Workflow de diseño

- ***Clases de Diseño***

Las clases de diseño son clases cuyas especificaciones se han completado hasta tal nivel que se pueden implementar. Una clase de diseño completa es una que está suficientemente detallada para servir como una buena base para crear código fuente.

El análisis trata sobre modelar lo que el sistema debería hacer.
El diseño es sobre modelar cómo el comportamiento se puede implementar.

Las clases de diseño proceden de dos lados:

- El ámbito del problema haciendo una mejora de las clases de análisis.
- El ámbito de la solución (librerías de clases de utilidad y componentes reutilizables).

¿Con que grado de detalle diseñamos las clases?
El método elegido de implementación determina el grado de totalidad que necesita en las especificaciones de la clase de diseño. Si se trata de generar código desde las clases de diseño con una herramienta de modelado apropiada, las especificaciones de clase de diseño deben estar completas en todos los aspectos.

¿Cómo convertimos las clases de análisis en clases de diseño?
Partiendo de las clases de análisis hay que realizar algunos ajustes para convertirlas en clases de diseño:

- Completar el conjunto de atributos y especificarlos incluyendo nombre, tipo, visibilidad y valor predeterminado si aplicara.
- Completar el conjunto de operaciones y especificarlas incluyendo nombre, lista de parámetros y tipo de retorno.

Clases de diseño bien creadas
Una clase de diseño bien creada debe cumplir las siguientes propiedades:

Totalidad
La característica de totalidad trata sobre proporcionar una clase de diseño que brinde todo lo que se podría esperar de la misma.

Suficiencia
La suficiencia trata sobre mantener la clase de diseño lo más sencilla y centrada posible.

La regla de oro para la totalidad y suficiencia es que una clase debería hacer o que los usuarios de la clase esperan, ni más ni menos.

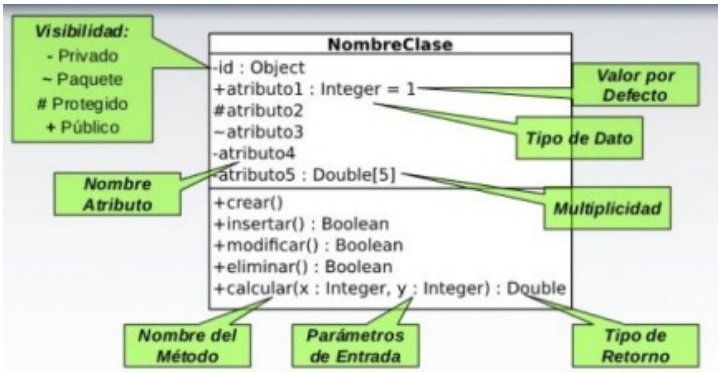
Sencillez
Una clase no debería ofrecer múltiples formas de realizar lo mismo.

Elementos del diagrama de clases

- ***Clase***

Una clase es un conjunto de objetos con la misma estructura y el mismo comportamiento.
Una clase es como un molde desde el que se crean los objetos.
Es una abstracción de un concepto que engloba a todos los objetos.

Atributos: propiedades relevantes de una clase. Representan su estructura.
Métodos: comportamiento asociado a una clase.



Visibilidad	Public	Private	Protected	Default*
Desde la misma clase	✓	✓	✓	✓
Desde una subclase	✓	✓	✓	⊘
Desde otra clase (no subclase)	✓	⊘	⊘	⊘

- ***Instancia***

Instancia es la particularización, realización específica u ocurrencia de una determinada clase, entidad o prototipo.
Instanciar es el proceso de generar instancias de una clase.
Objeto es una instancia de una clase.

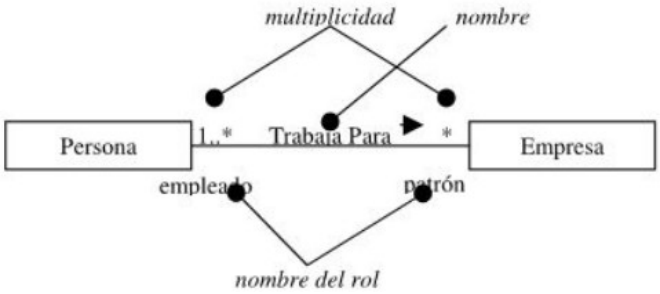
- ***Polimorfismo***

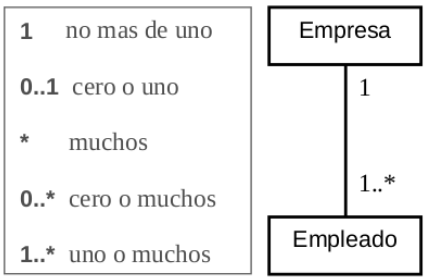
Es propiedad que tienen los métodos de mantener una respuesta unificada, con la misma semántica, aunque con distinta implementación, a través de la jerarquía de clases.

Relaciones entre clases

- ***Asociación***

Es una relación estructural que especifica que los objetos de un elemento se conectan a los objetos de otro.





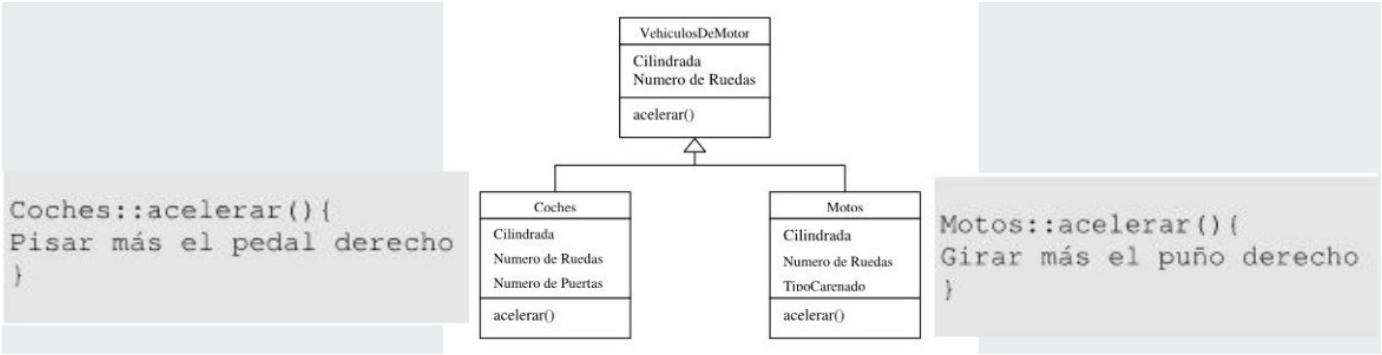
Nombre: Se utiliza para describir la naturaleza de la relación.

Rol: Cuando una clase participa en una asociación esta tiene un rol especifico que juega en dicha asociación.

Multiplicidad: En muchas situaciones del modelado es conveniente señalar cuantos objetos se pueden conectar a través de una instancia de la asociación.

- ***Herencia***

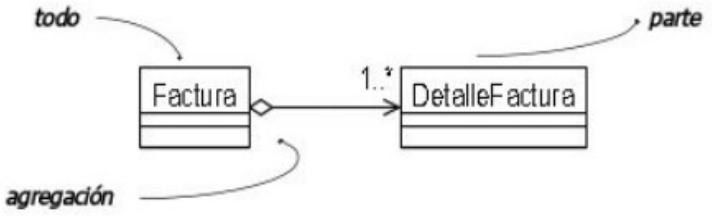
Es la relación entre una clase general y otra clase más específica.
Permite que cada clase pueda tener una clase ancestro y clases descendientes. Y también gracias a ella , cada clase hereda atributos y comportamiento de su clase ancestro.



- ***Agregación***

Es un tipo especial de asociación, que representa una relación completamente conceptual entre un “todo” y sus “partes”.

1. El conjunto puede existir algunas veces independientemente de las partes.
2. Las partes pueden existir independientemente el conjunto.
3. El conjunto está en cierto modo incompleto si faltan algunas de las partes.
4. Es posible tener propiedad compartida de las partes por varios conjuntos.
5. Las jerarquías de agregación son posibles.
6. El todo siempre sabe de las partes, pero si la relación es de un solo sentido, del todo a la parte, las partes no saben del todo.



- ***Composición***

Es una variación de la agregación simple, con una fuerte relación de pertenencia y vidas coincidentes de la parte con el todo.

1. Las partes pertenecen exactamente a un componente de cada vez.
2. El compuesto tiene la única responsabilidad de disponer de todas sus partes, esto significa responsabilidad para su creación y destrucción.
3. El compuesto puede también liberar partes, en tanto que la responsabilidad para éstas se asuma por otro objeto.
4. Si el compuesto se destruye, debe destruir todas las partes o pasar la responsabilidad de éstos a otro objeto.
5. Cada parte pertenece exactamente un compuesto por lo que solamente puede tener jerarquías de composición.

Interfaces y componentes

- ***Interfaz***

Es una colección de operaciones que especifican un servicio. Sirve para encapsular un conjunto de métodos, sin asignar esta funcionalidad a ningún objeto en particular ni expresar nada respecto del código que las va a implementar.
Una clase implementa una interfaz cuando la clase se compromete a implementar todos los métodos de la interfaz.
Las interfaces son la clave para el desarrollo basado en componentes. Si desea crear software flexible basado en componentes para lo que puede incorporar nuevas implementaciones, debe diseñar con interfaces.