En UML una clase se representa por un rectángulo con tres divisiones



- En UML los atributos pueden representarse con su nombre, o indicando también el tipo y valor por defecto.
- Los tipos básicos UML son:
  - Integer, String y Boolean
- Es necesario indicar la visibilidad del atributo:

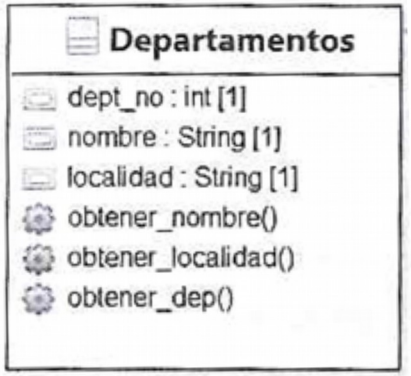
Público (public)	+
Protegido (protected)	#
Privado (private)	-
Paquete (package)	~

Diagrama de clases: métodos

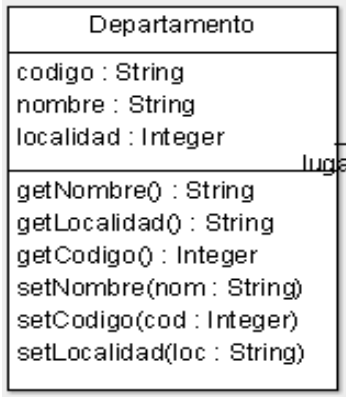
- Es la implementación de un servicio de la clase
- En UML los métodos se representan por su nombre, tipos de parámetros y tipo de dato devuelto.
- Igual que los atributos, la visibilidad del método:

Público (public)	+
Protegido (protected)	#
Privado (private)	-
Paquete (package)	~

Diagrama de clases

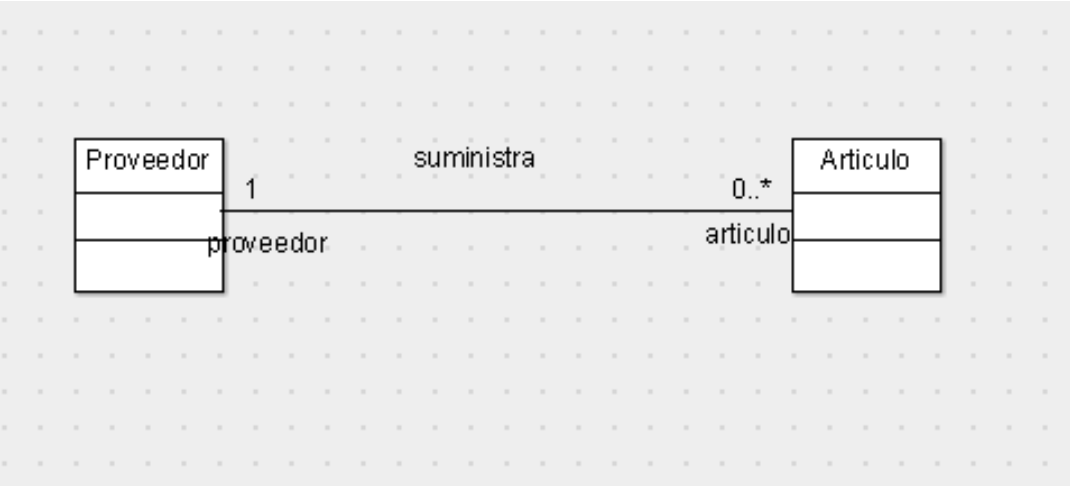


Netbeans UML



ArgoUML

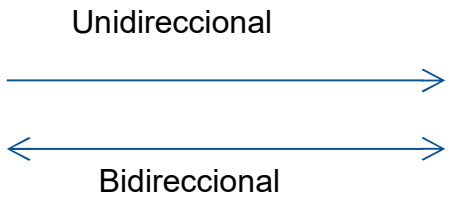
- En UML, los vínculos entre objetos se representan mediante asociaciones.
- Las asociaciones tienen un nombre y una cardinalidad



- Las multiplicidades destino mayores que 1 se implementan como un atributo de tipo array o colección.

Notación	Cardinalidad/Multiplicidad
0..1	Cero o una vez
1	Una y sólo una vez
*	De cero a varias veces
1..*	De una a varias veces
M..N	Entre M y N veces
N	N veces

- Tipos de relaciones
  - Asociación, dependiendo de si una clase conoce la existencia de la otra o no (*navegabilidad*) puede ser:
    - Bidireccional
      - En java cada una de las clases contendrá un objeto o set de objetos de la otra clase.
    - Unireccional
      - En java sólo la clase origen contendrá un objeto o set de objetos de la clase destino.



Asociación

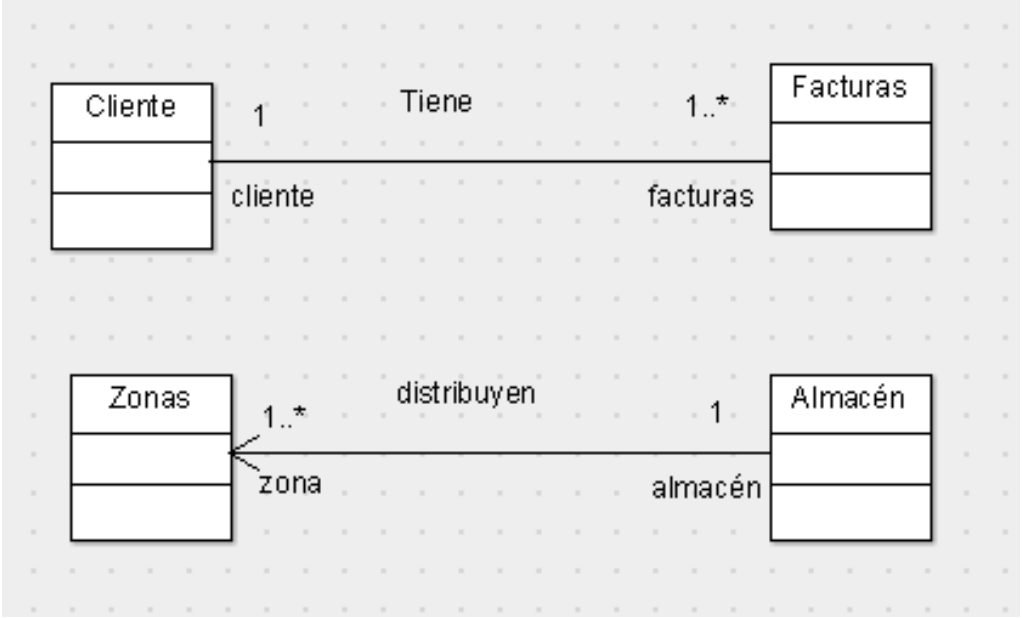

Cliente
1
Tiene
1..*
Facturas
cliente
facturas
Zonas
1..*
distribuyen
1
Almacén
zona
almacén

Diagrama de clases: Asociación

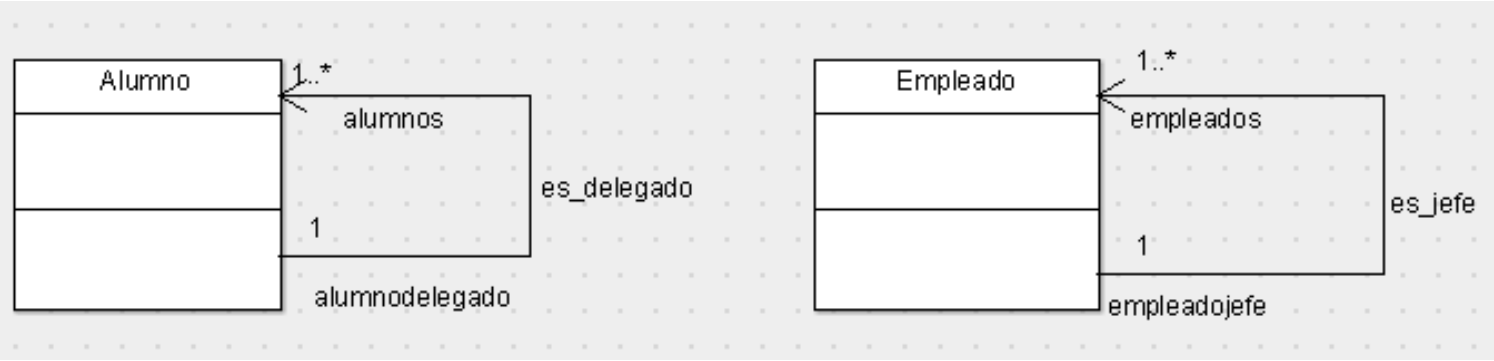
```
public class Cliente {     // 1 cliente tiene muchas facturas
   public HashSet<Facturas> facturas = new HashSet<Facturas>();

   public Cliente(){}
   public HashSet<Facturas> getFacturas(){
      return this.facturas;
   }

   public void setFacturas(HashSet<Facturas> nuevaFactura){
      this.facturas = nuevaFactura;
   }
}

public class Facturas {     // 1 factura pertenece a un sólo cliente
   public Cliente cliente = null;
   public Facturas(){}
   public Cliente getCliente(){
      return this.cliente;
   }
   public void setCliente(Cliente nuevoCliente){
      this.cliente = nuevoCliente;
   }
}
```

Diagrama de clases: Asociación

```
public class Zonas {     // No sabe de la existencia del almacén
   public Zona(){}
}

public class Almacen {     // 1 almacén distribuye en muchas zonas
   public HashSet<Zonas> zonas = new HashSet<Zonas>();

   public Almacen(){}
   public HashSet<Zonas> getZonas(){
      return this.zonas;
   }

   public void setZonas(HashSet<Zonas> nuevaZona){
      this.zonas = nuevaZona;
   }
}
```

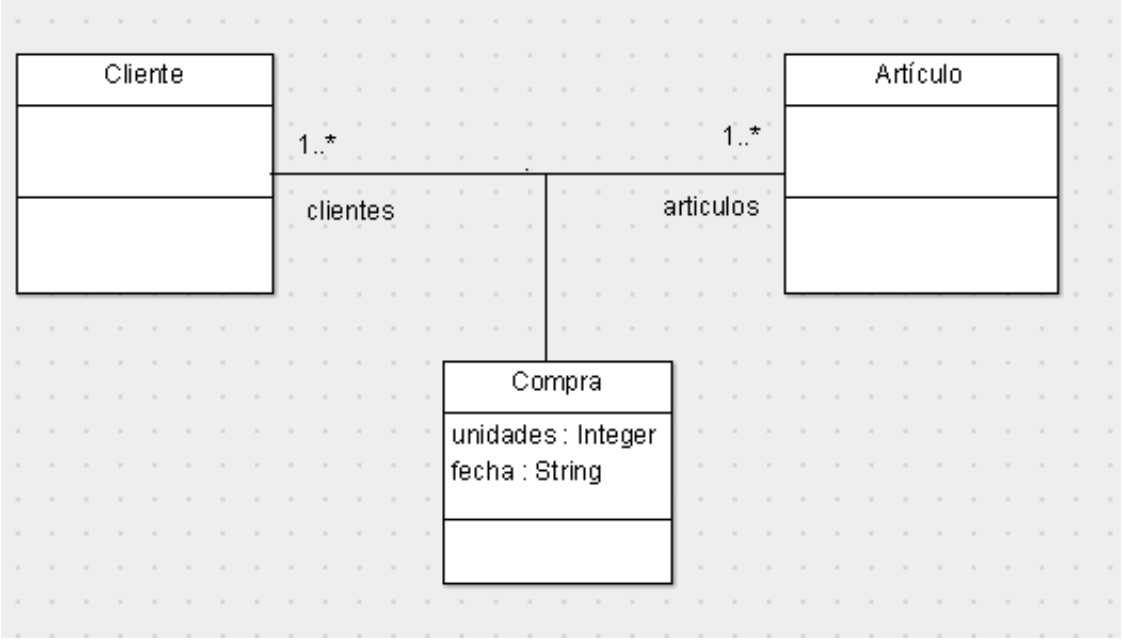
Asociación reflexiva

- Una clase se asocia consigo misma



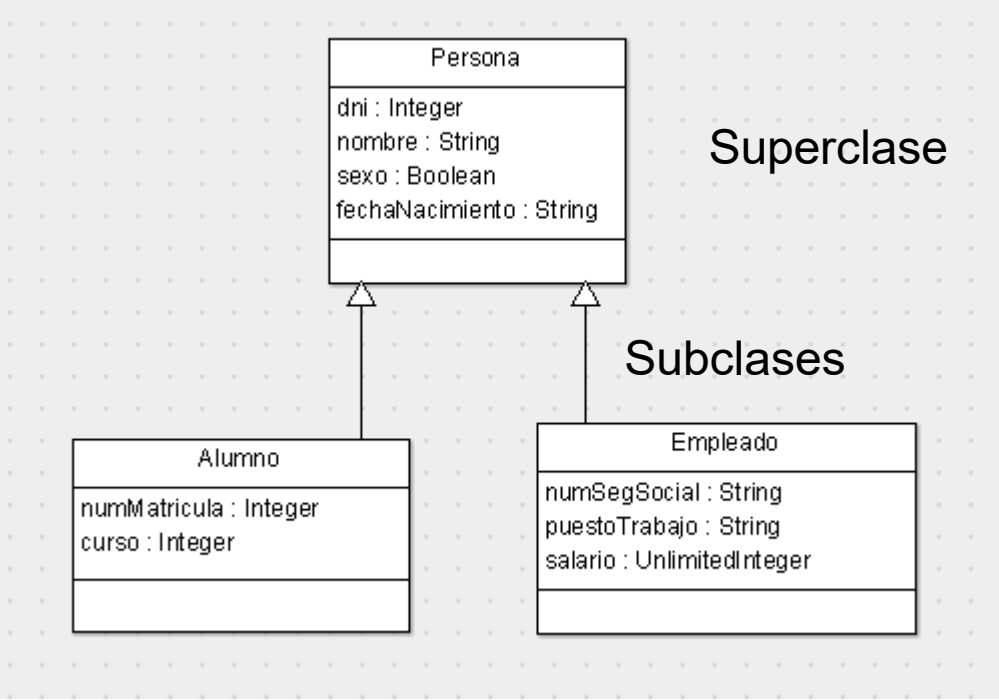
Clase ASOCIACIÓN

- Cuando una asociación entre dos clases posee sus propios atributos, se crea una nueva clase en el diagrama para dicha asociación.
- Pueden estar dotadas de atributos y métodos y asociarse, a su vez, con otras clases.



- Tipos de relaciones
  - Herencia
    - Abstracción para compartir similitudes entre clases (atributos y operaciones)
    - Se representa mediante una jerarquía donde:
      - La clase principal guarda las características comunes
      - Las subclases heredan y especializan las características de la clase principal.
    - Se representa en UML mediante una flecha que apunta a la clase principal.

- Tipos de relaciones
  - Herencia



- Tipos de relaciones
  - Herencia

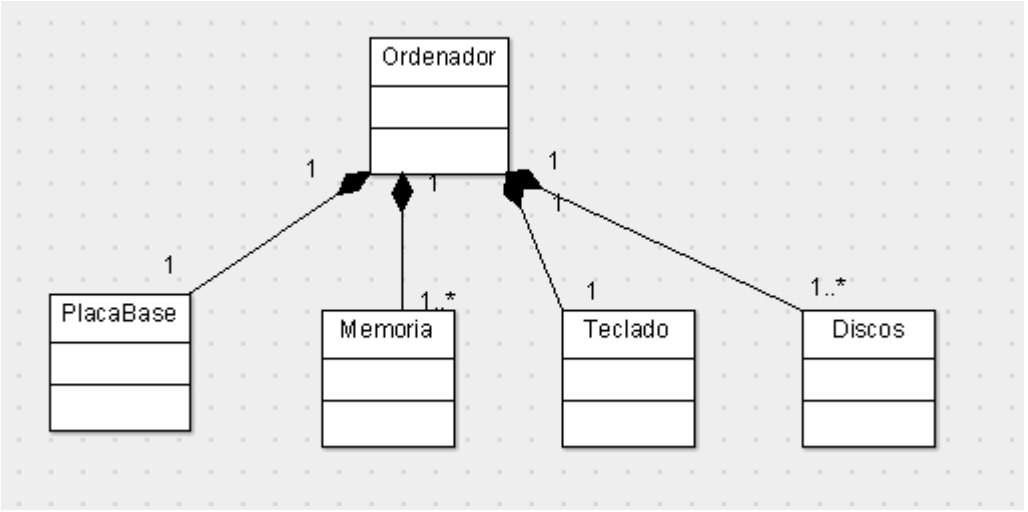
```
public class Persona {
   private int dni;
   private char nombre;
   private boolean sexo;
   private char fechaNacimiento;

   public Persona(){}
}

public class Alumno extends Persona{
   private int numMatricula;
   private int curso;

   public Alumno(){}
}
```

```
public class Empleado extends Persona{
   private char numSegSocial;
   private char puestoTrabajo;
   private int salario;

   public Empleado(){}

}
```

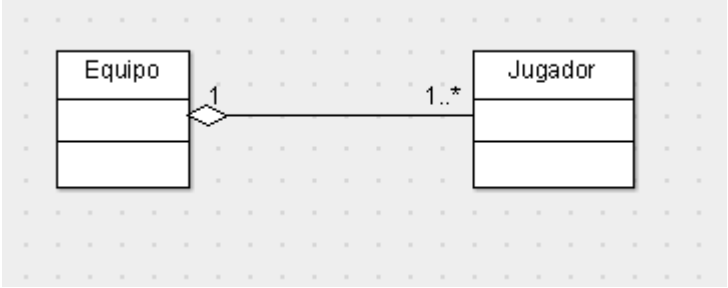
- Tipos de relaciones
  - Composición
    - Asocia un objeto complejo con los objetos que lo componen.
    - Existen dos formas de composición:
      - Composición fuerte : Los objetos forman parte sólo del objeto compuesto y no pueden participar en asociaciones con otros objetos compuestos.

      - Composición débil o agregación: Los componentes pueden ser compartidos por varios objetos compuestos

- Tipos de relaciones
  - Composición



  - Agregación



- Tipos de relaciones
  - Composición

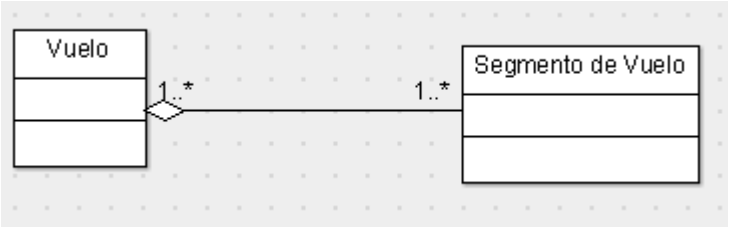
```
public class Ordenador {
   public HashSet<Memoria> memorias = new HashSet<memoria>();
   public HashSet<Disco> discos = new HashSet<disco>();
   public Teclado teclado = null;
   public Placabase placa = null;
   public Ordenador() {}
}
```

- Tipos de relaciones
  - Agregaciones típicas:

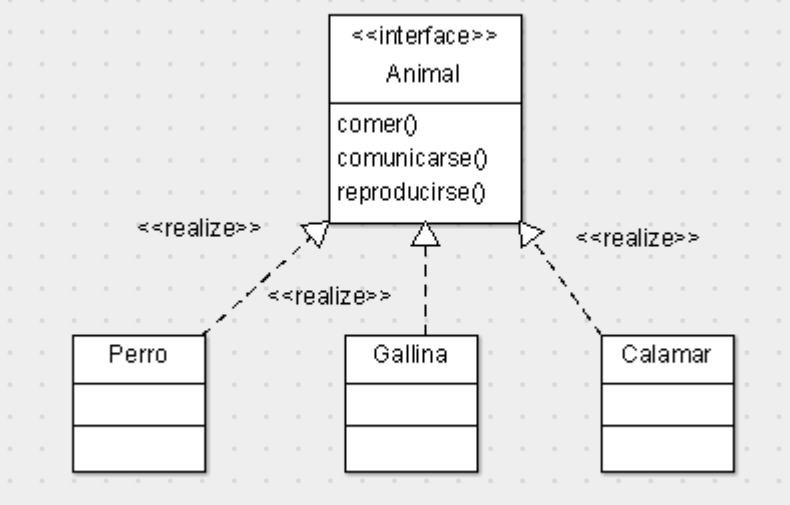
- Partes que componen un objeto de nivel superior

- Elementos contenidos en otro nivel superior

- Miembros de una colección o conjunto



- Tipos de relaciones
  - Realización
    - Es la relación de herencia entre una clase interfaz y la subclase que implementa dicha interfaz.
    - Son también las relaciones entre casos de uso y las colaboraciones que los realizan.
    - Se representan gráficamente con línea discontinua y flecha.



- Tipos de relaciones
  - Realización

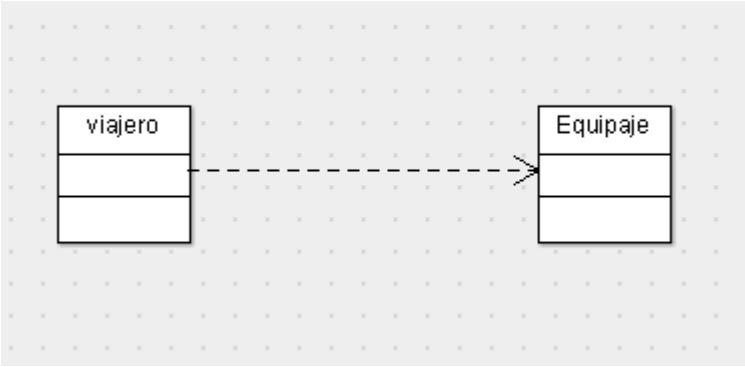
```
public interface Animal {
   public void comer();
   public void comunicarse();
   public void reproducirse();
}
```

```
public class Calamar implements Animal{
    public Calamar(){}
    public void comer(){}
    public void comunicarse(){}
    public void reproducirse(){}
}
```

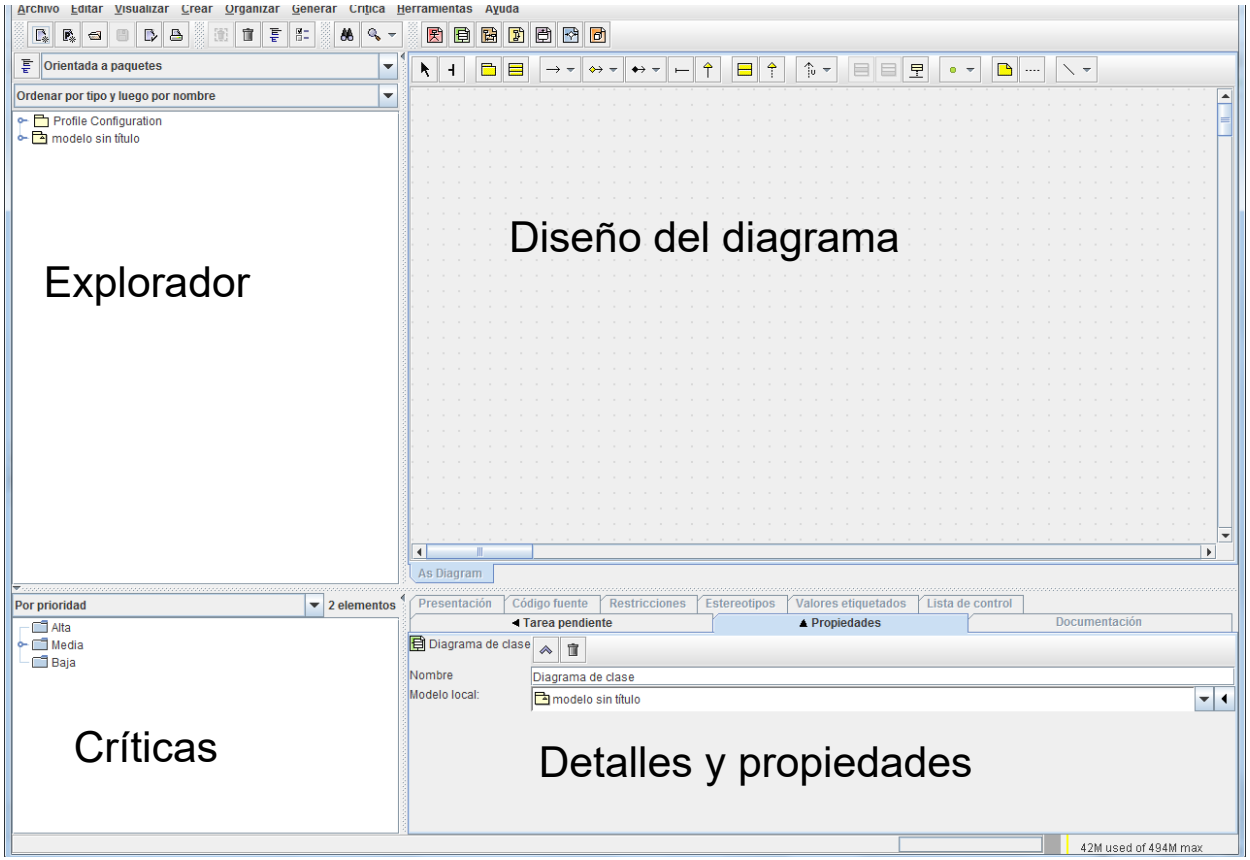
```
public class Perro implements Animal {
   public Perro(){}
   public void comer(){}
   public void comunicarse(){}
   public void reproducirse(){}
}
```

```
public class Gallina implements Animal {
public Gallina(){}
public void comer(){}
public void comunicarse(){}
public void reproducirse(){}
}
```

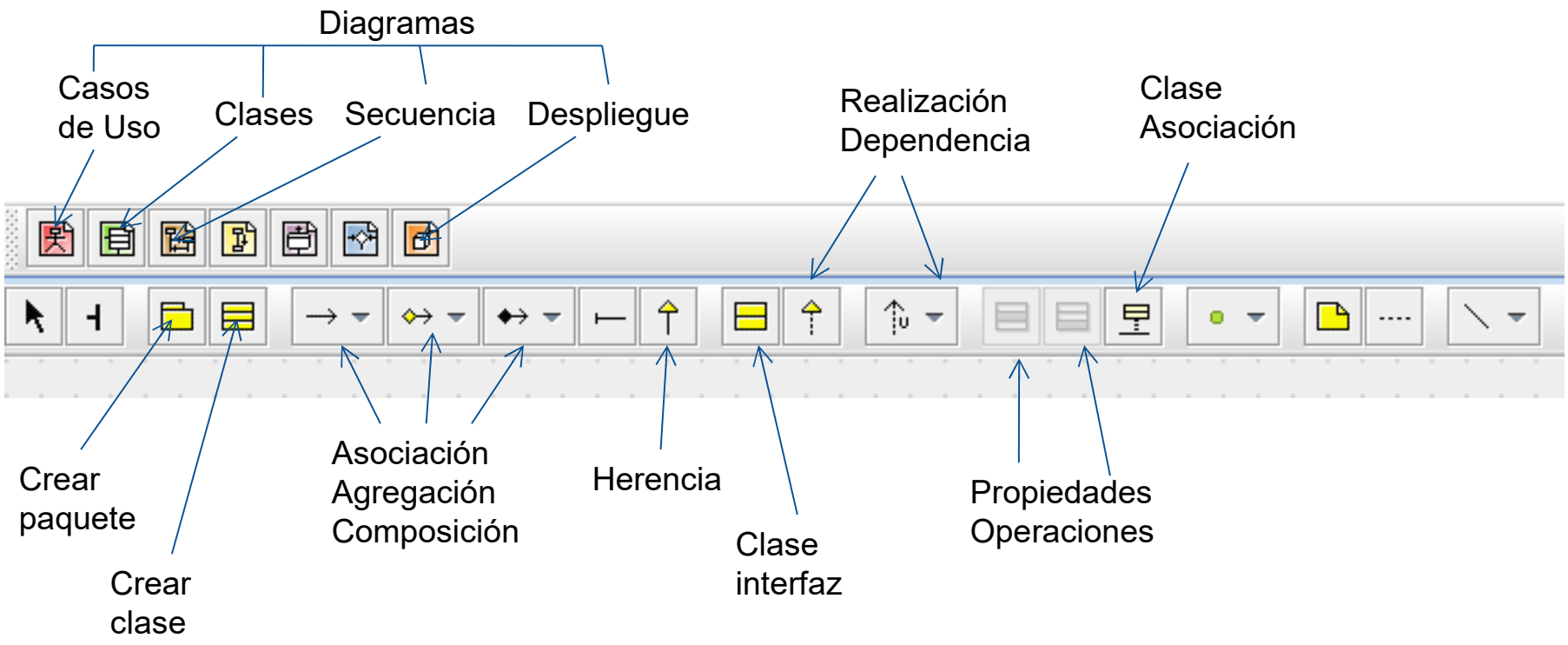
- Tipos de relaciones
  - Dependencia
    - Es la relación que se establece cuando una clase usa a la otra, es decir, la necesita para su cometido.
    - Se representan gráficamente con línea discontinua y flecha sin relleno.



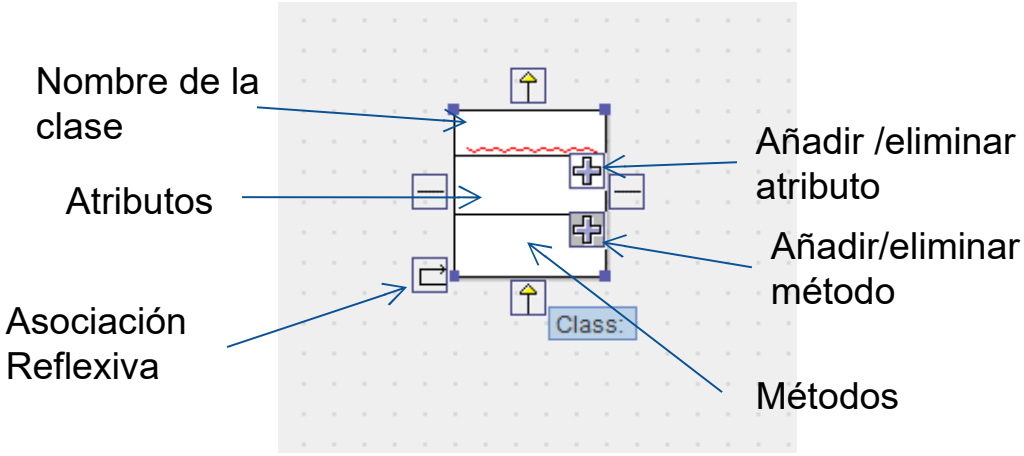
Ventana inicial



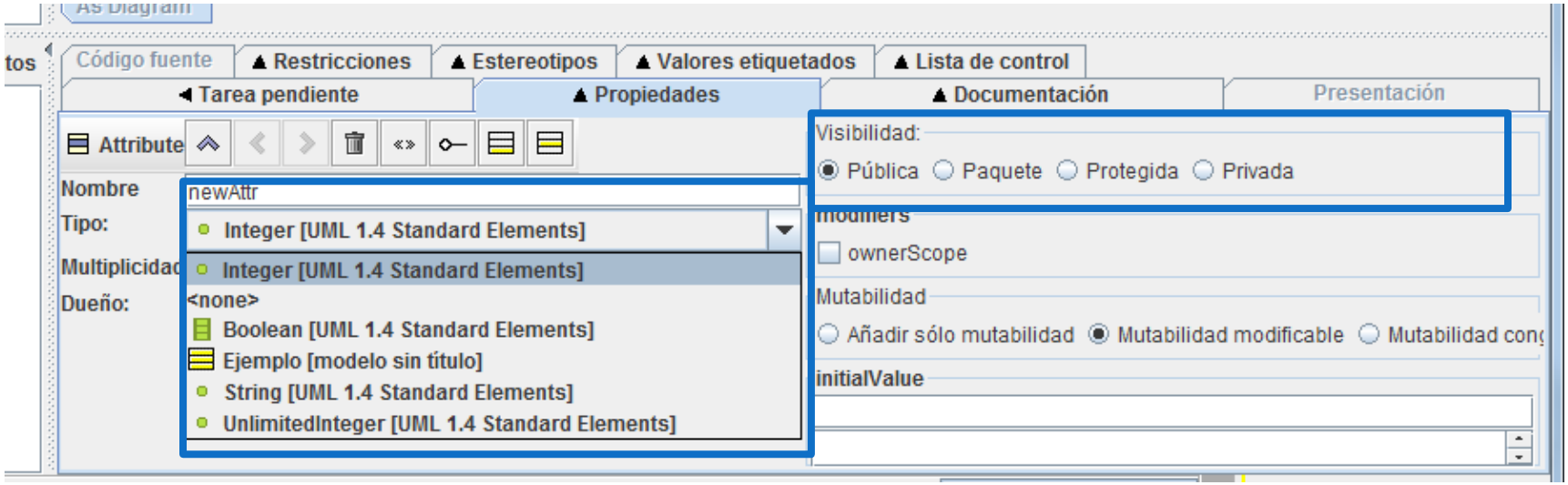
Barra de herramientas



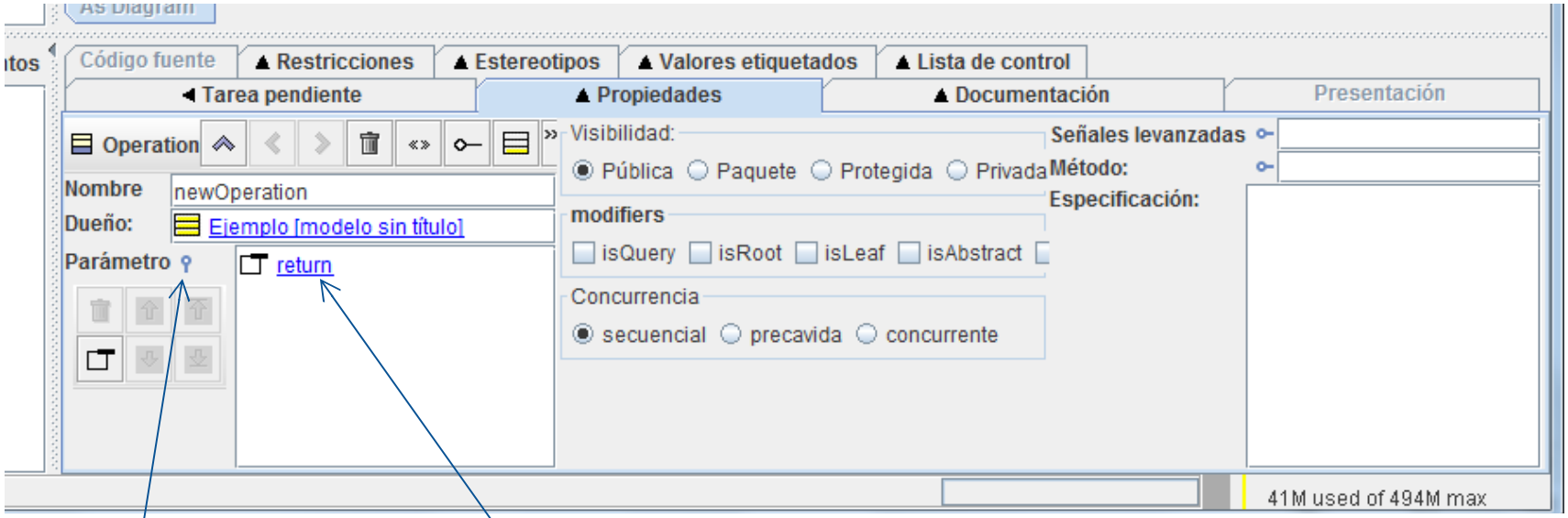
Clase


Nombre de la clase
Atributos
Asociación Reflexiva
Añadir /eliminar atributo
Añadir/eliminar método
Métodos
Class:

Propiedades de los atributos



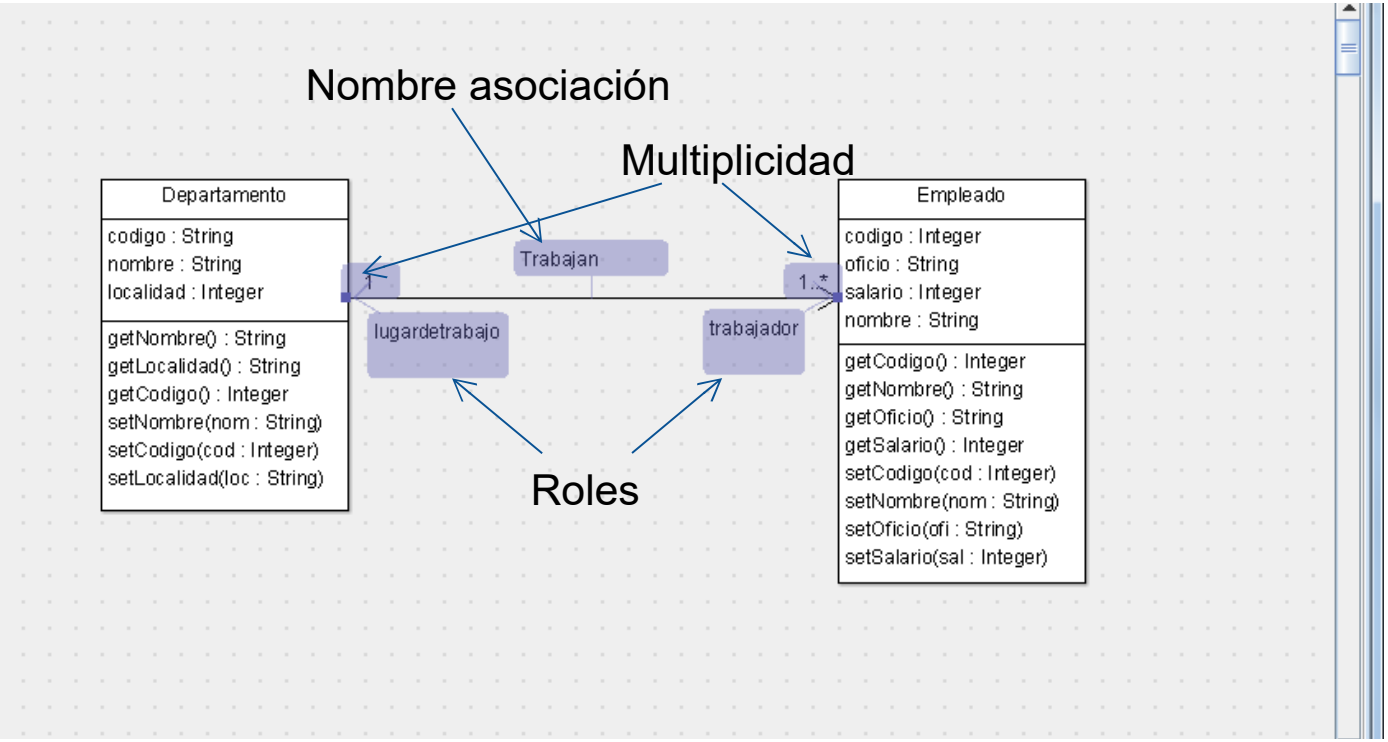
Parámetros en los métodos



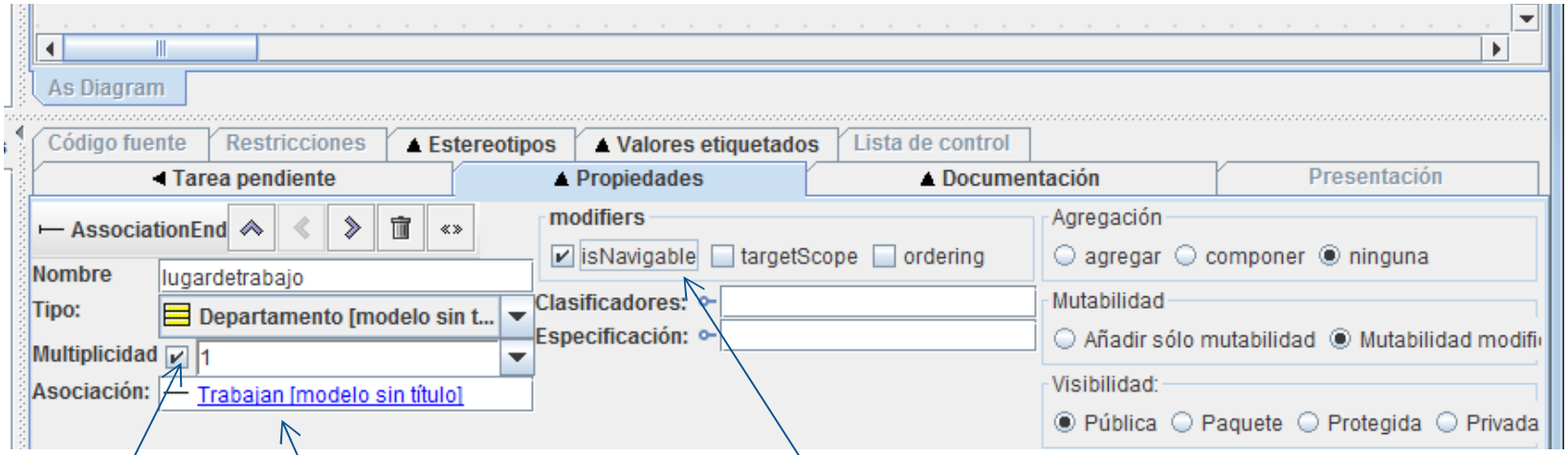
Añadir el tipo devuelto

Añadir un parámetro

Asociaciones



Propiedades de la asociación

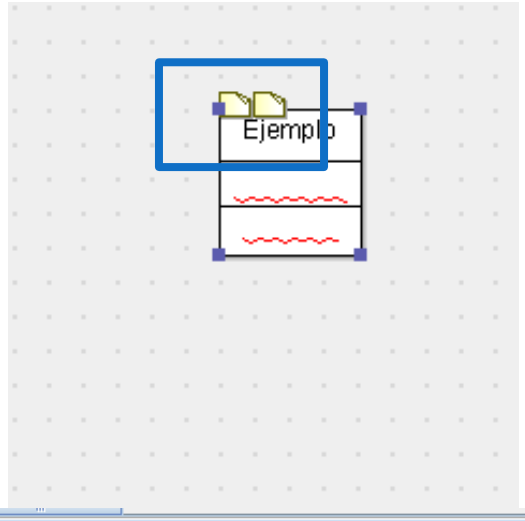


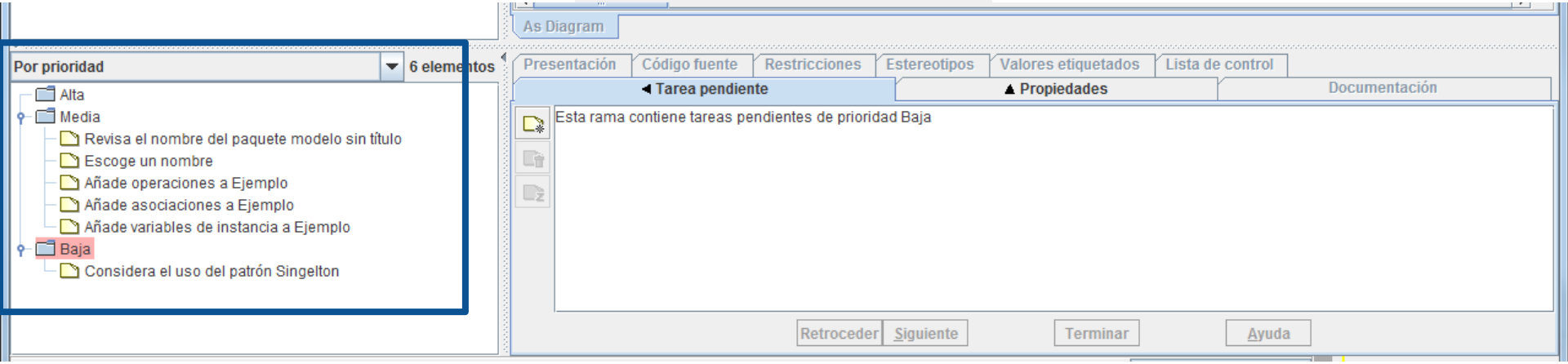
Multiplicidad

Roles

Dirección de la asociación

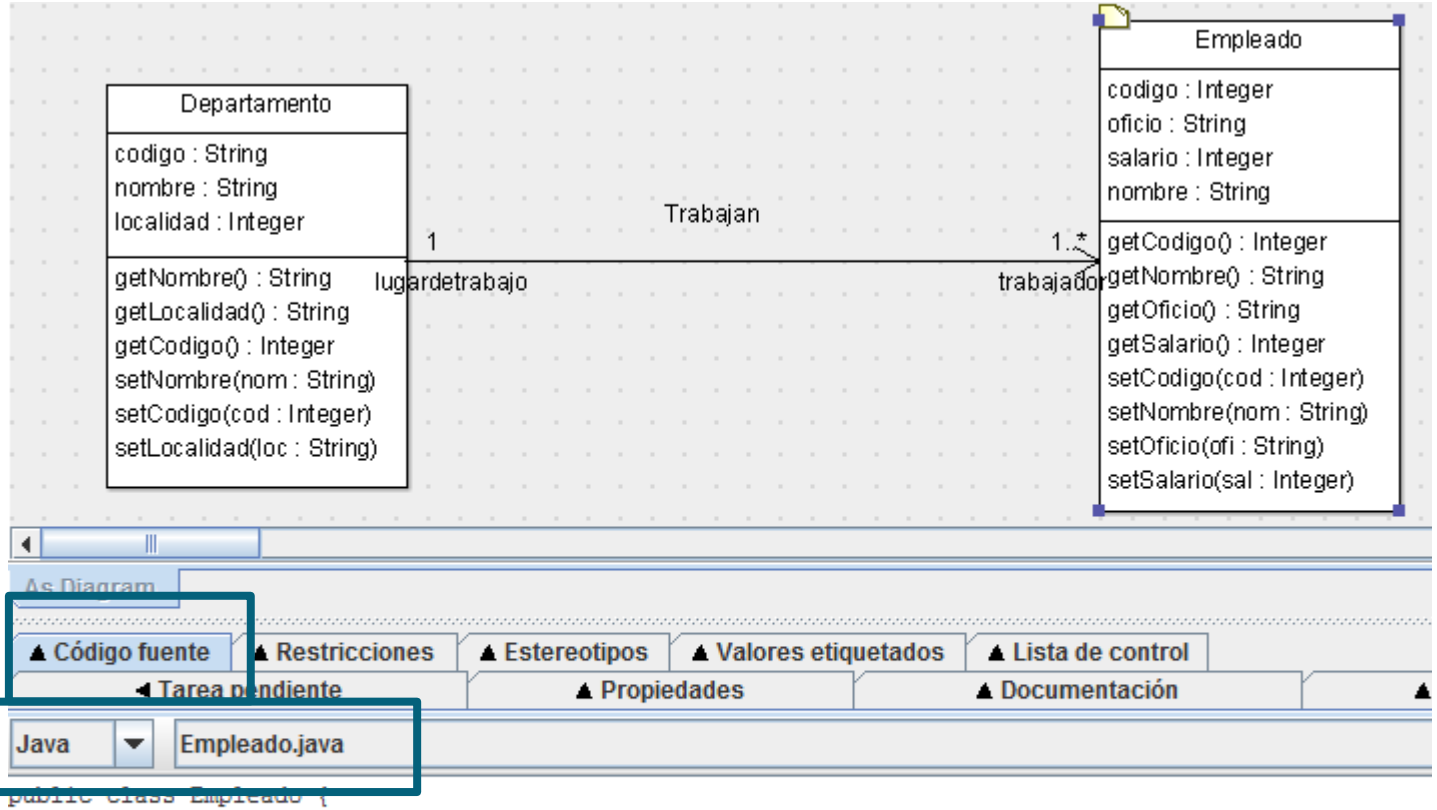
- Críticas y sugerencias





Generación de código fuente

- ArgoUML
  - Mostrar el código



- ArgoUML
  - Generar fichero fuente