SANTA ANA
Y SAN RAFAEL
Madrid

COLEGIOS
MARIANISTAS

DISEÑO ORIENTADO A OBJETOS

UD 5 DIAGRAMAS DE CLASE

ENTORNOS DE DESARROLLO – 1ero DAM

DISEÑO ORIENTADO A OBJETOS

DIAGRAMAS UML

- ✓ Para realizar labores de **diseño** es fundamental modelar y representar gráficamente la estructura de la aplicación o programa y su funcionalidad.
- ✓ En la **programación orientada a objetos**, y por tanto, en la fase de **diseño orientado a objetos**, utilizaremos representaciones gráficas de las clases y sus relaciones.
- ✓ La manera de unificar las representaciones de los diagramas es creando un estándar. Así surgió el estándar **UML**.
- ✓ El **Lenguaje Unificado de Modelado: UML *Unified Modeling Language***. Es un conjunto unificado de estándares para distintas necesidades de representación de diagramas.
- ✓ Son **diagramas de propósito general** que se presupone conocido por diseñadores, con **técnicas de notación** conocidas.

DISEÑO ORIENTADO A OBJETOS

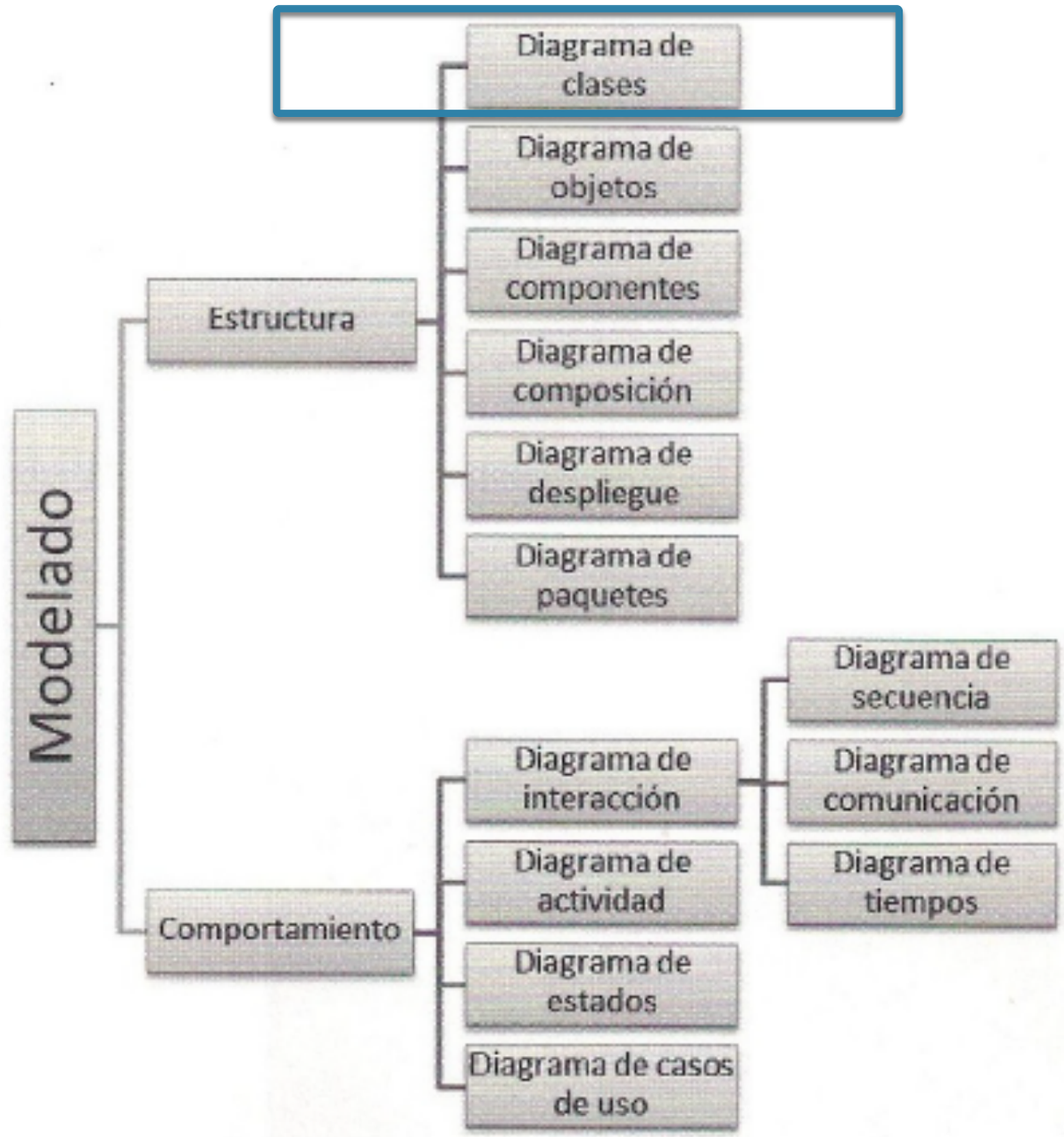
DIAGRAMAS UML

- ✓ El **UML** fue creado para forjar un **lenguaje de modelado visual** común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en **estructura** como en **comportamiento**.
- ✓ **UML** tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación.
- ✓ **UML** da soporte a una gran cantidad de metodologías de software. Define la semántica mediante una serie de reglas y notaciones pero no especifica cuál sería la metodología, el procedimiento o el lenguaje de programación a utilizar.
- ✓ Existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML y en particular, con el análisis y diseño orientado a objetos.

DISEÑO ORIENTADO A OBJETOS

DIAGRAMAS UML

- ✓ El **UML** se perfecciona continuamente.
- ✓ Actualmente estamos en la versión **UML 2.0** que extiende las especificaciones de UML para cubrir más aspectos de desarrollo, incluido **Agile**.
- ✓ En esta versión se ha definido un árbol completo de todos los tipos de diagramas UML, para definir un software completamente. Es lo que se denomina **superestructura de diagramas UML**.



DISEÑO ORIENTADO A OBJETOS

SUPERESTRUCTURA DE DIAGRAMAS UML

1) **Diagramas de estructura** (Qué componentes hay)

Diagrama de clases: Describe los diferentes tipos de objetos en un sistema y las relaciones existentes entre ellos.
Diagrama de objetos: (También llamado Diagrama de instancias) Foto de los objetos en un sistema en un momento del tiempo.
Diagrama de paquetes: Muestra la estructura y dependencia entre paquetes (librerías), los cuales permiten agrupar elementos (no solamente clases) para la descripción de grandes sistemas.
Diagrama de despliegue: Muestra la relación entre componentes o subsistemas software y el hardware donde se despliega o instala.
Diagrama de estructura compuesta: Descompone jerárquicamente una clase mostrando su estructura interna.
Diagrama de componentes: Muestra la jerarquía y relaciones entre componentes de un sistema software.

SUPERESTRUCTURA DE DIAGRAMAS UML

2) **Diagramas de comportamiento** (Cómo se comportan)

Diagramas de casos de uso: Permite capturar los requerimientos funcionales de un sistema.

Diagrama de estado: Permite mostrar el comportamiento de un objeto a lo largo de su vida.

Diagrama de actividad: Describe la lógica de un procedimiento, un proceso de negocio o *workflow*.

Diagramas de interacción (Subgrupo dentro de los diagramas de comportamiento): Describen cómo los grupos de objetos colaboran para producir un comportamiento. A su vez, se descomponen en los siguientes:

SUPERESTRUCTURA DE DIAGRAMAS UML

2) **Diagramas de comportamiento** (Cómo se comportan)

Diagramas de interacción: A su vez, se descomponen en los siguientes:

Diagrama de secuencia: Muestra los mensajes que son pasados entre objetos en un escenario concreto.

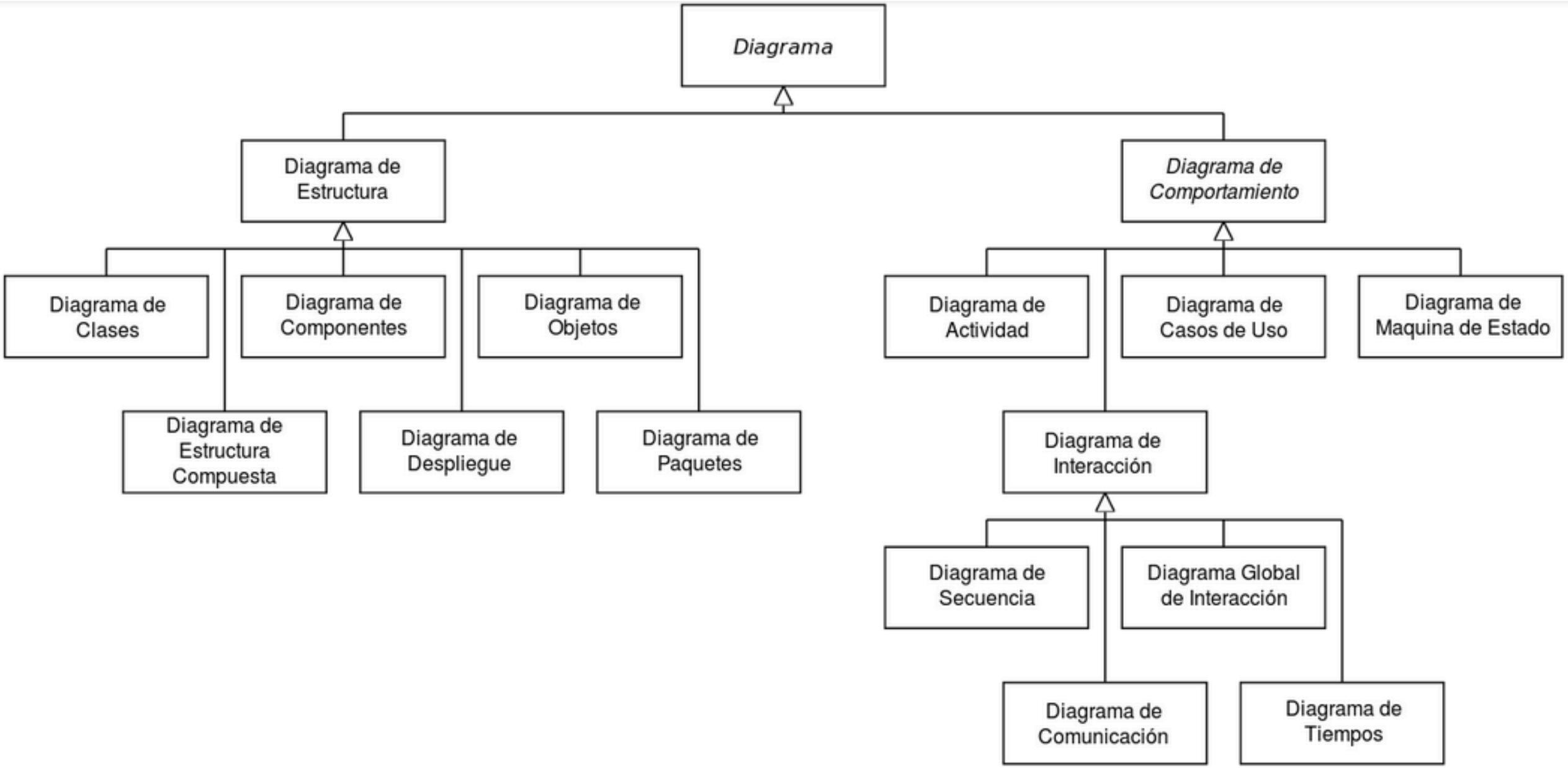
Diagrama de comunicación: Muestra las interacciones entre los participantes haciendo énfasis en la secuencia de mensajes.

Diagrama de (visión de conjunto o resumen de) interacción: Se trata de mostrar de forma conjunta diagramas de actividad y diagramas de secuencia.

Diagrama de tiempo: Pone el foco en las restricciones temporales de un objeto o un conjunto de objetos.

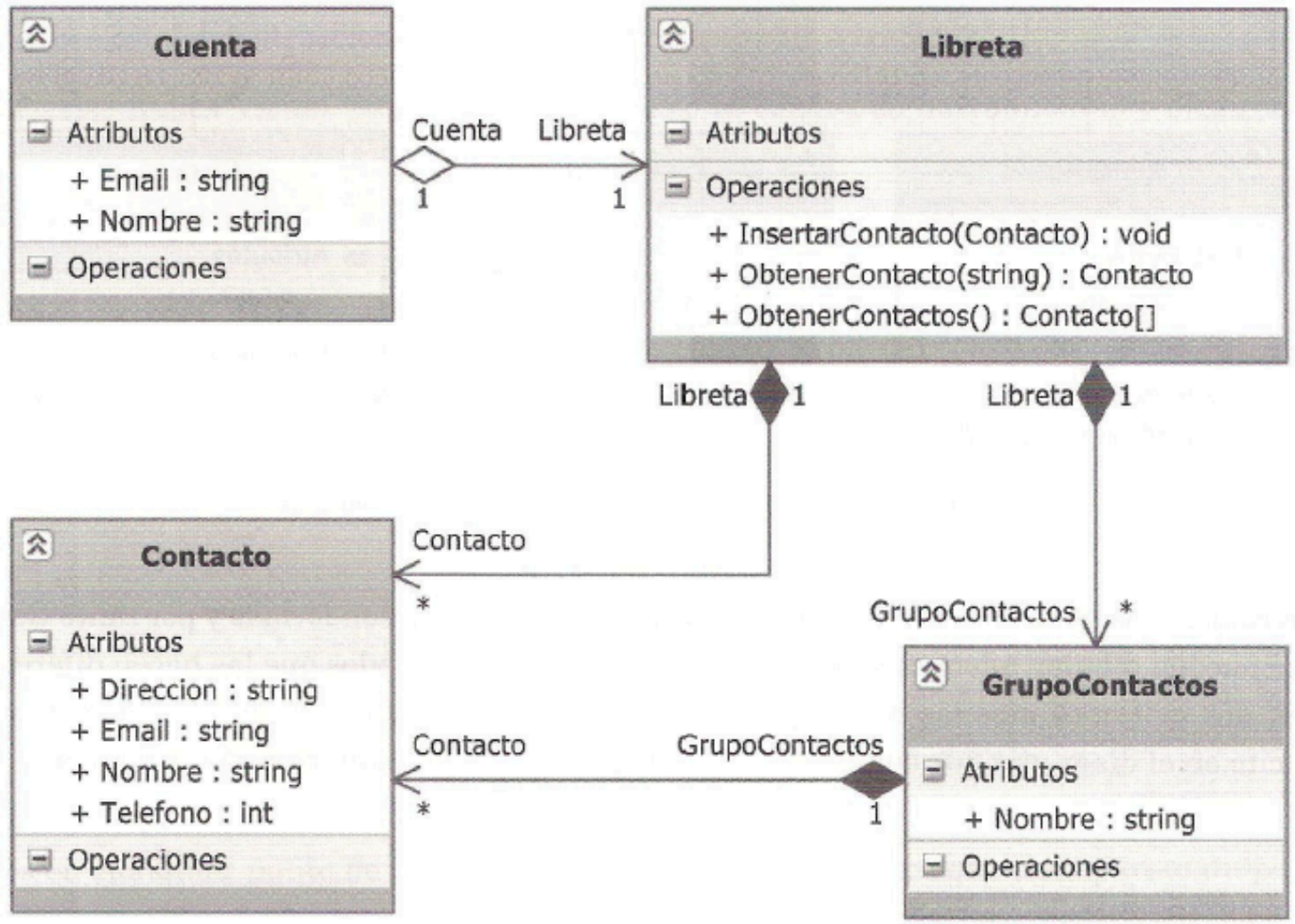
DISEÑO ORIENTADO A OBJETOS

DIAGRAMAS UML



DIAGRAMAS DE CLASE UML

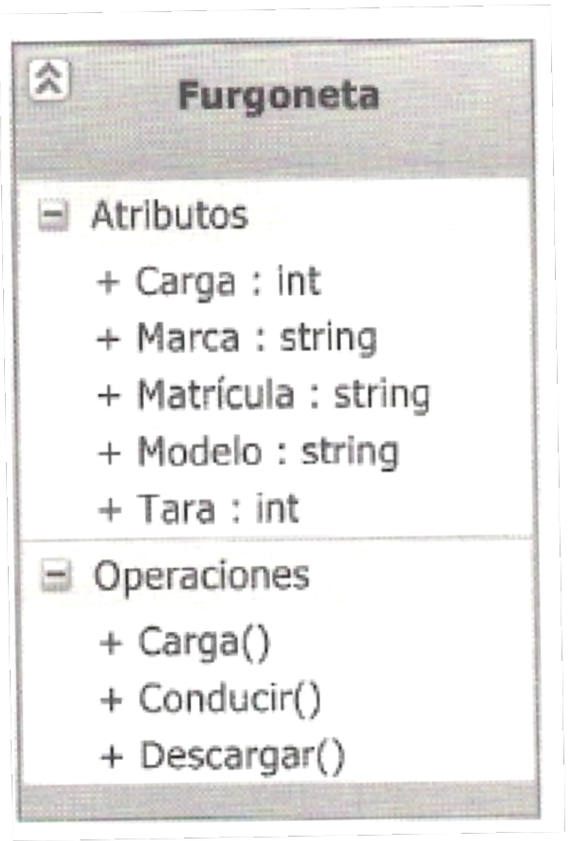
- ✓ Es el diagrama UML más básico.
- ✓ Tiene gran parecido con el diagrama clásico de entidad-relación.
- ✓ Se basan en ciertas reglas y notaciones sencillas para relacionar las clases y sus diferentes operaciones entre sí.
- ✓ En la POO es un recurso fundamental.

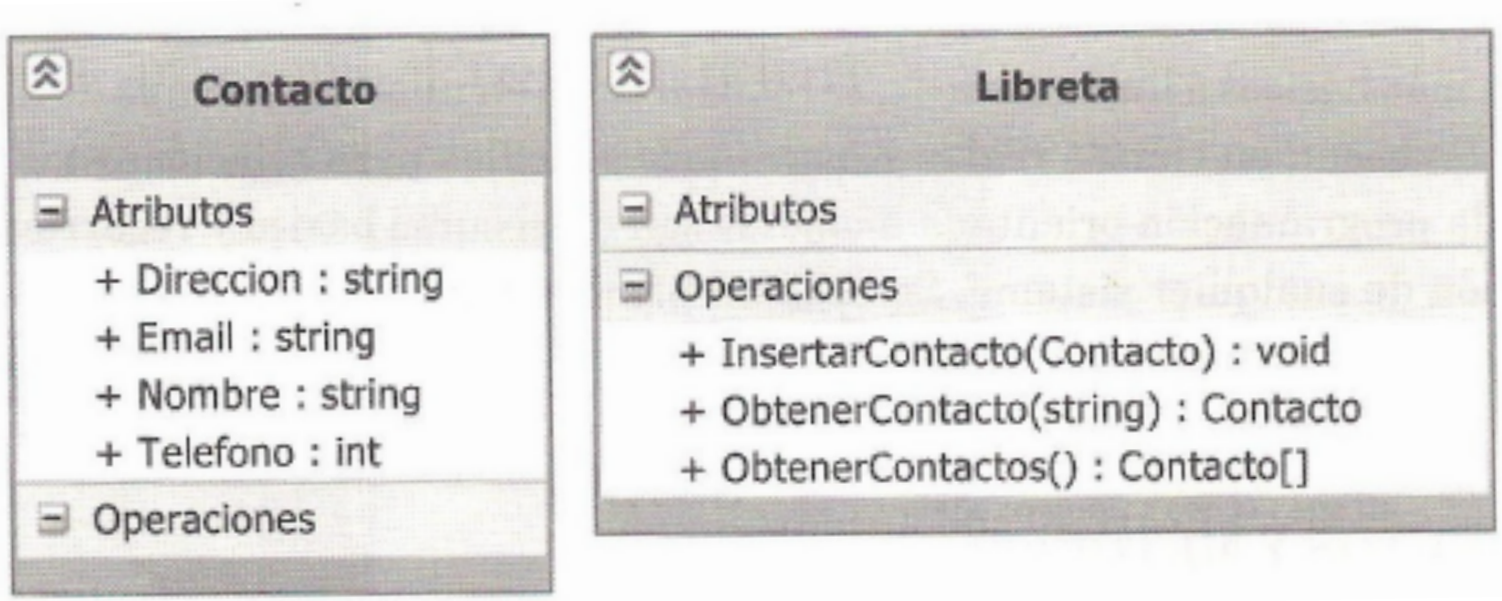


DISEÑO ORIENTADO A OBJETOS

CLASES, ATRIBUTOS Y MÉTODOS

- ✓ Cada bloque es una clase. Las **clases** representan nuestros **objetos**; los **atributos** definen las **propiedades** y características de los objetos; los métodos **especifican** las **acciones** que podemos realizar con dicho objeto.





- ✓ +: Elemento público
- ✓ -: Elemento Privado
- ✓ #: Elemento Protegido

DISEÑO ORIENTADO A OBJETOS

RELACIONES ENTRE CLASES

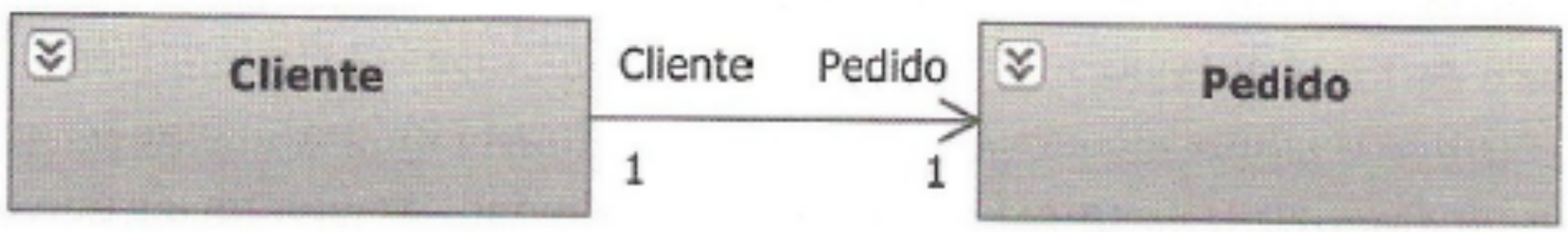
- ✓ Las relaciones se representan con flechas con una forma determinada.
- ✓ La relación posee una **cardinalidad** ; número de elementos de cada clase en cada relación. .

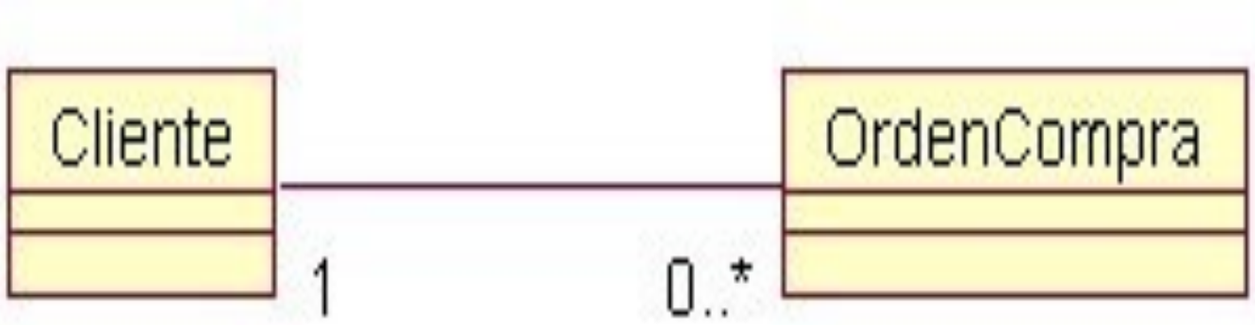
Cardinalidad	Significado
1	Uno y solo uno
0..1	Cero o uno
X..Y	Desde X hasta Y
*	Cero o Varios
0..*	Cero o Varios
1..*	Uno o Varios

DISEÑO ORIENTADO A OBJETOS

RELACIÓN DE ASOCIACIÓN

- ✓ Relación más básica que implica cualquier tipo de relación, por ejemplo un **uso de objeto asocia**do.
- ✓ La relación entre clases conocida como **Asociación**, permite asociar **objetos que colaboran entre si**. No es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.
- ✓ Se representa mediante una flecha simple
- ✓ Puede tener **cardinalidad**

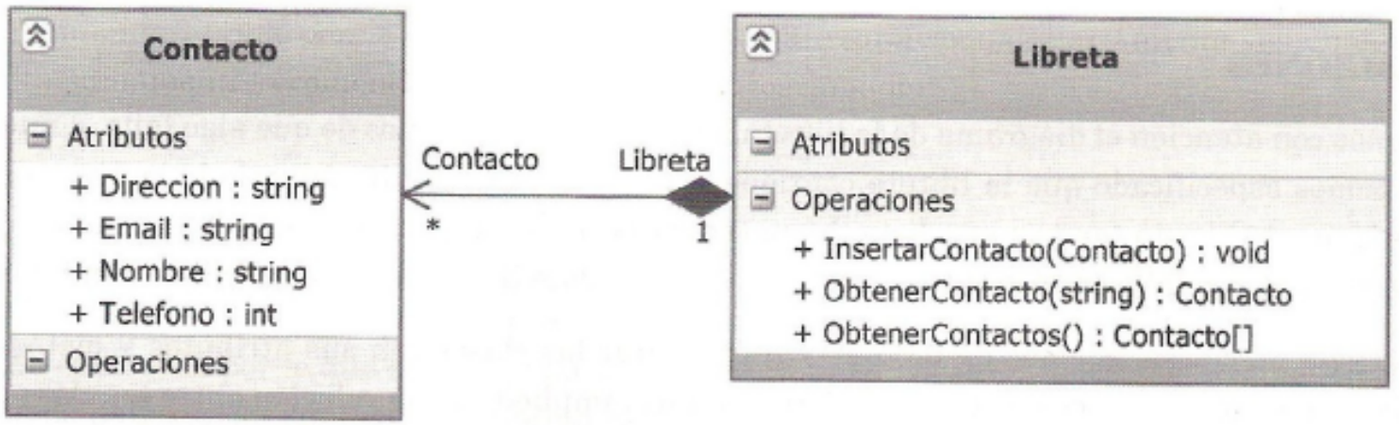


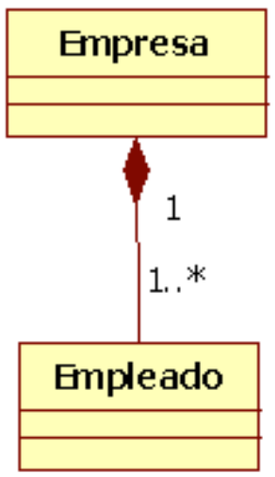


DISEÑO ORIENTADO A OBJETOS

RELACIÓN DE COMPOSICIÓN

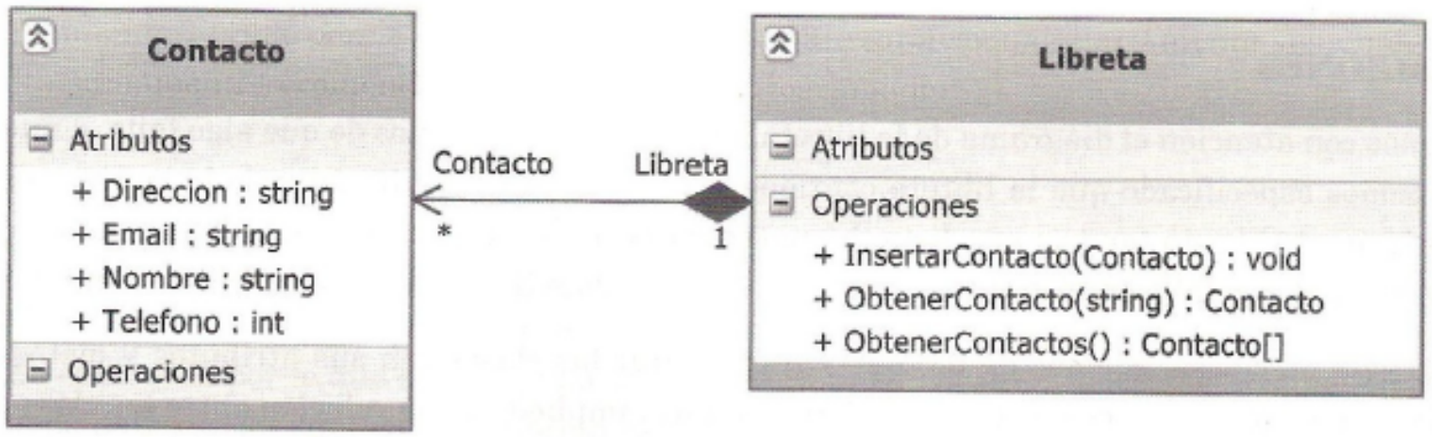
- ✓ La relación de **Composición** es una forma fuerte de asociación y define los componentes de los que se **compone otra clase**
- ✓ La vida de la **clase contenida** debe coincidir con la vida de la **clase contenedor,** ya que los componentes constituyen una parte del objeto compuesto y no tienen sentido en otra relación de asociación.
- ✓ La supresión del objeto compuesto conlleva la supresión de los componentes.
- ✓ El símbolo de composición es un diamante de color negro colocado en el extremo en el que está la clase que representa el "todo" (**Compuesto**).



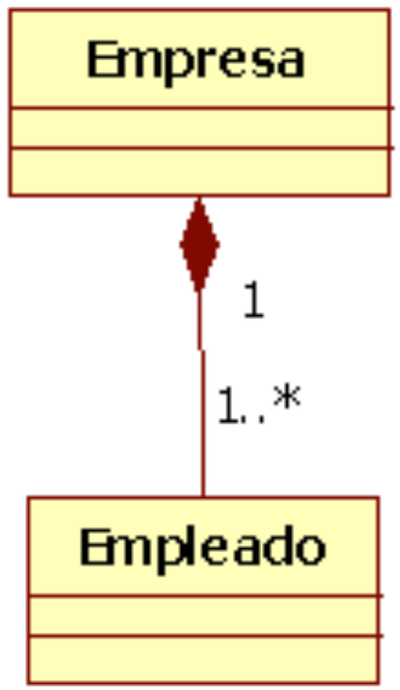


DISEÑO ORIENTADO A OBJETOS

RELACIÓN DE COMPOSICIÓN



Una **Libreta de direcciones** está compuesta por varios **Contactos** o ninguno (la libreta puede estar vacía).
Si no hay **Libreta**, no tiene sentido la clase **Contacto**.



Una **Empresa** está compuesta por uno o varios **Empleados**.

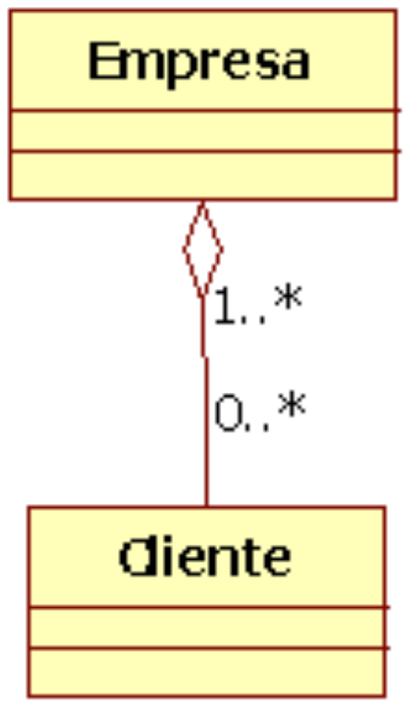
Si no hay **Empresa**, no tiene sentido la clase **Empleado** ya que un objeto es empleado por pertenecer a esa empresa.

Un **Empleado** no pertenece a otra **Empresa**.

RELACIÓN DE AGREGACIÓN

- ✓ **Agregación**: una clase es parte de otra clase (composición débil).
- ✓ Los agregados pueden ser compartidos por varios compuestos (de la misma asociación de agregación o de varias asociaciones de agregación distintas).
- ✓ La destrucción del compuesto no conlleva la destrucción de los componentes. Habitualmente se da con mayor frecuencia que la composición.
- ✓ Similar a la composición. Diferencia entre agregación y composición: la **agregación** no implica que la clase agregada necesite una instancia de la otra clase.

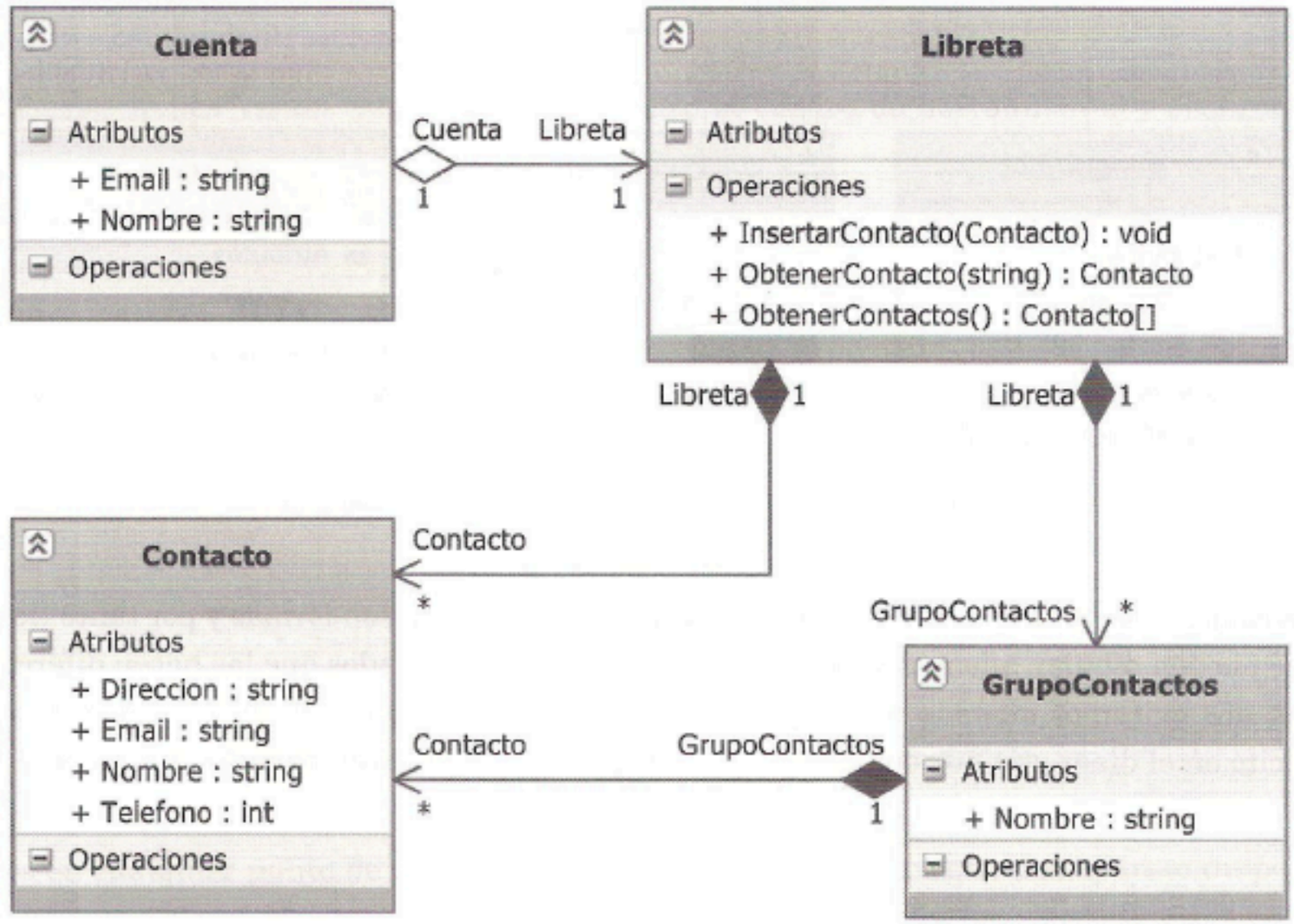
Una **Empresa** puede tener varios o ningún **Cliente**.
Si no hay **Empresa**, puede tener sentido la clase **Cliente** ya que puede pertenecer a otra relación.



DISEÑO ORIENTADO A OBJETOS

RELACIONES DE AGREGACIÓN Y COMPOSICIÓN

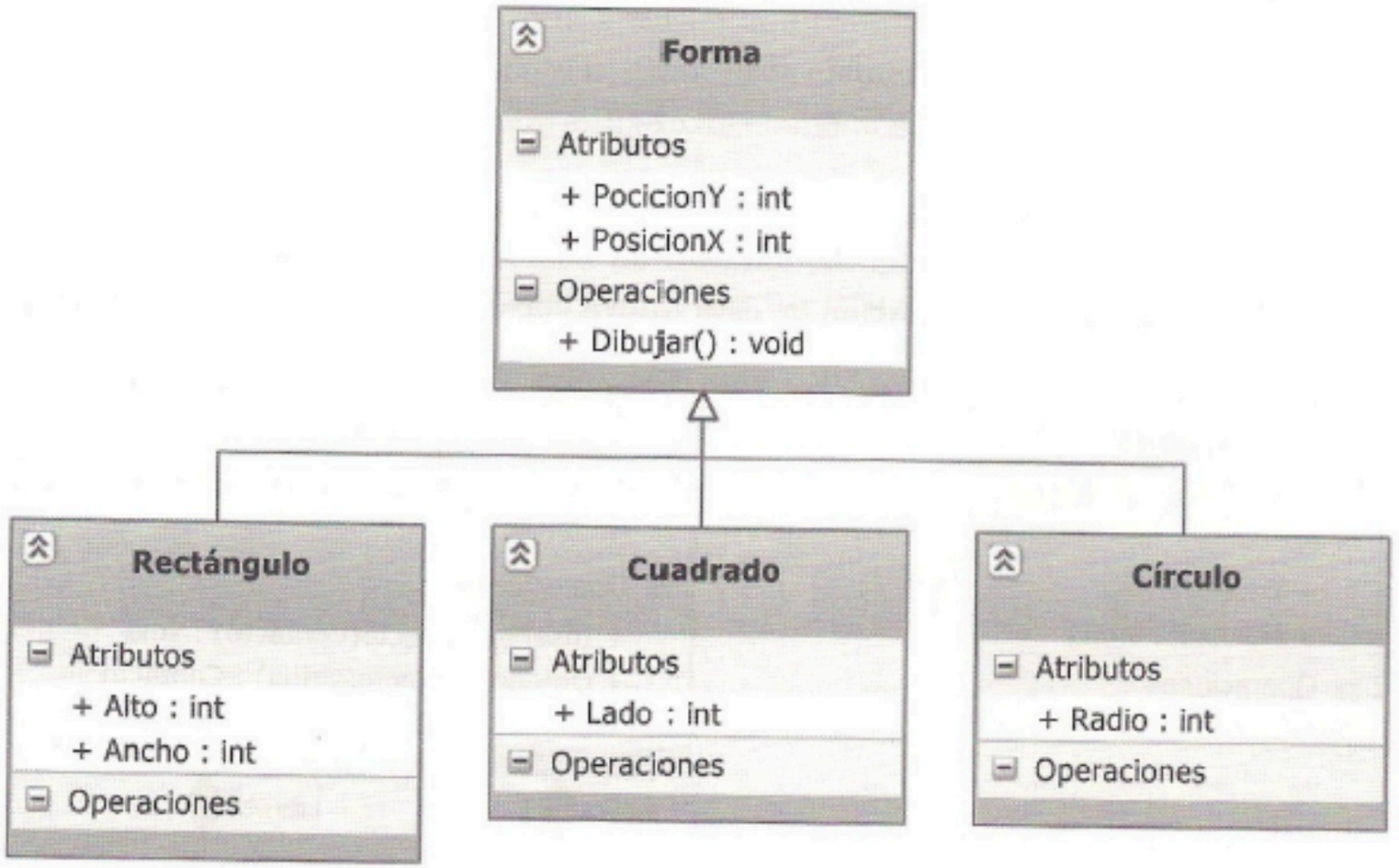
- ✓ Puede ser que estas relaciones inicialmente sea de asociación para más adelante “reforzarlas” mediante la relación apropiada.
- ✓ Las relaciones de composición y agregación son una versión más fuerte de las relaciones de asociación.



DISEÑO ORIENTADO A OBJETOS

RELACIÓN DE HERENCIA

- ✓ Representa clases y subclases, es decir, **clases más específicas de una general**.
- ✓ Se representa mediante una flecha con una punta triangular vacía.



HERRAMIENTAS UML

- ✓ **StarUML** es una herramienta UML de MKLab.
- ✓ El software tenía licencia bajo una versión modificada de **GNU GPL** hasta 2014, cuando se lanzó una versión reescrita 2.0.0 bajo una licencia propietaria.
- ✓ **StarUML** te permite trabajar con diagramas estándares **UML 2.x**, incluidos Clase, Objeto, Caso de uso, Componente, Implementación y muchos más.

- ✓ **https://staruml.io/download**

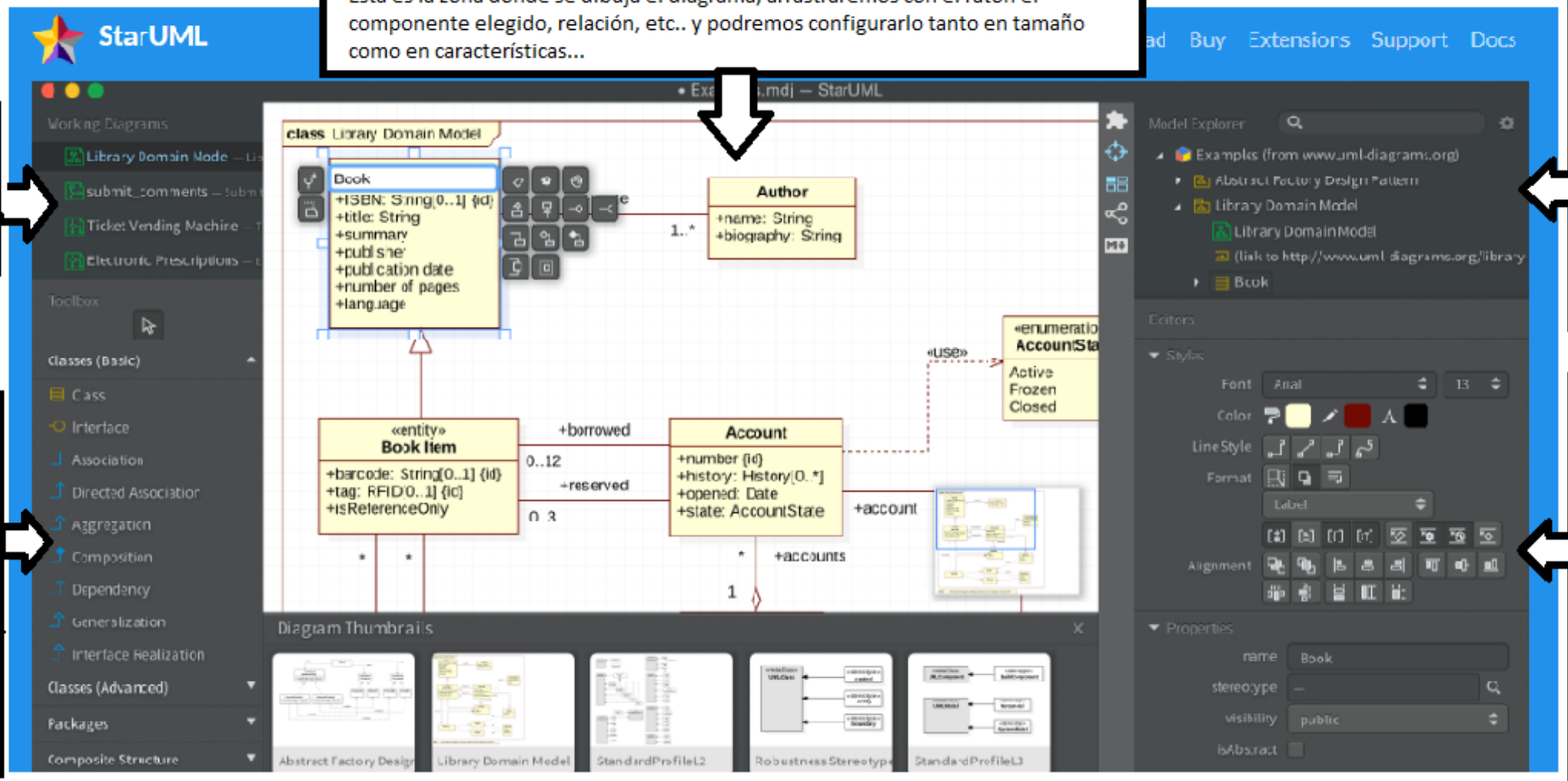
DISEÑO ORIENTADO A OBJETOS

HERRAMIENTAS UML

Esta es la zona donde se dibuja el diagrama, arrastraremos con el ratón el componente elegido, relación, etc.. y podremos configurarlo tanto en tamaño como en características...

Diagramas que tenemos abiertos y estan siendo usados.

En esta zona es donde damos de alta los diagramas, del tipo que necesitemos, en nuestro caso diagrama de clases.
Habrá que pulsar botón derecho encima de Main, "Add diagram", y el diagrama correspondiente.



En esta zona es donde estarán los distintos componentes que pueden ser usados en el diagrama.
Los componentes serán distintos dependiendo del diagrama con el que estemos trabajando.
Estos serían los correspondientes a un diagrama de clases.

aparecerán las características de cada componente, y es donde se podrán modificar (hay muchas), y cambiará según qué componente esté seleccionado.
Si no hay ningún componente seleccionado estará vacío.

GENERACIÓN DE CÓDIGO A PARTIR DE DIAGRAMAS DE CLASES

- ✓ Los diagramas de clase tienen una relación directa con el código.
- ✓ Se puede crear una traducción exacta de un código a un diagrama de clases y de un diagrama de clases a un código.
- ✓ Las herramientas de modelado ofrecen esta funcionalidad.
- ✓ La **ingeniería inversa** de software permite extraer los detalles estructurales y de comportamiento implícitos en el código de un producto de software y expresarlos de forma estándar, generalmente en diagramas UML.