



# El modelado en el desarrollo de Software

**DIIT**

Departamento de Ingeniería e  
Investigaciones Tecnológicas

**DISEÑO DE APLICACIONES WEB/MÓVILES**

# Índice

- Diagrama de Clases
- Componentes del Diagrama de Clases
- Clase
- Visibilidad
- Clases de Análisis
- Interface
- Relaciones

# Diagrama de Clases

Muestra una vista estática de la estructura del sistema, describiendo qué atributos y comportamiento debe poseer el mismo.

Son los diagramas más comunes en el modelado de sistemas orientados a objetos. Se utilizan para modelado de datos detallado traduciendo las clases directamente a código.

# Clase

Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

Se utiliza para capturar el vocabulario del sistema que se está desarrollando.

El comportamiento de una clase se describe mediante los mensajes que la clase es capaz de comprender.



# Clase

## Visibilidad (abstracción)

Se utiliza para ocultar los detalles de implementación.

Hay tres niveles disponibles:

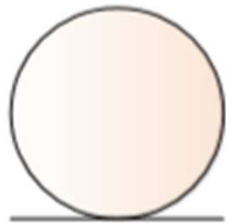
**Pública:** Cualquier clasificador externo con visibilidad hacia el clasificador dado puede utilizar la característica. Se representa con el símbolo “+”.

**Protegida:** Cualquier descendiente del clasificador puede utilizar la característica. Se representa con el símbolo “#”.

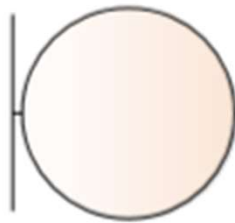
**Privada:** Sólo el propio clasificador puede utilizar la característica. Se representa con el símbolo “-”.

# Clases de Análisis

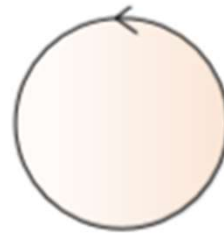
UML, tiene tres tipos de clases, denominadas clases de análisis.  
El propósito es lograr una estructura estable y mantenible del sistema.



Clase de Entidad



Clase de Interfaz



Clase de Control

# Clases de Análisis

## *Clase de Entidad*

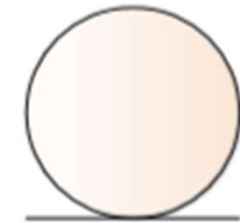
Modela información en el sistema que debería guardarse por mucho tiempo incluso luego de la ejecución del caso de uso

Se utilizan para modelar abstracciones del dominio del problema.

Identifican en el modelo del dominio del problema son un ejemplo.

Pueden identificarse desde los casos de uso.

La mayoría de las clases de entidad se encuentran pronto y son obvias.



Clase de Entidad

# Clases de Análisis



Clase de Interfaz

## ***Clase de Interface***

Modela el comportamiento e información que depende de la frontera del sistema con el ambiente.

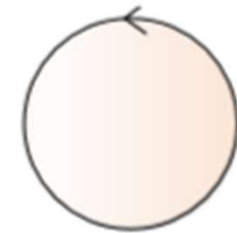
Cualquier vínculo entre el sistema y los actores, se ubica en una clase de interface.

Pueden describir comunicación bidireccional entre el sistema y sus usuarios.

Cada actor concreto necesita su propia interfaz para comunicarse con el sistema.



# Clases de Análisis



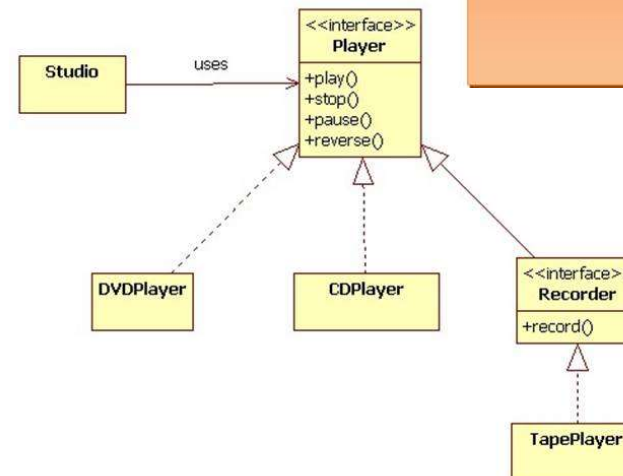
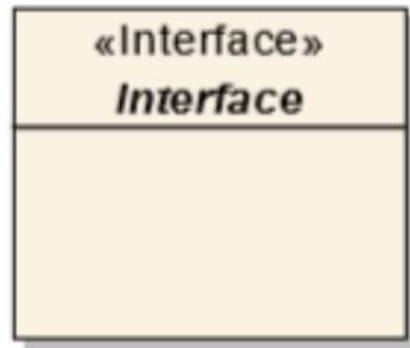
Clase de Control

## *Clase de Control*

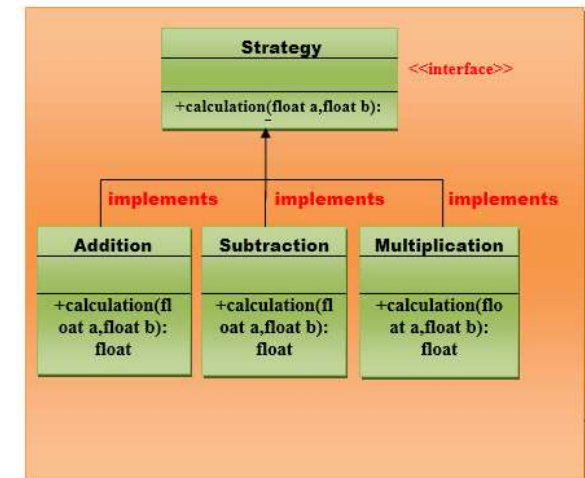
- Modela operaciones sobre varias clases de entidad diferentes, haciendo cálculos y retornando el resultado (a una clase de interfaz).
- Contiene comportamiento de la lógica de negocio definida en un caso de uso.
- Actúa como vínculo, que une los otros tipos de clase.
- Comúnmente duran mientras dura la ejecución de un caso de uso.
- Se encuentran directamente desde los casos de uso.
- En una primera vista asignaremos un objeto de control para cada caso de uso.
- Los objetos de control conectan cursos de eventos y así llevan adelante la comunicación con otros objetos.

# Interface

Es un tipo especial de clase que agrupa una colección de operaciones que especifican un servicio de una clase o componente.  
Sirve para crear una estructura o jerarquía de clase



UML for Strategy Pattern:



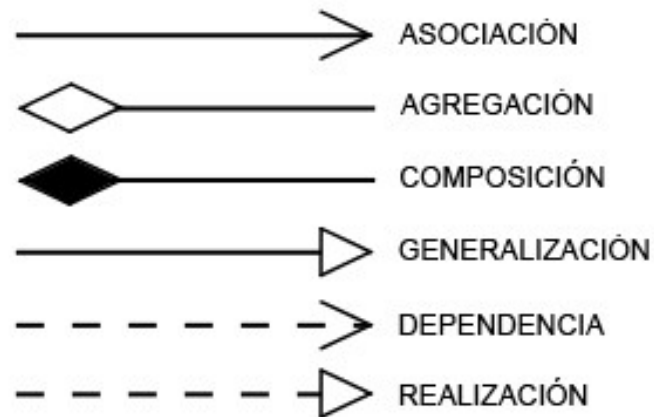
# Relaciones

Una relación es una conexión entre dos elementos.

Los elementos que se relacionan son clases e interfaces.

Los tipos de relaciones son:

*La asociación puede no tener flecha, ser una línea*



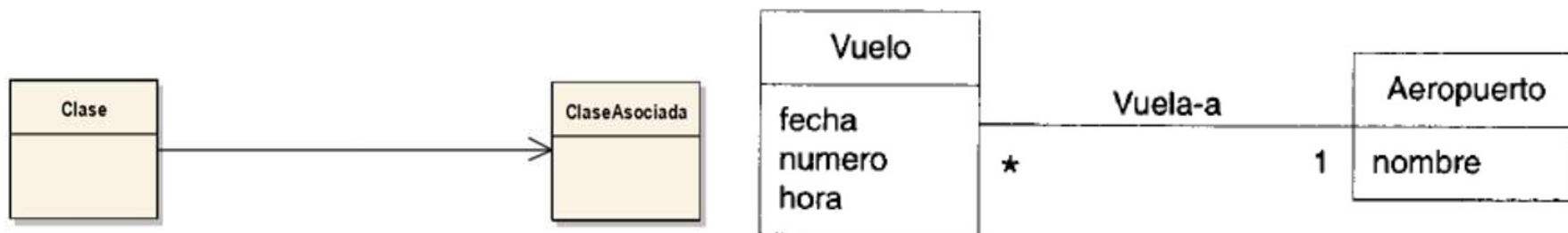
# Relaciones - Asociación

Especifica que los objetos de un elemento se conectan a los objetos de otro elemento.

Se implementa con una referencia en uno de los dos elementos que participan de la relación

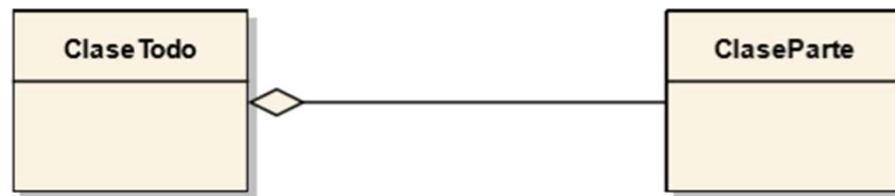
Pueden incluir multiplicidad, navegabilidad y el nombre del rol que se establece.

El rol de la asociación suele ser del tipo de uno de los elementos en la relación.



# Relaciones - Agregación

Representa una relación completamente conceptual entre un “Todo” y sus “Partes”.



## Agregación



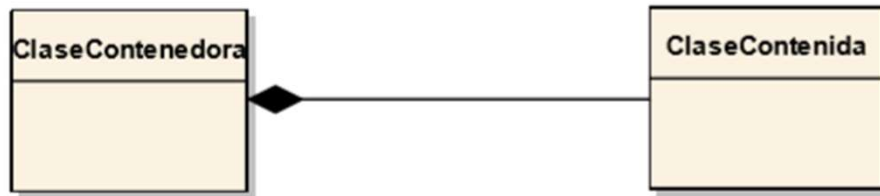
# Relaciones - Composición

Es una variación de la agregación que muestra una relación más fuerte entre el todo y sus partes.

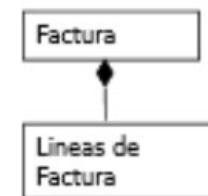
Se utiliza cuando existe una relación de contenedor-contenido entre dos elementos.

Usualmente una instancia de una parte suele pertenecer sólo a una instancia del todo.

Esta relación en general implica que al borrarse el todo se borran todas sus partes.



## Composición



# Relaciones - Generalización

Es una relación entre un elemento general (superclase o padre) y un tipo más específico de ese elemento (subclase o hijo).

El hijo hereda los métodos y atributos del padre, luego puede añadir nueva estructura y comportamiento, o modificar el comportamiento del padre.

Existen dos tipos de herencia:

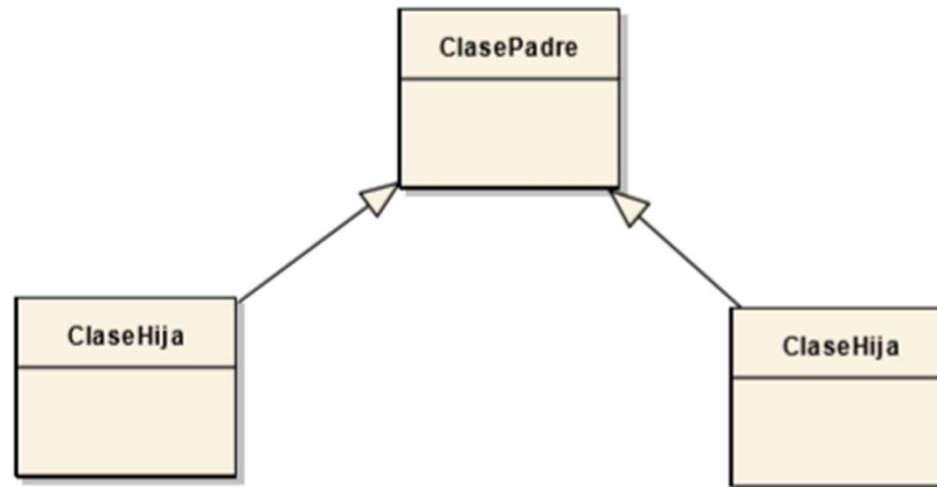
- Herencia Simple.
- Herencia Múltiple.

# Relaciones – Generalización

## Herencia Simple

Un hijo hereda de un único padre.

### Herencia





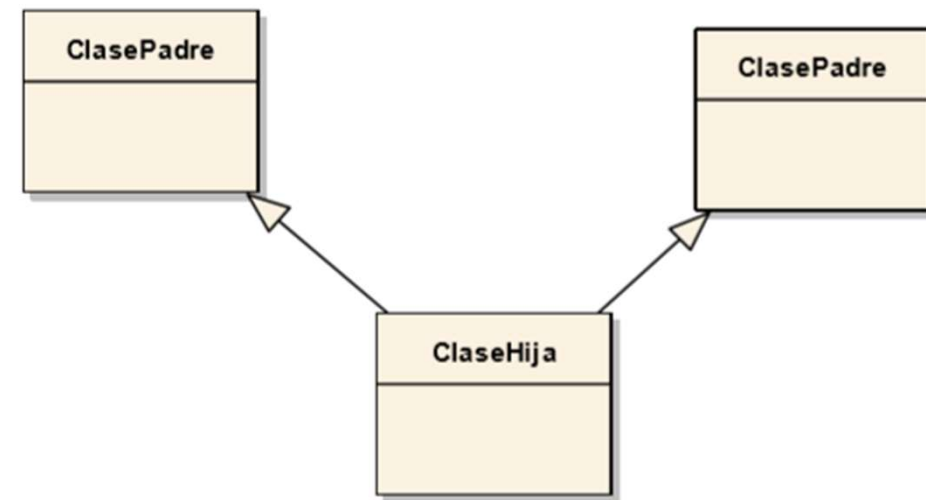
# Relaciones - Generalización

## Herencia Múltiple

Un hijo hereda de más de un padre.

Ha sido prácticamente reemplazada por clases de interfaz.

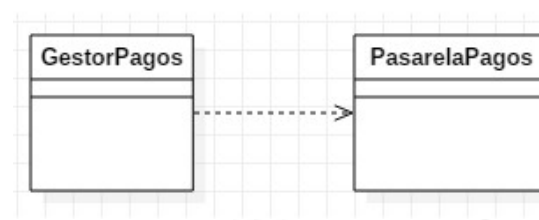
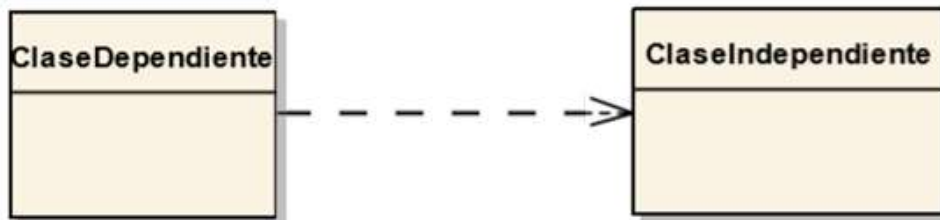
Muchos lenguajes de programación no lo soportan.



# Relaciones - Dependencia

Indica que un cambio en la especificación de un elemento puede afectar a otro elemento que lo utiliza.

Se utiliza este tipo de relación para **representar que una clase requiere de otra para ofrecer sus funcionalidades**. Es muy sencilla y se representa con una flecha discontinua que va desde la clase que necesita la utilidad de la otra flecha hasta esta misma.



# Relaciones - Realización

Implica que el elemento de donde parte la relación implementa el comportamiento definido en el otro elemento relacionado.

Indica trazabilidad entre los elementos.

Esta relación se suele utilizar entre una clase y una interfaz, cuando la clase implementa el comportamiento definido por la interfaz.

