UNIDAD 5

Análisis y diseño orientado a objetos. Elaboración de diagramas de clases

MÓDULO: Entornos de desarrollo

OBJETIVOS DEL TEMA

- Comprender los fundamentos del lenguaje UML.
- Conocer los tipos de diagramas que se pueden construir en el lenguaje UML.
- Saber cómo representar clases empleando el lenguaje UML.
- Reconocer y diferenciar los tipos de relaciones que se pueden establecer entre las clases.
- Conocer la manera de representar las diferentes relaciones entre clases empleando el lenguaje UML.
- Distinguir los tipos de clases de análisis de interfaz, de control y de entidad.
- Utilizar herramientas para la elaboración de diagramas de clases.
- Generar código a partir de un diagrama de clases.
- Crear un diagrama de clases a partir de código mediante ingeniería inversa.

INTRODUCCIÓN

- Durante las etapas de análisis y diseño, se deben obtener modelos gráficos que representen la información que es necesario manejar en la aplicación y las operaciones que se deben realizar sobre dicha información.

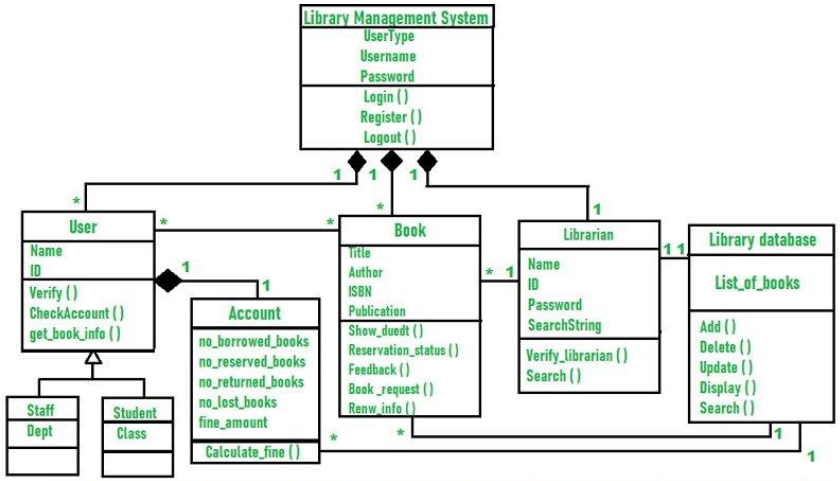
- La creación de estos modelos se debe realizar siguiendo ciertas normas y el estándar a nivel internacional para ello es el **lenguaje UML** (unified modeling language).

- El lenguaje UML establece las normas para crear muchos tipos diferentes de diagramas, de los cuales los más relevantes son los **diagramas de clases**.

5.1 EL LENGUAJE UML.

UNIFIED MODELING LANGUAGE™

Es un lenguaje estándar utilizado en el campo de la ingeniería de software para visualizar, especificar, construir y documentar los artefactos de un sistema de software.



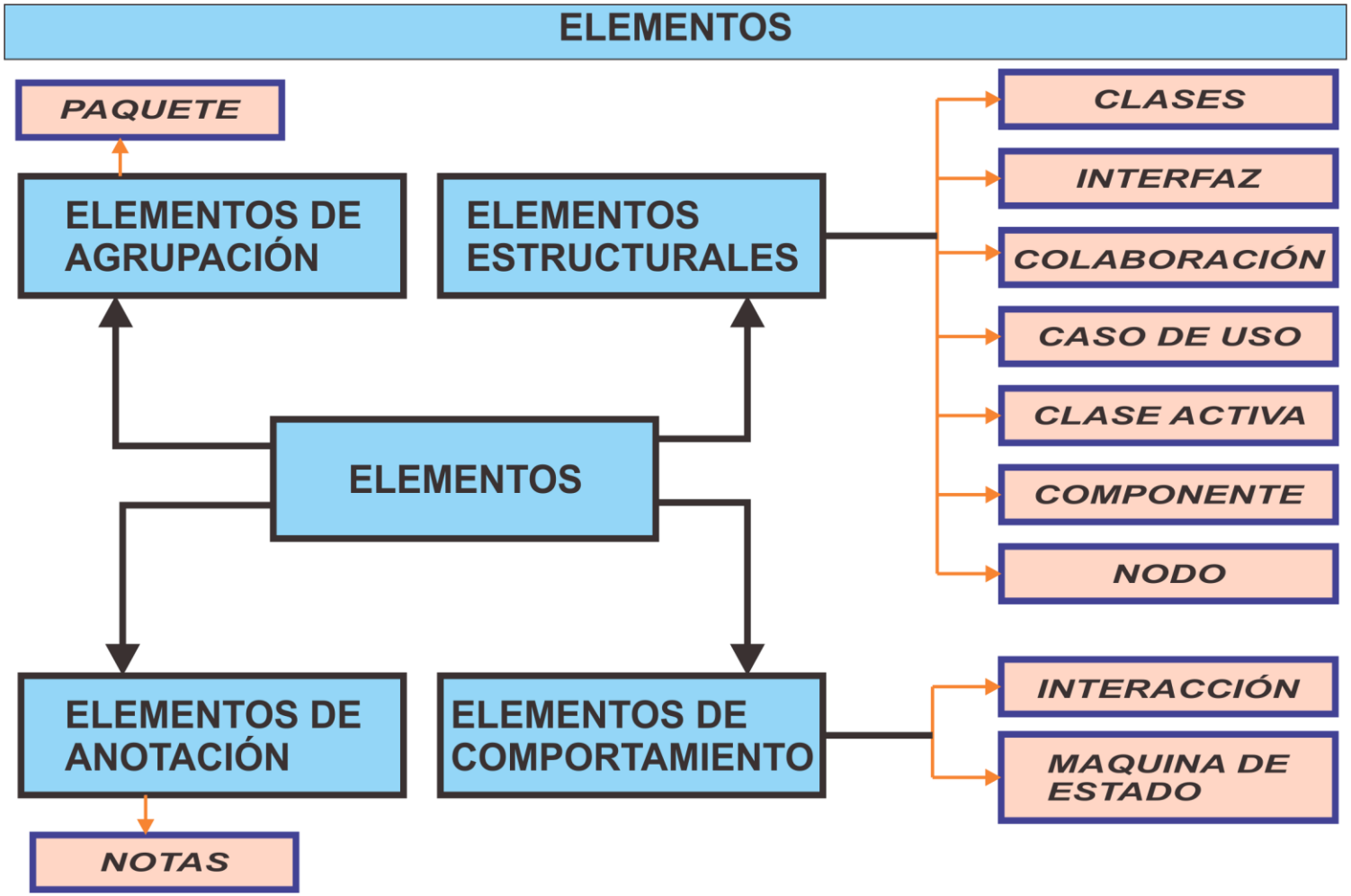
CLASS DIAGRAM FOR LIBRARY MANAGEMENT SYSTEM

5.1 EL LENGUAJE UML.

Los principios básicos:

- La elección acerca de qué modelos crear tiene una enorme influencia sobre cómo se acomete un problema y cómo se da forma a una solución.
- Todo modelo puede ser expresado con diferentes niveles de precisión.
- Los mejores modelos están ligados a la realidad.
- Un único modelo o vista no es suficiente. Cualquier sistema no trivial se aborda mejor mediante un pequeño conjunto de modelos casi independientes y desde múltiples puntos de vista.

5.1.1 TIPOS DE ELEMENTOS EN UML.



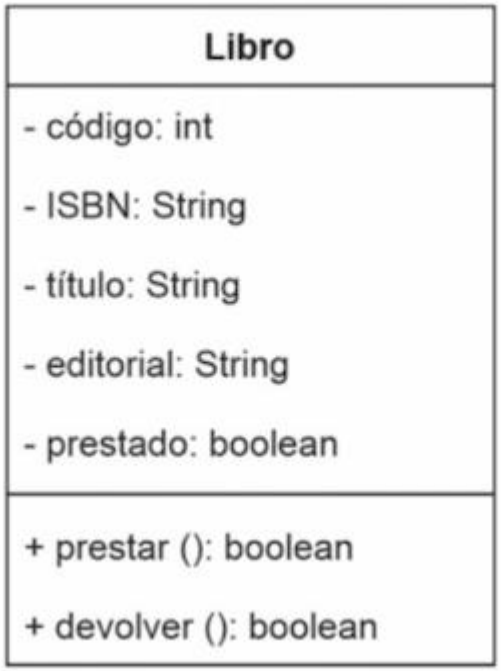
5.2.1 TIPOS DE ELEMENTOS EN UML.

Elementos estructurales: En su mayoría, los elementos estructurales son la parte estática de un modelo y representan conceptos o cosas materiales. Son los siguientes:

1. ***Clase:*** es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

2. ***Interfaz:*** es una colección de operaciones que especifican un servicio de una clase o componente. Por lo tanto, una interfaz describe el comportamiento visible desde el exterior de ese elemento.

5.2.1 TIPOS DE ELEMENTOS EN UML.



5.2.1 TIPOS DE ELEMENTOS EN UML.

Elementos estructurales:

3. ***Caso de uso:*** describe un comportamiento de un sistema, clase o componente desde el punto de vista de la persona usuaria. Describe un conjunto de secuencias de acciones que ejecuta un sistema y que produce un resultado observable de interés para un usuario o una usuaria particular.

5.2.1 TIPOS DE ELEMENTOS EN UML.

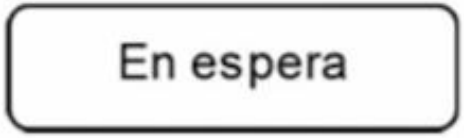
Elementos de comportamiento: Los elementos de comportamiento son las partes dinámicas de los modelos UML y representan un comportamiento en el tiempo y en el espacio. Suelen estar conectados semánticamente a elementos estructurales. Existen de tres tipos:

1. ***Interacción:*** comprende un conjunto de mensajes intercambiados entre un conjunto de objetos, dentro de un contexto particular y para un propósito específico.

5.2.1 TIPOS DE ELEMENTOS EN UML.

Elementos de comportamiento:

2. ***Máquina de estados:*** específica las secuencias de estados por los que pasa un objeto o una interacción durante su vida, en respuesta a eventos, junto con sus reacciones a esos eventos. Sirve para especificar el comportamiento de una clase individual o de una colaboración de clases. Una máquina de estados involucra los siguientes elementos: ***estados***, ***transiciones*** (paso de un estado a otro), ***eventos*** (disparadores de una transición) y ***actividades*** (la respuesta a una transición).



5.2.1 TIPOS DE ELEMENTOS EN UML.

Elementos de comportamiento:

3. ***Actividad:*** específica la secuencia de pasos que ejecuta un proceso computacional. En una máquina de estados, el énfasis se pone en el ciclo de vida de un objeto, mientras que en una actividad, lo importante es la secuencia o el flujo de pasos, sin importar qué objeto ejecuta cada paso. Cada paso de una actividad recibe el nombre de acción.

5.2.2 TIPOS DE DIAGRAMAS EN UML.

En UML se pueden construir hasta 14 tipos de diagramas distintos que se pueden agrupar en ***dos tipos de diagramas genéricos***: estructurales y de comportamiento.

Diagramas estructurales: Estos diagramas sirven para visualizar, construir, especificar y documentar los aspectos estáticos de un sistema.

Diagramas de comportamiento: Estos diagramas sirven para visualizar, especificar, construir y documentar los aspectos dinámicos de un sistema.

5.2.2 TIPOS DE DIAGRAMAS EN UML.

Diagramas estructurales:

1. ***Diagramas de clases:*** muestran un conjunto de clases y las relaciones entre ellas. Son los diagramas más comunes en el análisis y diseño de un sistema. Un diagrama de clases incluye básicamente clases (con atributos y métodos) y relaciones (asociación, agregación, composición, generalización y dependencia).
2. ***Diagramas de objetos:*** muestran un conjunto de objetos y sus relaciones. Se pueden considerar como un caso especial de diagramas de clases en los que se muestran objetos, es decir, instancias de clases en un momento del tiempo.
3. ***Diagramas de componentes:*** describen la estructura del software mostrando la organización y las dependencias entre un conjunto de componentes..

5.2.2 TIPOS DE DIAGRAMAS EN UML.

Diagramas estructurales:

4. ***Diagramas de despliegue.***
5. ***Diagramas de paquetes.***
6. ***Diagramas de perfiles.***
7. ***Diagramas de estructura compuesta.***

5.2.2 TIPOS DE DIAGRAMAS EN UML.

Diagramas de comportamiento:

1. ***Diagramas de casos de uso:*** ayudan a determinar la funcionalidad y las características del software desde el punto del usuario. En estos diagramas se muestran casos de uso, actores y las relaciones entre ellos. Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado de interés, como por ejemplo dar de alta a un empleado. Por su parte, un actor es un conjunto coherente de roles que desempeñan los usuarios de los casos de uso cuando interactúan con estos.

5.2.2 TIPOS DE DIAGRAMAS EN UML.

Diagramas de comportamiento:

2. ***Diagramas de actividades:*** muestran el flujo paso a paso de una computación. Una actividad muestra un conjunto de acciones, el flujo entre ellas y los valores producidos o consumidos. Se emplean para especificar una operación compleja, un proceso de negocio o un flujo de trabajo.

3. ***Diagramas de estados:*** estos diagramas muestran una máquina de estados, que consta de estados, transiciones (pasos de un estado a otro), eventos (que provocan transiciones) y actividades (tareas que se llevan a cabo al realizar una transición). Son especialmente importantes en el modelado de una clase o de una colaboración con comportamiento significativo. Así pues, podría afirmarse que muestran el comportamiento dirigido por los eventos de un objeto.

5.2.2 TIPOS DE DIAGRAMAS EN UML.

Diagramas de comportamiento:

4. ***Diagramas de Interacción:*** son un grupo especial de diagramas de comportamiento que muestran una interacción, es decir, muestran un conjunto de objetos o roles y los mensajes que se envían entre ellos. Se pueden distinguir cuatro tipos de diagramas de interacción:

- Diagramas de secuencia: muestran la secuencia cronológica de los mensajes entre objetos durante un escenario de un caso de uso.

- Diagramas de colaboración: muestran un conjunto de objetos, enlaces entre ellos y los mensajes enviados y recibidos entre ellos

5.2.2 TIPOS DE DIAGRAMAS EN UML.

Diagramas de comportamiento:

4. ***Diagramas de Interacción:***

■ Diagramas de tiempos: muestran los tiempos reales en la interacción entre diferentes objetos y roles, en ellos se representa el comportamiento de los objetos en un determinado periodo de tiempo.

■ Diagrama global de Interacciones: aportan una visión general del flujo de control de las interacciones. Es un tipo de diagrama híbrido entre el diagrama de secuencia y el diagrama de actividad.

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

Un **objeto** es una instancia de una clase, de modo que una clase está formada por un conjunto de objetos que poseen la misma estructura y el mismo comportamiento.

Por ejemplo, la clase Libro representa cualquier libro de una biblioteca, siendo un objeto cada uno de los ejemplares de libros disponibles en la biblioteca.

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

Una clase tiene tanto una serie de atributos como una serie de métodos, comunes a todos los objetos de la clase:

- Los ***atributos*** son las propiedades que posee una clase.

- Los ***métodos*** son las ***operaciones*** que se pueden llevar a cabo con los objetos de la clase y definen el ***comportamiento*** de la clase.

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

Representación de clases en UML: se representan mediante rectángulos con tres compartimentos:

- **Compartimento superior:** Nombre de la clase.
- **Compartimento central:** Atributos.
- **Compartimento inferior:** Operaciones

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

Visibilidad:

- La visibilidad de los atributos y operaciones se indica mediante símbolos:
  - +: Público
  - -: Privado
  - #: Protegido

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

El modificador de acceso hace referencia a la visibilidad del atributo, es decir, al ámbito desde el cual es visible el atributo en cuestión. Los modificadores de acceso más importantes son:

- ***Private:*** si un atributo o un método de una clase se define con el modificador de acceso prívate, quiere decir que ese atributo o método sólo es accesible desde métodos de la clase en la que está declarado el atributo o método. Este modificador se o simboliza con un guión (-).

- ***Public:*** si un atributo o un método de una clase se define con el modificador de acceso public, quiere decir que ese atributo o método es accesible desde métodos de cualquier clase. Este modificador se simboliza mediante el signo más (+).

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

- ***Protected:*** este modificador de acceso indica que el atributo o el método que lo tenga asignado es accesible desde métodos de la propia clase y desde métodos de su subclase o de sus subclases. Este modificador se simboliza por medio del símbolo almohadilla (#).

- ***Package:*** indica que el atributo o el método es visible en las clases incluidas en el mismo paquete. Este modificador se simboliza con el carácter tilde (~).

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

No obstante, a medida que se avanza en el proceso de desarrollo de software, se puede añadir más información a cada clase. Así, se pueden detallar más los diagramas de clases UML, indicando para cada atributo de una clase un ***modificador de acceso*** y el ***tipo de dato*** asociado a este, escribiendo:

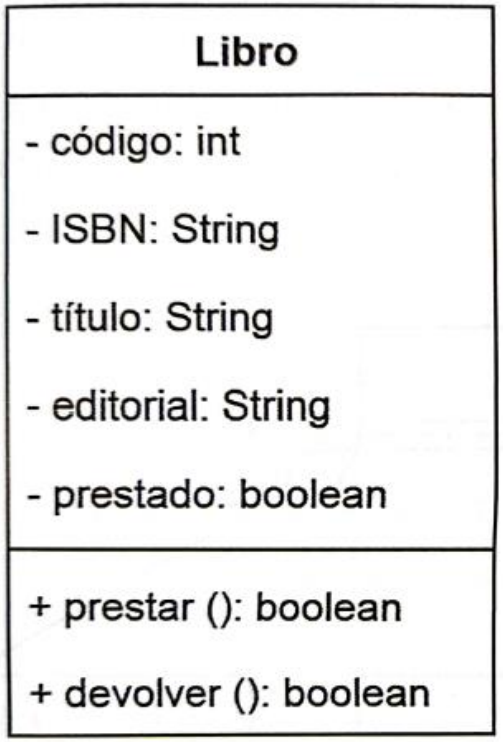
```
modificador atributo: tipoDato
```

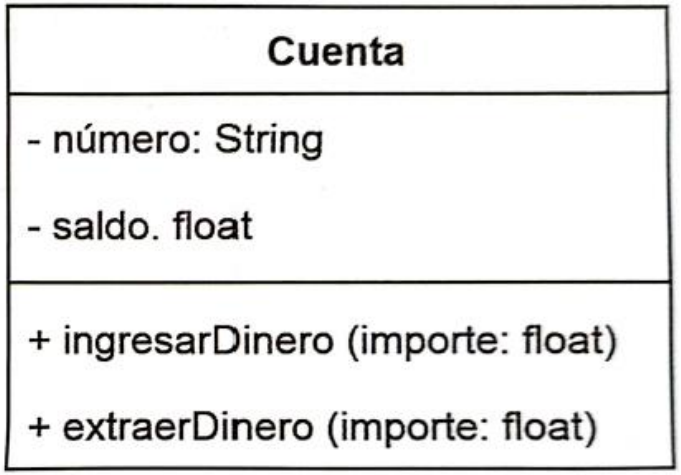
5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.

Además, en el caso de los métodos, se puede indicar el tipo del valor devuelto, si es que el método devuelve algún valor, y para cada uno de los datos que recibe el método (parámetro), el nombre del parámetro y su tipo de dato, siguiendo la siguiente sintaxis:

```
modificador método(parám1: tipoDato1, parám2: tipoDato2, …): tipoDatoDevuelto
```

5.3 CLASES, ATRIBUTOS, MÉTODOS Y VISIBILIDAD.





5.4 RELACIONES ENTRE CLASES.

Para todos los diagramas de clases, se han de establecer relaciones entre las clases, de manera similar a como se establecen relaciones entre las entidades de un diagrama entidad-relación. Existen distintos tipos de relaciones:

Agregación: Una agregación es un tipo de relación entre clases que representa un objeto compuesto por otros objetos. En esta relación, se distingue la parte que representa el todo, que recibe el nombre de ***compuesto***, y las diferentes partes que lo componen, que reciben el nombre de ***componente***.

La agregación implica que sólo puede existir una instancia del objeto agregado si existe al menos una instancia relacionada de alguno de los componentes. Por ejemplo, no tiene sentido hablar de un plan de estudios si no tiene al menos una asignatura.

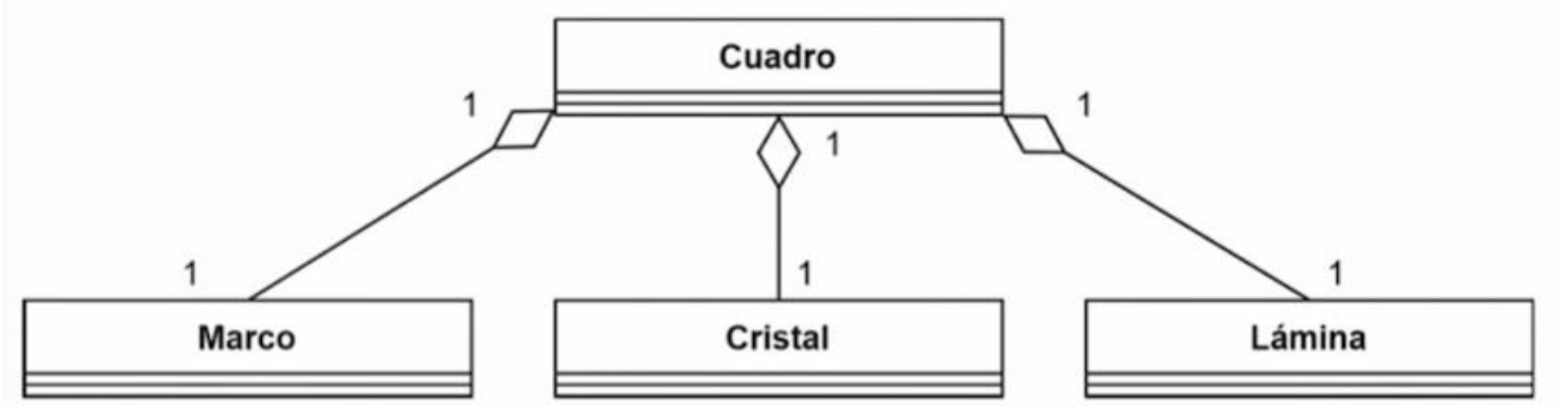
5.4 RELACIONES ENTRE CLASES.

Agregación:

Existen dos tipos de agregaciones: la ***agregación débil***, que se conoce generalmente como ***agregación*** a secas y la ***agregación fuerte***, conocida como ***composición***.

Las relaciones de agregación se representan uniendo el compuesto con sus partes componentes mediante una línea colocando un robo con fondo blanco al lado del compuesto.

5.4 RELACIONES ENTRE CLASES.



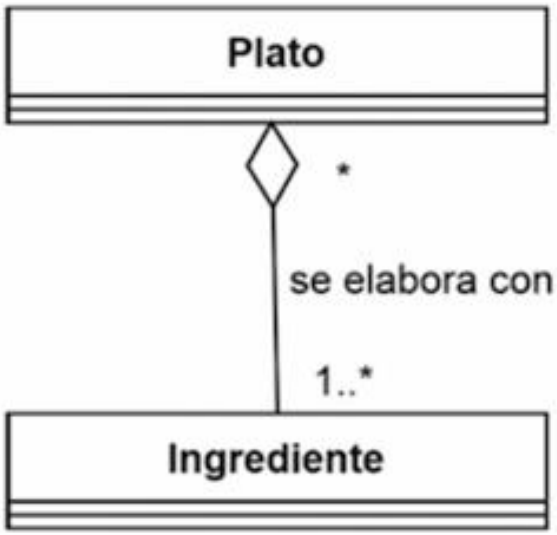
5.4 RELACIONES ENTRE CLASES.

Agregación:

Los números 1 al lado de las clases reciben el nombre de ***multiplicidades*** o ***cardinalidades*** e indican el número de objetos de una clase que pueden estar vinculados con cada objeto de la otra clase. Estos cardinales pueden ser:

- 1: uno
- 0..1: cero o uno
- 0..*: cero o varios (también puede ser empleado *)
- 1..*: uno o varios
- N: un número específico
- N..M: entro los números N y M

5.4 RELACIONES ENTRE CLASES.



5.4 RELACIONES ENTRE CLASES.

Composición: Las relaciones de composición se representan como las de agregación, pero el rombo tiene fondo negro. La agregación fuerte o composición es un tipo especial de agregación en la que la multiplicidad al lado de la clase que representa el compuesto es siempre 1 y en la que la existencia de los componentes depende del compuesto. De esto se puede deducir que, a diferencia de la agregación, la composición impone dos restricciones:

1. Cada componente sólo puede estar presente en un compuesto.

2. Si se elimina el compuesto, hay que eliminar todos los componentes vinculados a él.

5.4 RELACIONES ENTRE CLASES.

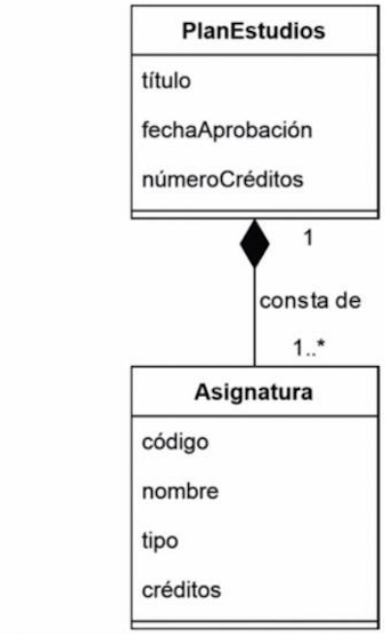
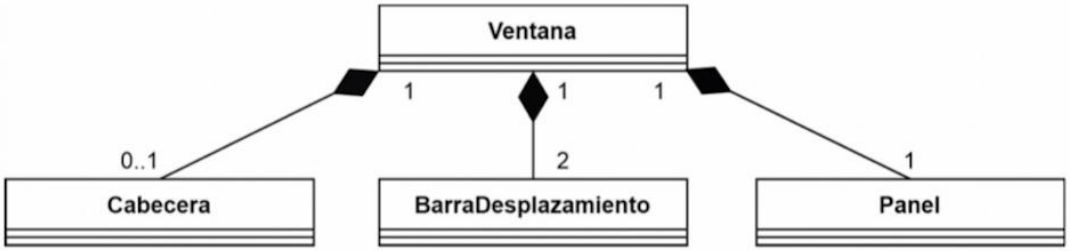


Figura 5.18. *Diagrama UML que muestra una relación de composición entre la clase 'compuesto' PlanEstudios y sus asignaturas.*



5.4 RELACIONES ENTRE CLASES.

ACTIVIDAD 1:

Agregación o composición.

En un programa, se quiere manejar información sobre países y las regiones en que se dividen esos países. Como sabes, un país consta de varias regiones y una región pertenece a un único país. Si se dispone de las clases País y Región, ¿Qué tipo de relación (agregación o composición) se debe establecer entre estas dos clases?, ¿por qué?

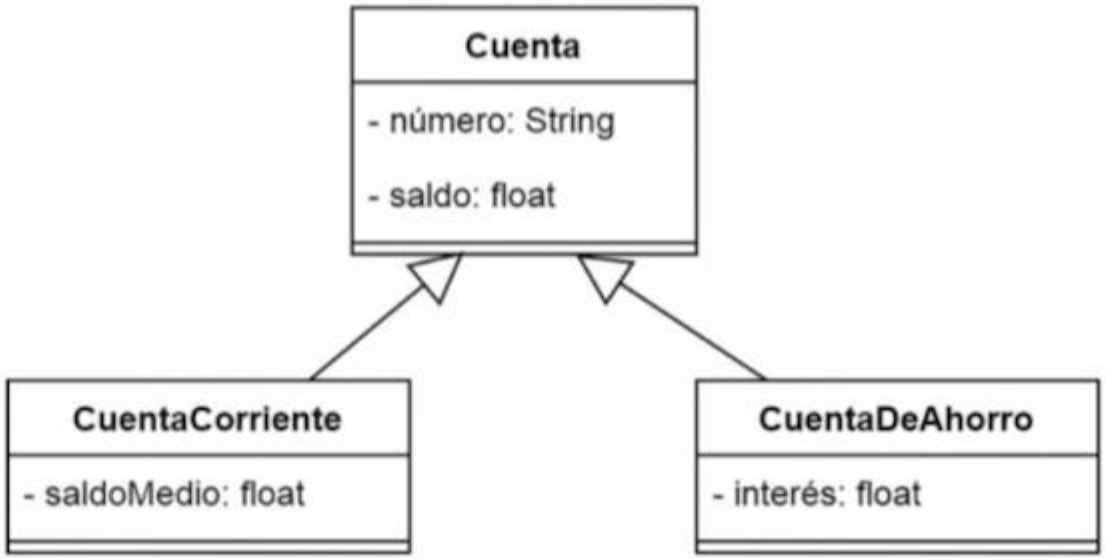
5.4 RELACIONES ENTRE CLASES.

Generalización y especialización: Las relaciones de generalización-especialización ocurren cuando se establece una jerarquía de clases y subclases motivada por el hecho de que hay varias clases que poseen atributos y/o métodos comunes. En este caso, se deben asignar los atributos y métodos comunes a la superclase, dejando los atributos y métodos específicos en las subclases. Este tipo de relaciones se representan uniendo mediante una línea de cada subclase con su superclase y esta línea lleva en el extremo un triángulo apuntando a la superclase.

5.4 RELACIONES ENTRE CLASES.

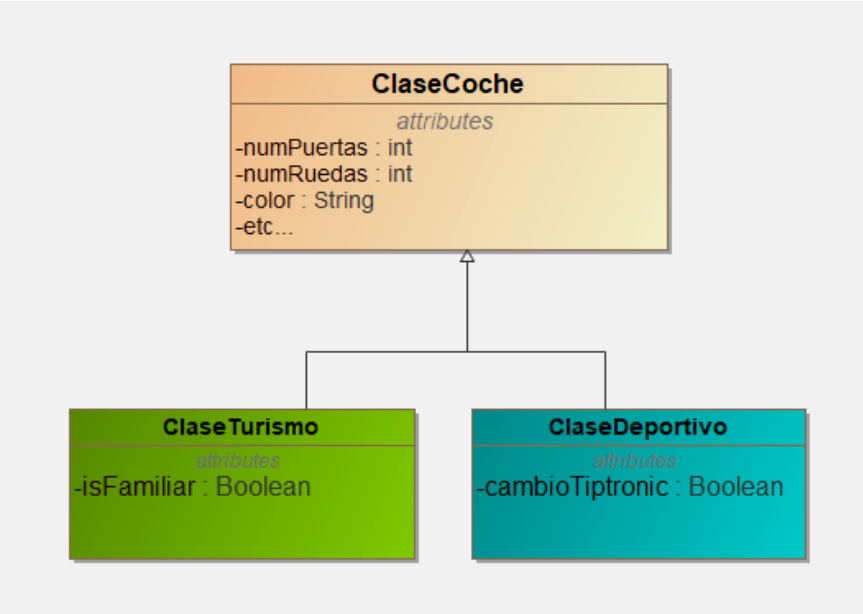
Generalización y especialización: En este contexto, se habla de ***generalización*** porque se puede decir que las subclases se generalizan en una superclase, dado que los atributos y métodos comunes a las subclases se asignan a la superclase. Por otro lado, se habla de ***especialización*** porque la superclase se especializa en una o varias subclases, las cuales poseen atributos y/o métodos específicos además de los de la superclase.

5.4 RELACIONES ENTRE CLASES.



5.4 RELACIONES ENTRE CLASES.

La ***herencia*** hace referencia al hecho de que las superclases poseen no solo sus atributos o métodos propios, sino que también heredan los de la superclase correspondiente. Así, la subclase CuentaCorriente tiene un saldo medio (atributo propio), pero también tiene un número de cuenta y un saldo, atributos que hereda de la superclase Cuenta.



5.4 RELACIONES ENTRE CLASES.

Asociación: Entre las clases pueden existir relaciones genéricas, que reciben el nombre de asociaciones. Cabe definir las ***asociaciones*** como relaciones entre clases del dominio del problema que no tienen las características de la agregación ni de la relación de generalización-especialización. También es común referirse a las asociaciones con el nombre genérico de ***relación***, por lo que se emplearán ambos términos indistintamente.

Las asociaciones pueden ser de diferentes tipos en función del número de clases que vinculan. El número de clases que se vinculan por medio de una asociación recibe el nombre de ***grado de la relación***.

5.4 RELACIONES ENTRE CLASES.

Asociación:

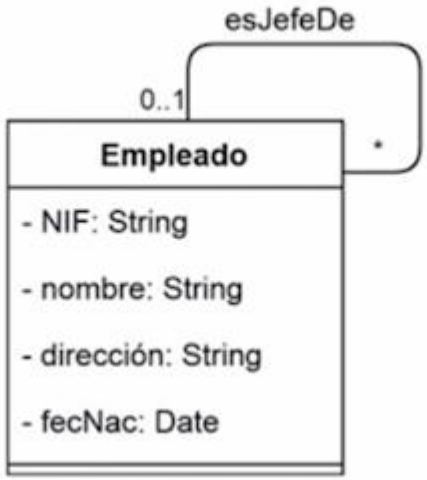
En función de su grado, existen los siguientes tipos de relaciones:

- ***Reflexivas:*** son las relaciones de grado 1, es decir, aquellas que vinculan a una clase consigo misma.
- ***Binarias:*** son las relaciones de grado 2, esto es, aquellas que vinculan dos clases.
- ***Ternarias, cuaternarias***...: son las relaciones de grado 3, 4..., es decir, aquellas que vinculan tres, cuatro... clases, respectivamente.

5.4 RELACIONES ENTRE CLASES.

Asociación: La asociación se representa mediante líneas. Si la relación es:

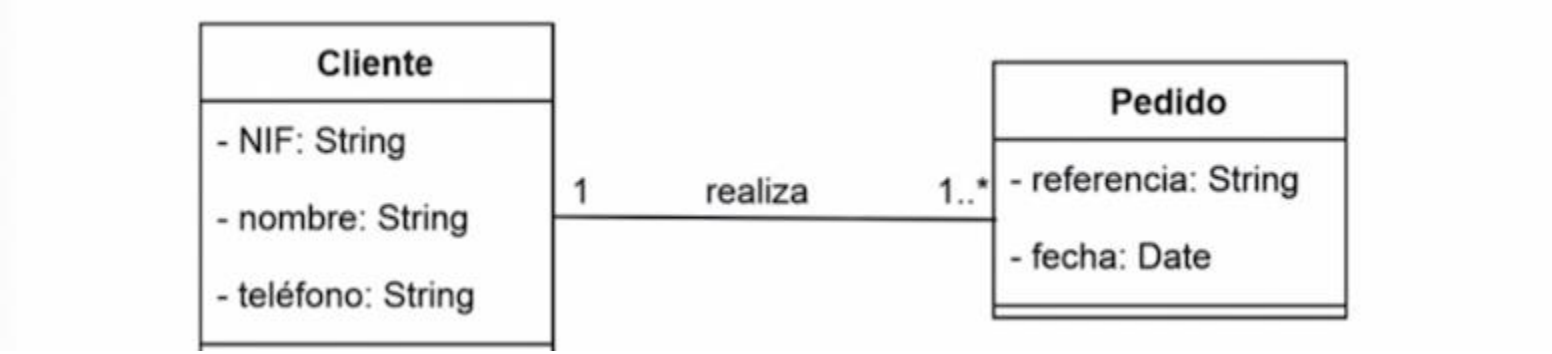
- Reflexiva: la línea une la clase consigo misma



5.4 RELACIONES ENTRE CLASES.

Asociación: La asociación se representa mediante líneas. Si la relación es:

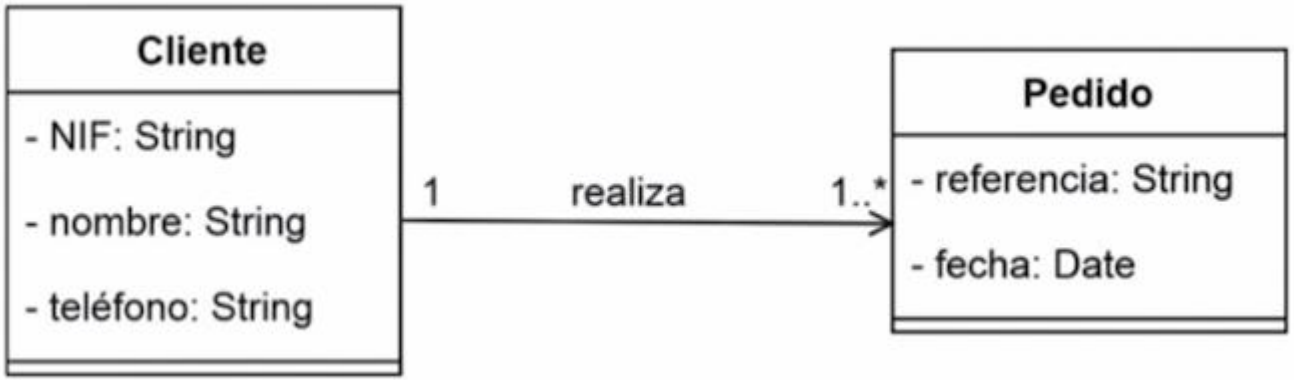
- Binaria: se representa mediante una línea que une dos clases



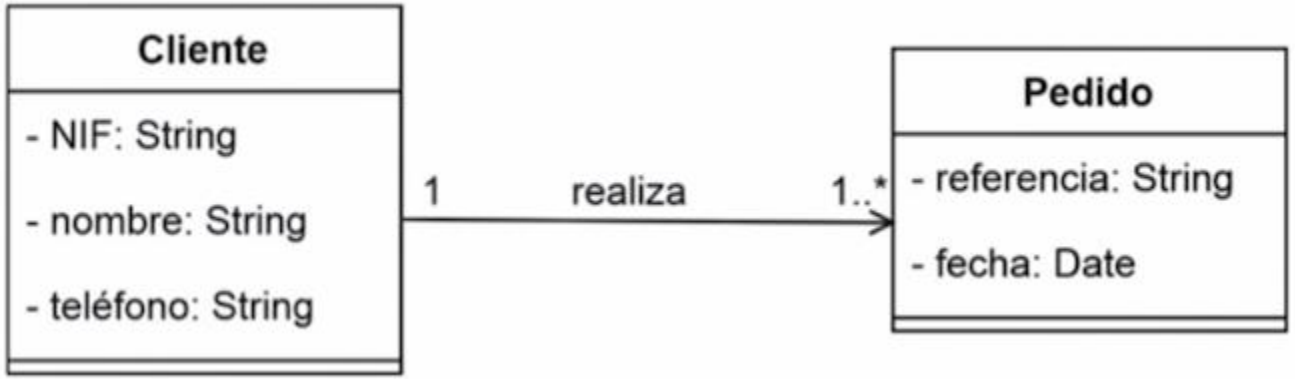
5.4 RELACIONES ENTRE CLASES.

Para las asociaciones, agregaciones y composiciones, también se puede indicar su ***navegabilidad***, esto es, el sentido en el que se puede acceder a información desde una de las clases intervinientes en la asociación. Se representa mediante una punta de flecha

5.4 RELACIONES ENTRE CLASES.



5.4 RELACIONES ENTRE CLASES.

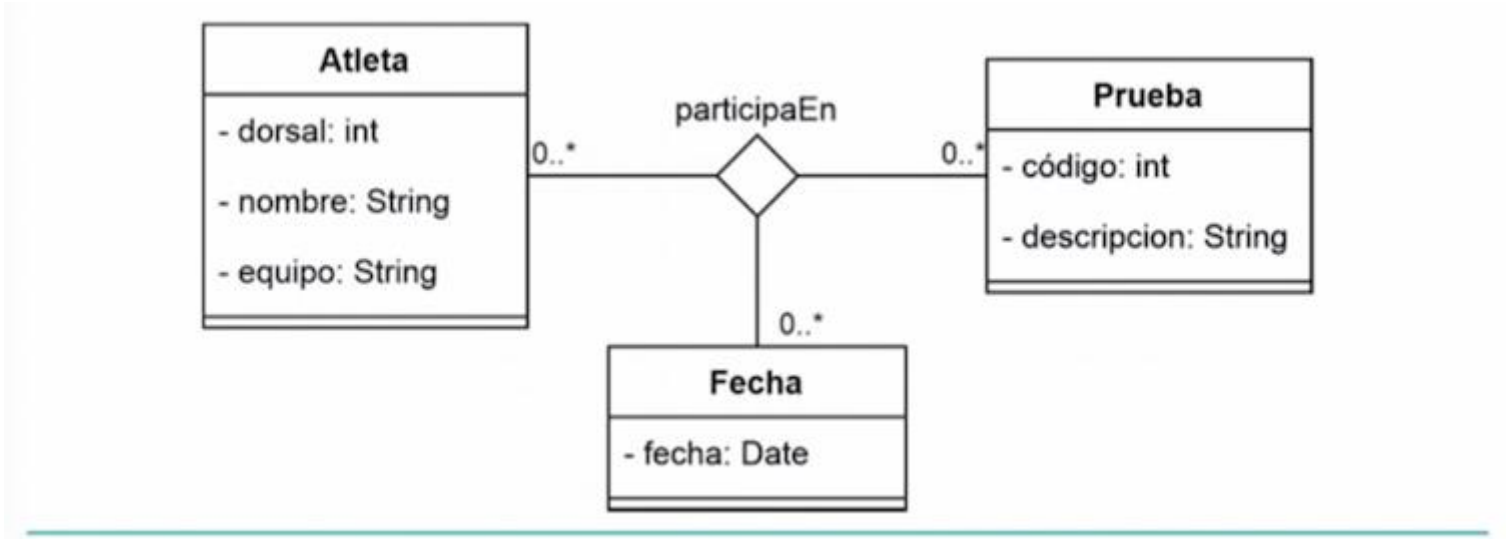



```
public class Cliente {
    private String NIF;
    private String nombre;
    private String teléfono;
    private List<Pedido> pedidos = new ArrayList<Pedido> ();
…
}
public class Pedido {
    private String referencia;
    private Date fecha;
…
}
```

5.4 RELACIONES ENTRE CLASES.

Asociación: La asociación se representa mediante líneas. Si la relación es:

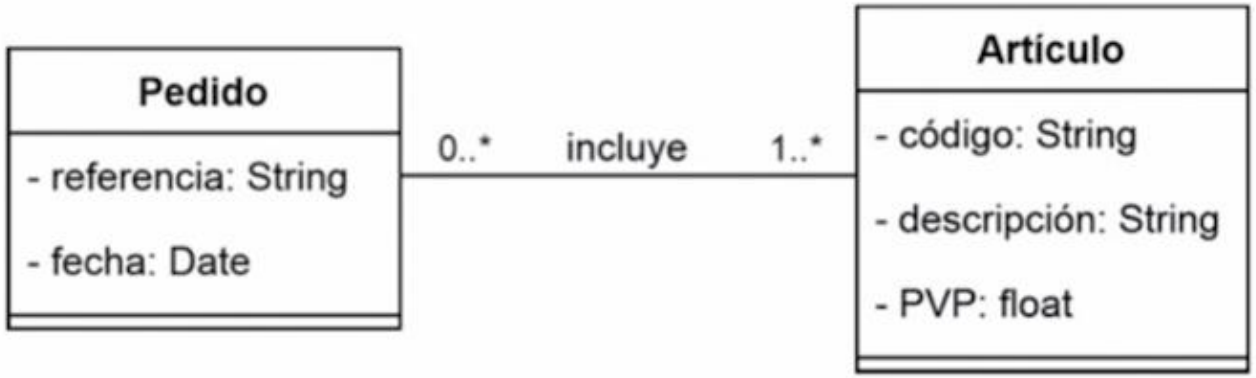
- Relaciones de grado mayor que dos (ternaria, cuaternaria…): se representa mediante un rombo que une a las clases relacionadas.



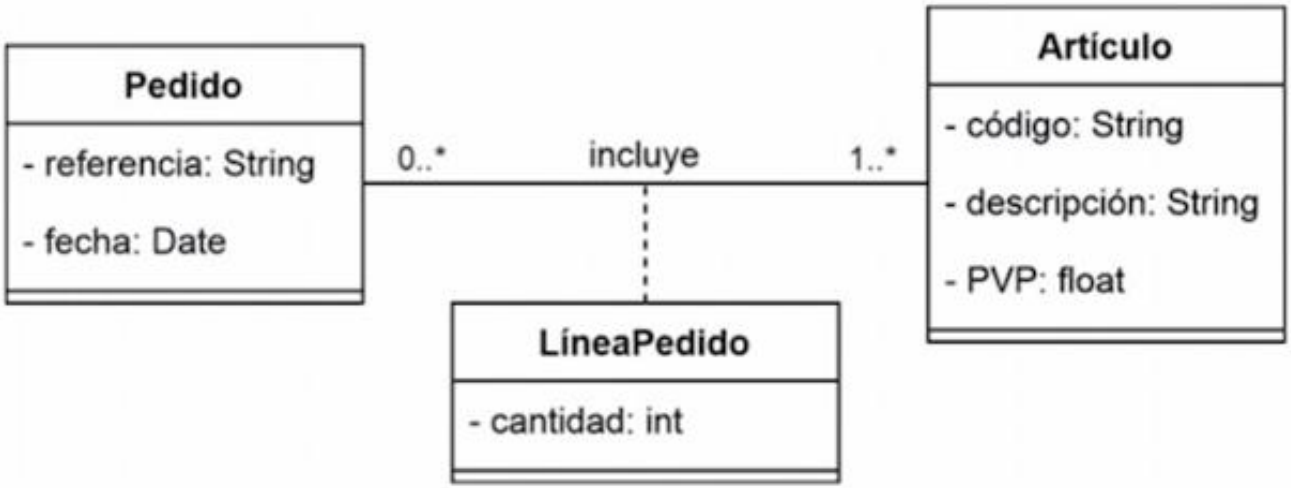
5.4 RELACIONES ENTRE CLASES.

Hay algunas ocasiones en las que es necesario almacenar información acerca de las asociaciones. La manera de reflejar esto es por medio de la creación de lo que se conoce como una ***clase asociativa***.

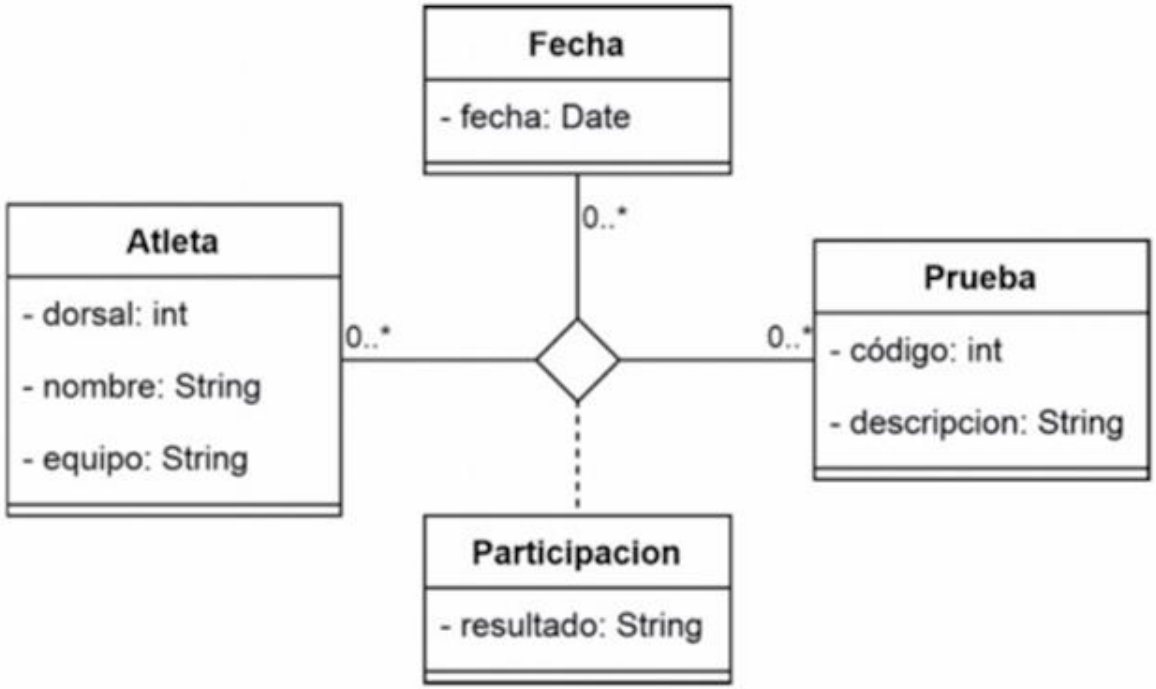
5.4 RELACIONES ENTRE CLASES.



5.4 RELACIONES ENTRE CLASES.



5.4 RELACIONES ENTRE CLASES.



5.4 RELACIONES ENTRE CLASES.

ACTIVIDAD 5:

Relación reflexiva con clase asociativa vinculada

En una empresa fabricante de vehículos, se precisa que esta disponga de información sobre las piezas necesarias para fabricar cada tipo de vehículo. Para desarrollar la aplicación se necesita crear un diagrama de clases:

■ Cada tipo de vehículo consta de varias piezas (carrocería, motor, ruedas, etc.) y cada una de estas piezas se puede descomponer en piezas más pequeñas, y así sucesivamente, hasta llegar a piezas indivisibles, como por ejemplo, un espejo o un tornillo.

5.4 RELACIONES ENTRE CLASES.

ACTIVIDAD 5:

- Se consideran piezas todos los componentes de un tipo de vehículo independientemente de su tamaño, por lo que se considera que un vehículo es una pieza en sí misma, pero también lo es un simple tornillo.

- Dada una pieza, puede que esta no forme parte de ninguna pieza superior (tal es el caso de un vehículo) o que forme parte de varias piezas superiores.

5.4 RELACIONES ENTRE CLASES.

ACTIVIDAD 5:

Se desea conocer:

- Por cada pieza, su código, nombre, existencias, existencias mínimas deseables, en qué pieza o piezas está incluida (si es el caso) y de qué pieza o piezas consta (si es el caso).

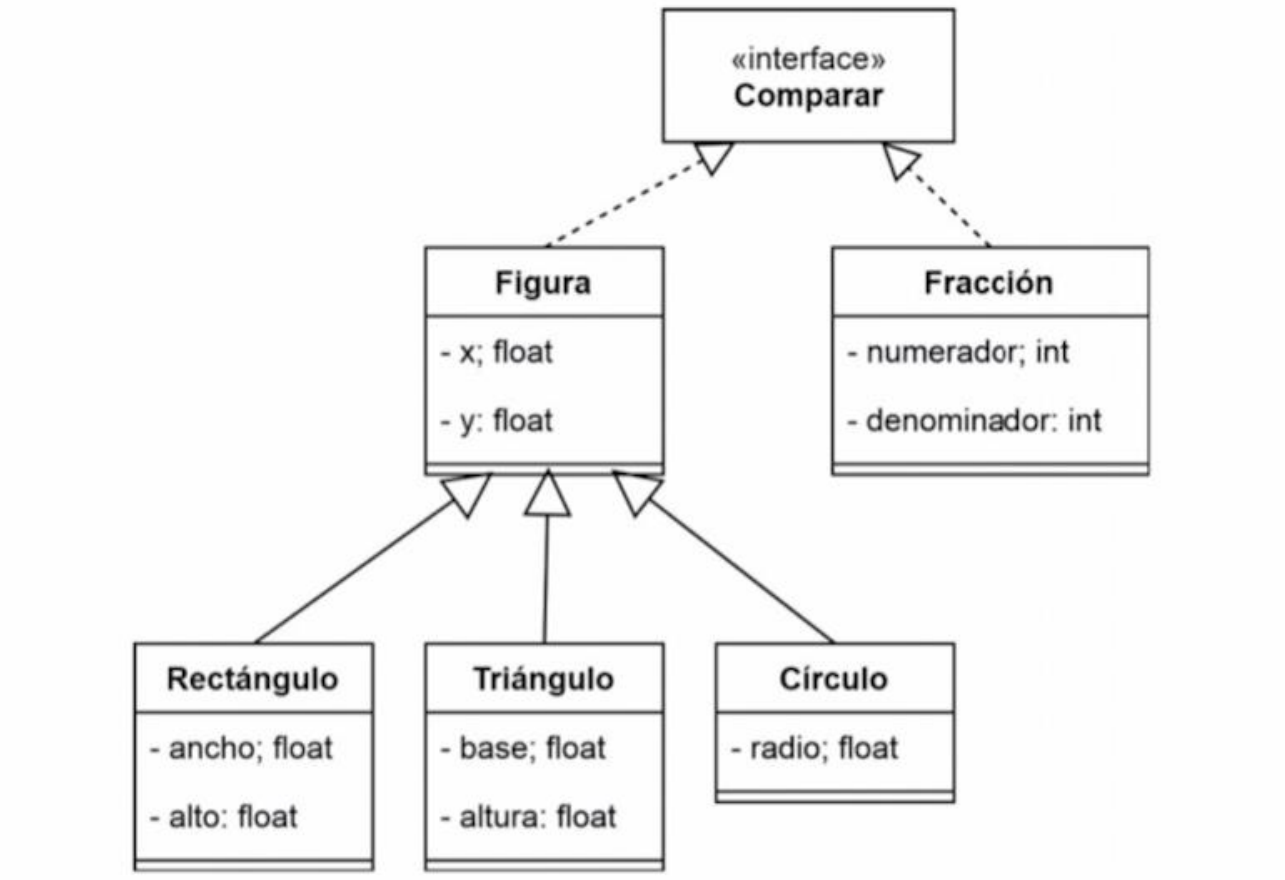
- Por cada pieza que consta de piezas inferiores, el número de unidades de cada pieza inferior que incluye cada pieza superior; así, no solo interesa saber que un determinado tipo de vehículo consta de motor y de ruedas, sino también cuántos motores tiene ese tipo de vehículo (uno) y cuántas ruedas (4, por ejemplo).

5.4 RELACIONES ENTRE CLASES.

Realización: Una relación de realización es la que se establece entre una clase Interface y las clases que implementan esa interfaz. Una clase Interface es un mecanismo que se puede emplear en Java para permitir la herencia múltiple, es decir, que una clase herede de varias superclases. El motivo es que, en Java, una clase sólo puede heredar de una superclase, pero puede implementar una o varias interfaces.

Las interfaces suelen contener métodos comunes a varias clases, de forma que todas las clases que implementan una interfaz o que establecen una relación de realización con una interfaz, heredan de dicha interfaz todos sus métodos.

5.4 RELACIONES ENTRE CLASES.



Resumen

Composición	Rombo negro	Alto grado dependencia	Al crear la clase contenedora, obligatoriamente se crean el resto. Los componentes no existen sin la contenedora.
Agregación	Rombo blanco	Menor grado dependencia	Puede existir cualquiera de las dos clases, sin existir la otra. Pueden tener tiempos de vida diferentes
Asociación	Con cardinalidad Con verbo	0 grado de dependencia	