UDLAP
UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

Diagramas de clases UML: Repaso

Dr. José Luis Zechinelli Martini
joseluis.zechinelli@udlap.mx
LDS – 1101

El material presentado está basado en los libros de [Booch *et al*.] y [Joyanes].

Unified Model Language (UML)

- UML es un **lenguaje estándar de modelado** para el desarrollo de sistemas y de software:
  - Permite gestionar la complejidad en el diseño de sistemas
  - Permite representar y modelar la información durante las fases de análisis y diseño de un sistema
- Un **modelo**:
  - Es una abstracción de cosas reales
  - Ignora los detalles irrelevantes
  - Es una simplificación del sistema real

UML (continuación)

- La **notación** de UML:
  - Son signos convencionales utilizados para expresar conceptos
  - Está definida por diferentes elementos: Pseudocódigo, código escrito en algún lenguaje de programación, dibujos, programas, descripciones, etc.
- El **bloque básico de construcción** de UML es un diagrama
- Tipos de **diagramas**:
  - De propósito general (*e.g.*, diagramas de clases)
  - De propósito específico (*e.g.*, diagramas de tiempo)

DIAGRAMAS DE CLASES

Clases: Definición

- Una clase es un tipo definido por el usuario:
  - Es un **conjunto de objetos** que comparten una estructura y comportamiento comunes [Booch et al.]
  - Contiene la especificación de los **datos** que describen un objeto junto con la descripción de las **acciones** que un objeto conoce cómo ha de ejecutar
- Una clase puede representar un concepto:
  - **Tangible y concreto** tal como un avión
  - **Abstracto** tal como un documento
  - **Intangible** tal como inversiones de alto riesgo

Clases: Representación (1)

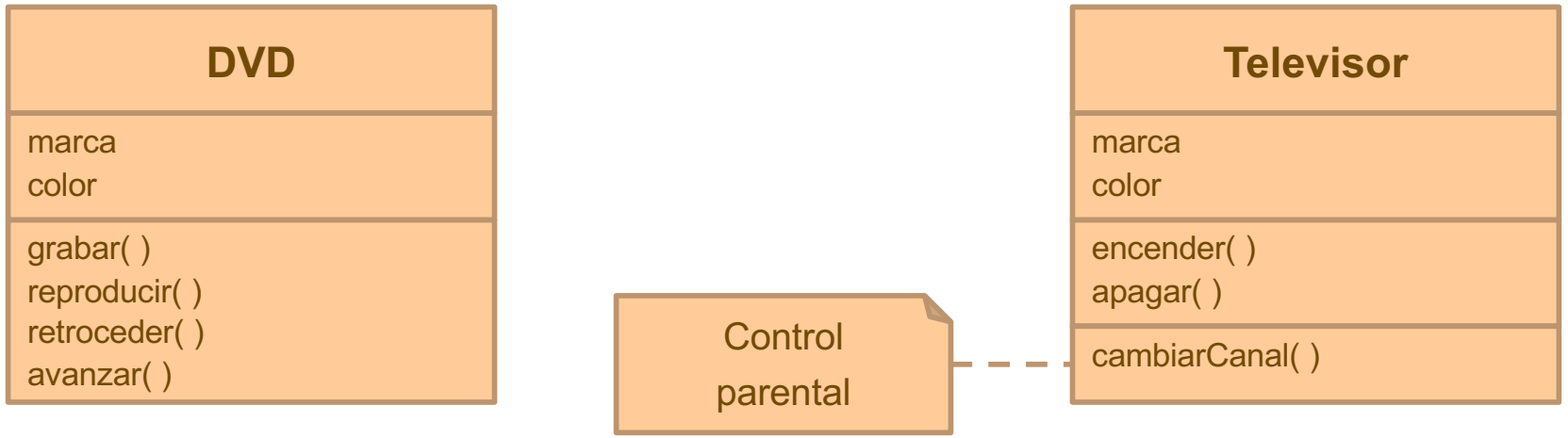
- Una clase **se representa** con una caja rectangular dividida en tres compartimentos (secciones o bandas):
  - El primer compartimento contiene el nombre de la clase
  - El segundo contiene los atributos o propiedades
  - El tercero se utiliza para las operaciones
- Se puede **ocultar o quitar** cualquier compartimento de la clase para aumentar la legibilidad del diagrama:
  - Cuando se oculta un compartimento, no significa que esté vacío
  - Se pueden agregar compartimentos para mostrar información adicional, tal como excepciones o eventos

Clases: Representación (2)

- En UML se propone que el **nombre** de una clase:
  - Comience con una letra mayúscula
  - Esté centrado en el compartimento superior
  - Sea escrito en tipo de letra (fuente) negrita
  - Sea escrito en cursivas cuando la clase sea abstracta
- Ejemplos de **diagramas** para especificar una clase:

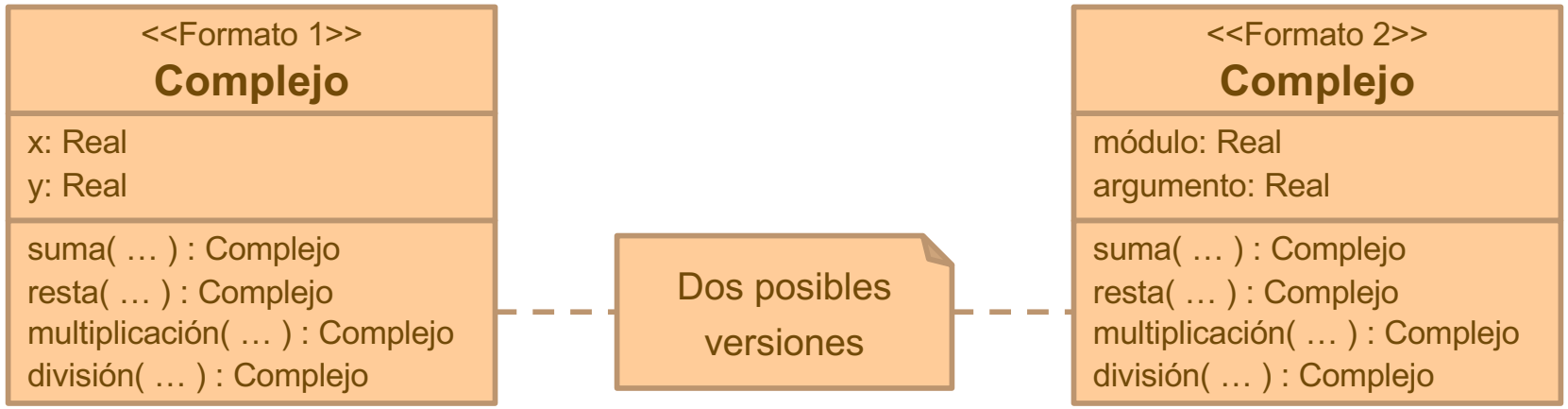
Clases: Ejemplos (1)

- DVD: Reproductor y grabador de videos digitales
- Televisor:
  - Utilizado por adultos y niños
  - Ofrece un alto grado de abstracción gracias a sus operaciones elementales



Clases: Ejemplos (2)

- Números complejos*:
  - En un espacio vectorial se representan usando dos componentes, la primera componente, **x**, se denomina parte real y, la segunda, **y**, parte imaginaria
  - También se pueden representar usando la forma polar o forma **módulo-argumento**



* http://wmatem.eis.uva.es/~matpag/CONTENIDOS/Complejos/complejos.htm

Clases: Atributos

- Un atributo es una propiedad o característica de una clase y describe un rango de valores que la propiedad podrá contener en los objetos:
  - Una clase puede contener **varios atributos o ninguno**
  - Los atributos de una clase son las partes de información que **representan de manera general** el estado de cualquier objeto
- Un atributo se define en un diagrama de clase:
  - **Formato**: *nombre : tipo = valor_por_defecto*
  - Se puede mostrar **en línea** o **en relaciones** entre clases

Clases: Métodos

- Un método es una operación (acción):
  - Que los objetos de la clase **pueden realizar** o
  - Que **se puede hacer** sobre los objetos de la clase
  - **Formato**: *nombre(parámetros) : tipo_retorno*
- Los métodos son las operaciones de la clase que especifican el modo de invocar un comportamiento específico:
  - Un método específica **qué hace una clase**, pero no necesariamente **cómo lo hace**
  - UML hace una **diferencia clara** entre la invocación de un método y su implementación

Clases: Restricciones

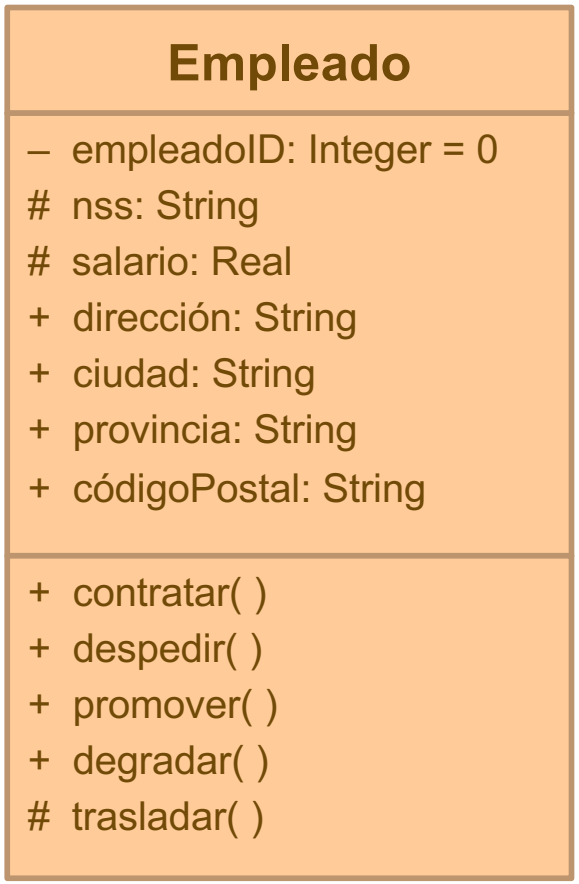
- Responsabilidad:
  - Es un **contrato** o una obligación
  - Debe incluir **información** suficiente para describir una clase sin ambigüedades
    - Se escriben debajo del compartimento de operaciones
- Restricción:
  - Es una **manera formal** de eliminar la ambigüedad
  - **Formato**: { *atributo* θ *valor(es)* }, donde $\theta \in$ { =, <, >, <=, >=, != }
    - UML permite definir restricciones usando el lenguaje OCL (*Object Constraint Language*)

Clases: Reglas de visibilidad (1)

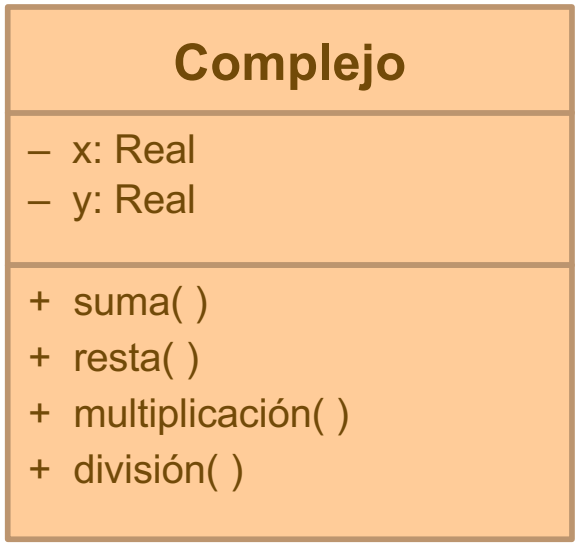
- Complementan o refinan el concepto de encapsulamiento
- Permiten especificar el nivel de visibilidad de los atributos y de las operaciones definiendo qué clases:
  - Tienen acceso a visualizar y cambiar los atributos
  - Pueden ejecutar las operaciones de una clase

Nivel de visibilidad	Símbolo UML	Descripción
Público	+	Es visible a todas las clases
Privado	–	No es visible a ninguna otra clase
Protegido	#	La clase y sus descendientes tienen acceso

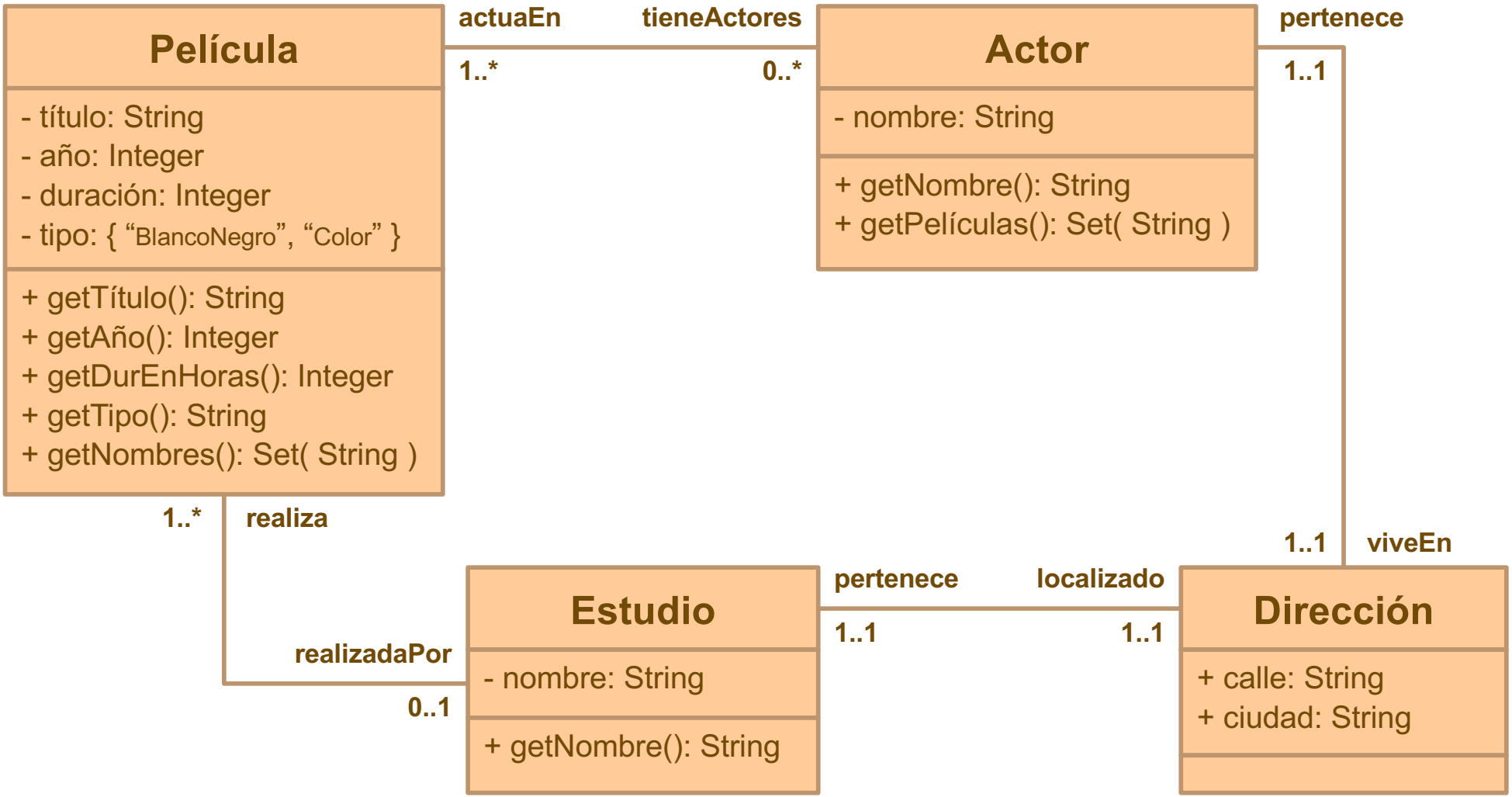
Clases: Reglas de visibilidad (2)



- En general, se recomienda visibilidad privada o protegida para los atributos



Clases: Reglas de visibilidad (3)



Clases: Declaración de objetos

- En algunos lenguajes de programación se requiere un proceso de dos etapas:
  - Declaración:
    - Empleado **emp**;
    - La variable **emp** no define un objeto simplemente una variable que puede referir un objeto de tipo Empleado
  - Asignación:
    - emp = **nuevo** Empleado();
    - El operador "**nuevo**" asigna un espacio de memoria, cuya referencia es almacenada en la variable emp