Agenda Octubre

- 2/10: Diagrama de clases
- 7/10: Consulta
- 9/10: Diagrama de interacción
- 14/10: Consulta
- 16/10: Sprint #1 Review
- 21/10: Sprint #2 Planning
- 23/10: Consulta
- 28/10: Parcial
- 30/10: Revisión y consulta

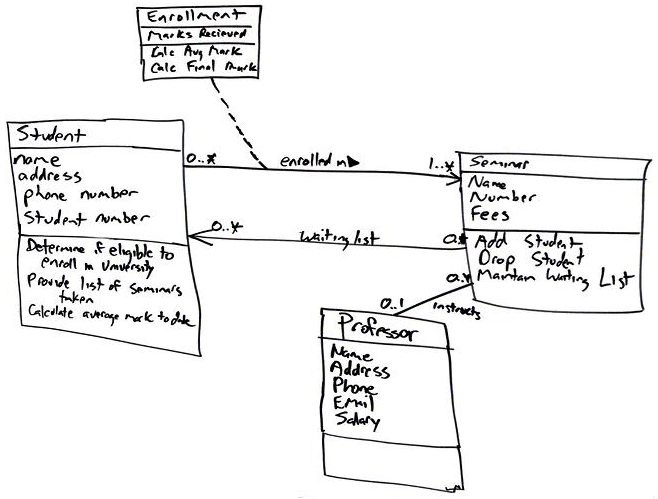
Diagrama de Clases

Dr. Santiago Vidal

ISISTAN-CONICET

Diagrama de Clases

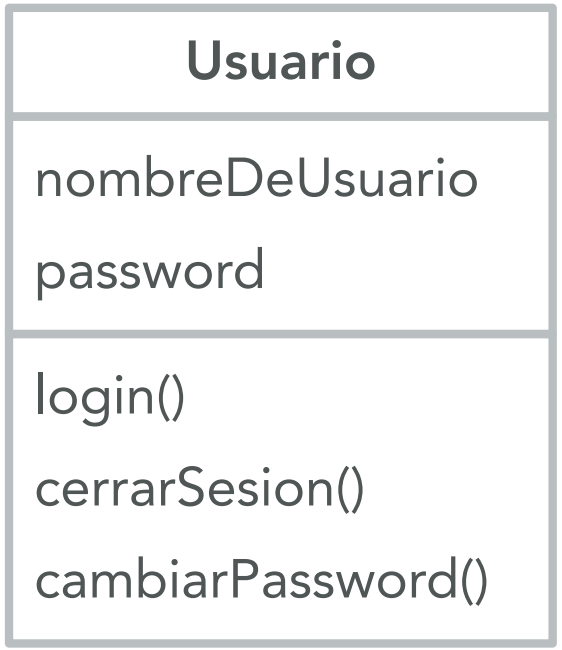
- Se utilizan para:
  - Explorar conceptos del dominio
  - Analizar requerimientos
  - Mostrar el diseño detallado de software orientado a objetos
- Generalmente contiene:
  - Clases
  - Interfaces
  - Relaciones



Clases

- Una clase es la descripción de un conjunto de objetos que comparten los mismos:
  - Atributos
  - Operaciones
  - Semantica
- Las clases se captura el vocabulario del sistema que se está desarrollando
- Representan objetos del mundo real y elementos conceptuales (ej. Conversacion, politica de ordenamiento, etc.)

Clases en UML



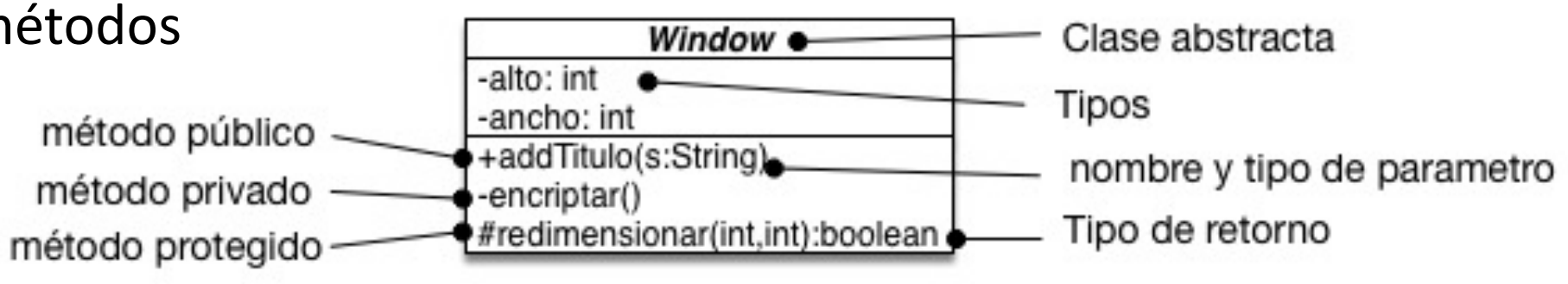
Nombre (CamelCase. Primer letra en mayúsculas)

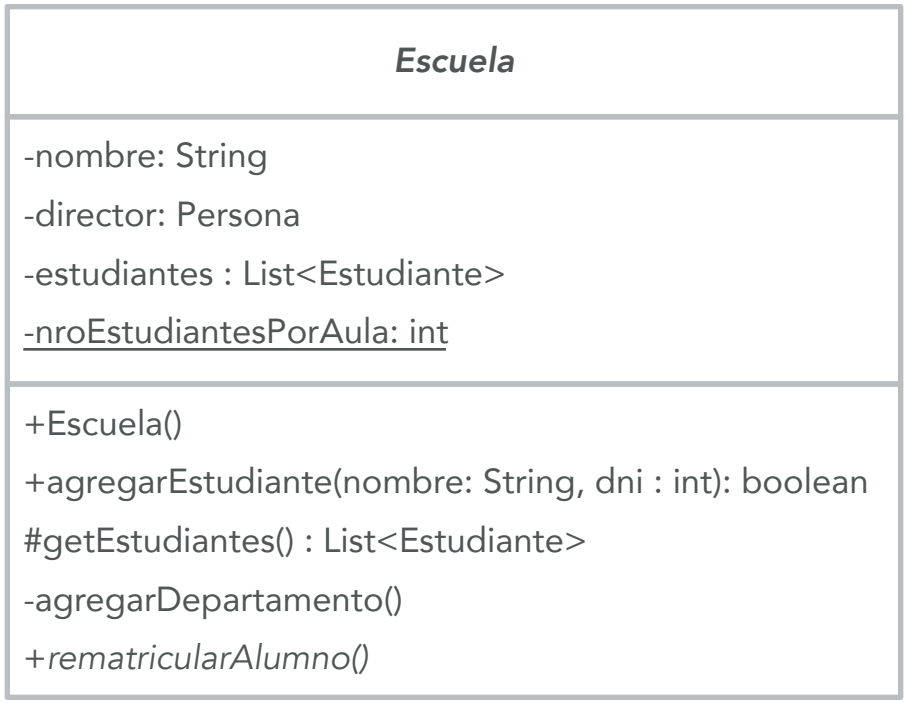
Atributos (CamelCase. Primer letra en minúsculas)

Operaciones (CamelCase. Primer letra en minúsculas)

Mayor detalle de la clase

- Accesibilidad de atributos y operaciones:
  - Public: +
  - Private: -
  - Protected: #
- Clases abstractas: su nombre se pone en cursiva
- Tipo de los atributos: luego del nombre del atributo se debe poner “:” y el tipo
- Parámetros: similar a la declaración de los atributos (pero sin accesibilidad)
- Retornos de los métodos




Escuela
-nombre: String
-director: Persona
-estudiantes : List<Estudiante>
-nroEstudiantesPorAula: int
+Escuela()
+agregarEstudiante(nombre: String, dni : int): boolean
#getEstudiantes() : List<Estudiante>
-agregarDepartamento()
+rematricularAlumno()

Relaciones entre clases

- Las relaciones son conexiones entre clases
- Modelan la colaboración entre objetos
- Tipos:
  - Generalización
  - Realización
  - Asociación
  - Agregación
  - Composicion
  - Clase de asociación
  - Dependencia

TIENEN QUE SER ESTRUCTURALES

LAS ROSAS TIENEN QUE SER EL TODO Y LAS PARTES.
LA COMPOSICION TIENE "TIEMPO DE VIDA LIGADO"

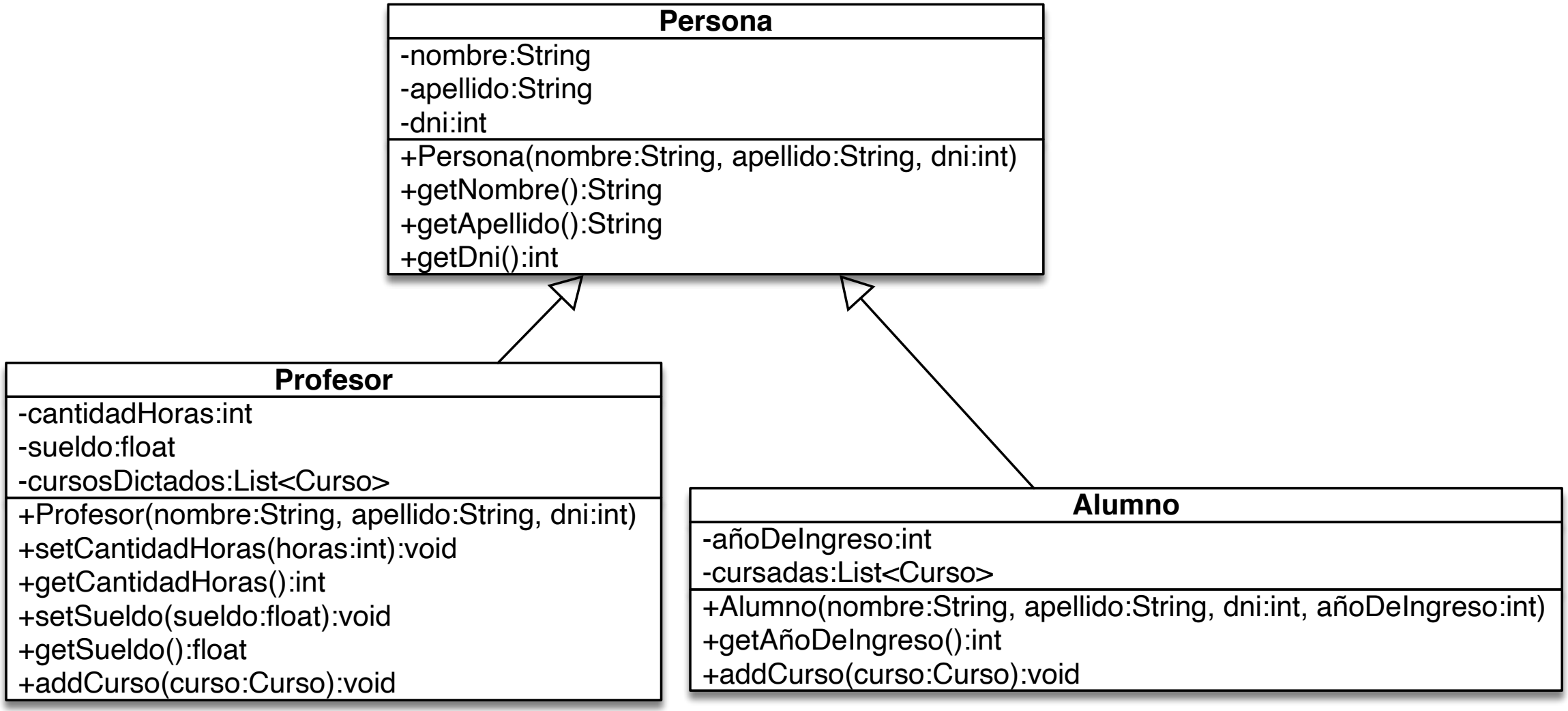
Relaciones entre clases

- Dos clases A y B están relacionadas si:
  - Un objeto de la clase **A envía un mensaje a** un objeto de la clase **B**
  - Un objeto de la clase **A crea un objeto de** la clase **B**
  - Un objeto de la clase **A tiene un atributo** cuyo **tipo** es **B** o que es una colección de objetos de tipo B
  - Un objeto de la clase **A recibe un mensaje con un objeto de** la clase **B** como parámetro
  - La clase **A es superclase de B**
  - La clase **A implementa la clase B**

Relación de Generalización

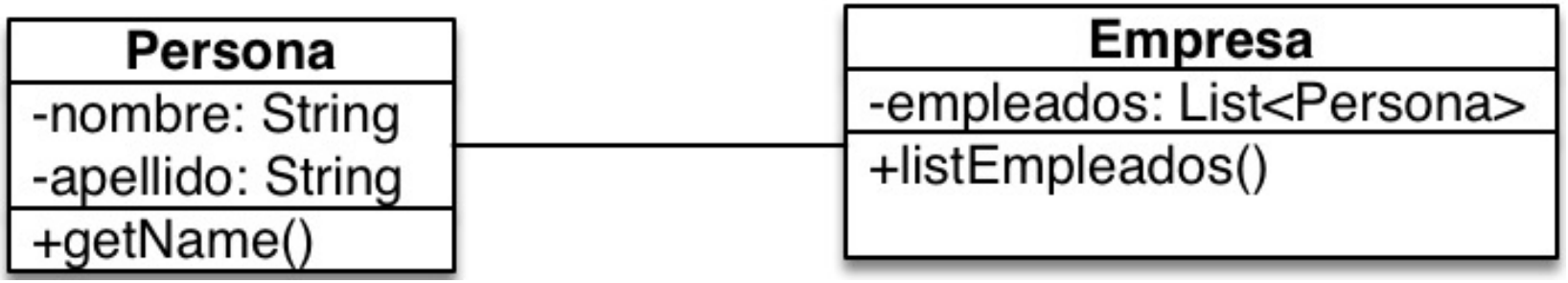
ES LO QUE CONOCEMOS COMO HERENCIA

- Indica una relación entre una clase general (superclass) y un tipo más específico de esa clase (subclass)
- Usamos generalizaciones para modelar conexiones del tipo: “Es un tipo de”
- Una clase A es del tipo de otra clase B en caso que:
  - La clase A es subclass de B
- Los atributos, operaciones y relaciones comunes se muestran en la superclass


Persona
-nombre:String
-apellido:String
-dni:int
+Persona(nombre:String, apellido:String, dni:int)
+getNombre():String
+getApellido():String
+getDni():int
Profesor
-cantidadHoras:int
-sueldo:float
-cursosDictados:List<Curso>
+Profesor(nombre:String, apellido:String, dni:int)
+setCantidadHoras(horas:int):void
+getCantidadHoras():int
+setSueldo(sueldo:float):void
+getSueldo():float
+addCurso(curso:Curso):void
Alumno
-añoDeIngreso:int
-cursadas:List<Curso>
+Alumno(nombre:String, apellido:String, dni:int, añoDeIngreso:int)
+getAñoDeIngreso():int
+addCurso(curso:Curso):void

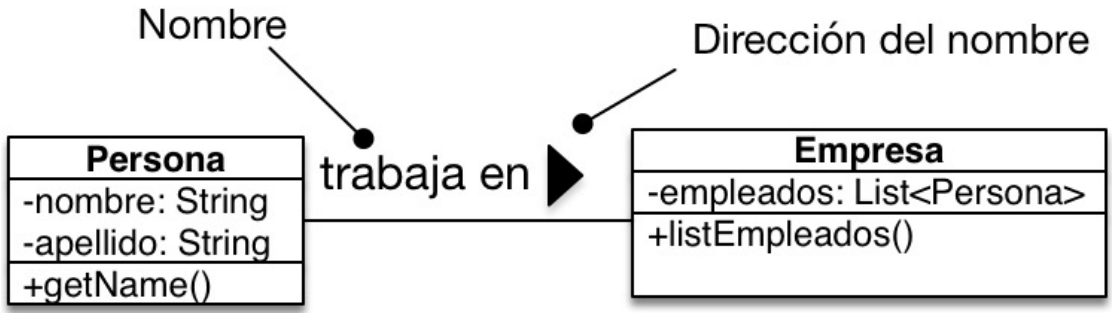
Relación de asociación

- Es una relación **estructural**
- Dos clases A y B están asociadas si:
  - Un objeto de la clase **A tiene un atributo cuyo tipo es B** o que es una colección de objetos de tipo B
- Usamos asociaciones para modelar conexiones del tipo: “tiene”, “es de”, “conoce”
- Se dibuja como una línea entre dos clases



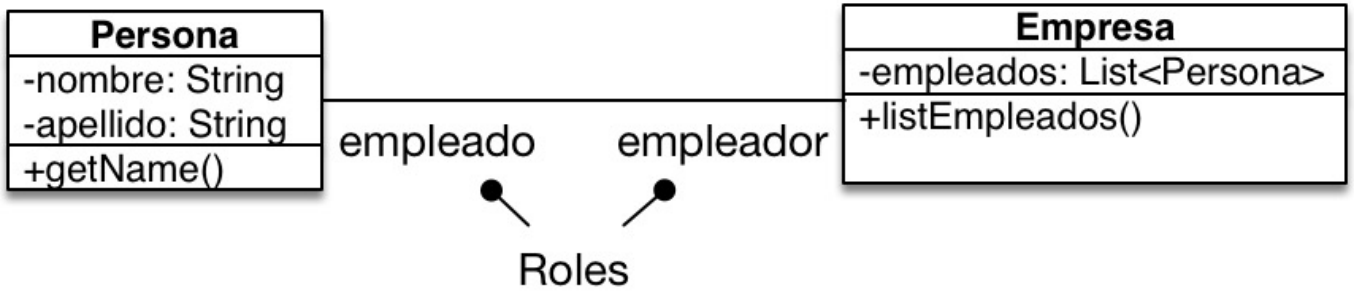
Adornos de Asociaciones

- Las asociaciones pueden tener adornos que agregan mas información a la relación
  - Nombre que describe la naturaleza de la relación



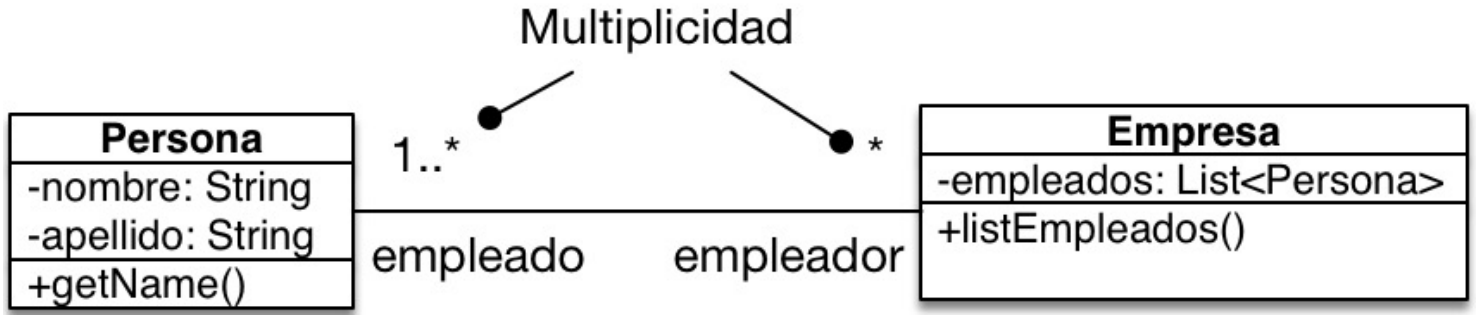
Adornos de Asociaciones

- Rol que cumple una clase en la relación (La misma clase puede jugar el mismo o diferentes roles en otras asociaciones)



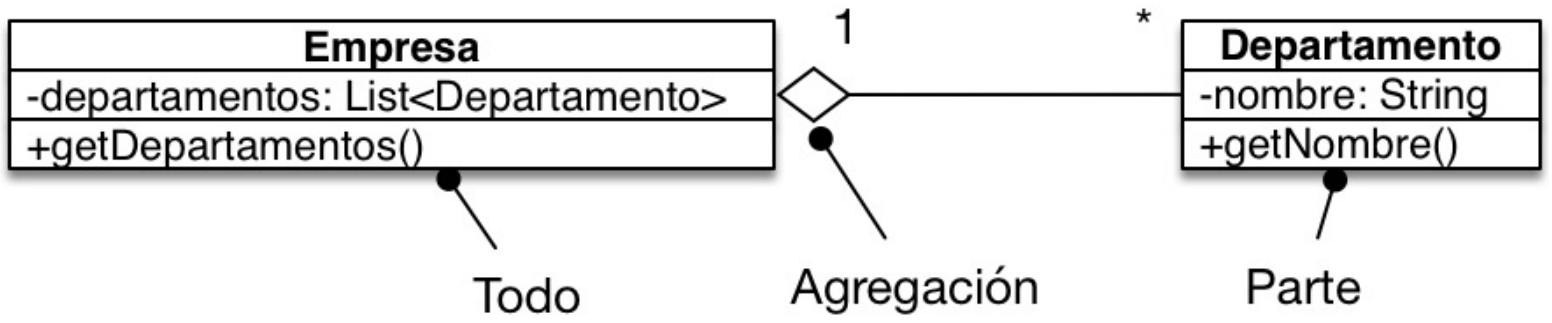
Adornos de Asociaciones

- La multiplicidad indica cuántos elementos de una instancia se relacionan con otra. Por ejemplo:
  - 0..1: entre 0 y 1 objetos
  - 3..4: entre 3 y 4 objetos
  - 6..*: 6 o mas objetos
  - 0..1, 3..4, 6..*: cualquier numero de objetos que no sea 2 o 5



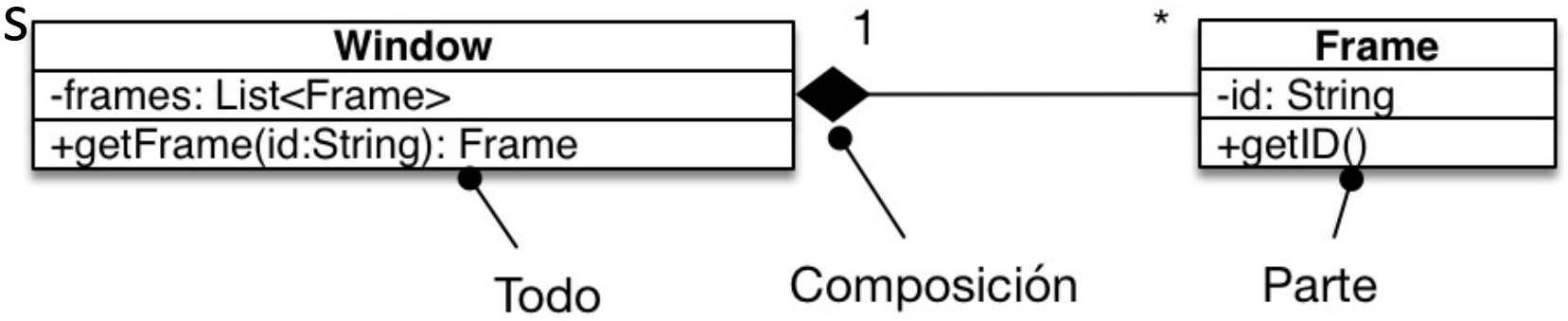
Relación de Agregación

- Es un tipo especial de relación de asociación
- Se utiliza solamente cuando una de las clases representa **el "todo" y la/s** otra/s la/s **"partes"**
- Modelan conexiones del tipo: "Esta formado por"
- Se dibuja como una línea entre dos clases con un rombo sobre la clase que representa el "todo"
- Puede incluir adornos



Relación de Composición

- Es un tipo especial de relación de agregación
- También representa el **“todo” y la/s “partes”** pero el **tiempo de vida** de las partes esta **ligada al** del **todo**
- Las partes se pueden crear después del todo pero cuando se destruye el todo también se destruyen las partes
- Se dibuja como una línea entre dos clases con un rombo lleno sobre la clase que representa el “todo”
- Puede incluir adornos

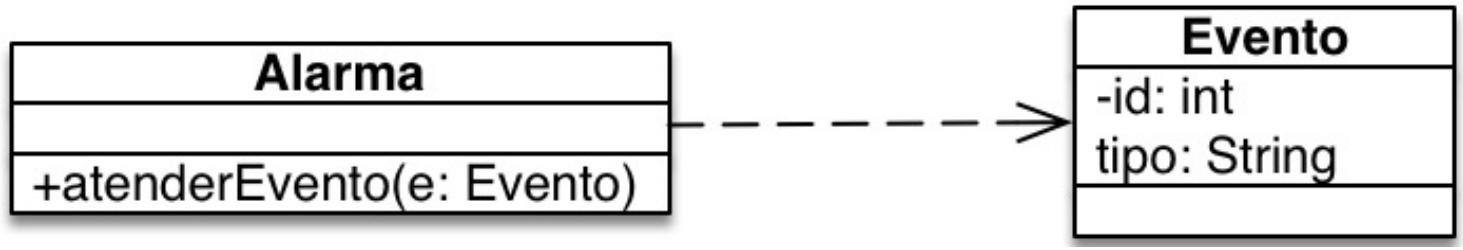


Relación de Dependencia

- Es una relación para indicar que una clase **usa** a otra clase
- Una clase A depende de otra clase B en alguno de los siguientes casos:
  - Una operación del objeto de la clase **A define una variable cuyo tipo es B** o que es una colección de objetos de tipo B
  - Un objeto de la clase **A envía un mensaje a** un objeto de la clase **B**
  - Un objeto de la clase **A crea un objeto de** la clase **B**
  - Un objeto de la clase **A recibe un mensaje con un objeto** de la clase **B** como parámetro

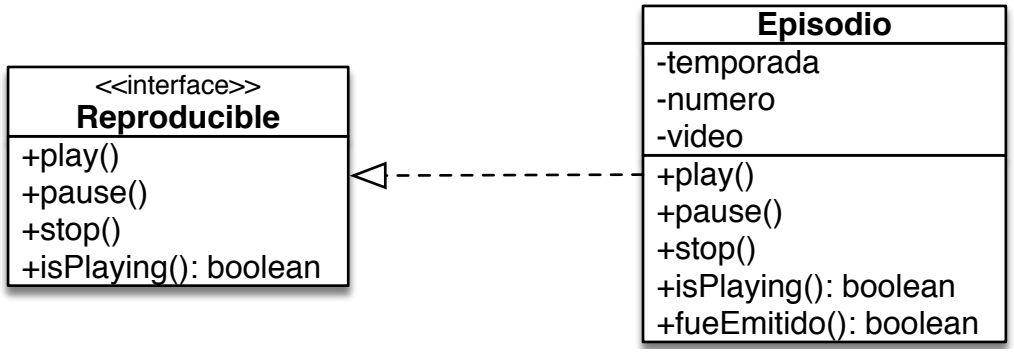
Relación de Dependencia

- Usamos dependencias para modelar conexiones del tipo: “usa”
- Se dibuja como una línea punteada entre dos clases con una flecha que indica de que clase se depende



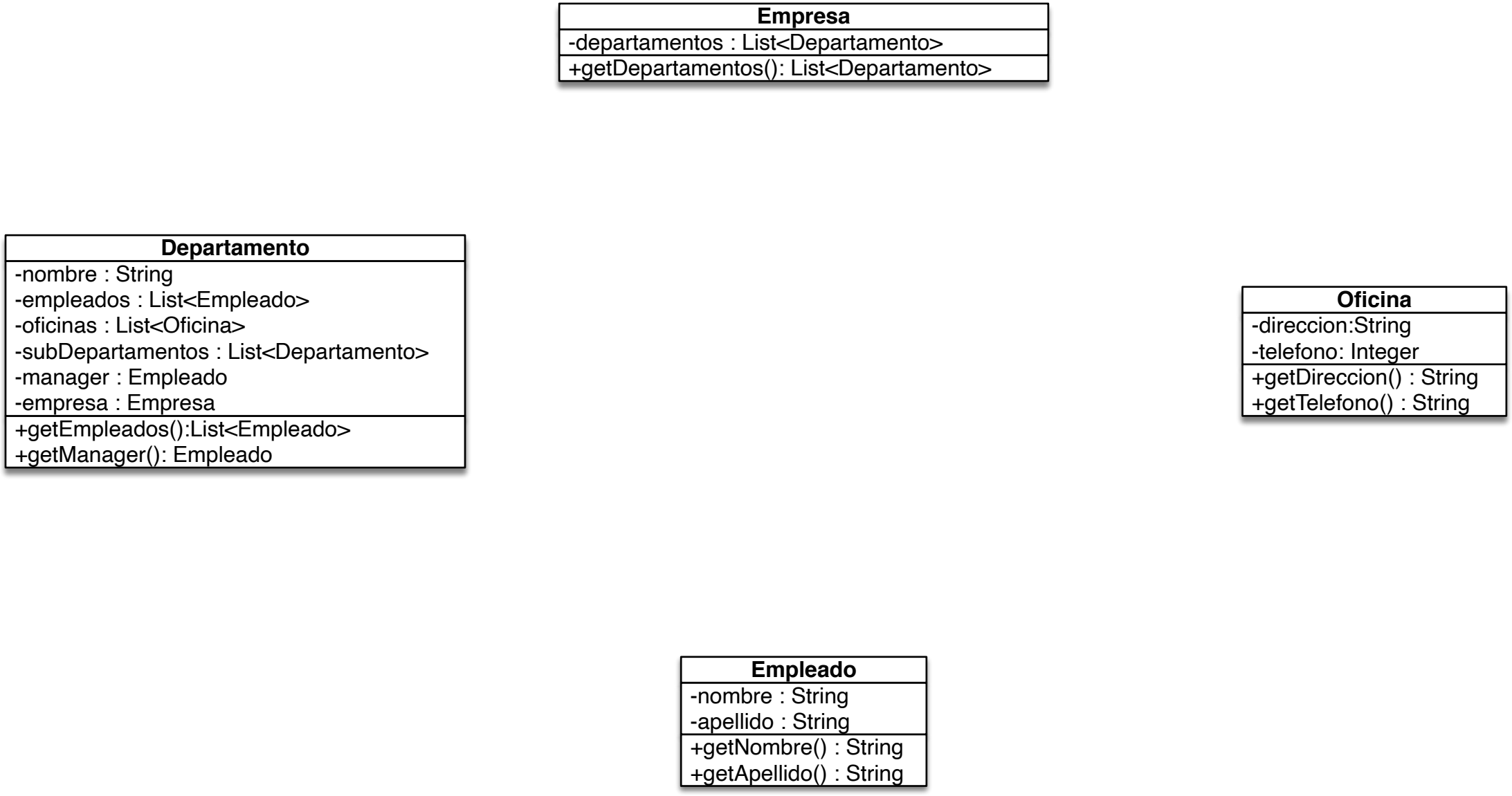
Relación de Realización

- Indica una relación donde una de las partes especifica un “contrato” y la otra parte garantiza llevarlo a cabo
- Es una mezcla entre dependencia y generalización
- Usamos realizaciones para modelar conexiones del tipo: “Implementa”
- Se usan principalmente para especificar la relación entre una interface y la clase que provee una operación para ella



Ejercicio – Constuyendo el diagrama

- Se desea modelar el sistema de personal de una empresa. La empresa esta formada por departamentos. Los departamentos pertenecen a una única empresa. En cada departamento trabajan uno o mas empleados (cada empleado trabaja en un único departamento) y uno de ellos es el jefe. Cada departamento tiene una o mas oficinas pero una oficina puede corresponder a mas de un departamento. Ademas, los departamentos pueden estar compuestos por otros departamentos.

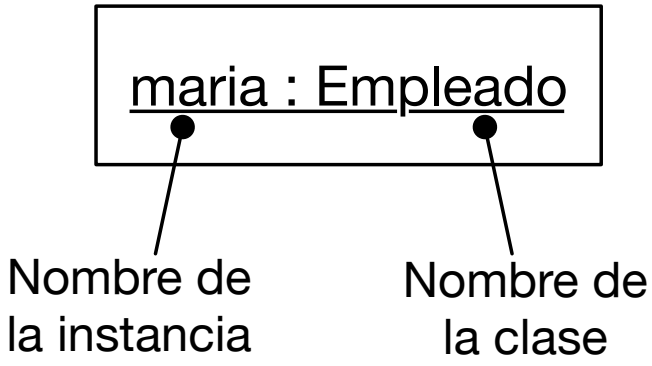

Empresa
-departamentos : List<Departamento>
+getDepartamentos(): List<Departamento>
Departamento
-nombre : String
-empleados : List<Empleado>
-oficinas : List<Oficina>
-subDepartamentos : List<Departamento>
-manager : Empleado
-empresa : Empresa
+getEmpleados():List<Empleado>
+getManager(): Empleado
Oficina
-direccion:String
-telefono: Integer
+getDireccion() : String
+getTelefono() : String
Empleado
-nombre : String
-apellido : String
+getNombre() : String
+getApellido() : String

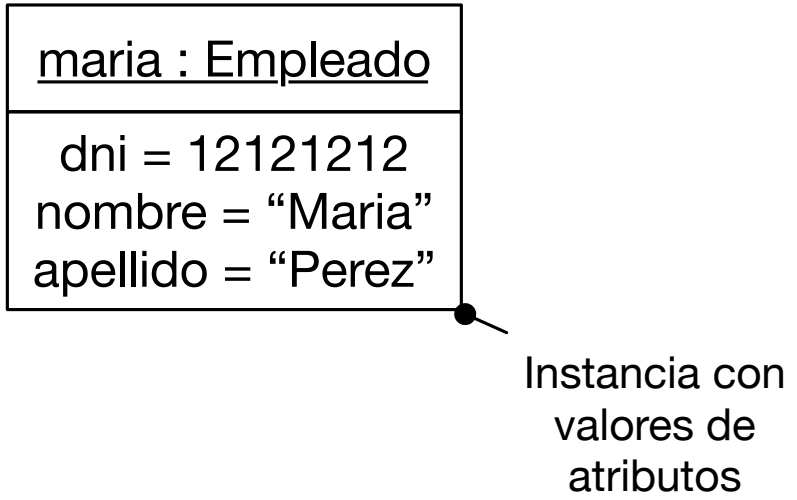
Ejercicio – Ingenieria reversa

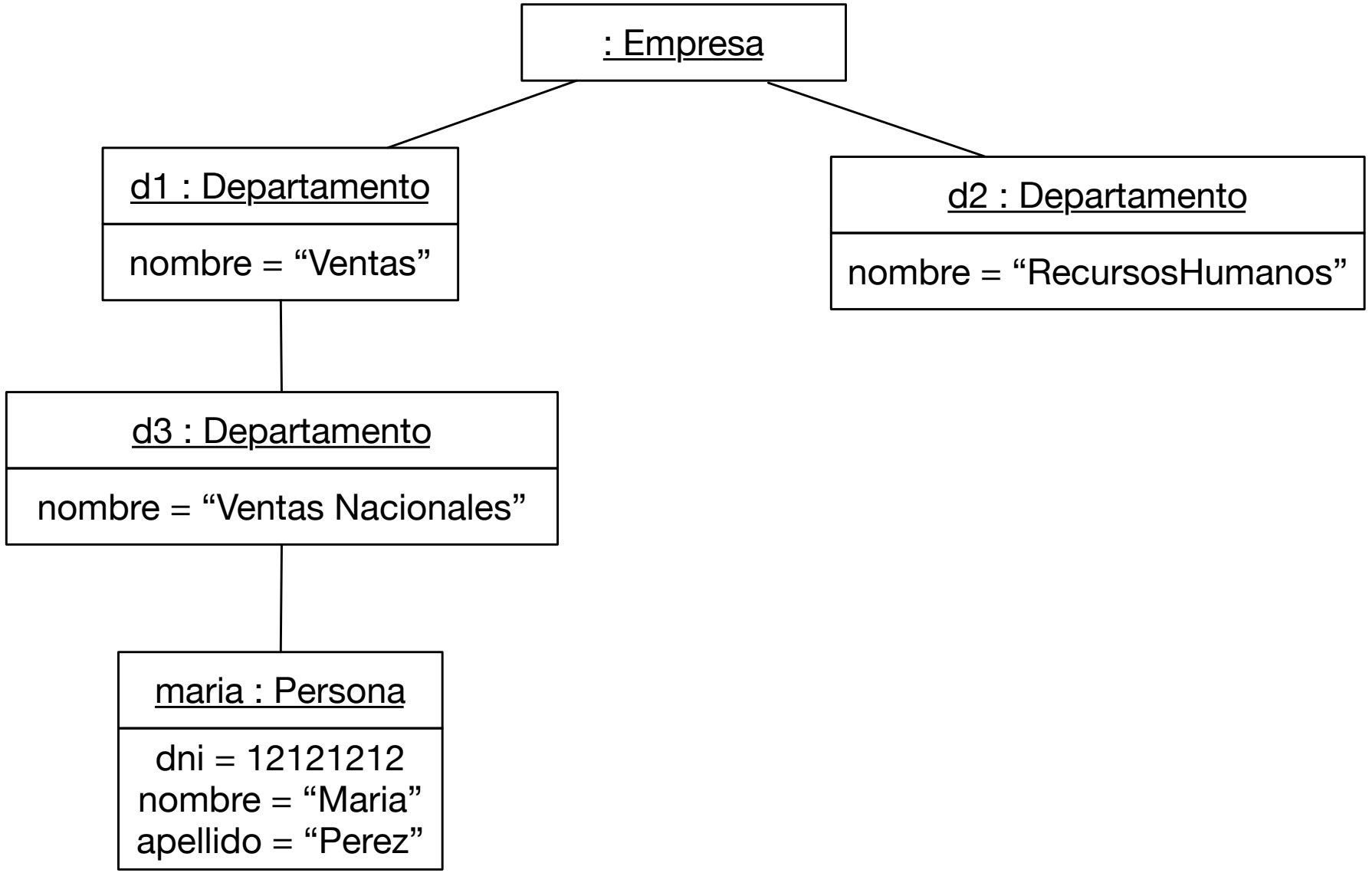
- Realice el diagrama de clases a partir del código fuente

Diagrama de objetos

- Modelan las instancias de las clases contenidas en un diagrama de clases
- Muestra un conjunto de objetos y sus relaciones en un momento dado
- Gráficamente el objeto se dibuja como un rectángulo con su nombre de instancia y clase subrayados






: Empresa
d1 : Departamento
nombre = “Ventas”
d2 : Departamento
nombre = “RecursosHumanos”
d3 : Departamento
nombre = “Ventas Nacionales”
maria : Persona
dni = 12121212
nombre = “Maria”
apellido = “Perez”