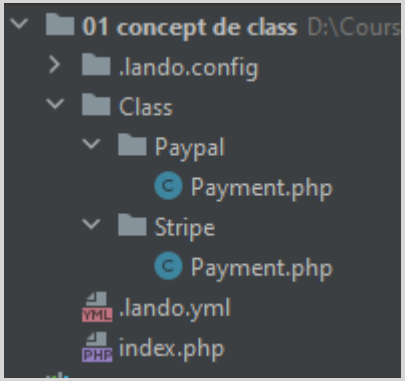
Namespace

1. Namespace? C’est quoi?

Le namespace est une espèce de chemin d’emplacement virtuel dans lequel on va venir placer nos classes. L’avantage est que 2 classes pourront avoir le même nom et ne rentrerons pas en collision lors des instanciations et ainsi éviter les conflits.

On va partir d’un nouveau projet et on va créer notre architecture de code avec 2 Classes du même nom

Architecture



Class Payment (les 2)

```php
<?php

class Payment
{

}
```

index.php

```php
<?php
require_once ('./Class/Paypal/Payment.php');
require_once ('./Class/Stripe/Payment.php');

$payment = new Payment();

var_dump($payment);
```

2. Utilisation des namespaces

Lorsqu'on crée une classe on peut le situé dans son namespace, la norme veut qu'on utilise son chemin absolu comme namespace séparé par des antislash (« \ »). S'il n'y a pas de namespace, on dit qu'il est sur le namespace global. Pour nos classes ca va donner:

Class Payment dans Stripe

```php
<?php

namespace Class\Paypal;

class Payment
{

}
```

Class Payment dans Paypal

```php
<?php

namespace Class\Stripe;

class Payment
{

}
```

On peut instancier nos 2 classes

```php
<?php
require_once ('./Class/Paypal/Payment.php');
require_once ('./Class/Stripe/Payment.php');

$paymentPaypal = new Class\Paypal\Payment();
$paymentStripe = new Class\Stripe\Payment();

var_dump($paymentPaypal , $paymentStripe);
```

```
/app/index.php:8:
object(Class\Paypal\Payment)[1]

/app/index.php:8:
object(Class\Stripe\Payment)[2]
```

On voit qu'à chaque instanciation d'une classe (new Class) on réécrit tout le namespace, on va donc utiliser le « use » pour indiquer ou se trouve la Class et pouvoir l'instancier juste avec son nom

```php
<?php
require_once ('./Class/Paypal/Payment.php');
require_once ('./Class/Stripe/Payment.php');

use \Class\Paypal\Payment;
use \Class\Stripe\Payment;

$paymentPaypal = new Payment();
$paymentStripe = new Payment();

var_dump($paymentPaypal , $paymentStripe);
```

On voit qu'on se retrouve comme avant avec un conflit car il dit que je n'ai pas le droit d'utiliser Payment car il est déjà utilisé.
Pour remédier à ca on peut (et on doit) utiliser un alias avec « as »

```php
<?php
require_once ('./Class/Paypal/Payment.php');
require_once ('./Class/Stripe/Payment.php');

use \Class\Paypal\Payment as PaypalPayment;
use \Class\Stripe\Payment;

$paymentPaypal = new PaypalPayment();
$paymentStripe = new Payment();

var_dump($paymentPaypal , $paymentStripe);
```

On créer une nouvelle Class User qui sera dans un dossier Users mais on ne va pas lui mettre de namespace. Cette nouvelle classe aura une fonction construct qui génère un id aléatoire.

```php
class User
{
    private int $id;

    public function __construct()
    {
        $this->id = rand();
    }
}
```

On va maintenant essayer d'instancier cette classe dans un constructeur d'une class Payment de Paypal

```php
namespace Class\Paypal;

class Payment
{
    public function __construct()
    {
        var_dump(new User());
    }
}
```

On a une erreur car comme on a pas mis de namespace à User il se retrouve dans le namespace globale.
En l'instanciant dans Paypal\Payment, il s'attend à le trouver :dans Paypal\Payment\User

On doit donc spécifier que User se trouve dans le namespace global. On a 2 possibilités, soit on fait:

- New \User() (on met un antislash devant User)
- Use User (on utilise use et on met juste User car pas de namespace)

Et la ca fonctionne require_once ('./Class/Users/User.php') dans index.php
=> On va indiquer le namespace à User (namespace Class\Users)

Namespace

```php
namespace Class\Users;

class User
{
```

on « use »

```php
namespace Class\Paypal;

use \Class\Users\User;

class Payment
{
    public function __construct()
    {
        var_dump(new User());
    }
}
```

Maintenant si je dois utiliser une classe native de PHP (ex: DateTime) comment je dois l'instancier?

Exercice: dans le var_dump du constructeur de Paypal\Payment remplacer User par la classe native PHP DateTime

Réponse:

```php
namespace Class\Paypal;

use DateTime;

class Payment
{
    public function __construct()
    {
        var_dump(new DateTime());
    }
}
```

OU

```php
<?php

namespace Class\Paypal;

class Payment
{
    public function __construct()
    {
        var_dump(new \DateTime());
    }
}
```

Bon a savoir: Si vous utiliser PhpStorm, il sait très bien importer les namespaces
Si vous utiliser Visual Code téléchargé l'extension « PHP namespace resolver »
En faisant clic droit vous aurez l'option « importer class »