# Curso de Programación en PHP

# Nivel I

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología – Oro Verde – v2.1

## Capítulo 5: POO y patrones en PHP

**Repaso con un ejemplo de POO en PHP**

**Herencia**

**Clases abstractas en PHP**

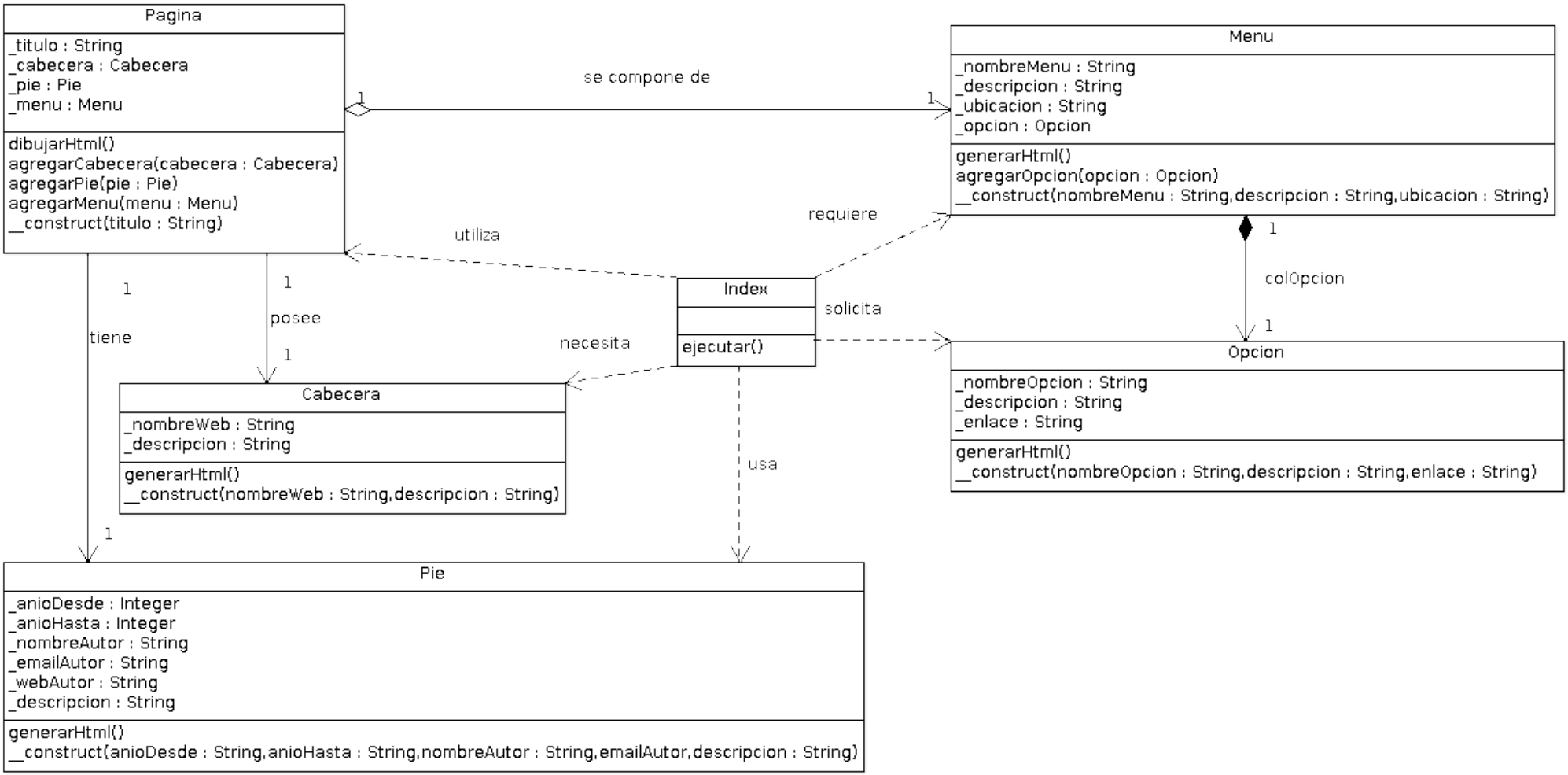
**Sobre-escritura de métodos y constructores**

**La palabra reservada final y static**

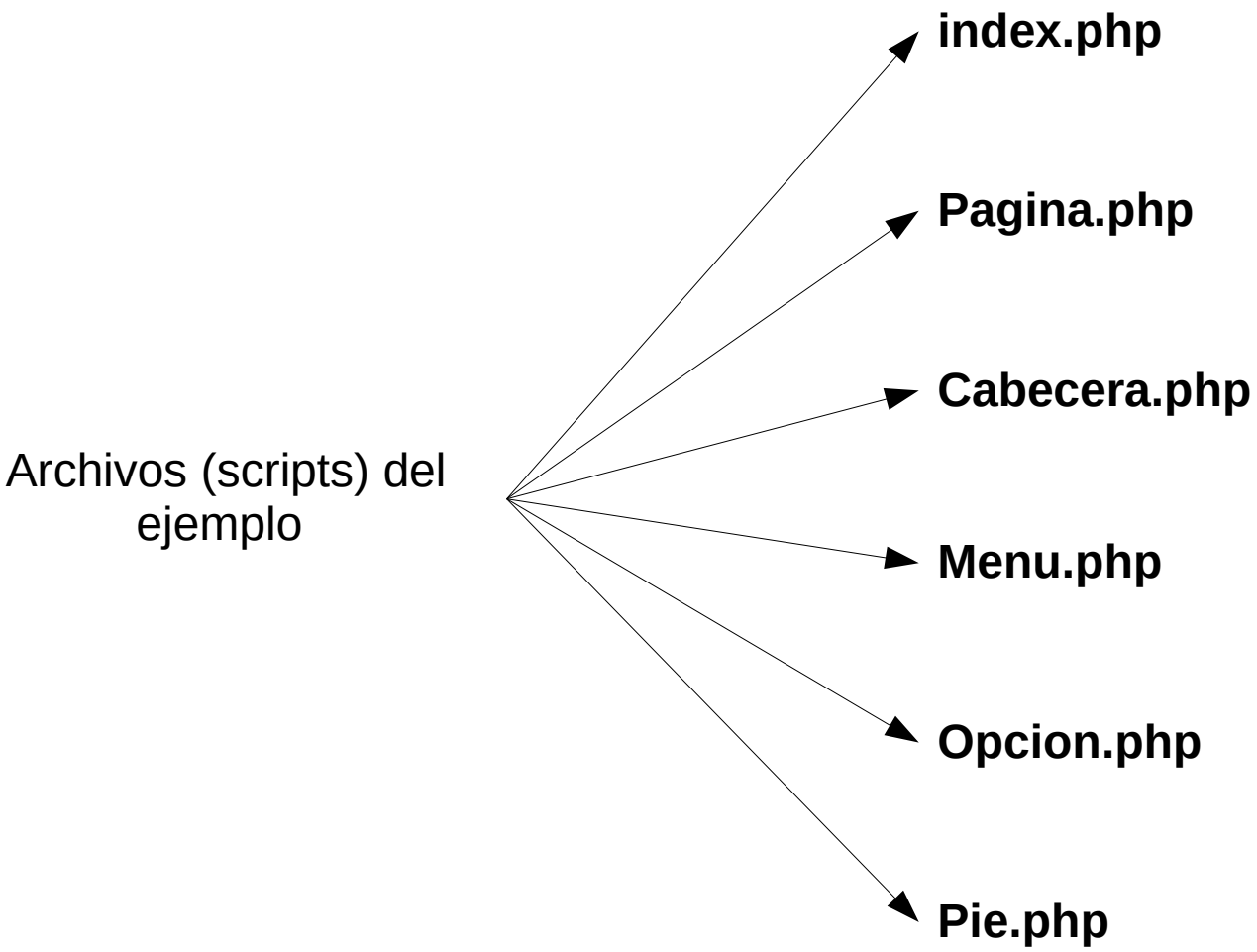
**Polimorfismo**

**Interfaces**

# Diagrama de clases UML del ejemplo



# Codificación en PHP del ejemplo



## Clase 9: POO en PHP

**index.php**

```
<?php
require_once 'Pagina.php';
require_once 'Cabecera.php';
require_once 'Pie.php';
require_once 'Menu.php';
require_once 'Opcion.php';

class Index
{
	public function ejecutar()
	{
		// Se crea el objeto de clase Pagina con un título
		$oPagina = new Pagina('Prueba');
		// Se crea la cabecera de la Página con nombre y descripción
		$oCabecera = new Cabecera('Página Prueba', 'descripción de página');
		// Se crea el menú de la página
		$oMenu = new Menu('Principal', 'Menú principal', 'horizontal');
		// Se crean las opciones de menú para la página
		$oOpcionInicio = new Opcion('Inicio', 'Página de Inicio', 'index.php');
		...
	}
}
```

## Clase 9: POO en PHP

**Pagina.php**

```
<?php
require_once 'Cabecera.php';
require_once 'Pie.php';
require_once 'Menu.php';
class Pagina
{
	private $_titulo;
	private $_cabecera;
	private $_pie;
	private $_menu;
	public function __construct($titulo)
	{
		$this->_titulo = $titulo;
	}
	public function agregarCabecera(Cabecera $cabecera)
	{
		$this->_cabecera = $cabecera;
	}
	public function dibujarHtml()
	{
		$cadena = '';
		...
```

## Clase 9: POO en PHP

**Cabecera.php**

```php
<?php

class Cabecera
{
	private $_nombreWeb;
	private $_descripcion;
	public function __construct($nombre, $descripcion)
	{
		$this->_nombreWeb = $nombre;
		$this->_descripcion = $descripcion;
	}
	public function generarHtml()
	{
		$cadena = '<h1>'.$this->_nombreWeb.'</h1>';
		$cadena .= '<br/>';
		$cadena .= '<h3>'.$this->_descripcion.'</h3>';
		$cadena .= '<br/>';
		$cadena .= '<hr/>';

		return $cadena;
	}
}
```

## Clase 9: POO en PHP

**Menu.php**

```
<?php
require_once 'Opcion.php';
class Menu
{
	private $_nombreMenu;
	private $_ubicacion;
	private $_descripcion;
	private $_opcion = array();
	public function __construct($nombeMenu, $ubicacion='horizontal', $descripcion)
	{
		$this->_nombreMenu = $nombreMenu;
		$this->_ubicacion = $ubicacion;
		$this->_descripcion = $descripcion;
	}
	public function agregarOpcion(Opcion $opcion)
	{
		$this->_opcion[] = $opcion;
	}
	public function generarHtml()
	{
		...
```

## Clase 9: POO en PHP

**Opcion.php**

```
<?php
class Opcion
{
	private $_nombreOpcion;
	private $_descripcion;
	private $_enlace;

	public function __construct($nombreOpcion, $descripcion, $enlace)
	{
		$this->_nombreOpcion = $nombreOpcion;
		$this->_descripcion = $descripcion;
		$this->_enlace = $enlace;
	}

	public function generarHtml()
	{
		$cadena = '';
		$cadena .= "<a href='".$this->_enlace."' title='".$this->_descripcion."'>".$this->_nombreOpcion."</a>";
		return $cadena;
	}
}
```

## Clase 9: POO en PHP

**Pie.php**

```
<?php
class Pie
{
	private $_anioDesde;
	private $_anioHasta;
	private $_nombreAutor;
	private $_emailAutor;
	private $_webAutor;
	private $_descripcion;
	public function __construct($anioDesde, $anioHasta, $nombreAutor, $emailAutor,
$webAutor, $descripcion)
	{
		$this->_anioDesde = $anioDesde;
		$this->_anioHasta = $anioHasta;
		$this->_nombreAutor = $nombreAutor;
		$this->_emailAutor = $emailAutor;
		$this->_webAutor = $webAutor;
		$this->_descripcion = $descripcion;
	}
	public function generarHtml()
	{
		...
```

## Capítulo 5: POO y patrones en PHP

~~Repaso con un ejemplo de POO en PHP~~

**Herencia**

**Clases abstractas en PHP**

**Sobre-escritura de métodos y constructores**

**La palabra reservada final y static**

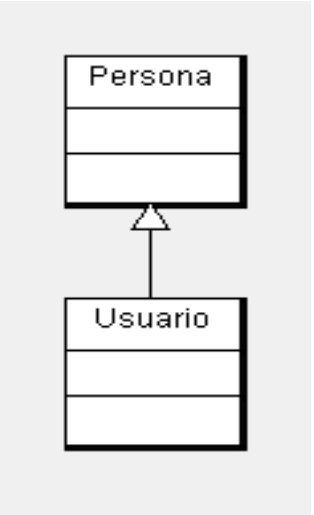
**Polimorfismo**

**Interfaces**

## Clase 9: POO en PHP

# Herencia

**Definición:** es una relación donde una clase nueva se crea a partir de una clase existente.



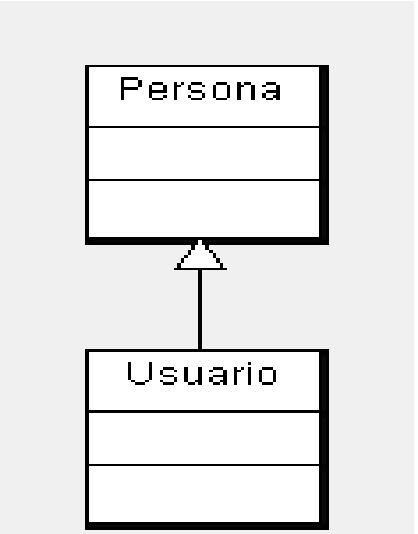

# UML (diagrama de clases)

# Codificación en PHP:

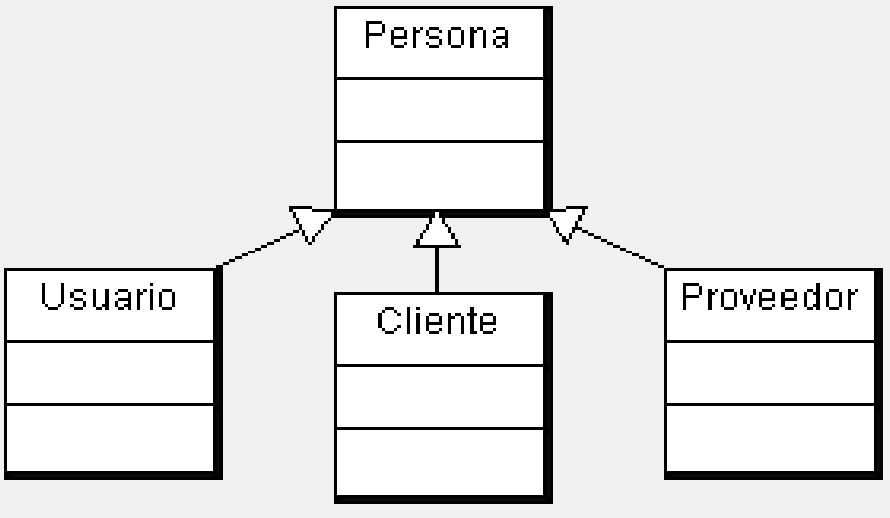
```
<?php
require_once 'Persona.php';
    class Usuario extends Persona
    {
    }
```

## Clase 9: POO en PHP

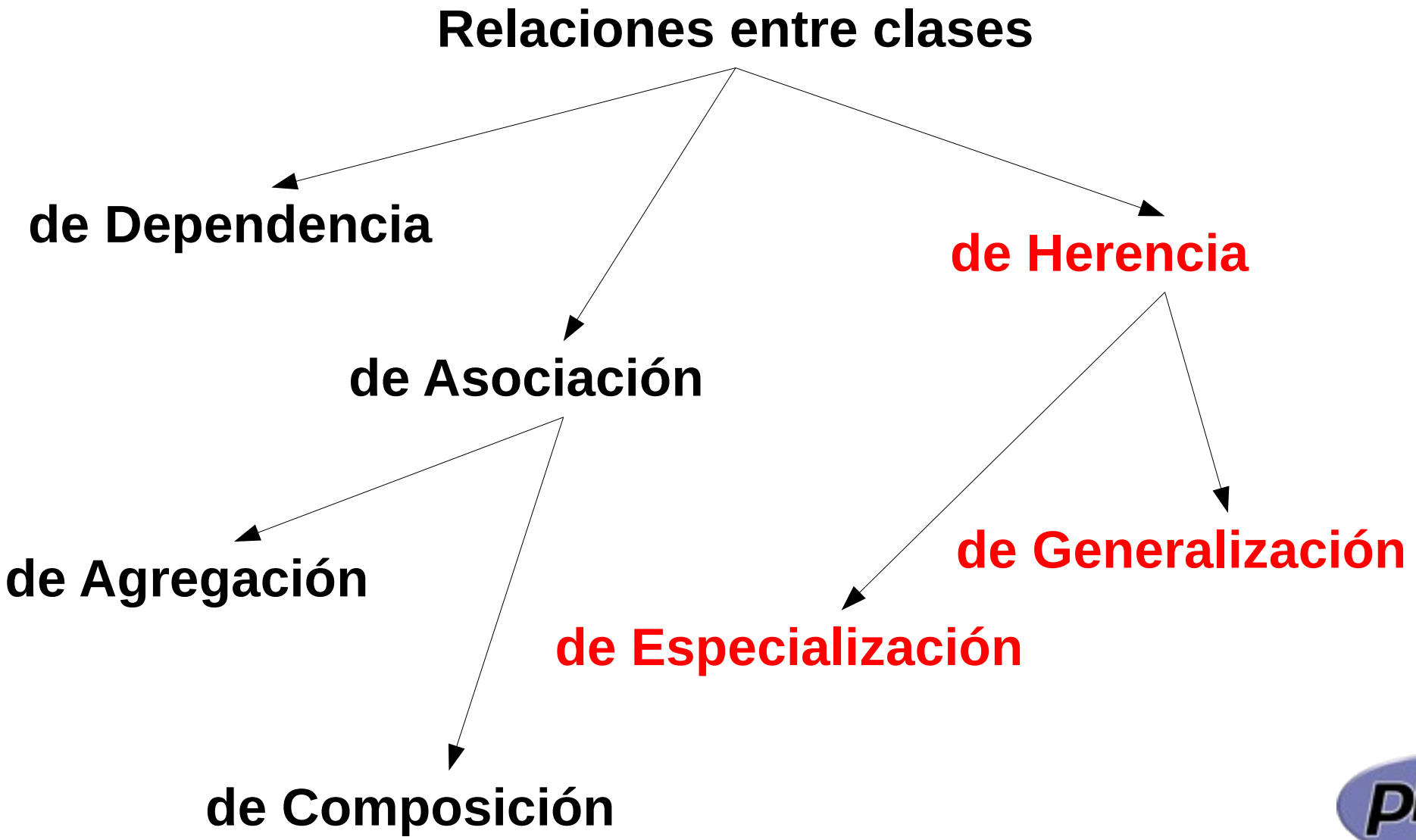
**Herencia (**Especialización**)**



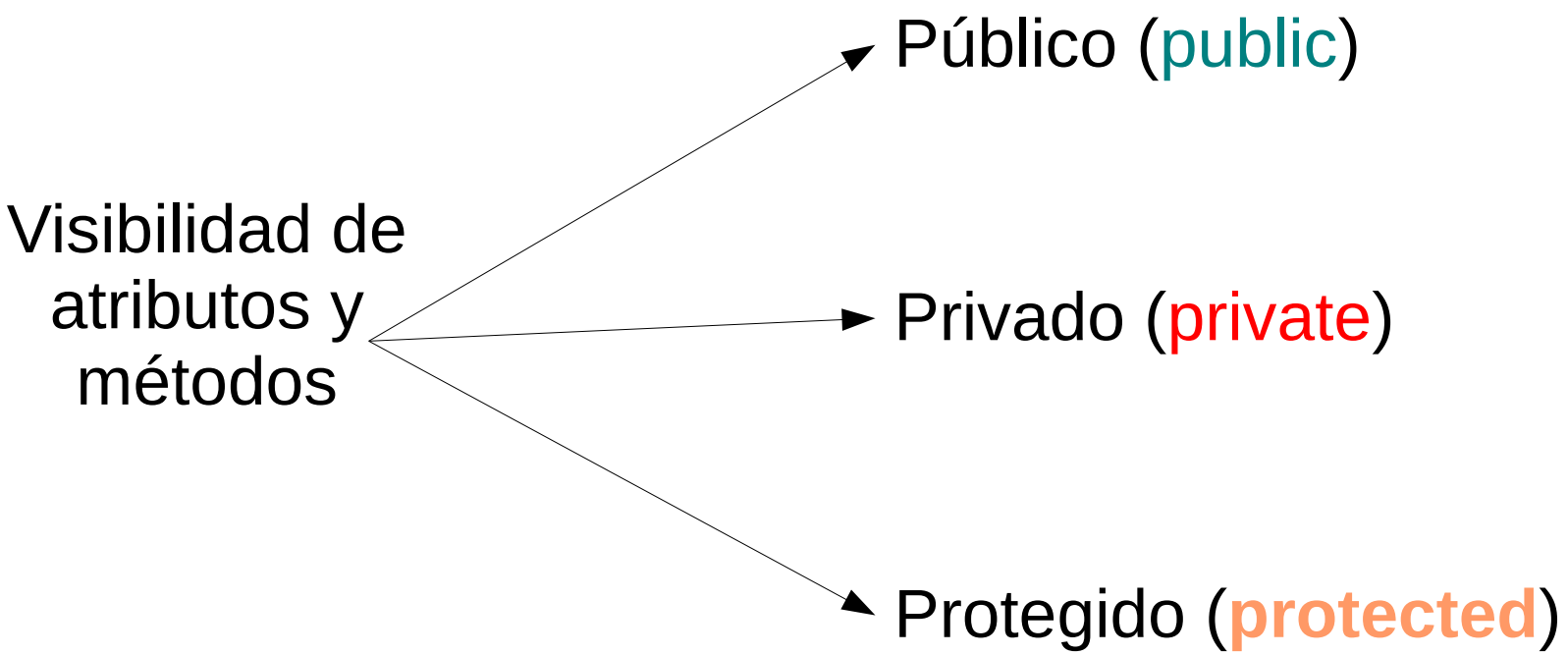

**Herencia (**Generalización**)**



## Clase 9: POO en PHP



## Visibilidad teniendo en cuenta la Herencia



## Visibilidad teniendo en cuenta la Herencia

Visibilidad Pública (**public**)
Las clases hijas u otras pueden acceder sin restricciones a los atributos y métodos de su clase padre.

Visibilidad Privada (**private**)
La clase padre puede acceder sin restricciones a los atributos y métodos pero no así sus clases hijas u otras.

Visibilidad Protegida (**protected**)
La clases hijas pueden acceder sin restricciones a los atributos y métodos de su clase padre pero no así para otras clases.

## Ejemplo de Visibilidad de atributos (Herencia)

```
class A
{
	public $publica = 'pub';
	private $_privada = 'priv';
	protected $protegida = 'prot';

}
```

```
class B extends A
{
	public function imprimir()
	{
		echo $this->publica;
		echo $this->_privada;
		echo $this->protegida;
	}
}
```

```
$oB = new B();
$oB->imprimir();
```

**Salida:** pubprot

## Otro ejemplo de Visibilidad de atributos (Herencia)

```
class A
{
	public $publica = 'pub';
	private $_privada = 'priv';
	protected $protegida = 'prot';

}
```

```
class B extends A
{
	public function imprimir()
	{
		echo $this->publica;
		echo $this->_privada;
		echo $this->protegida;
	}
}
```

```
$oB = new B();
echo $oB->publico;
echo $oB->_privada;
echo $oB->protegida;
```

**Salida:** pub

## Capítulo 5: POO y patrones en PHP

~~Repaso con un ejemplo de POO en PHP~~

~~Herencia~~

**Clases abstractas en PHP**

**Sobre-escritura de métodos y constructores**

**La palabra reservada final y static**

**Polimorfismo**

**Interfaces**

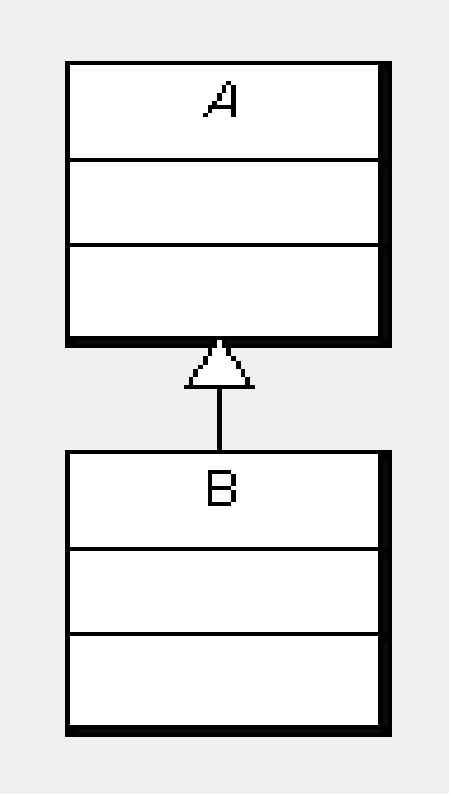
## Qué es una clase abstracta?

Es un esqueleto básico que permite especificar un tipo de entidad.

No pueden ser ***instanciadas***. No podemos usar la construcción **new**.

Sirven de molde para crear otras clases que si se podrán instanciar.

## Clase abstracta en UML



## Ejemplo de clase abstracta

```
abstract class A
{
    public $publica = 'pub';
    private $_privada = 'priv';
    protected $protegida = 'prot';

}
```

```
class B extends A
{
    public function imprimir()
    {
        echo $this->publica;
        echo $this->_privada;
        echo $this->protegida;
    }
}
```

```
$oB = new B();
$oB->imprimir();

$oA = new A(); // No instancia
```

**Salida:** pubprot

## Capítulo 5: POO y patrones en PHP

~~Repaso con un ejemplo de POO en PHP~~

~~Herencia~~

~~Clases abstractas en PHP~~

**Sobre-escritura de métodos y constructores**

**La palabra reservada final y static**

**Polimorfismo**

**Interfaces**

## Sobre-escritura de métodos y constructores

La sobre-escritura de métodos nos permite cambiar el comportamiento de un objeto.

Es posible extender la funcionalidad del método o sobre-escribirlo por completo.

Para poder ejecutar el contenido de la clase padre debemos hacerlo mediante la palabra reservada **parent**

# Ejemplo de como sobre-escribir un método en PHP

```
class ClasePadre
{
    protected function miFuncion()
    {
        echo ' Método Padre ';
    }
 }
class Extendida extends ClasePadre
{
    public function miFuncion()
    {
        parent::miFuncion();
        echo ' Método hijo';
    }
}
```

## Capítulo 5: POO y patrones en PHP

~~Repaso con un ejemplo de POO en PHP~~

~~Herencia~~

~~Clases abstractas en PHP~~

~~Sobre-escritura de métodos y constructores~~

**La palabra reservada final y static**

**Polimorfismo**

**Interfaces**

## La palabra reservada *final*

La palabra clave ***final*** aparece con la versión 5 de PHP e ***impide*** que las ***clases hijas sobrescriban un método***, antecediendo su definición con la palabra final.

Si la propia clase se define como ***final***, entonces no se podrá heredar de ella.

Las ***propiedades no pueden*** declararse como final. Sólo pueden las clases y los métodos.

## Ejemplo de uso de la palabra clave final

```
<?php
class Base
{
   public function prueba() {
      echo "llamada a Base::prueba()";
   }
   final public function otraPrueba() {
      echo "llamada a Base::otraPrueba()";
   }
}
class Hija extends Base {
   public function otraPrueba() {
      echo "llamada a Hija::otraPrueba()";
   }
}
// Devuelve: Cannot override final method Base::otraPrueba()
```

## Otro ejemplo de uso de la palabra clave final

```
<?php
final class Base
{
   public function prueba() {
      echo "llamada a Base::prueba()\n";
   }
   // Aquí no importa si definimos una función como final o no
   final public function otraPrueba() {
      echo "llamada a Base::otraPrueba()\n";
   }
}
class Hija extends Base {
}
// Devuelve un error Fatal: Class Hija may not inherit from final class (Base)
```

# La palabra reservada static

Declarar ***propiedades o métodos*** de clases como ***estáticos***, pasan a ser accesibles ***sin*** necesidad de una ***instanciación*** de la clase.

Una propiedad declarada como *static* ***no puede ser accedida*** con un ***objeto de clase instanciado*** (pero si se puede con métodos estáticos).

Debido a que los métodos estáticos se pueden invocar sin tener creada una instancia del objeto, ***$this no está disponible*** dentro de los métodos declarados como ***static***.

Las propiedades estáticas no pueden ser accedidas a través del objeto utilizando el operador flecha (**->**).

## Ejemplo de uso de la palabra clave static

```
<?php
class A
{
    public static $my_static = 'valor A';

    public function staticValue() {
        return self::$my_static;
    }
}
class B extends A
{
    public function aStatic() {
        return parent::$my_static;
    }
}
```

## Ejemplo de uso de la palabra clave static (continuación...)

```
print A::$my_static;                    // valor de A

$oA = new A();
      print $oA->staticValue();              // valor de A
      print $oA->my_static;             // Undefined "Property" my_static


      print B::$my_static;              // valor de A
      $oB = new B();
      print $oB->aStatic();                  // valor de A
```

# Otro ejemplo de uso de la palabra clave static

```
<?php

class Prueba {
    public static function unMetodoEstatico() {
        // ...
    }
}

Prueba::unMetodoEstatico();

$classname = 'Prueba';
$classname::unMetodoEstatico(); // A partir de PHP 5.3.0
```

## Capítulo 5: POO y patrones en PHP

~~Repaso con un ejemplo de POO en PHP~~

~~Herencia~~

~~Clases abstractas en PHP~~

~~Sobre-escritura de métodos y constructores~~

~~La palabra reservada final y static~~

**Polimorfismo**

**Interfaces**

## Clase 9: POO en PHP

Capacidad para que varias clases derivadas de una antecesora utilicen un mismo método de forma diferente.

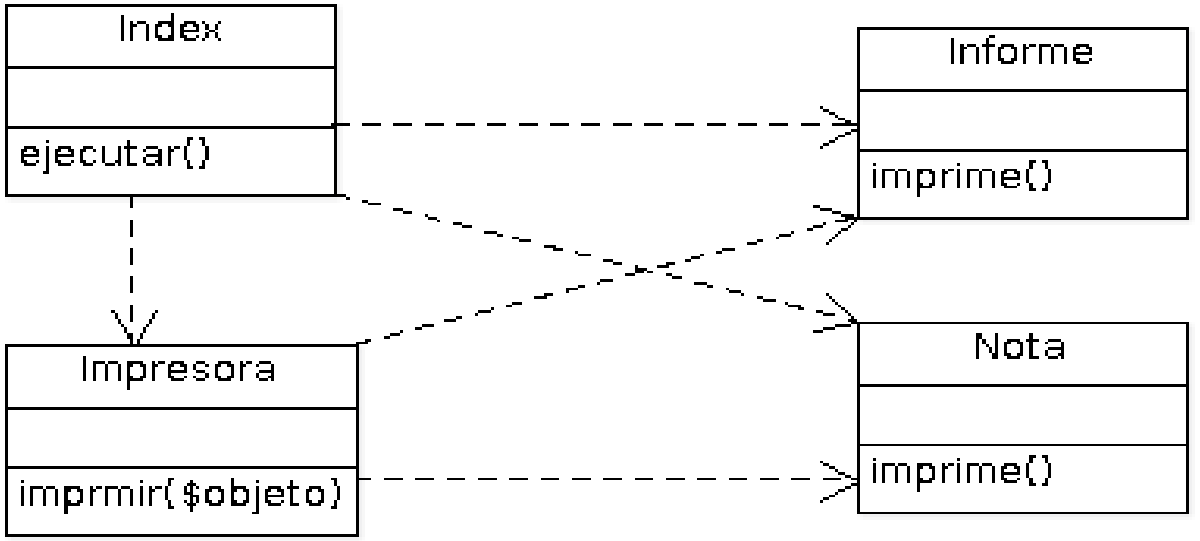
Es la capacidad de poder tomar varias clases que poseen operaciones iguales

**Polimorfismo**

Débil *(sobrecarga):* dos métodos en la misma clase se llamen igual, siempre que sus firmas sean diferentes.

*Por herencia:* implica definir métodos en una clase base y reemplazarlos con nuevas implementaciones en clases derivadas

# Polimorfismo con UML



# Ejemplo de polimorfismo en PHP

```
class Impresora
{
    public function imprimir($objeto)
    {
        echo $objeto->imprime();
    }
 }

class Informe
{
    public function imprime()
    {
        return 'Se imprime informe';
    }
}
```

```
class Nota
{
    public function imprime()
    {
        return 'Se imprime nota';
    }
}
```

## Capítulo 5: POO y patrones en PHP

~~Repaso con un ejemplo de POO en PHP~~

~~Herencia~~

~~Clases abstractas en PHP~~

~~Sobre-escritura de métodos y constructores~~

~~La palabra reservada final y static~~

~~Polimorfismo~~

**Interfaces**

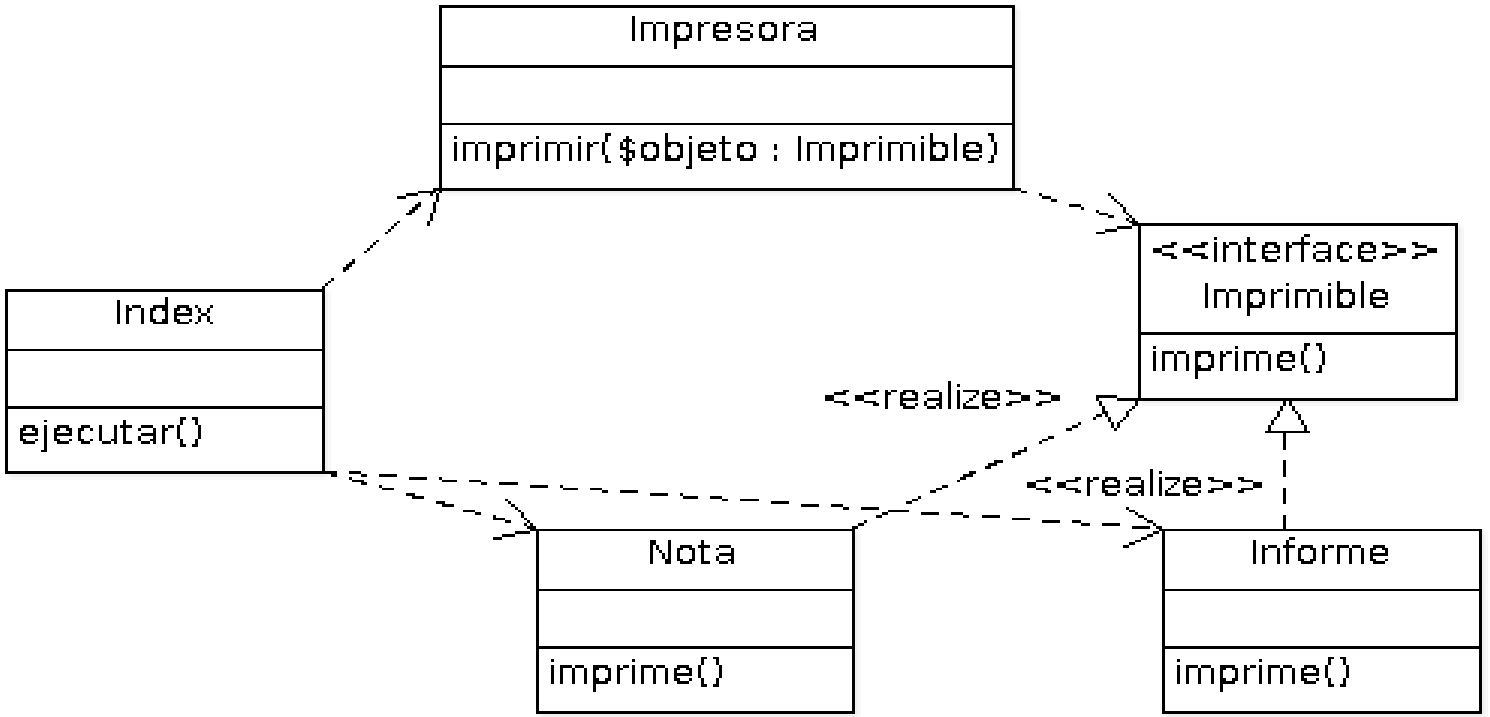
## Qué es una interfaz?

Se puede definir como *declaraciones de funcionalidades* que tienen que *cubrir las clases que implementen dicha interfaces*.

Se *definen* un juego de *métodos* que deben *codificar las clases que implementan dicha interfaz*.

Se declaran métodos sin especificar ningún código fuente asociado. Luego, las clases que implementen esa interfaz serán las encargadas de proporcionar un código a los métodos que contiene esa interfaz.

# Interfaz con UML



# Ejemplo de interfaz en PHP

ImprimibleInterfaz.php

```
interface Imprimible
{
    public function imprime();
}
```

Impresora.php

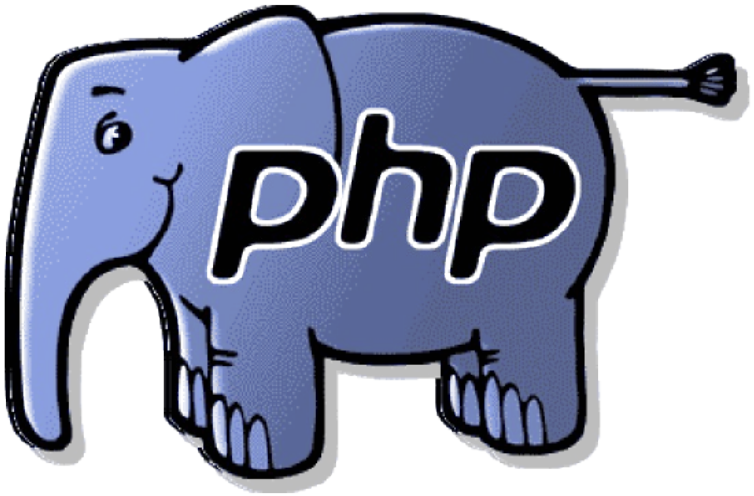
```
class Impresora
{
    public function imprimir(Imprimible $objeto)
    {
    echo $objeto->imprime();
    }
}
```

Informe.php

```
require_once 'ImprimibleInterfaz.php';

class Informe implements Imprimible
{
    public function imprime()
    {
        return 'Se imprime informe';
    }
}
```

¿Dudas?




¿Consultas?

# Información de contacto

**Web:**

http://www.gugler.com.ar

http://campusvirtual.gugler.com.ar

http://www.facebook.com/gugler.com.ar

http://www.twitter.com/cgugler

**Mail:**

contacto@gugler.com.ar

academica@gugler.com.ar

administracion@gugler.com.ar

Versión 2.1