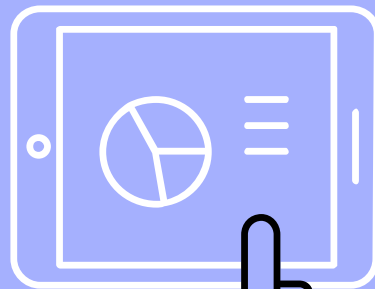
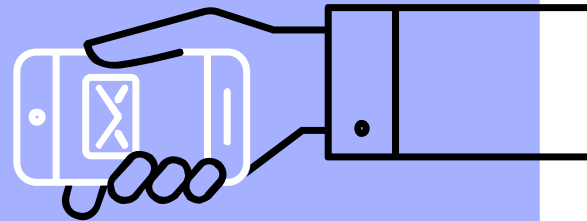
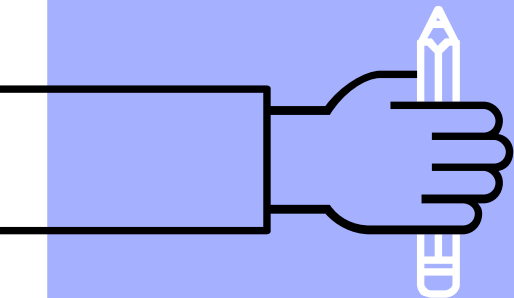
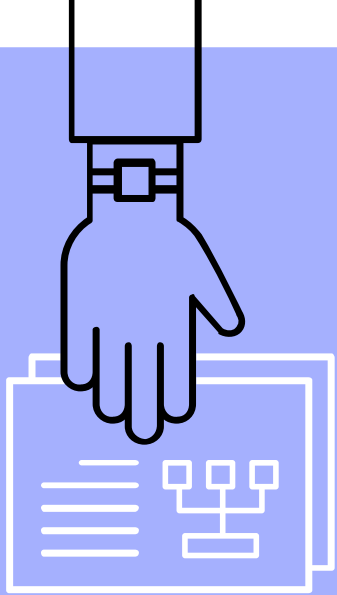
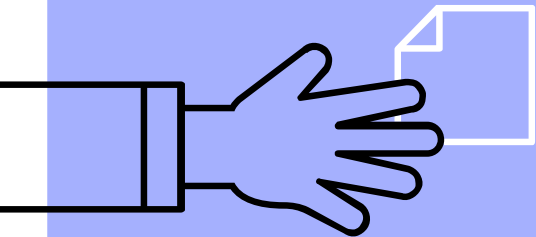
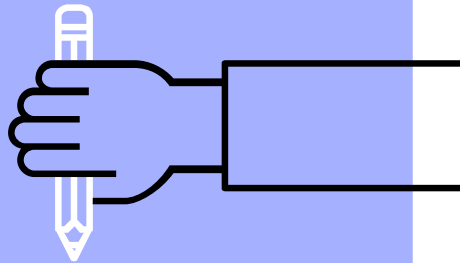


Les bases de la Programmation Orientee Objet



1. QU'EST-CE QUE LA POO ?

Petite introduction à un
principe vaste

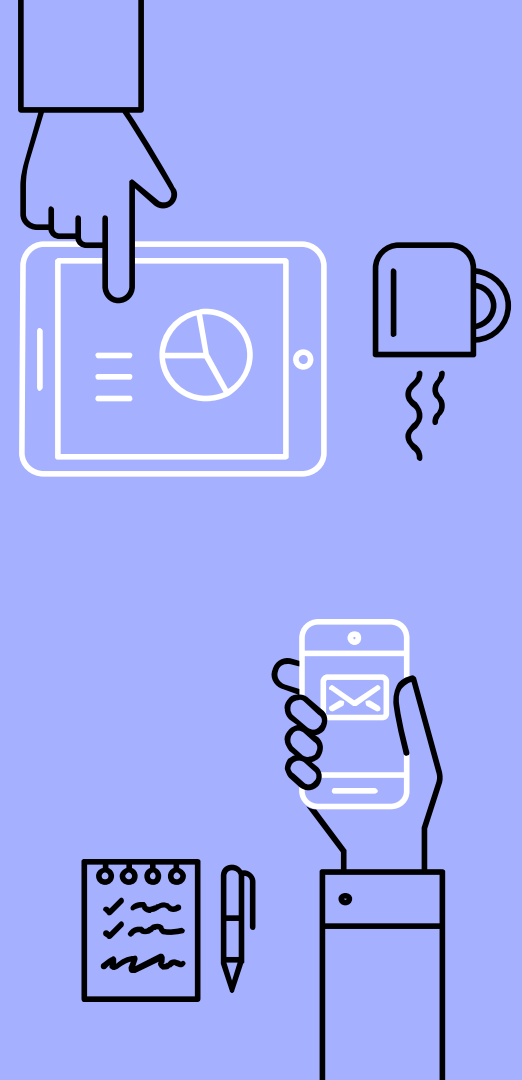


“

*La P00 est un paradigme
basé sur le concept
d'objets.*

ORIENTÉS OBJET ?

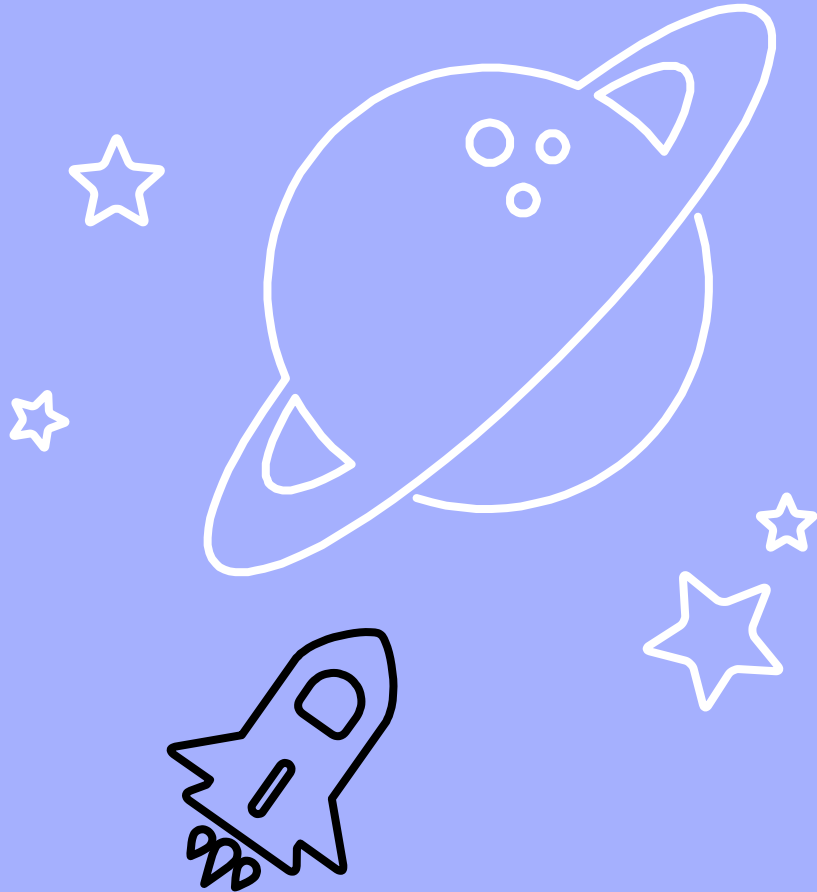
- C++
- C#
- Java
- JavaScript
- PHP
- Python
- Ruby
- Swift
- ...



OBJET

CONCEPT

Entité informatique
qui possède des
caractéristiques et
des **actions**



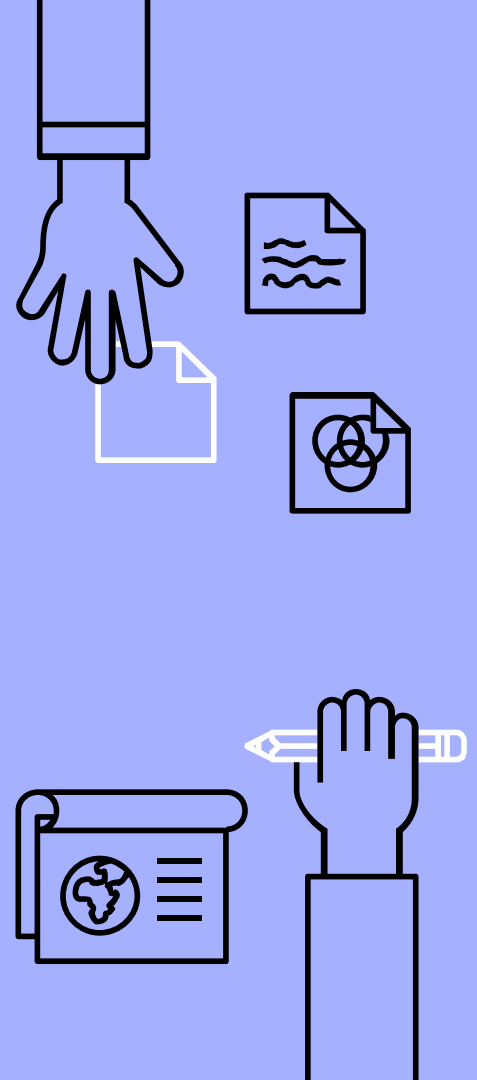
QU'EST-CE QUI COMPOSE UN OBJET ?

Attributs

- Nom
- Taille
- Poids
- Couleur
- Vitesse
- ...

Méthodes

- Walk ();
- Grow ();
- Eat ();
- Fly ();
- Wait ();
- ...



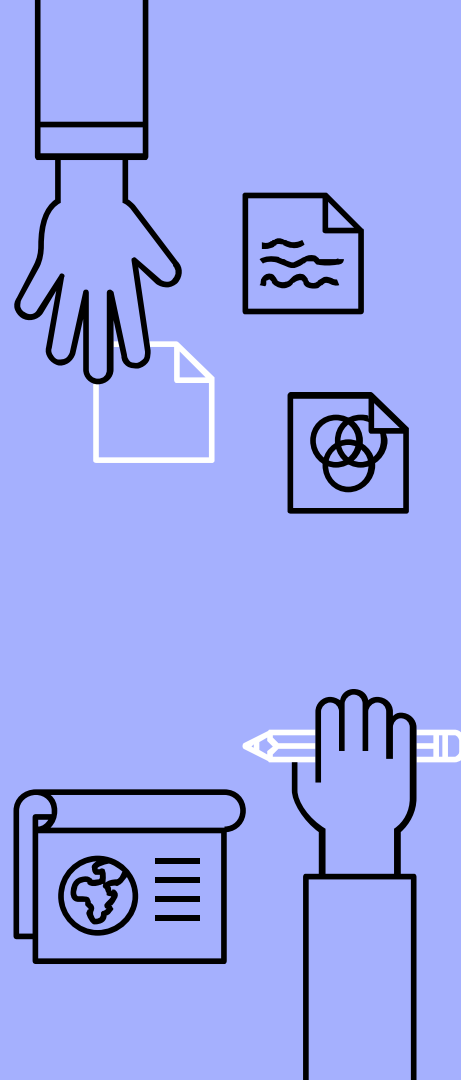
QU'EST-CE QUI COMPOSE UN OBJET ?

Attributs

Méthodes

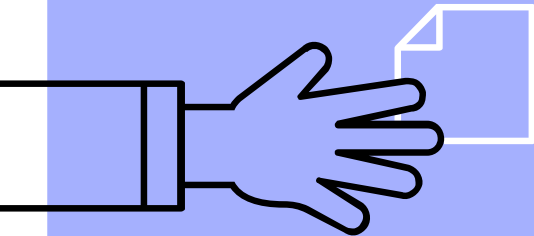
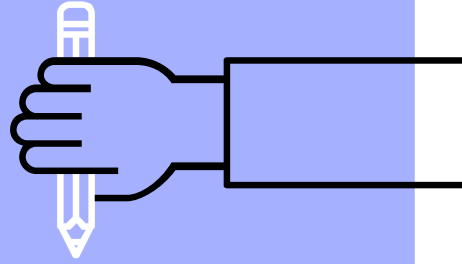
L'objet est un **excellent moyen de gérer et classer** de la donnée

On peut **instancier** autant d'objets qu'on souhaite à partir de la même **classe**.



2. LES CLASSES

Rentrons dans le vif du sujet



A QUOI SERT UNE CLASSE ?

Elles **instancient** des objets d'un certain type, comme un moule.

Composées:

- d'un constructeur
- d'attributs et de méthodes

() Dog.js



() Dog.js > ...

```
1
2  class Dog {
3      constructor() {
4          this.name = "Beethoven";
5          this.race = "St. Bernard";
6          this.speed = 2;
7          this.weigth = 105;
8      }
9
10     inspect() {
11         console.log(`This is ${this.name}, a ${this.race}`);
12     }
13
14     bark() {
15         console.log("Woof!\n");
16     }
17
18     play() {
19         this.speed *= 2;
20     }
21 }
```

A QUOI SERT UNE CLASSE ?

Elles **instancient** des objets d'un certain type, comme un moule.

Composées:

- d'un constructeur
- d'attributs et de méthodes

Dog.py ×

Dog.py > ...

```
1
2 class Dog:
3     def __init__(self):
4         self.name = "Beethoven"
5         self.race = "St. Bernard"
6         self.speed = 2
7         self.weigth = 105
8
9     def inspect(self):
10        print("This is {self.name}, a {self.race}")
11
12    def bark(self):
13        print("Woof!\n")
14
15    def play(self):
16        self.speed *= 2
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

A QUOI SERT UNE CLASSE ?

Elles **instancient** des objets d'un certain type, comme un moule.

Composées:

- d'un constructeur
- d'attributs et de méthodes

Dog.php X

Dog.php > ...

```
1  <?php
2  class Dog {
3      public $name;
4      public $race;
5      public $speed;
6      public $weight;
7
8      function __construct() {
9          $this->name = "Beethoven";
10         $this->race = "St. Bernard";
11         $this->speed = 2;
12         $this->weight = 105;
13     }
14
15     function inspect() {
16         echo("This is {self.name}, a {self.race}\n");
17     }
18
19     function bark() {
20         echo("Woof!\n");
21     }
22
23     function play() {
24         $this->speed *= 2;
25     }
26 }
```

LE CONSTRUCTEUR

Il sert à **initialiser** les **attributs** de l'objet et/ou à lui donner des valeurs.

Il est appelé lors de **l'instanciation** de l'objet.

```
() Dog.js X
() Dog.js > ...
1
2 class Dog {
3   constructor() {
4     this.name = "Beethoven";
5     this.race = "St. Bernard";
6     this.speed = 2;
7     this.weigth = 105;
8   }
9 }
10
11 var beeth = new Dog();
12
13
```

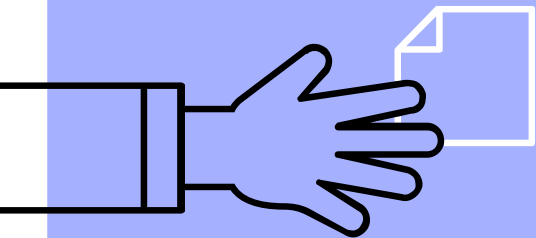
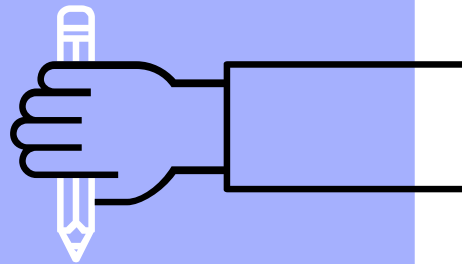
LE CONSTRUCTEUR

Il sert à **initialiser** les **attributs** de l'objet et/ou à lui donner des valeurs.

Il est appelé lors de **l'instanciation** de l'objet.

```
() Dog.js ×
() Dog.js > ...
1
2 class Dog {
3   constructor(name, race, speed, weight) {
4     this.name = name;
5     this.race = race;
6     this.speed = speed;
7     this.weight = weight;
8   }
9 }
10
11 var beeth = new Dog("Beethoven", "St. Bernard", 2, 105);
12
13
14
15
16
```

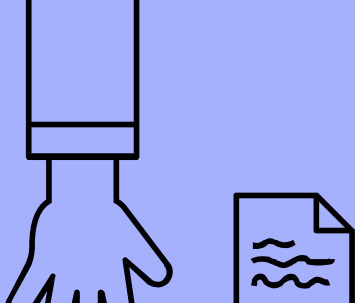
3. LES ATTRIBUTS ET LES MÉTHODES



Apprendre à modifier les
données

L'UTILISATION DES ÉLÉMENTS DE L'OBJET

Les **attributs** et les **méthodes** sont facilement utilisables à partir de **l'objet instancié**.



```
Dog.php X
Dog.php > ...
1  <?php
2  > class Dog { ...
26 }
27
28 $beeth = new Dog("Beethoven", "St. Bernard", 2, 105);
29
30 echo $beeth->name . "\n";
31 $beeth->inspect();
32 $beeth->bark();
33

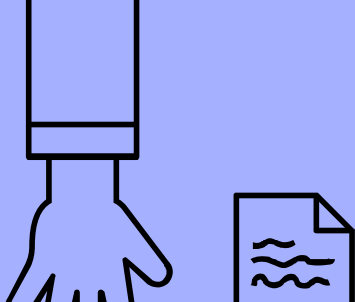
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

carra@iGilgamesh:~/Cours_Live/Objet$ php Dog.php
Beethoven
This is Beethoven, a St. Bernard
Woof!
carra@iGilgamesh:~/Cours_Live/Objet$
```

L'UTILISATION DES ÉLÉMENTS DE L'OBJET

La variable objet instanciée sert de base aux **manipulations sur ses données**.

Tout changement effectué dans cet objet est **enregistré** dans la **mémoire**.



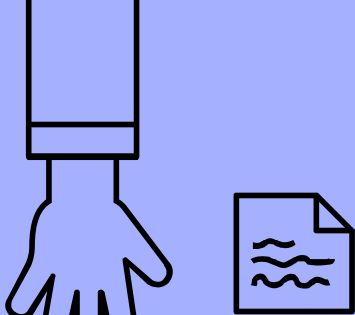
```
Dog.php X
Dog.php > ...
1  <?php
2  > class Dog { ...
26 }
27
28 $beeth = new Dog("Beethoven", "St. Bernard", 2, 105);
29
30 $beeth->inspect();
31 $beeth->name = "Mozart";
32 $beeth->inspect();
33

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

carra@iGilgamesh:~/Cours_Live/Objet$ php Dog.php
This is Beethoven, a St. Bernard
This is Mozart, a St. Bernard
carra@iGilgamesh:~/Cours_Live/Objet$
```


L'ENCAPSULATION

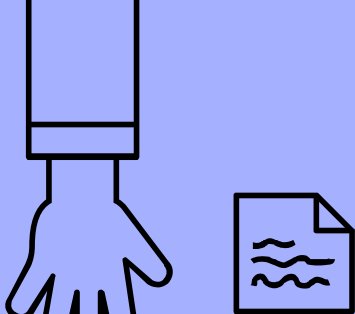
- Empêcher l'utilisateur d'accéder aux attributs directement
- **Keywords**
 - private
 - public



```
Dog.php X
Dog.php > ...
1  <?php
2  class Dog {
3      private $name;
4      private $race;
5      private $speed;
6      private $weight;
7
8      function __construct($name, $race, $speed, $weight) {
9          $this->name = $name;
10         $this->race = $race;
11         $this->speed = $speed;
12         $this->weight = $weight;
13     }
14
```

L'ENCAPSULATION

- **Cannot access private property Dog::\$name**
- Garantit l'intégrité des données contenues dans l'objet



```
Dog.php x
Dog.php > ...
1  <?php
2  > class Dog {
26 }
27
28 $beeth = new Dog("Beethoven", "St. Bernard", 2, 105);
29
30 $beeth->inspect();
31 $beeth->name = "Mozart";
32 $beeth->inspect();
33

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

carra@iGilgamesh:~/Cours_Live/Objet$ php Dog.php
This is Beethoven, a St. Bernard
PHP Fatal error:  Uncaught Error: Cannot access private property Dog::$name in /Users/carra/Cours_Live/Objet/Dog.php:32
Stack trace:
#0 {main}
  thrown in /Users/carra/Cours_Live/Objet/Dog.php on line 32

Fatal error: Uncaught Error: Cannot access private property Dog::$name in /Users/carra/Cours_Live/Objet/Dog.php:32
Stack trace:
#0 {main}
  thrown in /Users/carra/Cours_Live/Objet/Dog.php on line 32
carra@iGilgamesh:~/Cours_Live/Objet$
```

SETTERS ET GETTERS

Ce sont des fonctions d'interface.

Ils permettent d'accéder aux données d'une instance d'objet.

- **Setter** — modifie
- **Getter** — récupère

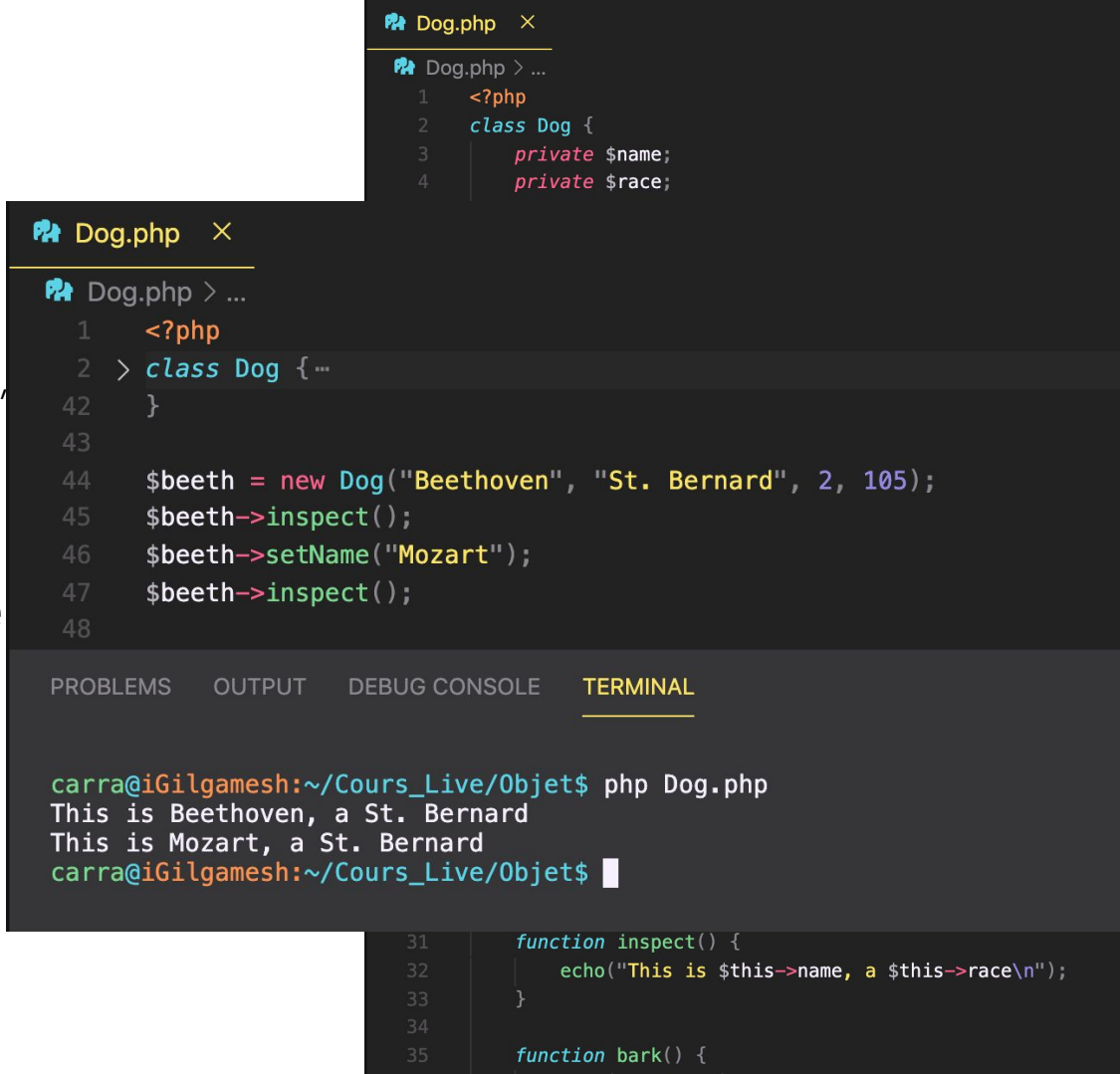
```
Dog.php X
Dog.php > ...
1  <?php
2  class Dog {
3      private $name;
4      private $race;
5      private $speed;
6      private $weight;
7
8      function __construct($name, $race, $speed, $weight) {
9          $this->name = $name;
10         $this->race = $race;
11         $this->speed = $speed;
12         $this->weight = $weight;
13     }
14
15     function getName() {
16         return $this->name;
17     }
18
19     function setName($name) {
20         $this->name = $name;
21     }
22
23     function getRace() {
24         return $this->race;
25     }
26
27     function setRace($race) {
28         $this->race = $race;
29     }
30
31     function inspect() {
32         echo("This is $this->name, a $this->race\n");
33     }
34
35     function bark() {
```

SETTERS ET GETTERS

Ce sont des fonctions d'

Ils permettent d'accéder
données d'une instance

- **Setter** – modifie
- **Getter** – récupère



```
Dog.php X
Dog.php > ...
1  <?php
2  class Dog {
3      private $name;
4      private $race;

Dog.php X
Dog.php > ...
1  <?php
2  > class Dog { ...
42  }
43
44  $beeth = new Dog("Beethoven", "St. Bernard", 2, 105);
45  $beeth->inspect();
46  $beeth->setName("Mozart");
47  $beeth->inspect();
48

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

carra@iGilgamesh:~/Cours_Live/Objet$ php Dog.php
This is Beethoven, a St. Bernard
This is Mozart, a St. Bernard
carra@iGilgamesh:~/Cours_Live/Objet$

31  function inspect() {
32      echo("This is $this->name, a $this->race\n");
33  }
34
35  function bark() {
```

ET APRÈS ?

De nombreuses autres notions

- Héritage
- Polymorphisme
- Abstraction
- Interface
- Design Patterns
- ...



La suite lors
d'un prochain
cours

