# WEB & MOBILE APPLICATIONS

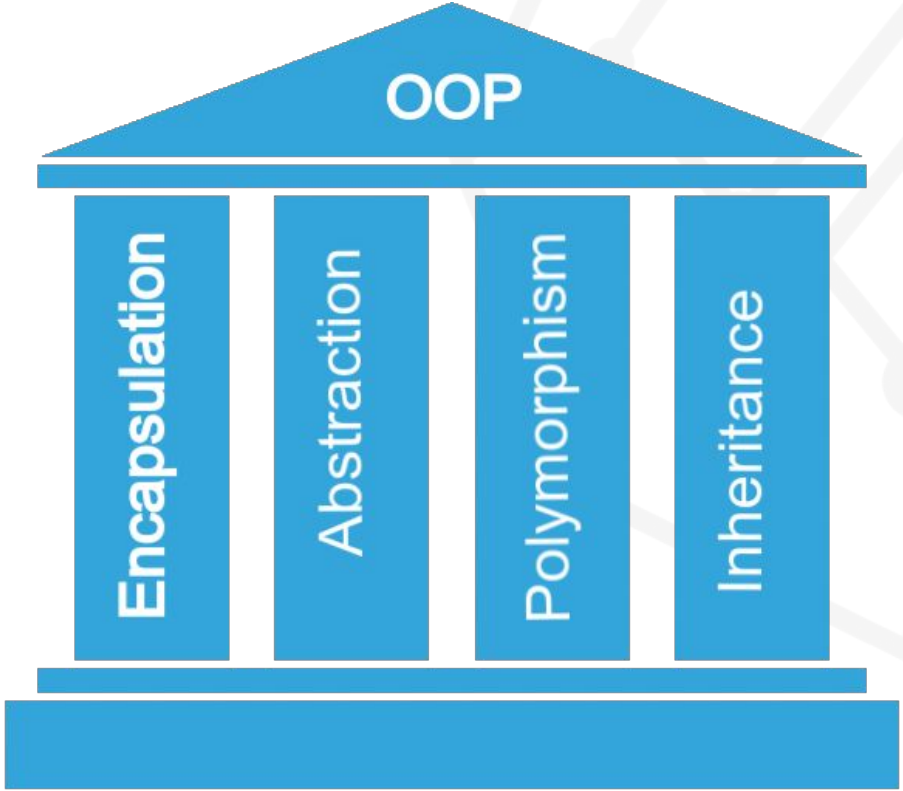
Object-Oriented Programming in PHP

# Outline

- I quattro pilastri della programmazione orientata agli oggetti
- OOP in PHP

# I pilastri dell’OOP

# I 4 pilastri dell'OOP



# Incapsulamento

- Per incapsulamento si intende il meccanismo di considerare determinate funzionalità, composte da dati e logica che agisce sui dati, all'interno di unità specifiche, ed evitare un unico blocco di codice monolitico che contiene tutte le funzionalità.
- Ad esempio, si può pensare ai componenti di un computer:
  - la CPU si occupa dell'elaborazione
  - la scheda video si occupa di quella parte di elaborazione relativa al video
  - l'alimentatore contiene l'hardware necessario ad alimentare il PC attraverso varie tensioni
  - il disco rigido si occupa di salvare le informazioni [...]
- Si incapsula quindi un funzionamento specifico in una "classe", ovvero una determinata entità appositamente progettata. Ciò che succede dentro questa classe è nascosto al mondo esterno.

# Astrazione

- L’astrazione si lega fortemente all’incapsulamento ed è la modalità con cui suddividiamo il nostro codice in varie entità, facendone emergere solo le macro funzionalità necessarie all’interazione tra le stesse
- L’astrazione è legata ad un processo mentale: semplifico la progettazione dei componenti di un determinato sistema non considerandone gli aspetti implementativi
- *L'idea alla base dell'astrazione è concentrarsi su cosa piuttosto che su come. L'incapsulamento nasconde la meccanica interna di come.*

# Ereditarietà

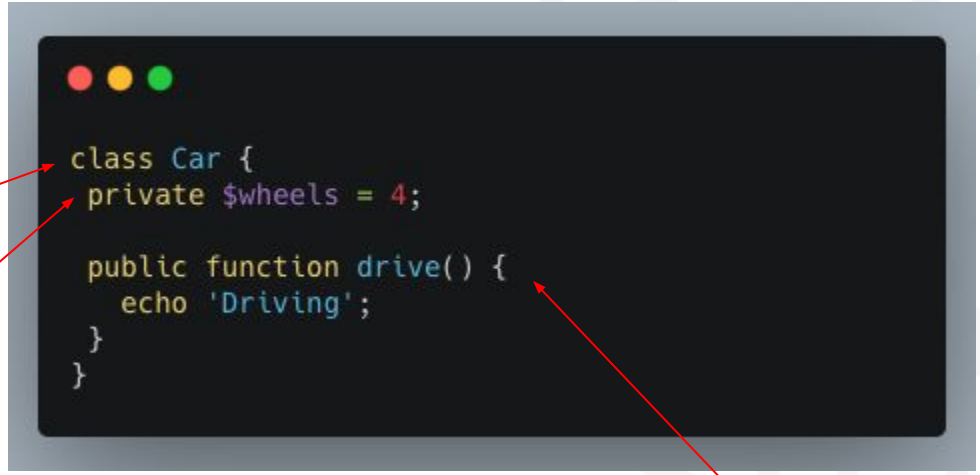
- Per ereditarietà si intende la capacità di una classe di “ereditare” variabili e funzioni (chiamate **proprietà** e **metodi**) da un’altra classe “padre”
- Per fare ciò, la classe figlia “estende” la classe padre. Sulla classe figlia sarà possibile eseguire i metodi, nonché accedere alle proprietà, della classe padre.

# Polimorfismo

- Fortemente legato all’ereditarietà, il polimorfismo è la capacità di modificare, in tutto o in parte, il funzionamento di un metodo ereditato, lasciando però inalterata la stessa **signature** del metodo padre (ovvero il nome, il numero e il tipo di parametri)

# OOP in PHP

# OOP in PHP - classi, proprietà e metodi



# OOP in PHP - ereditarietà

```php
class Car {
 private $wheels = 4;

 public function drive() {
   echo 'Driving';
 }
}

class Tesla extends Car {
 public function chargeEngine() {
  echo 'Charging engine';
 }
}
```

```php
$tesla = new Tesla();
$tesla->drive();
```

# OOP in PHP - polimorfismo

```php
class Car {
 private $wheels = 4;

 public function drive() {
   echo 'Driving with '.$this->wheels.' wheels';
 }
}

class Tesla extends Car {
 public function chargeEngine() {
  echo 'Charging engine';
 }

 public function drive() {
     parent::drive();
     echo ' - AUTOPILOT!';

 }
}
```

```php
$tesla = new Tesla();
$tesla->drive();
```

# Q & A