

# CLASSES - OBJETS

IMPLÉMENTATION D'UNE CLASSE EN PHP

**BTS SIO1 – ANNÉE 2022/2023 – LYCEE PERGAUD**

# CONCEPTEUR / UTILISATEUR



Je suis un  
développeur qui  
**conçoit** des classes



Je suis un  
développeur qui  
**utilise** des classes

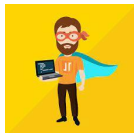
# CLASSE

## → Déclaration d'une **classe**

```
class NomClasse {  
    ...  
}
```



**NomClasse.php**



**Rectangle.php**

```
1 class Rectangle {  
2  
3 }
```

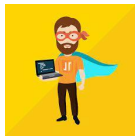
**TIP**

**1 classe = 1 fichier**

# ATTRIBUT

## → Déclaration d'un **attribut**

```
class NomClasse {  
    private type $nomAttribut;  
}
```



### ● ● ● Rectangle.php

```
1 class Rectangle {  
2     private int $longueur;  
3     private int $largeur;  
4 }
```



Les **attributs** se déclarent  
dans la **partie privée**  
de la **classe**

# MÉTHODE

## → Déclaration d'une **méthode**

```
class NomClasse {
```

```
signature    public function nomMethode([liste_param]) : [type_retour]  
              {  
                implémentation  
              }  
}
```



Les **méthodes** se déclarent  
dans la **partie publique**  
de la **classe**

# MÉTHODES

## → Exemple : calcul de la surface

● ● ● **Rectangle.php**

```
1 class Rectangle {  
2     private int $longueur;  
3     private int $largeur;  
4  
5     public function calculerSurface( ) : int {  
6  
7     }  
8 }
```



## ATTRIBUTS - MÉTHODES

- **Attributs** : variables directement accessibles dans toutes les méthodes de la classe
- Ce sont des variables globales à la classe : ils sont connus de l'ensemble des méthodes



Il n'est donc pas nécessaire de passer les **attributs** comme arguments des **méthodes**

# ATTRIBUTS - MÉTHODES

→ Accès à un **attribut** dans une méthode :  
**`$this->nomAttribut`**

● ● ● Rectangle.php

```
1 class Rectangle {  
2     private int $longueur;  
3     private int $largeur;  
4  
5     public function calculerSurface( ) : int {  
6         return $this->longueur * $this->largeur;  
7     }  
8 }
```






## CLASSE - INSTANCIATION

- **Classe** = type
- Possibilité de créer des variables de ce type
- Variables = instances = objets
- Processus de création = **Instanciation**
- Un utilisateur d'une **classe** doit donc pouvoir l'utiliser en créant des instances de la **classe**

# CONSTRUCTEUR

- L'**instanciation** d'une **classe** est réalisée par une **méthode spécifique** : le **constructeur**
- C'est une **méthode** dédiée uniquement à la création d'une **instance**

 `class NomClasse {`

`public function __construct( ? )` — *Le nom est imposé par PHP*

`{`

`}` — *Ne retourne aucune valeur !*

# CONSTRUCTEUR

## → Exemple



*Permet de créer une instance et d'initialiser les attributs de l'instance avec des valeurs passées en paramètres*



### Rectangle.php

```
1 class Rectangle {  
2     private int $longueur;  
3     private int $largeur;  
4  
5     public function __construct(int $longueur, int $largeur) {  
6         $this->longueur = $longueur;  
7         $this->largeur = $largeur;  
8     }  
9  
10    ...  
11 }
```



# INSTANCIATION

## → Création d'une instance

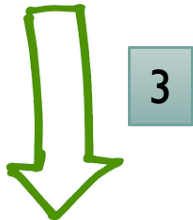
**\$nomInstance** = new **NomClasse**( ... );



référence

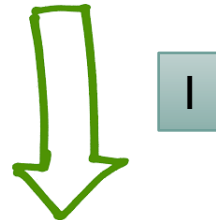
Opérateur  
d'allocation mémoire

Appel du  
constructeur



3

Permet de **référer**  
l'instance nouvellement  
créée afin de l'utiliser  
par la suite



1

Permet de **réserver** en  
mémoire un espace  
afin de stocker  
l'instance à créer



2

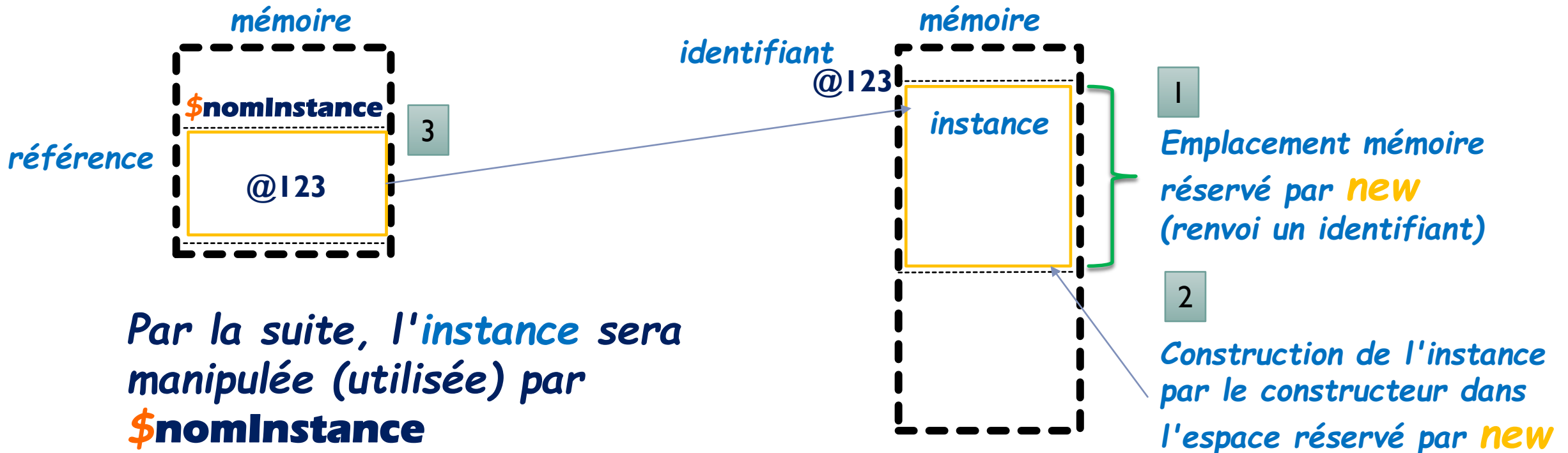
Permet de **construire** (créer  
et initialiser) une nouvelle  
instance dans l'emplacement  
mémoire réservé par new



# INSTANCIATION

## → Création d'une instance

**\$nomInstance** = new **NomClasse**( ... );



# INSTANCIATION



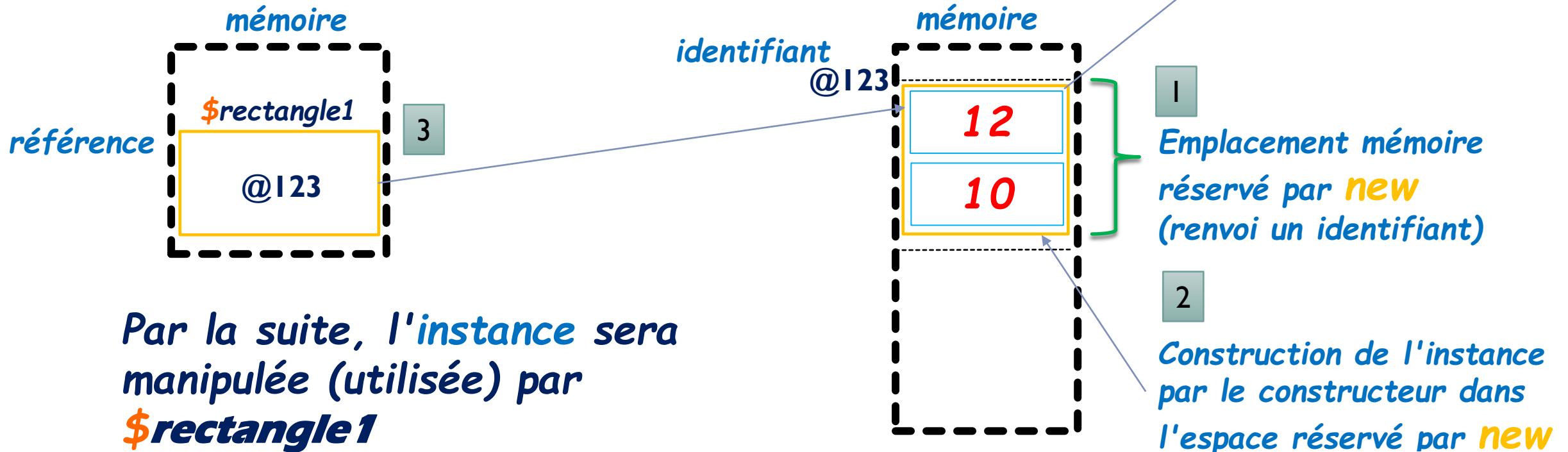
Création d'une instance de la classe **Rectangle**



**\$rectangle1** = new **Rectangle**(12,10);



instance



Par la suite, l'instance sera manipulée (utilisée) par **\$rectangle1**

# INSTANCIATION

→ Création d'une instance de la classe **Rectangle**

test.php



```
1 require_once "../classes/Rectangle.php";  
2  
3 // Instanciation de la classe Rectangle  
4 $rectangle1 = new Rectangle(12,10);
```

## IDENTIFIANT D'INSTANCE

- Une instance est accessible via une référence (variable) qui contient l'identifiant de l'instance
- Cet identifiant est unique et invariable durant la durée de vie de l'instance

```
test.php
1 require_once "../classes/Rectangle.php";
2
3 // Instanciation de la classe Rectangle
4 $rectangle1 = new Rectangle(12,10);
5 echo spl_object_hash($rectangle1);
```



## UTILISATION D'UNE INSTANCE

→ Utiliser une instance : invoquer une **méthode** de l'**interface** de la **classe** de l'instance

**\$nomInstance** -> **nomMethode**(...);



Appel de la méthode **nomMethode()**  
sur l'instance dont l'identifiant est  
stocké dans la variable **\$nomInstance**

# UTILISATION D'UNE INSTANCE

## → Exemple

● ● ● test.php

```
1 require_once "./classes/Rectangle.php";  
2  
3 // Instanciation de la classe Rectangle  
4 $rectangle1 = new Rectangle(12,10);  
5 // Invocation (appel) de la méthode calculerSurface sur l'instance $rectangle1  
6 echo $rectangle1->calculerSurface();
```



# PRINCIPE D'ENCAPSULATION

→ Accès direct à un **attribut**

● ● ● test.php

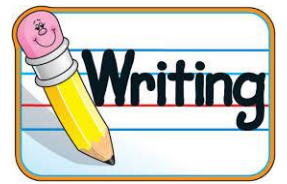
```
1 require_once "../classes/Rectangle.php";  
2  
3 // Instanciation de la classe Rectangle  
4 $rectangle1 = new Rectangle(12,10);  
5 // Invocation (appel) de la méthode calculerSurface sur l'instance $rectangle1  
6 echo $rectangle1->calculerSurface();  
7 // Accès direct à l'attribut $longueur  
8 echo $rectangle1->$longueur;
```



Respect du principe d'encapsulation !

## PRINCIPE D'ENCAPSULATION

- Accès à un **attribut** : **méthode** spécifique
- Définir dans l'**interface** de la **classe** des **méthodes** permettant d'accéder aux **attributs**
- 2 types d'accès
  - Accès en lecture → **consultation**
  - Accès en écriture → **modification**



## ACCÈS EN LECTURE

→ Méthode d'accès en lecture : **accesseur**

→ Retourner la valeur d'un **attribut**

**getter !**

```
class NomClasse {
    private type $nomAttribut;

    public function getNomAttribut() : type
    {
        return $this->nomAttribut;
    }
}
```

Diagram illustrating the getter method: An arrow points from the type in the attribute declaration to the type in the function signature, with an equals sign (=) in between. Another arrow points from the type in the function signature to a small icon of a calculator.



# ACCÈS EN LECTURE

## ●●● Rectangle.php



```
1 class Rectangle {
2     private int $longueur;
3     private int $largeur;
4
5     public function __construct(int $longueur, int $largeur) {
6         $this->longueur = $longueur;
7         $this->largeur = $largeur;
8     }
9
10    public function getLongueur(): int
11    {
12        return $this->longueur;
13    }
14
15    public function getLargeur(): int
16    {
17        return $this->largeur;
18    }
```

## ACCÈS EN ÉCRITURE

- Méthode d'accès en écriture : **mutateur**
- Modifier la valeur d'un **attribut**

**setter !**

```
class NomClasse {  
    private type $nomAttribut;  
  
    public function setNomAttribut(type $param)  
    {  
        $this->nomAttribut = $param;  
    }  
}
```



# ACCÈS EN ÉCRITURE

## ●●● Rectangle.php



```
1 class Rectangle {
2     private int $longueur;
3     private int $largeur;
4
5     public function __construct(int $longueur, int $largeur) {
6         $this->longueur = $longueur;
7         $this->largeur = $largeur;
8     }
9
10    public function setLongueur(int $longueur): void
11    {
12        $this->longueur = $longueur;
13    }
14
15    public function setLargeur(int $largeur): void
16    {
17        $this->largeur = $largeur;
18    }
```



# ATTRIBUT

➔ Pour chaque **attribut** :

- Un accesseur
- Un mutateur



C'est moi qui  
décide !

**NOT  
MANDATORY**

# ATTRIBUT

● ● ● **test.php**

```
1 require_once "./classes/Rectangle.php";  
2  
3 // Instanciation de la classe Rectangle  
4 $rectangle1 = new Rectangle(12,10);  
5 // Invocation (appel) de la méthode calculerSurface sur l'instance $rectangle1  
6 echo $rectangle1->calculerSurface();  
7 // Affichage de la longueur de $rectangle1  
8 echo $rectangle1->getLongueur();  
9 // Modification de la longueur de $rectangle1  
10 $rectangle1->setLongueur(15);  
11 // Calcul de la surface de $rectangle1  
12 echo $rectangle1->calculerSurface();
```



# TRAVAUX PRATIQUES



1



Ajouter une **méthode** dans la  
classe **Rectangle** permettant de  
calculer le périmètre d'un rectangle



Afficher le périmètre d'un rectangle

2



Ajouter un **attribut** dans la classe **Rectangle** permettant de préciser la couleur de fond d'un rectangle. Définir un **accesseur** et un **mutateur**. Modifier le **constructeur**.

Ajouter une **méthode** permettant de retourner sous la forme d'une chaîne de caractères les données d'un rectangle (longueur, largeur et couleur de fond)



Créer un rectangle avec une longueur, une largeur et une couleur de fond

Afficher la couleur de fond du rectangle

Afficher les données du rectangle