Programación Orientada a Objetos: Asociación vs Composición

En lo personal, conceptos que me parecieron bastante difíciles de comprender cada vez que trataba de estudiar **Programación Orientada a Objetos**. Por eso trataré de crear una explicación sencilla para los que ahora se ven en mi situación. Porque el que domines un lenguaje de programación no te garantiza que harás un buen diseño del sistema.

Los dos conceptos que debes conocer cómo mínimo cuando intentas descifrar la forma en que tus objetos deben interactuar son **Asociación** y **Composición**.

Asociación

La asociación se podría definir como el momento en que dos objetos se unen para trabajar juntos y así, alcanzar una meta.

Un punto a tomar muy en cuenta es que ambos objetos son independientes entre sí, veremos un poco más adelante qué implicación tiene esto. Para validar la asociación, la frase *"Usa un"*, debe tener sentido:

- **El ingeniero** *usa* **una computadora**
- **El cliente** *usa* **tarjeta de crédito**.

Composición

En caso contrario, la composición es un tipo de relación **dependiente** en dónde un objeto más complejo es conformado por objetos más pequeños. En esta situación, la frase "*Tiene un*", debe tener sentido:

- **El auto** *tiene* **llantas**
- **La portátil** *tiene* **un teclado**.

Y como ésta mini guía no va a mencionar nada de **UML**. Vamos a ver directamente en código cómo se verían representadas ambos tipos de relaciones.

El código es Java, pero funciona para cualquier lenguaje de programación orientado a objetos.

Cómo implementar *Asociación*

Representaremos la relación: **El cliente** *usa* **tarjeta de crédito**.

Código :

```
public class Customer {


    private int id;

    private String firstName;

    private String lastName;

    private CreditCard creditCard;


    public Customer() {

        //Lo que sea que el construtor haga

    }


    public void setCreditCard(CreditCard creditCard) {

        this.creditCard = creditCard;

    }


    // Más código aquí

}
```

La explicación viene más adelante para darles oportunidad que hagan sus propias comparaciones.

Cómo implementar *Composición*

Representaremos la relación: **La portátil** *tiene un* **teclado**.

Código :

```
public class Laptop {

    private String manufacturer;

    private String model;

    private String serviceTag;

    private KeyBoard keyBoard = new KeyBoard();


    public Laptop() {

        //Lo que sea que el constructor haga

    }




}
```

Muy similar, pero hay una gran diferencia: **Podemos crear un objeto de tipo Customer y asignarle un CreditCard más tarde mediante el método setCreditCard**.

Pero si creamos un objeto **Laptop**, de entrada sabremos que tendrá un teclado ya creado, puesto que la variable de referencia **keyBoad** es declarada e inicializada al mismo tiempo.

Llamaremos a las clases **Customer** y **Laptop**, *clases contenedoras*.

De ambos casos podemos deducir que:

En la asociación:

1. **Customer** es independiente de **CreditCard**, puesto que el cliente puede existir sin necesidad de tener asignada una tarjeta de crédito. Démosle tiempo para que la tramite, ¡Pero no lo dejemos ir!

2. Se puede asignar o retirar la tarjeta de crédito, sin que la existencia del Cliente se vea afectada (No debería verse afectada, esto significa que **Customer**no debe tronar si no hay un **CreditCard** presente).

En la composición:

1. Los objetos que componen a la clase contenedora, deben existir desde el principio. (También pueden ser creados en el constructor, no sólo al momento de declarar las variables como se muestra en el ejemplo).
2. No hay momento (No debería) en que la **clase contenedora** pueda existir sin alguno de sus objetos componentes. Por lo que la existencia de estos objetos no debe ser abiertamente manipulada desde el exterior de la clase.

Tiempo de vida de un objeto

Para que quede más clara la diferencia entre Asociación y Composición, entendamos además, lo que es el **tiempo de vida de un objeto**.

Se define como el tiempo que transcurre desde que un objeto es creado hasta que se destruye.

Aplicando esto a la asociación, tenemos que los tiempos de vida de ambos objetos se cruzan mientras están trabajando juntos, esto es, mientras se encuentran asociados, pero no significa que se hayan creado al mismo tiempo.

En el ejemplo del cliente, puede ser que primero se cree el cliente, después la tarjeta de crédito y luego viene la asociación. O incluso se puede crear antes la tarjeta de crédito. Sus tiempos de vida se cruzan sólo mientras la tarjeta de crédito está asociada al cliente.

En la composición, tanto los objetos componentes como la clase contenedora, nacen y mueren al mismo tiempo. Esto es, tienen el mismo tiempo de vida.

Al ser una relación demasiado dependiente, si cualquier objeto muere, se lleva consigo a todos los demás. En el ejemplo de la portátil, si mi teclado se descompone, mi laptop ya no es funcional.

Y no vengan con que puedo reemplazar el teclado, y que puedo seguir trabajando con la misma portátil y un teclado distinto. Tienes que analizarlo de la siguiente forma:

Si a tu auto, se le poncha una llanta, podrás reemplazarlas siempre y cuando lo tengas estacionado (Es como modificar el código de la clase, el sistema no está en funcionamiento). Pero ¿Qué pasa si tu auto estuviera en marcha?, ¿puedes cambiarla al vuelo e impedir que el auto se detenga? No se puede, por lo tanto tu auto deja de cumplir

su objetivo en ese momento y para fines prácticos, ya no sirve. Entonces es lo mismo con los objetos ya creados, no puedes reemplazarles componentes al vuelo porque no existe (no debería) mecanismo alguno en la definición de la clase, que te lo permita.

¿Asociación o Composición? … depende

Habrá casos en que será difícil determinar qué tipo de relación usar cuando ambas encajan:

- **Un reloj** *tiene* **manecillas**
- **Un reloj** *usa* **manecillas (Para dar la hora, claro)**.

Así que debes tomar en cuenta qué tanta flexibilidad te daría implementar una u otra.

Desde el punto de vista de fabricante de relojes, necesito tener control sobre cada una de las piezas que conforman mis relojes; así, si alguna pieza sale defectuosa, puedo reemplazarla antes que mi producto llegue al mercado. Me conviene la asociación.

Pero desde el punto de vista de Consumidor final, Si mis manecillas se friegan, pues tiro el reloj entero y me compro uno nuevo. Lo vería como composición.

Y terminaré diciendo lo mismo que dicen la mayoría de las lecturas que tratan este tema:

Todo depende del cristal con que se mire (Más propiamente dicho, el que necesites).

Pero espero y haya logrado darles una perspectiva más clara de cómo y cuándo aplicar **Asociación** y **Composición**.