Object Oriented Programming

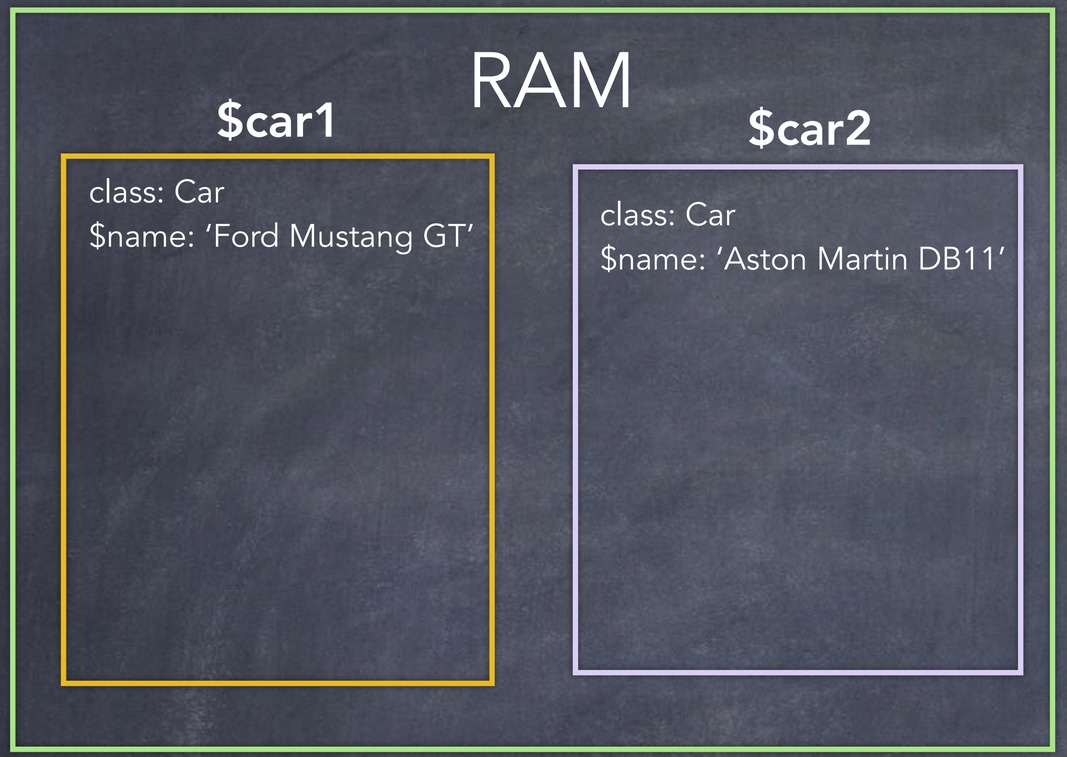
OOP Les 3

Lesdoelen voorgaande les

- Je begrijpt de lesstof van les 1 beter
- Je leert al iets meer visueel te denken en minder in code
- Je begrijpt wat $this is en doet

Lesdoelen vandaag

- Je begrijpt wat protected en public zijn en doen
- Je weet wat Encapsulation is
- Je kunt zelf een functie toevoegen aan een class
- Je weet hoe je een object vanuit je class maakt
- Je kunt functies binnen objecten aanroepen




```php
<?php
class Car
{
    protected $name;
    protected $position;

    public function setName($new_name)
    {
        $this->name = $new_name;
    }

    public function driveTo($finish)
    {
        while ($this->position < $finish) {
            $this->position++;
            echo $this->name . ' is at position ' . $this->position . "\n";
        }
    }
}

$car1 = new Car();
$car1->setName(‘Ford Mustang’);

$car2 = new Car();
$car2->setName(‘Aston Martin DB11’);
```

$car1 zit in zijn eigen beschermde “**blok**” geheugen. Je kunt aan de buitenkant alleen “**public**” attributen en functies aanroepen.

Vandaar $car1->setName(‘Ford Mustang’) ipv $car1->name = ‘Ford Mustang’.

Voor $car2 geldt hetzelfde natuurlijk

Maar waarom niet public $name?

Encapsulation

Encapsulation

Encapsulation

Maar waarom niet public $name?

Basisbegrip: “Encapsulation”

Bij OOP willen we **complexiteit verbergen.** Je moet de classes kunnen gebruiken zonder de achterliggende werking te kennen.

Waarom?

```php
$pete = new Person();
$pete->setLastName('Peterson');
echo $person->getLastName();
```

```php
class Person
{
    protected $lastName;

    public function setLastName($new_name)
    {
        if ($new_name != null && strlen($new_name) > 0) {
            $new_name = filter_var($new_name, FILTER_SANITIZE_STRING);
            $this->lastName = $new_name;
            error_log('lastName changed to ' . $new_name);
        } else {
            error_log("Invalid name");
        }
    }

    public function getLastName()
    {
        error_log(date("Y-m-d h:i:sa") . ': lastName requested, current value: ' . $this->lastName);

        return $this->lastName;
    }
}
```

Maar waarom niet public $name?

Basisbegrip: “Encapsulation”

Bij OOP willen we **complexiteit verbergen.** Je moet de classes kunnen gebruiken zonder de achterliggende werking te kennen.

Waarom? Vergelijk het met een auto: een bestuurder hoeft niet te weten hoe een motor intern werkt om de auto te besturen.

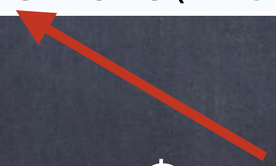
Bijvoorbeeld een front-end developer hoeft zo alleen maar de beschikbare functionaliteit te kennen, zonder de achterliggende implementatie ervan te snappen.

```php
class Robot
{
    protected $name;

    public function setName($new_name)
    {
        $this->name = $new_name;
    }
}

$robot1 = new Robot();
$robot2 = new Robot();

$robot1->setName('Robo 1');
$robot2->setName('Robo 2');
```



$name is protected, die kun je niet aanroepen.

Dus moet er een public function zijn die je wel kunt aanroepen

Opdracht

- challenges/Robot.php :-)