ENTORNOS DE DESARROLLO

UNIDAD 5: Lenguaje Unificado de Modelado (UML)

INDICE

Introducción a UML

Para realizar labores de diseño de software, es vital realizar modelados o diagramas representando gráficamente la estructura de la aplicación y su funcionalidad, de una manera fácil de entender y rápida de crear. Antiguamente, los diagramas de cada diseñador eran únicos, ya que, salvo un par de diagramas conocidos por todos (como los diagramas de flujo o los diagramas de entidad-relación), cada diseñador o analista realizaba los diagramas de la manera que mejor los entendía, por lo que podría suponer un problema si varias personas tenían que entender los diagramas que uno o varios diseñadores les presentaban para definir el software.

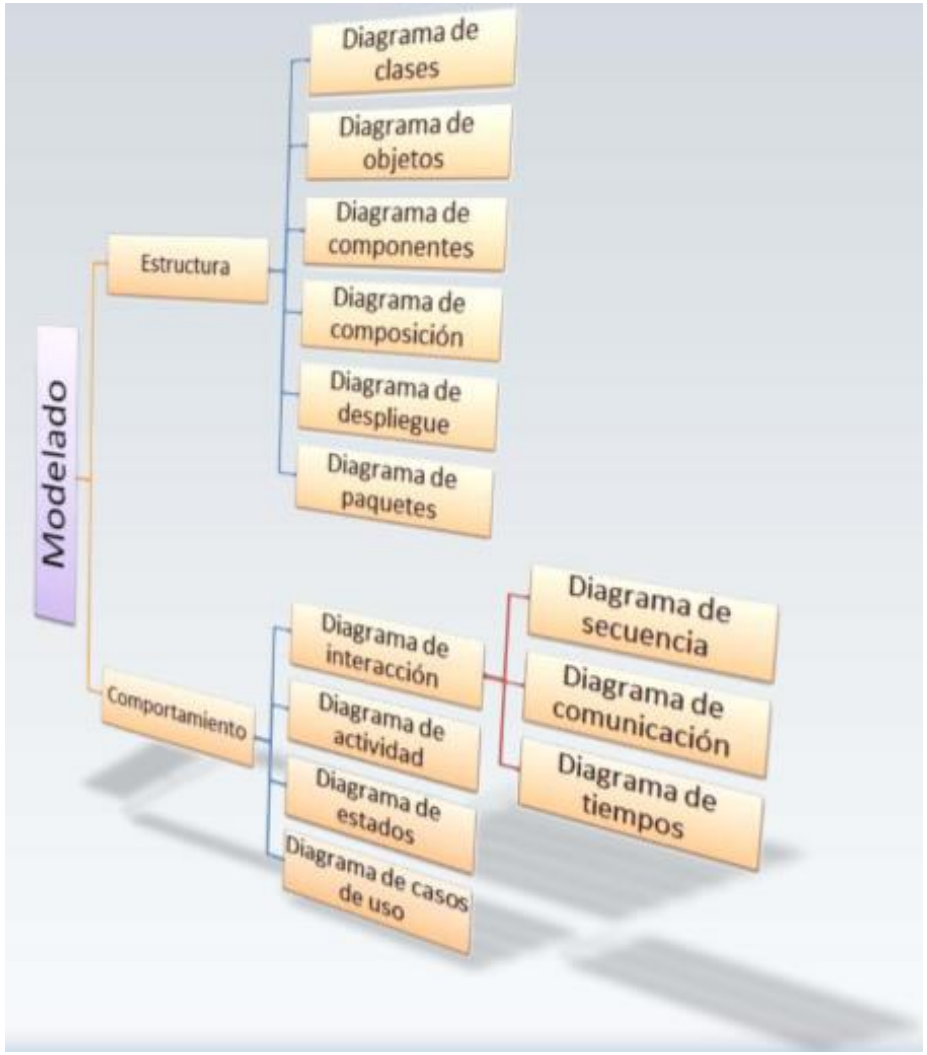
Con el fin de evitar ese problema se creó UML (Unified Modeling Language), un conjunto unificado de estándares para las diferentes necesidades y usos que un diseñador pudiera tener a la hora de plantear una representación gráfica de un programa. Son diagramas de propósito general que se presuponen conocidos por todos, con unas técnicas de notación conocidas, de modo que cualquiera pueda crear diagramas entendibles por todos.

Introducción a UML. UML 2.0

En la versión 2.0 de UML se definió un completo árbol de superestructuras donde se relacionaban y complementaban todos los tipos de diagramas UML a la hora de definir un software por completo.

Teniendo presente la superestructura de diagramas de UML, podríamos pensar que la tarea de definir y realizar dichos diagramas (y que por supuesto sean coherentes entre sí) sería una tarea abrumadora, pero no es necesario ni se suelen realizar todos los diagramas para modelar un software, por norma general se utilizan solo una serie de diagramas para modelarlo, lo más clásico sería la tríada diagrama de clases, de secuencia y de casos de uso.

Introducción a UML. UML 2.0



Diagramas de CLASES

Diseño de clases en UML

Estos diagramas son particularmente útiles en el desarrollo de la capa del modelo, para mapear de un modo gráfico las entidades y sus relaciones en nuestra aplicación.

ELEMENTOS:

- **CLASES.** Agrupan conjunto de objetos con características comunes, llamadas **ATRIBUTOS** y su comportamiento llamados **METODOS.** Tendrán una visibilidad.
- **RELACIONES.** Entre los elementos del sistema que componen las CLASES. Pueden ser ASOCIACIÓN, AGREGACIÓN, COMPOSICIÓN y GENERALIZACIÓN.
- **NOTAS.** Es un cuadro para escribir comentarios para ayudar al entendimiento del diagrama.
- **ELEMENTOS DE AGRUPACIÓN.** Se utiliza para modelar un sistema grande, agrupando en paquetes que se relacionan entre si.

Diseño de clases en UML

En primer lugar, debemos percatarnos de que las clases se definen mediante cuadrados, y dichos cuadrados se dividen en tres segmentos horizontales.

- 1º) tendríamos el nombre de la clase
- 2º) Seguidamente irían los atributos
- 3º)Al final, los métodos de la clase.

Es importante resaltar que si alguna clase no posee atributos propios o métodos propios, se deberá seguir dibujando el segmento que le corresponde, incluso si está vacío.

El carácter que precede al nombre del atributo o método se corresponde con su grado de comunicación o **visibilidad**.

+ Público, el elemento será visible tanto desde dentro como desde fuera de la clase.

- Privado, el elemento de la clase solo será visible desde la propia clase.

#Protegido, el elemento no será accesible desde fuera de la clase, pero podrá ser manipulado por los demás métodos de la clase o de las subclases.

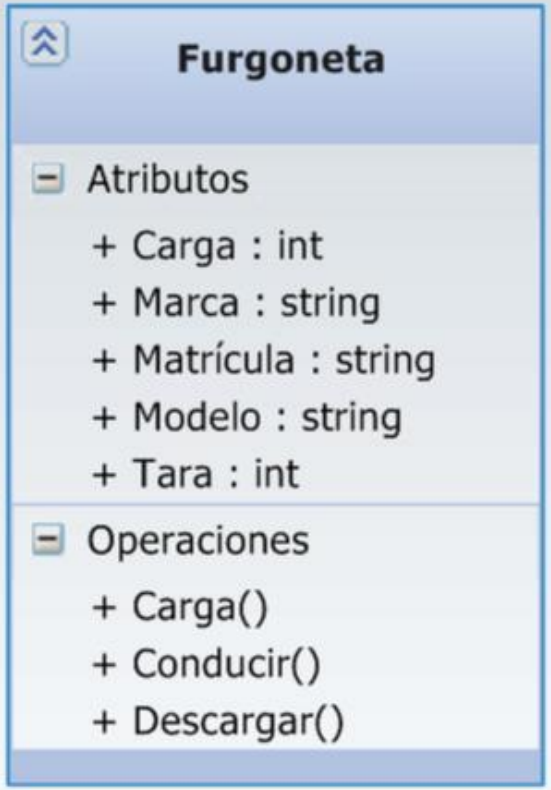
~ paquete/defecto, define la visibilidad del paquete que puede ser utilizada por otra clase mientras esté en el mismo paquete.

Diseño de clases en UML

DISEÑO DE CLASES EN UML

Cada bloque dentro del diagrama representa a una clase; una clase, como sabemos, dispone de unos atributos y de unos métodos asociados. Representaríamos las clases como cajas en donde escribiríamos sus atributos y sus métodos.

Las clases representan a nuestros objetos, los atributos definen las propiedades y características del objeto y los métodos especifican las acciones que podemos realizar con dicho objeto.



Diseño de clases en UML

Relaciones

En el diagrama de clases, además de dibujar y representar las clases con sus atributos y métodos, también se representan las relaciones.

Las relaciones se representan con flechas con una forma determinada, y, además, poseen una cardinalidad.

La cardinalidad es un número o símbolo que representa al número de elementos de cada clase en cada relación, pudiendo usar el comodín “*” para definir un número indeterminado de elementos.

Diseño de clases en UML

Relaciones

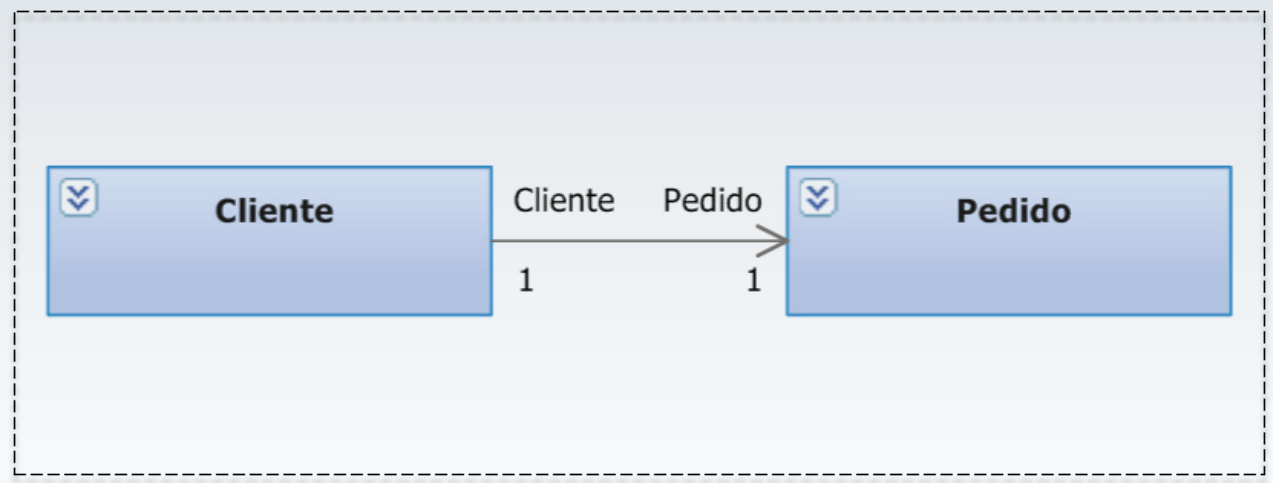
Cardinalidad	Significado
1	Uno y solo uno
0..1	Cero o uno
X..Y	Desde X hasta Y
*	Cero o varios
0..*	Cero o varios
1..*	Uno o varios

Diseño de clases en UML

Relaciones

Asociación

La asociación es la más básica de las relaciones, no tiene un tipo definido y puede ser tanto una composición como una agregación, además una asociación podría implicar únicamente un uso del objeto asociado. Se representan mediante una flecha simple que también puede tener una cardinalidad.

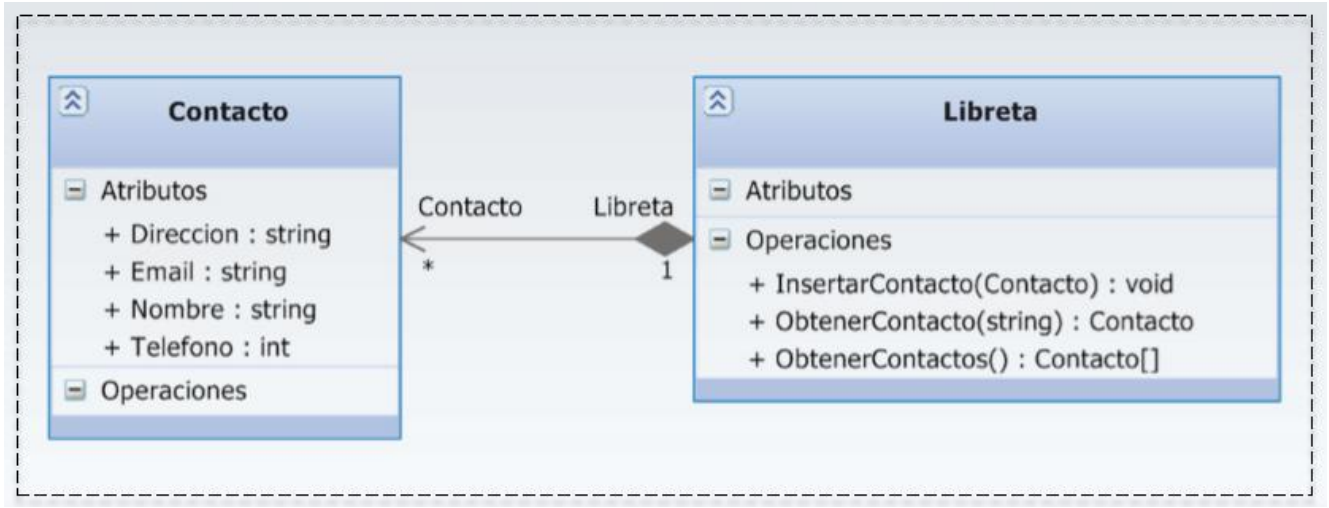


Diseño de clases en UML

Relaciones

Composición

La composición define los componentes de los que se compone otra clase, se define además que la clase que contiene la composición no tiene sentido de existencia si la(s) agregada(s) desaparece(n). Mediante esta relación, denotada por una flecha con un rombo relleno en una de sus puntas, se indica la presencia de las clases origen en la clase destino, donde la clase destino es apuntada por el rombo de la relación.



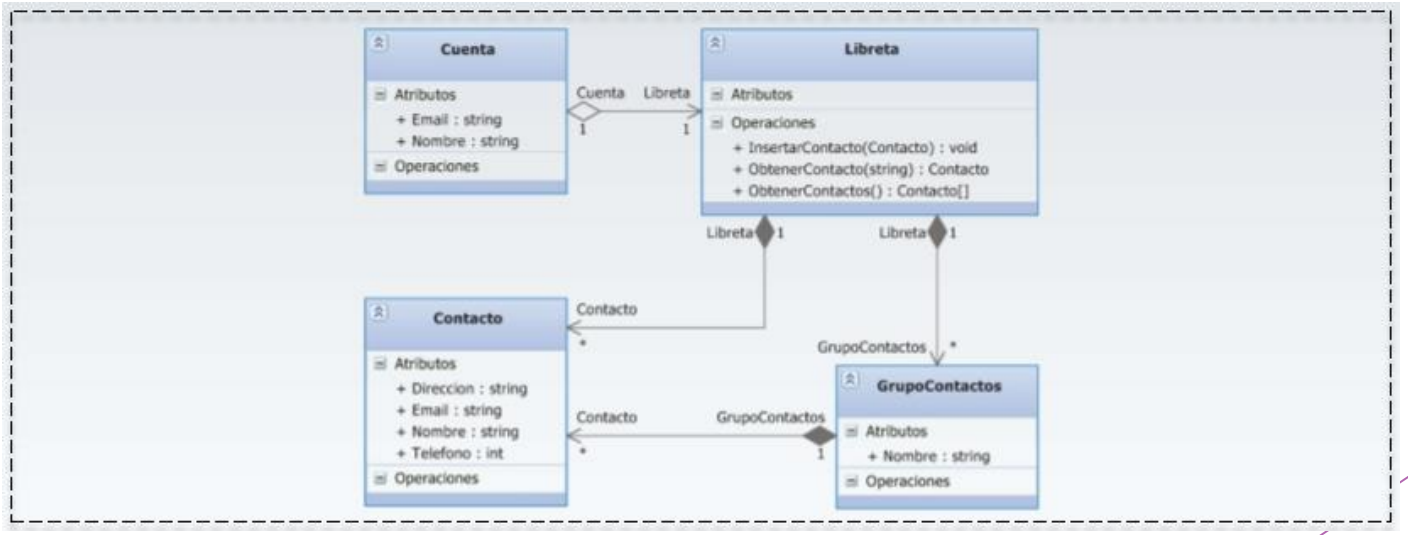
Diseño de clases en UML

Relaciones

Agregación

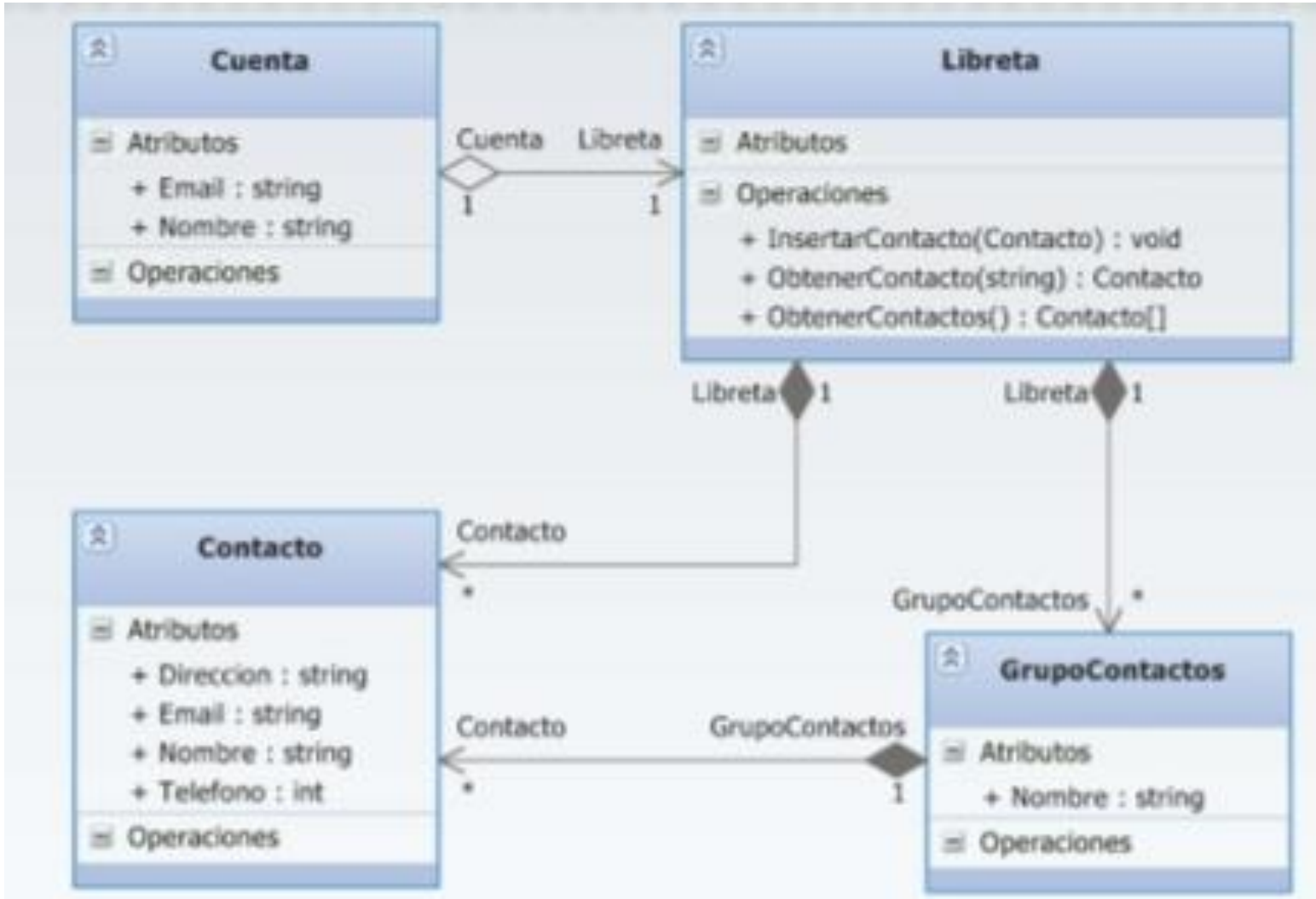
La agregación es un tipo de asociación que indica que una clase es parte de otra clase. A diferencia que en la composición, la destrucción del compuesto no conlleva la destrucción de los componentes. Habitualmente se da con mayor frecuencia que la composición.

La agregación se representa en UML mediante un rombo de color blanco colocado en el extremo en el que está la clase que representa el “todo”



Diseño de clases en UML

Relaciones: AGREGACION

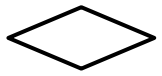


Diseño de clases en UML

Relaciones

Agregación vs Composición

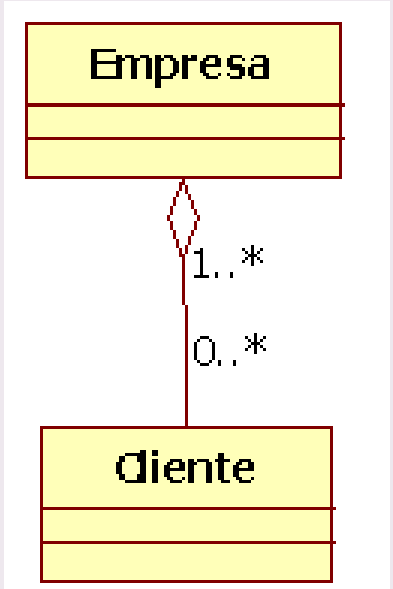
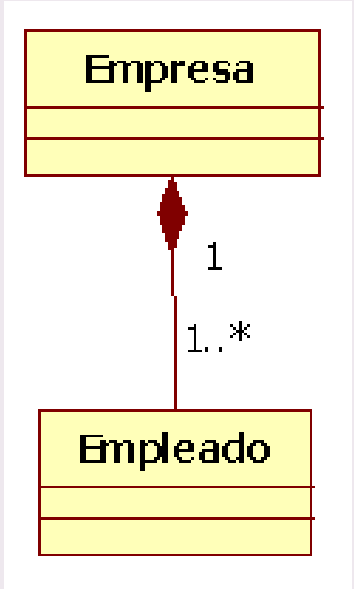
	Agregación	Composición
Varias asociaciones comparten los componentes	Sí	No
Destrucción de los componentes al destruir el compuesto	No	Sí
Cardinalidad a nivel de compuesto	Cualquiera	0..1 ó 1
Representación	Rombo transparente	Rombo negro



Diseño de clases en UML

Relaciones

Agregación vs Composición

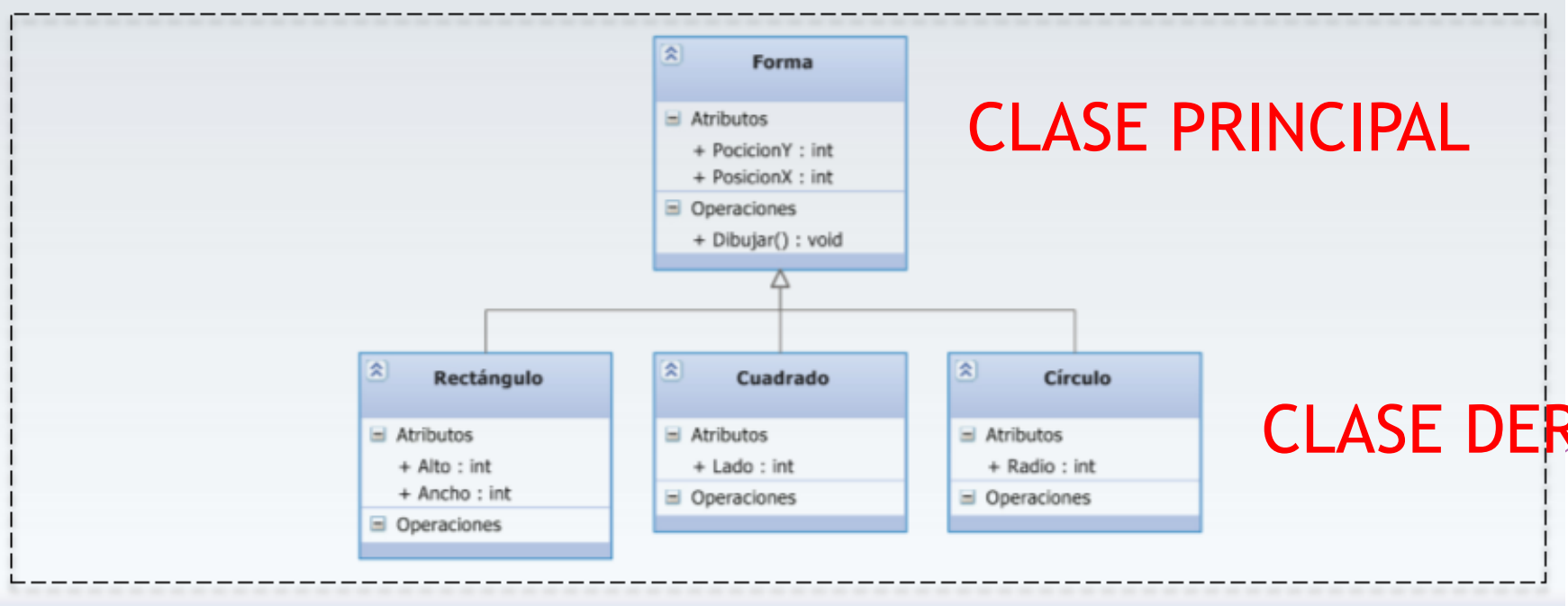
Agregación	Composición
	

Diseño de clases en UML

Relaciones

Herencia

La herencia es un modo de representar clases y subclases, es decir, clases más específicas de una general, se representan mediante una flecha con una punta triangular vacía.

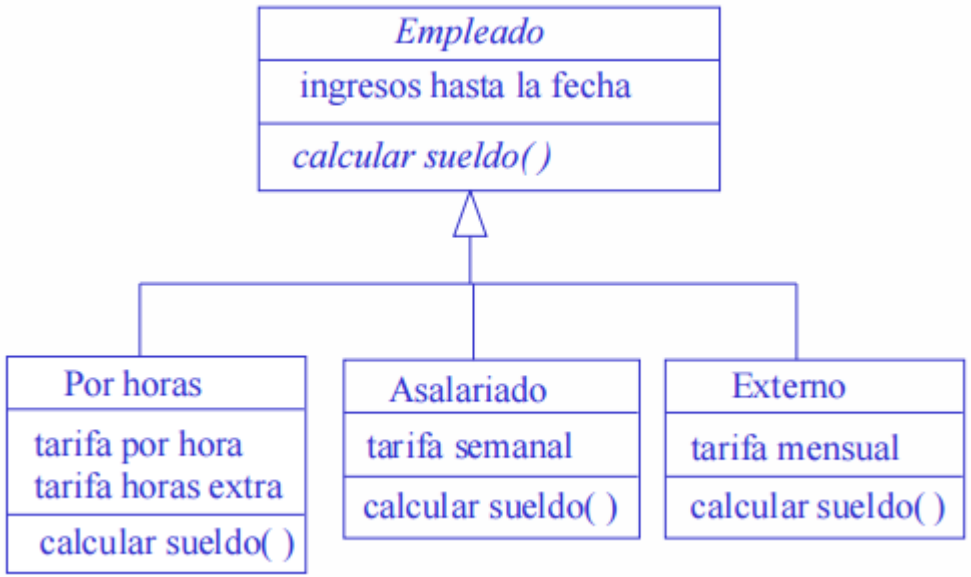


Diseño de clases en UML

Clases abstractas

Una clase abstracta es una clase sin instancias directas, es decir que no se puede instanciar. Una clase abstracta es una clase que contiene uno o más métodos abstractos, es decir, que no están implementados. La clase heredera tiene que implementar todos los métodos abstractos de la clase abstracta o ser también una clase abstracta.

Para representar una clase abstracta en UML escribiremos su nombre en cursiva.



Diseño de clases en UML

Visibilidad

Para especificar la visibilidad de un miembro de la clase, atributo o método, se coloca uno de los siguientes signos delante:

+	Público
-	Privado
#	Protegido
/	Derivado (se puede combinar con otro)
~	Paquete

Diseño de clases en UML

Ejemplo 01- Tienda de Música

Diseño de clases en UML

Ejemplo 02- EMPRESA

Representa mediante un diagrama de clase las siguientes especificaciones:

- Una aplicación necesita almacenar información sobre empresas, sus empleados y sus clientes. Ambos se caracterizan por su nombre y edad.
- Los empleados tienen un sueldo bruto, los empleados que son directivos tienen una categoría, así como un conjunto de empleados subordinados.
- De los clientes además se necesita conocer su teléfono de contacto.
- La aplicación necesita mostrar los datos de empleados y clientes.

Diseño de clases en UML

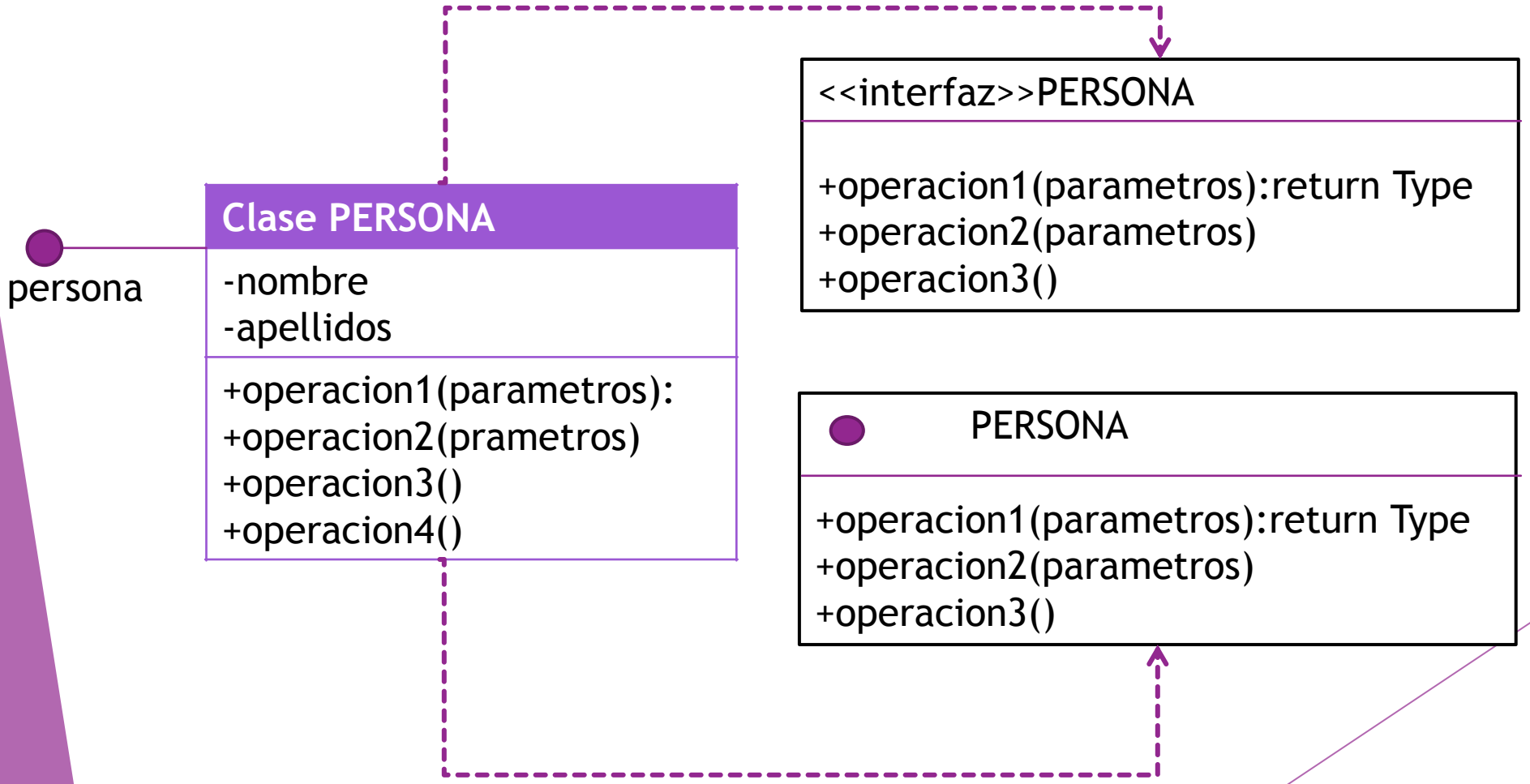
Interfaces

Una interfaz (interface) contiene la declaración de un conjunto de operaciones sin su implementación correspondiente y se usa para especificar un servicio proporcionado por una clase o un componente. Una interfaz no tiene instancias.

Se representa con el símbolo de un clasificador (rectángulo), precediendo el nombre con el estereotipo <<interface>>, o con una línea con un círculo en el extremo, etiquetado con el nombre de la interfaz.

Diseño de clases en UML

3 Formas de representar los Interfaces



FIN