Asociación _Relaciones

- *Una clase `A` tiene una relación de asociación con una clase `B`, si `A` utiliza en alguno de sus métodos un objeto de tipo `B`.*
  - *Puede ser tanto como parámetro, tipo de retorno, o internamente en alguna operación dentro de un método.*
- *Una relación de asociación implica que `A` no necesita imperiosamente conocer a `B` y que no puede subsistir sin ella.*
  - *Existe dependencia `A` de sobre `B`.*
- *La relación de asociación es sólo desde la clase que utiliza a la otra.*
  - *La clase `B` no tiene relación de asociación con la clase `A`.*
  - *La clase `B` es independiente de la clase `A`.*

Simbología

Acoplamiento _Relaciones

- *Cada vez que una clase invoca un método sobre otra, se establece una relación entre ambas.*
  - *La clase que invoca (cliente) debe conocer el tipo de la clase donde se invoca el método (servidor).*
  - *Se dice que la clase cliente está acoplada a la clase servidor.* → El client conoce al servidor
- *Mientras más necesite saber la clase cliente de la clase servidor más acoplada estará a ella.*
- *El acoplamiento rigidiza la estructura de las clases.*
  - *Es difícil cambiar el servidor porque implica cambiar el cliente también.*
  - *Imagínense si una clase es servidor de muchos clientes!*

Ejemplo

Un avión tiene sistemas de medición de parámetros operacionales (altura, velocidad, etc). Estos sistemas, cuando el parámetro sale de cierto rango deben gatillar el sistema de alarma del avión para que el piloto se de cuenta de la situación.

- *Modelando este problema, encontraríamos que existe la clase `Avion` y `Sensor`.*
- *Un avión contiene sistemas de medición.*
- *Además vemos que los sistemas de medición tienen que ser capaces de notificar al avión.*
- *¿Entonces el sensor podría tener una referencia al avión que lo contiene?*

¿Qué problemas ven en esta solución?

- *`Avion` necesita conocer a `Sensor`, de manera de poder usar los sensores.*
  - *Avion tiene una relación de composición con Sensor.*
- *Sensor necesita conocer a `Avion` para notificarlo.*
  - *Sensor tiene una relación de composición con `Avion`.*
- *Si cambiamos algo en `Sensor` o `Avion` deberemos cambiar la otra clase.*
- *Y, si queremos usar `Sensor` para algo que no sea aviones?*
- ***Exceso de acoplamiento.***

Tenemos una dependencia circular o cíclica.

Estas relaciones causan muchos problemas de mantenibilidad.

Es un problema porque si x hace un cambio en una x tiene que hacer otro en la otra.

Ejercicio: