OOP

Object Oriented Programming

Introducció

- Paradigma de programació
- La seva base són els objectes i les seves interaccions
- En PHP els objectes són instàncies d'una classe

OOP with PHP

Type Hinting (1/2)

- PHP determina el tipus de les variables dinàmicament
- Podem canviar o convertir el tipus de les variables
  - cast
  - funcions natives

Type Hinting (2/2)

Type	Description	Minimum PHP version
Class/interface name	The parameter must be an *instanceof* the given class or interface name.	PHP 5.0.0
self	The parameter must be an *instanceof* the same class as the one the method is defined on. This can only be used on class and instance methods.	PHP 5.0.0
array	The parameter must be an array.	PHP 5.1.0
callable	The parameter must be a valid callable.	PHP 5.4.0
bool	The parameter must be a boolean value.	PHP 7.0.0
float	The parameter must be a floating point number.	PHP 7.0.0
int	The parameter must be an integer.	PHP 7.0.0
string	The parameter must be a string.	PHP 7.0.0

Magic methods

- Funcions que es criden automàticament quan es produeixen certs events
- Podeu consultar el [llistat complert]

```php
class User
{
    public function __construct()
    {
    /**
     * Es crida automàticament al fer una nova instància
     * de la classe User
     */
        ...
    }
}
```

Objectes (1/2)

- Dades agrupades amb comportament associat
- Instàncies d'una classe
- Les classes defineixen:
  - propietats
  - mètodes
  - comportament

Objectes (2/2)

```php
/**
* User
*/
class User
{
    private $name;

    public function __construct(string $name)
    {
        $this->name = $name;
    }
    public function name()
    {
        return $this->name;
    }
}

$newUser = new User('Jaume');
echo $newUser->name();
```

Visibility (1/2)

- Les propietats i mètodes d'una classe poden ser:
  - private
  - protected
  - public

Visibility (2/2)

```php
/**
* Rectangle
*/
class Rectangle
{
    private $height;
    private $width;

    public function __construct($height, $width)
    {
        $this->height = $height;
        $this->width = $width;
    }

    public function calculateArea()
    {
        echo $this->height * $this->width;
    }
}

$rectangle = new Rectangle(20, 20);
$rectangle->calculateArea(); // Outputs 400
$rectangle->height; // Error
$rectangle->width; // Error
```

Inheritance (1/2)

- Una classe hereta tots els mètodes i propietats definides com *public* o *protected* de la classe pare
- La classe filla pot sobreescriure aquests mètodes i propietats

Inheritance (1/2)

```php
class A
{
    private $message;

    public function __construct(string $message)
    {
        $this->message = $message;
    }

    public function showMessage()
    {
        echo $this->message();
    }
}

/**
* B
*/
class B extends A
{
    public function showMessage()
    {
        echo "This is the message from the class B: " . $this->message;
    }
}

$classB = new B('Cool message');
$classB->showMessage(); // This is the message from the class B: Cool Message
```

Scope

- Un mètode s'executa en l'àmbit (*scope*) de l'objecte on s'ha definit
- PHP ens permet modificar el *scope* actual:
  - Paamayim Nekudotayim ::
  - self, parent i static

Classes abstractes (1/2)

- No es poden instanciar
- Si una classe té un mètode abstracta s'ha de declarar com a tal
- Les classes que heretin d'una abstracta han d'implementar tots els mètodes declarats com a tal

Classes abstractes (2/2)

```php
abstract class Test
{
    abstract protected function getValue();

    public function print() {
        echo $this->getValue() . "\n";
    }
}

class ChildTest extends Test
{
    protected function getValue() {
        return "Child";
    }
}

$test = new Test(); //Error
$childTest = new ChildTest();
echo $childTest->print(); // Prints Child
```

Interface (1/2)

- Defineixen contractes
- Les classes que les implementin han de complir el contracte
- Tots els mètodes declarats han de ser públics

Interface (2/2)

```php
interface Repository {
    public function getUser($id);
}

class MySQLRepository implements Repository {
    public function getUser($id) {
        // MySQL implementation
    }
}
```

Hand on

Exercici 13

In the next episode

- Use namespaces
- Package management with Composer