Introducción POO

Introducción **P**rogramación
Orientada a **O**bjetos

Evolución de los Lenguajes

- Implementaciones más simples, flexibles y portables
- Impulsada:
  - avances tecnológicos
  - avances conceptuales
  - avances en cuanto a enfoque de la programación

Evolución Tecnológica

Lenguajes de bajo nivel (Ensambladores): Un programa escrito en lenguaje ensamblador solo podrá ser ejecutado en la máquina donde fue diseñado.

Lenguaje de alto nivel: cada instrucción puede equivaler a varias decenas e incluso cientos de instrucciones en ensamblador.

Evolución Conceptual

- Programación Lineal: las instrucciones se ejecutan en el mismo orden en que las escribimos
- Programación Estructurada: enfocada en procedimientos
- Programación Orientada a Objetos

Evolución de enfoque

Programación procedural: Java, C,Pascal, BASIC, Cobol, Fortran, APL, RPG, Clipper, etc

Programación declarativa: declaran hechos, para extraer las conclusiones que resulten inferibles de esos hechos. Ej ProLog

Programación orientada a objetos: se definen clases que implementan las características y comportamiento de los objetos.

Paradigma Orientado a Objetos

- Conceptos que se introducen en POO: Objetos/instancias, atributos/variables instancias, comportamiento/métodos y clases.
- Los objetos tienen estado y un comportamiento, ya que los valores que tengan los atributos de una instancia determinan el estado actual de esa instancia, y los métodos denidos en una clase determinan cómo se va a comportar ese objeto

POO – PHP

POO (clase) - PHP

- Definición e implementación de una clase, permite la creación de objetos de esa clase
- Las instancias / objetos de una clase, tienen las mismas caracteristicas comportamiento

```
class UnaClase{

    $unObj = new UnaClase($param);

}
```

POO (atributos) - PHP

- Los atributos de una clase, definen las características
- Los métodos de una clase, definen el comportamiento

```
class UnaClase{
        private var_1 ;
        ...
        private var_n :

         public function c_1 (..){
          }
        ....
        public function c_n (..){
         }
}
```

POO (métodos) - PHP

- Método constructor: crear objetos de una clase.
- Método destructor: objetos inaccesibles.
- Métodos acceso para cada atributo:
  - getVar_1()
  - setVar_2($valor)
- Metodo __toString: retornar un string represente a los objetos de la clase.

```
class UnaClase{
        public function __construct(..){  }
        public function __destruct(..){  }
        public getVar_1(){          } ;
        public setVar_1($valor1){          } ;
        ……
        public getVar_n(){          } ;
        public setVar_n($valor_n){          } ;
        public function toString (){
            …
            return $cadena;
        }
}
```

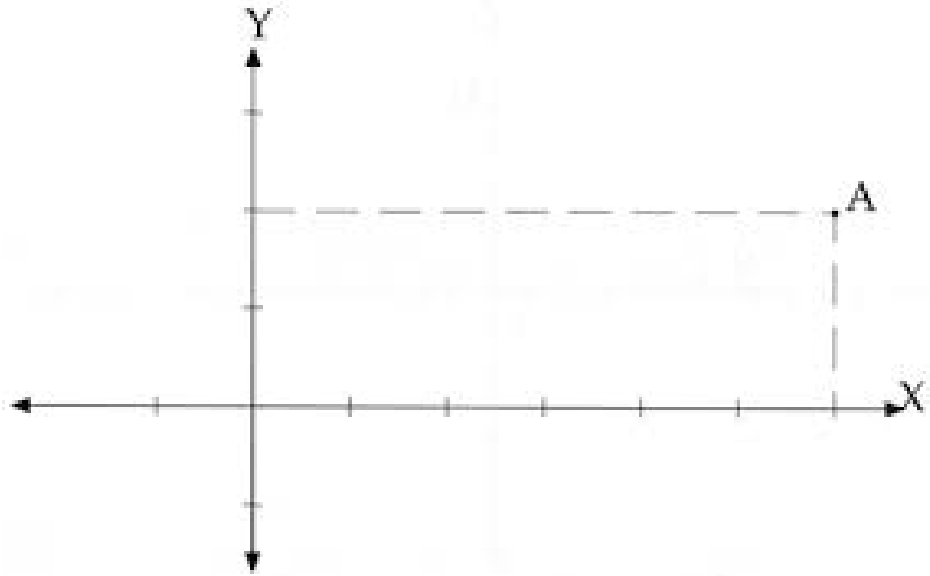
POO (Acceso) - PHP

- Se pueden definir distintos tipo de accesos sobre las variables instancias y métodos:
  - public
  - private
  - protected

```
class UnaClase{
        private var_1 ;
        ...
        public var_n :

         public function c_1 (..){
          }
        ....
        public function c_n (..){
         }
}
```

Ejemplo




```php
<?php
class Punto{
    private  $coordenada_x ;
    private $coordenada_y ;
    public function  __construct($x, $y){
        $this->coordenada_x = $x;
        $this->coordenada_y = $y;
    }
    public function getCoordenada_x(){
        return $this->coordenada_x;
    }
    public function getCoordenada_y(){
        return $this->coordenada_y;
    }
    public function setCoordenada_y($y){
        $this->coordenada_y=$y;
    }
    public function setCoordenada_x($x) {
        $this->coordenada_x=$x;
    }
    public function distancia($otroPunto){
            $dx = $this->getCoordenada_x() - $otroPunto->getCoordenada_x();
            $dy = $this->getCoordenada_y() - $otroPunto->getCoordenada_y();
            $laDistancia = pow(($dx*$dx + $dy*$dy),0.5);
            return $laDistancia;
    }
    public function restar($otroPunto){
            $x = $this->getCoordenada_x() - $otroPunto->getCoordenada_x();
            $y = $this->getCoordenada_y() - $otroPunto->getCoordenada_y();
            $nuevoPunto =  new Punto($x,$y);
            return $nuevoPunto;
    }
    public function __toString(){
        return "(".$this->getCoordenada_x().",".$this->getCoordenada_y().")\n";
    }
}?>
```

Repaso

- Objeto
- Clase
- Atributos
- Métodos
- Acceso a atributos y métodos

Atributos y métodos clases

- Pertenecen a la clase, en lugar de pertenecer a una instancia de la misma.
- El valor de un atributo de clase es el mismo para todas las instancias.
- La invocación de un método de clase se hace a partir de la clase.

```
<?php
    Static $atributo_1;
    ...
    Static $atributo_n;
    Static function c_1 (..){
     }
    ...
    Static function c_n (..){
    }
?>
```

Preguntas ?