Programación orientada a objetos

2.1 Paradigma orientado a objetos

Metodología de desarrollo de aplicaciones en la cual éstas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representan una instancia de alguna clase, y cuyas clases son miembros de jerarquías de clases unidas mediante relaciones de herencia.

La POO está compuesta por:

- Clase

Una clase es un **modelo** que se utiliza para crear objetos que comparten un mismo comportamiento, estado e identidad.

```
class Persona {
    # Propiedades
    # Métodos
}
```

Los objetos son entidades que combinan estado (atributo),comportamiento (método) e identidad:

- El estado está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).

```
$persona = new Persona();
/*
El objeto, ahora, es $persona,
que se ha creado siguiendo el modelo de la
clase Persona
*/
```

- El comportamiento está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.

```
function caminar() {
        #...
}
```

- **Evento y Mensaje**

Un evento es un suceso en el sistema mientras que un mensaje es la comunicación del suceso dirigida al objeto.

La identidad es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador (concepto análogo al de identificador de una variable o una constante).

```
$nombre = 'Juan';
$edad = '25 años';
$altura = '1,75 mts';
```

2.1.1 Encapsulamiento, herencia, polimorfismo

•**Encapsulamiento:** Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema.

```php
<?php

class Algo {
// En OOP se utilizan generalmente nombres comenzados con mayúscula.
var $x;

function setX($v) {
// Para los métodos se utilizan generalmente nombres en minúscula y sólo
// se utiliza mayúscula para separar las palabras, por ej. getValueOfArea()
$this->x=$v;
}

function getX() {
return $this->x;
}
}

?>
```

- **Herencia:** las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen.

```php
<?php

class Another extends Something {
var $y;
function setY($v) {
// Para los métodos se utilizan generalmente nombres en minúscula y sólo
// se utiliza mayúscula para separar las palabras, por ej. getValueOfArea()
$this->y=$v;
}

function getY() {
return $this->y;
}
}

?>
```

Polimorfismo: permite a un mismo nombre de método representar código diferente, en consecuencia puede expresar muchos comportamientos distintos.

```
function niceDrawing($x) {
//Supongamos que este es un método de la clase Board.
$x->draw();
}

$obj=new Circle(3,187);
$obj2=new Rectangle(4,5);

$board->niceDrawing($obj); //Podemos llamar al método draw de circulo.
$board->niceDrawing($obj2); //Podemos llamar al método draw de rectángulo.
```

2.1.3 Diseño orientado a objetos (paso a paso)

Es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema.

Los pasos para un buen diseño son los siguientes:

- Diagrama de Estructura Estática.
- Diagrama de Casos de Uso.
- Diagrama de Secuencia.
- Diagrama de Colaboración.
- Diagrama de Estados.

2.1.4 UML

Es un lenguaje de modelado que permite la representación conceptual y física de un sistema.

Objetivos

- Un lenguaje visual de modelado, expresivo y sencillo en su uso.
- Mantener una independencia de los métodos y de los lenguajes de programación.
- Establecer bases formales.
- Imponer un estándar mundial.
- Integrar las mejores prácticas.
- Modelar sistemas, y no únicamente software.
- Establecer las relaciones entre modelos conceptuales y ejecutables.
- Crear un lenguaje de modelado utilizable tanto por máquinas como por hombres.

2.2 Funcionalidades de php

2.2.1 Instanciando objetos

Los objetos son instancias a una clase de una clase existente, por lo tanto cada objeto es único.

En php creamos un objeto usando la palabra reservada que es el operador **new** instancia una clase asignando memoria para el objeto nuevo del tipo indicado. **new** necesita un sólo argumento, que es una llamada a alguno de los métodos constructores que tenga la clase que se debe instanciar. Los métodos constructores son métodos especiales que proporciona cada clase y son los responsables de la inicialización de los nuevos objetos de ese tipo. El operador new crea el objeto y el constructor lo inicia.

2.2.2 Asociación, Agregación 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:

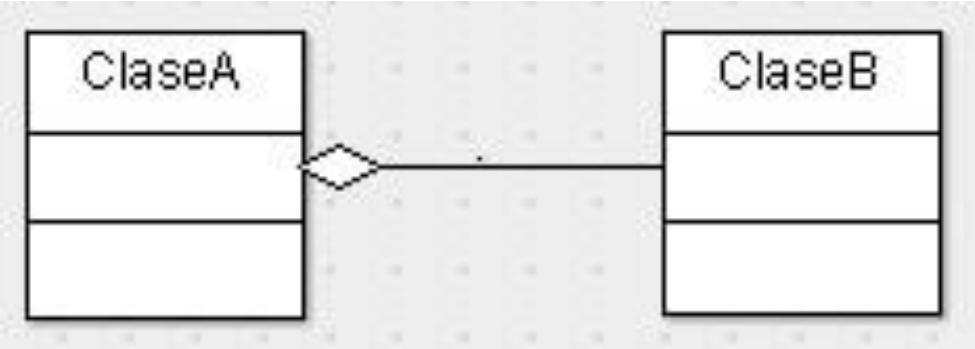
- **El ingeniero** *usa* **una computadora**
- **El cliente** *usa* **tarjeta de crédito**.

Agregación

Es muy similar a la relación de Asociación solo varía en la multiplicidad ya que en lugar de ser una relación "uno a uno" es de "uno a muchos".

Representación UML

Se representa con una flecha que parte de una clase a otra en cuya base hay un rombo de color blanco.

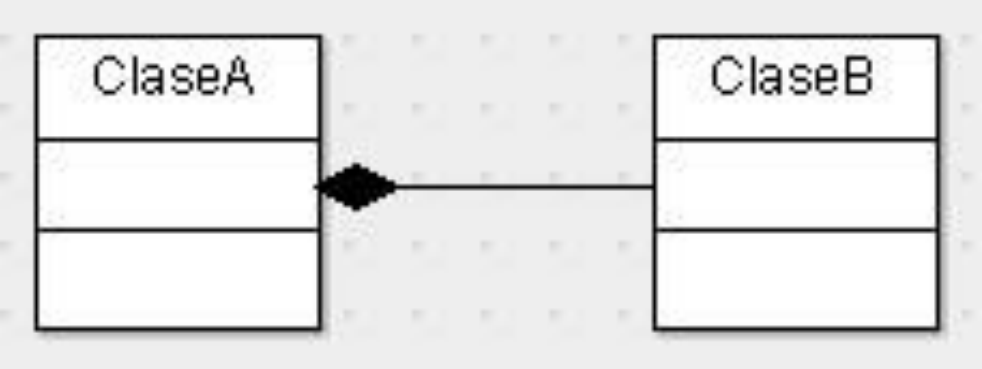


Composición

Similar a la relación de Agregación solo que la Composición es una relación mas fuerte. Aporta documentación conceptual ya que es una "relación de vida", es decir, el tiempo de vida de un objeto está condicionado por el tiempo de vida del objeto que lo incluye.

Representación UML

Se representa con una flecha que parte de una clase a otra en cuya base hay un rombo de color negro.



2.2.4 Sobrecarga y sobreescritura

Sobrecarga

La sobrecarga en PHP ofrece los medios para "crear" dinámicamente propiedades y métodos. Estas entidades dinámicas se procesan por los métodos mágicos que se pueden establecer en una clase para diversas acciones.

Se invoca a los métodos de sobrecarga cuando se interactúa con propiedades o métodos que no se han declarado o que no son visibles en el ámbito activo. A lo largo de esta sección usaremos los términos "propiedades inaccesibles" y "métodos inaccesibles" para referirnos a esta combinación de declaración y visibilidad.

Sobreescritura

Una subclase declara atributos nuevos y operaciones sobre una superclase. Es posible y en muchos casos útil sobrescribir las mismas operaciones o atributos declarados en la superclase. Esto se hace para dar a un atributo un valor diferente en la subclase que el que tiene en la superclase o en el caso de las operaciones para cambiar la funcionalidad de estas.