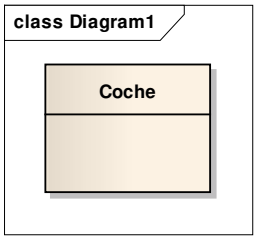
Patrones Software
Tema 1-1:
Conceptos de POO
y
Diagrama de clases

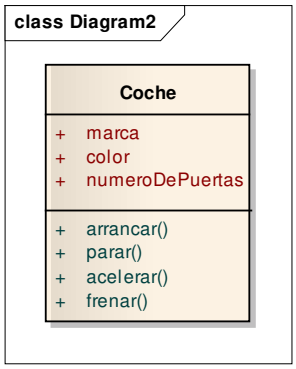
Clases y objetos

- Una clase es un contenedor de datos (atributos) junto con las operaciones para manipular esos datos.
- **Clase = atributos + operaciones**
- Una **clase** es una **plantilla** de la cual se pueden crear **instancias** que serán los **objetos**.
- **Una clase es un tipo de datos y un objeto es una variable de ese tipo**.

Clase

- Una clase se puede representar:
  - como una caja que solo contenga el nombre de la clase
  - como una caja dividida en 3 compartimentos que contendrán:
    - El nombre de la clase
    - Los atributos (Una clase puede tener varios o ningún atributo)
    - Las operaciones o métodos



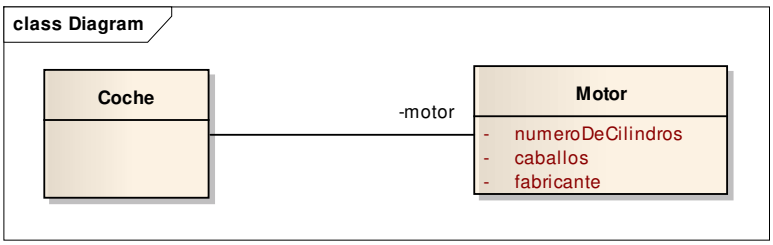


Clase

- **Atributos**
  - Un **atributo** es una propiedad o **característica de una clase**.
  - Todo objeto de la clase tiene un valor específico en cada atributo.
  - **Los atributos representan el estado de un objeto**.
  - Una clase puede tener varios o ningún atributo.
- **Operaciones**
  - Las operaciones permiten manipular los datos.

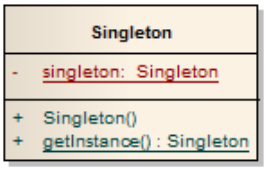
Clase: Atributos

- Notación:
  - "En línea": los atributos se muestran dentro de la clase (notación clásica).
  - "Atributos por relación": los atributos no se muestran dentro de la clase. Esta notación produce un diagrama de clase más grande, pero muestra un mayor detalle para atributos complejos.



Clase: Elementos estáticos

- Son compartidos por todos los objetos de una clase: **todos los objetos de una clase comparten la misma copia de un atributo u operación.**
- Un ejemplo de uso de atributos y operaciones estáticos es el patrón de diseño **Singleton,** que asegura que **solo se podrá construir un objeto de una determinada clase**.
- No es necesario instanciar la clase para poder usarlos, ya que no dependen de la instancia, sino de la clase.
- Para indicar que un atributo u operación es estático se subraya.



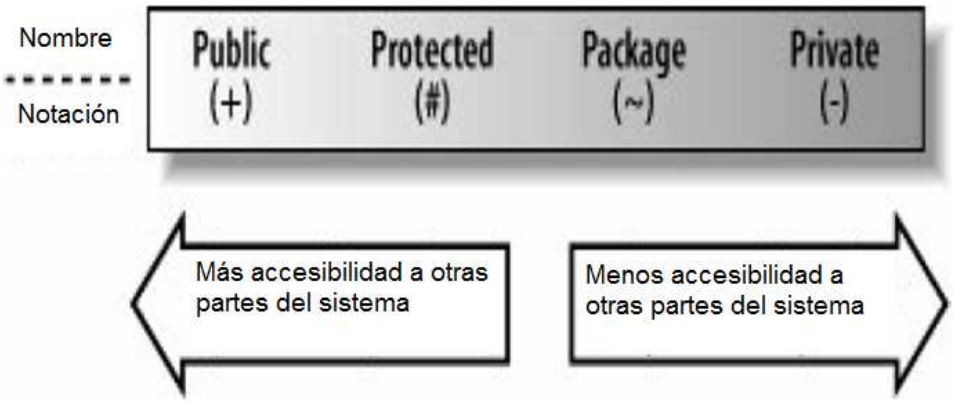
Clase: Modificadores de acceso

- **Public (+):** Todo el mundo puede acceder al elemento.

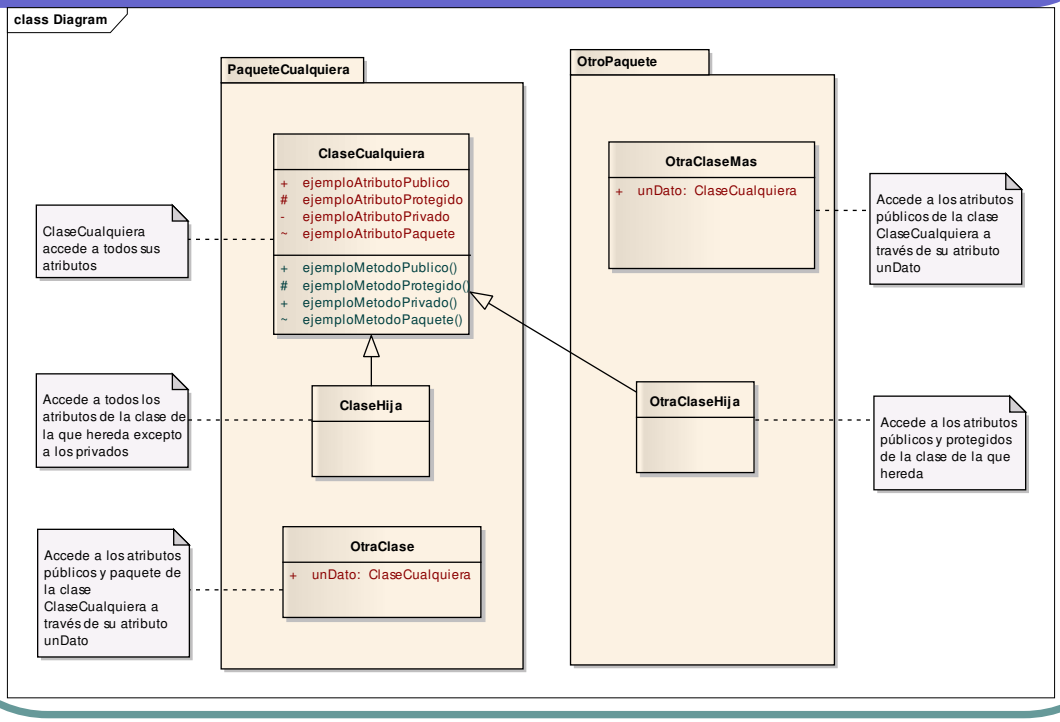
  Si es un atributo, todo el mundo puede ver el elemento, es decir, usarlo y asignarlo.

  Si es un método todo el mundo puede invocarlo.
- **Private (-):** Sólo se puede acceder al elemento desde la propia clase.
- **Protected (#):** es una combinación de los accesos que proporcionan los modificadores public y private. Protected proporciona:
  - acceso público para las clases derivadas
  - acceso privado (prohibido) para el resto de clases
- **Package (~):** se puede acceder al elemento desde cualquier clase del paquete donde se define la clase.

Clase: Visibilidad



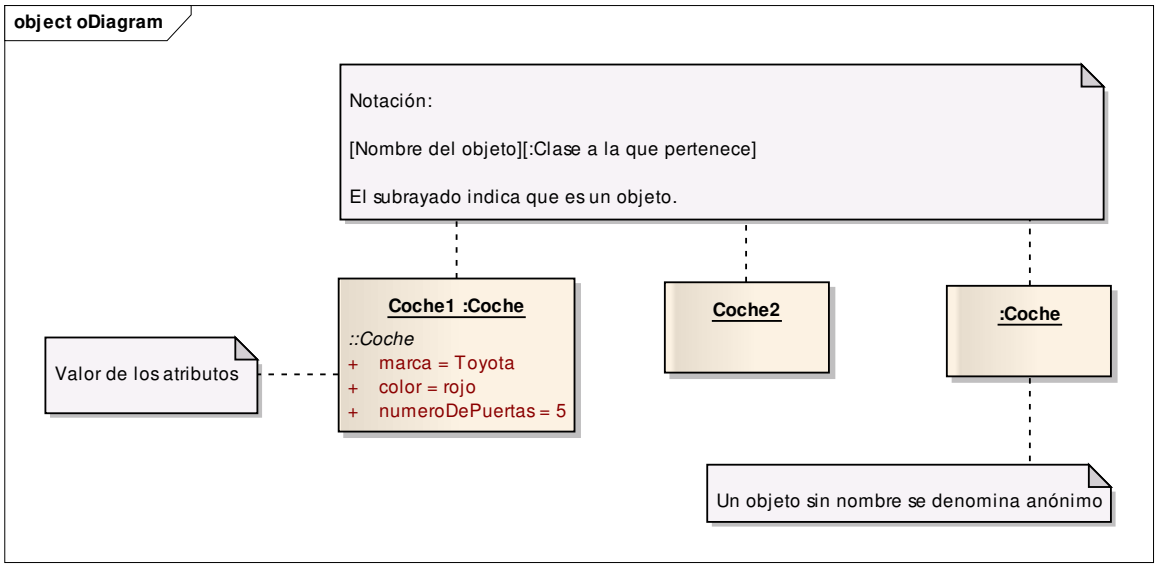
Clase: Ejemplo de visibilidad



Objetos

- Un objeto es una instancia de una clase:
  - Con la que se puede **interactuar**: se le pueden enviar mensajes y éste reaccionará ante ellos. Un mensaje es "algo" que le hacemos a un objeto. **Enviar un mensaje = Llamar a un método u operación.**
  - Que tiene **estado**: un objeto lo constituyen todos los datos (**atributos o variables de instancia**) que encapsula en un momento determinado cada uno de los cuales tiene un valor. Los atributos pueden ser constantes o cambiar de valor.
  - Que tiene **comportamiento**: el objeto puede reaccionar de manera diferente a un mensaje en función de su estado.
  - Que tiene **identidad**: al objeto se le hace referencia por un nombre (excepto en los objetos anónimos).

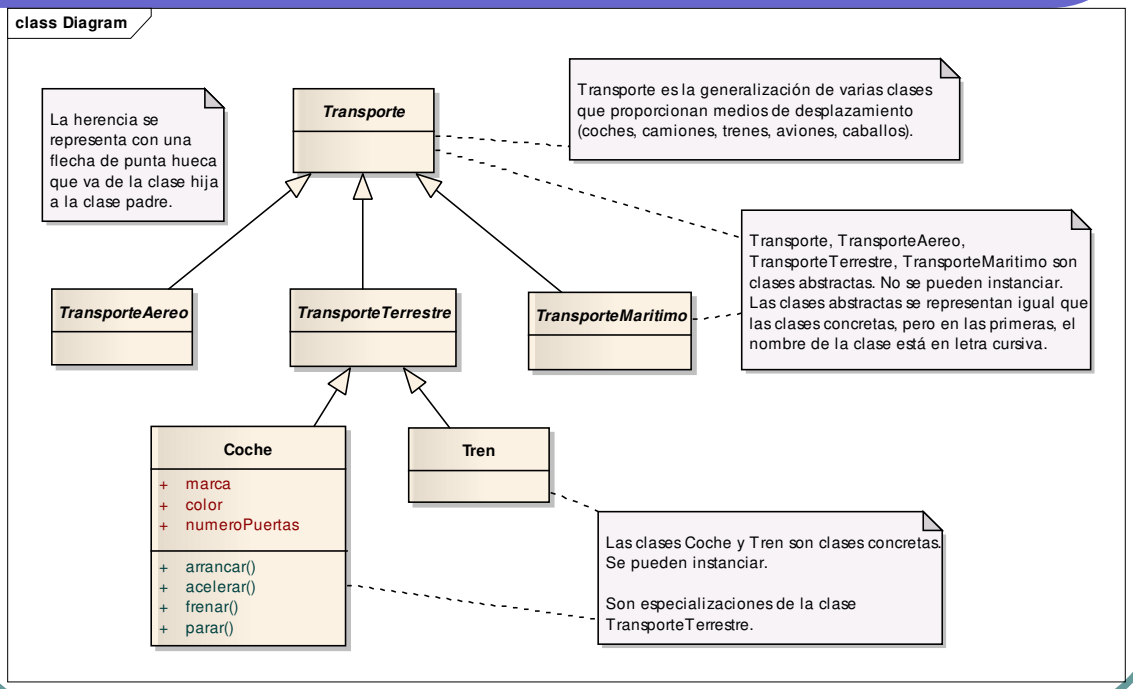
Objetos



Generalización/Especialización

- **Generalización (clase padre):**
  - Identifica y define los atributos y operaciones comunes en una colección de objetos.
  - **Se implementa a través de la herencia.**
- **Especialización (clase hija):**
  - Algunas veces necesitamos especializar una clase, modificar la implementación de una parte de su comportamiento.
  - Para personalizar los miembros de una subclase, se deben sobrescribir los miembros de la superclase.
  - **La especialización es la herencia con la adición y modificación de métodos para resolver problemas específicos.**

Generalización/Especialización

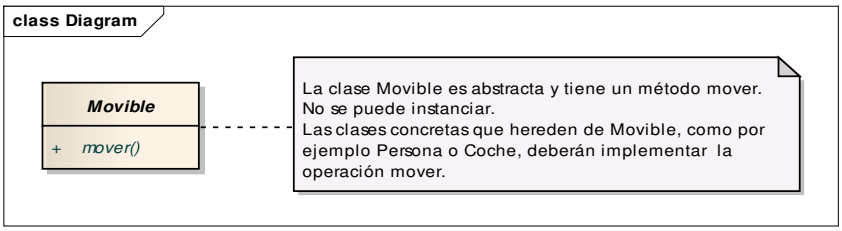


Métodos

- Un **método** es la implementación de una operación.
- Cada clase provee una implementación para sus operaciones o la hereda de una superclase (**herencia**).
- Si la clase no provee una implementación para una operación y esta tampoco es provista por su superclase, la operación se considera abstracta (**clases abstractas**).

Clases abstractas

- Las clases abstractas definen atributos, y operaciones que deben ser implementadas.
- **No se pueden instanciar** ya que falta la implementación de una o varias operaciones.
- **Se utilizan como clase base de otras clases**.
- Son las subclases las que deben implementar las operaciones.
- Una clase es abstracta si tiene alguna operación abstracta.
- Para indicar que una clase es abstracta se pone el nombre de la clase en cursiva.



Interfaces

- **Una interfaz es una clase abstracta pura**, es decir, ninguna de las operaciones tiene implementación.
- Una interfaz tiene
  - Atributos (lenguajes como Java no soportan atributos).
  - Operaciones
- Es un conjunto de operaciones que una clase presenta a otras (contrato).
- Definen un comportamiento común para varias clases aunque estas no desciendan de una clase común.
- Son un mecanismo para implementar herencia múltiple que evita los problemas derivados de esta.
- Java no permite herencia múltiple, pero si implementar cualquier número de interfaces.

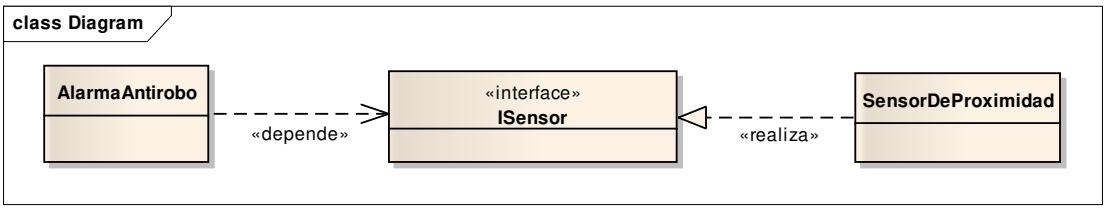
Interfaces

- **Relación interfaz-clase:**
  - **Realización**: Es la relación que se produce entre una clase y una interfaz cuando una clase implementa esa interfaz. La clase es proveedora de la interfaz.

    Que una interfaz declare atributos no significa que necesariamente su realización tenga que implementar dicho atributo. Estos pueden aparecer solo como parte de la interfaz para que sean vistos por observadores externos.
  - **Dependencia:** Es la relación entre una clase y una interfaz en la que la clase usa la interfaz. La clase requiere la interfaz.

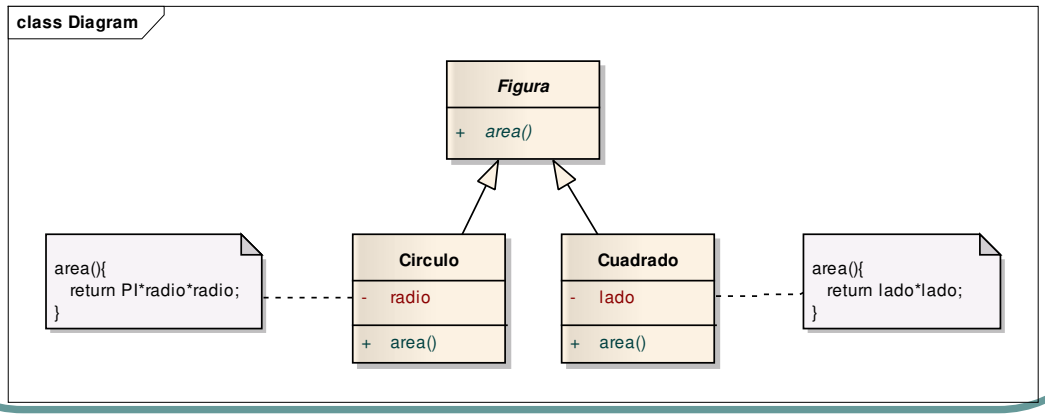
Interfaces

- Notación clásica: Clase estereotipada como <<interface>> y flechas
  - Interfaz ofrecido o provisto: flecha discontinua con punta hueca
  - Interfaz requerido: flecha de dependencia



Polimorfismo

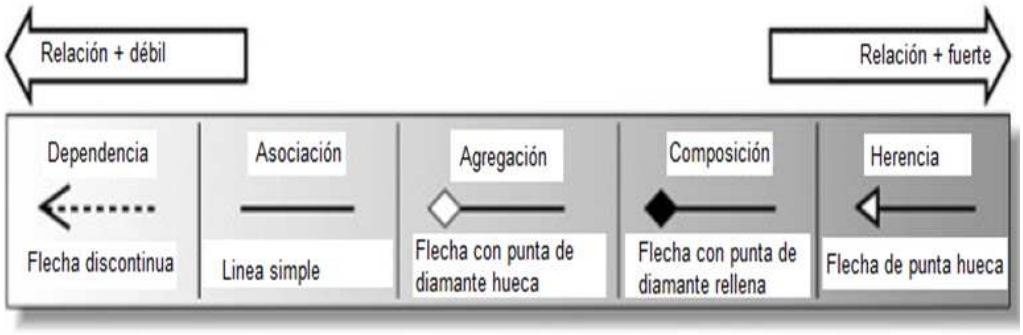
- Característica que permite implementar múltiples formas de un mismo método.
- Permite implementar una operación heredada en una subclase.
- Esto hace que se pueda acceder a una variedad de métodos distintos (todos con el mismo nombre) utilizando exactamente el mismo medio de acceso.



Relaciones entre clases

- Las clases no trabajan solas, sino que se relacionan con otras clases para cumplir con su objetivo.
- Tipos de relaciones de menor a mayor acoplamiento:
  - Dependencia o instanciación
  - Asociación
  - Agregación
  - Composición
  - Herencia

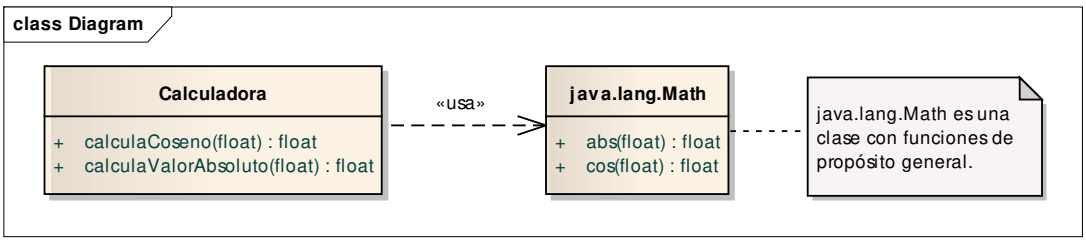
Relaciones entre clases



Cuando objetos de una clase usan objetos de otra clase por un periodo de tiempo breve.	Cuando objetos de una clase trabajan con objetos de otra clase por un periodo de tiempo prolongado.	Es un tipo de asociación en el que una clase está formada por otras clases. El tiempo de vida del objeto incluido es independiente del que lo incluye.	Es un tipo de asociación en el que una clase está formada por otras clases. El tiempo de vida del objeto incluido es dependiente del que lo incluye.	Una clase es un tipo de otra clase

Dependencia o Instanciación (uso)

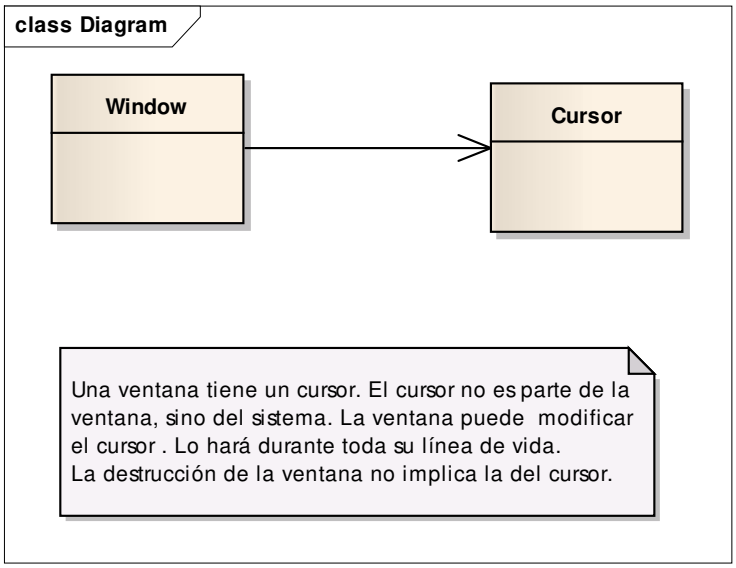
- Es un tipo de relación en el que una clase utiliza a otra, es decir, **una clase es instanciada por otra**.
- La relación de dependencia se usa a menudo cuando hay clases que proveen un conjunto de funciones de propósito general. Ejemplo: clase java.lang.Math de Java.
- La dependencia se representa con una flecha discontinua.



Asociación

- La **asociación es una relación más fuerte que la dependencia**.
- Por lo general indica que una clase mantiene una **relación con otra clase durante un período prolongado de tiempo**.
- **Las líneas de vida** de dos objetos vinculados por una asociación probablemente **no están unidas** (la destrucción de uno no implica la destrucción del otro).
- En una asociación, una clase contiene una referencia a un objeto u objetos de otra clase en forma de atributos.
- Las asociaciones se suelen leer como **"… tiene un …"**. No confundir con composición y agregación (que se leen como "…está formado por…")

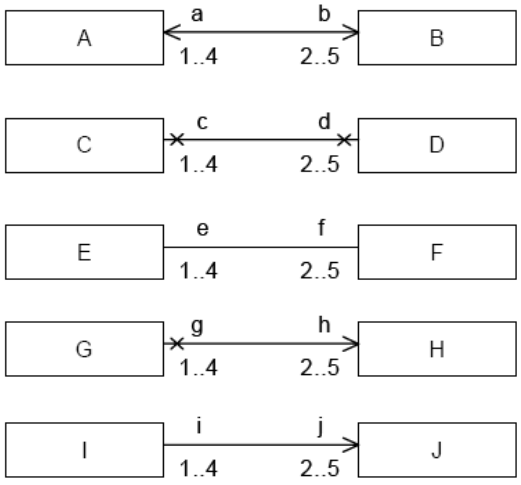
Asociación



Asociación

- **Características de la asociación:**
  - **Navegabilidad:** se aplica a menudo a una asociación para indicar qué clase contiene el atributo que soporta la relación.
  - **Nombre** de la asociación (opcional): frase para indicar el contexto de la asociación.
  - **Multiplicidad** de la asociación (opcional): indica la cantidad de objetos de una clase que pueden relacionarse con un objeto de una clase asociada

Asociación Navegabilidad



La asociación es navegable en ambos sentidos. A tendrá un atributo de tipo B y B tendrá un atributo de tipo A.

La asociación no es navegable.

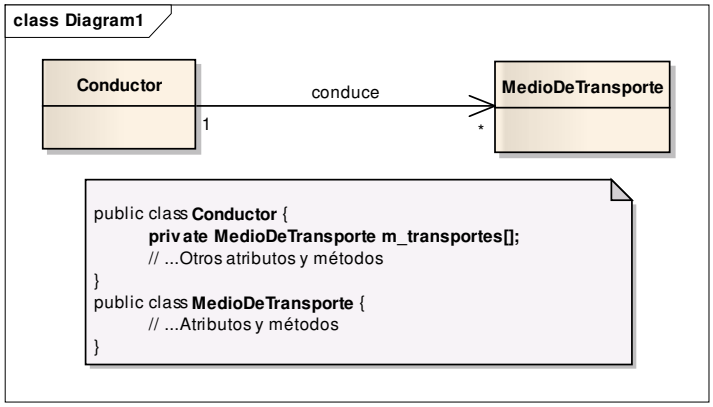
No se especifica la navegabilidad.

La asociación es navegable en un solo sentido. G tendrá un atributo de tipo H.

La asociación es navegable en uno de los sentidos y en el otro no se especifica la navegabilidad.
I tendrá un atributo de tipo J, J puede tener o no un atributo de tipo I.

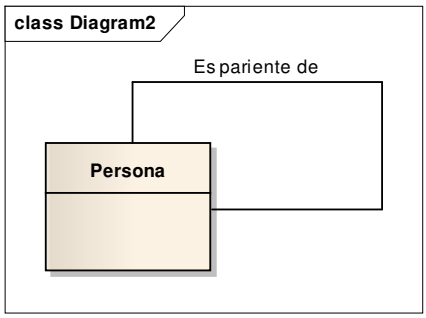
Asociación
Nombre y multiplicidad

- **Nombre** de la asociación: frase para indicar el contexto de la asociación.
- La **multiplicidad** de la asociación: indica la cantidad de objetos de una clase que pueden relacionarse con un objeto de una clase asociada.



Asociaciones reflexivas

- En algunas ocasiones una clase se asocia consigo misma.

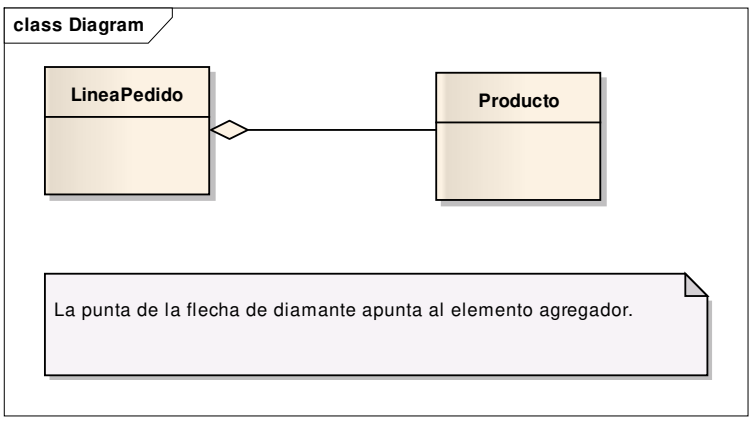


Agregación

- **Agregación:**
  - **Es un tipo especial de asociación**.
  - Se produce **cuando una clase está formada por otras clases**. Se lee "**…es parte de…**" o "…está formado por…"
  - El tiempo de vida del objeto incluido es **independiente** del que lo incluye.
  - Se representa con una línea con punta de diamante hueca. La punta de diamante está en el lado del agregador.

Agregación

- Ejemplo: una línea de pedido está formada por productos. Si la línea de pedido se elimina, el producto sigue existiendo, no se elimina.

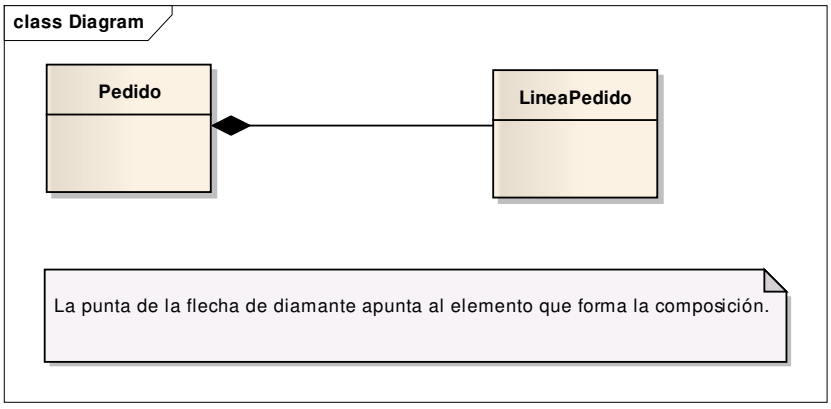


Composición

- **Composición:**
  - **Es un tipo especial de asociación** (como agregación).
  - Se produce **cuando una clase está formada por otras clases** (como agregación).
  - El tiempo de vida del objeto incluido es **dependiente** del que lo incluye (distinto de agregación).
  - Se representa con una línea con punta de diamante rellena.

Composición

- Ejemplo: un pedido está formado por líneas de pedido. Si el pedido se elimina, se eliminaran las líneas asociadas al pedido.



Ejemplo de Agregación y Composición

- Ejemplo: un pedido está formado por líneas de pedido. A su vez, una línea de pedido está formada por productos.