OO做了名詞上的改變如：

Function=>Method將呼叫

Call=>Invoke

解釋

(類別 class) =>藍圖

(物件 object) => 實體

\_\_CLASS\_\_ => 魔術常數

unset() => 摧毀物件

Abstract Class => 當類別（Class）之間有共同的方法(function)或屬性（attribute）時，可以將這些共用的地方寫成「抽象類別（Abstract Class）」，讓其他的「子類別（Class）」去繼承

Abstract =>類似實體跟Interface 的結合

abstract class不能直接new 出來只要class 裡面有 abstract function，就必須被宣告為 abstract class繼承(extends)時，子類必須定義父類中的所有abstract function子類所繼承的function所需引用的參數只能增加(可選參數)，不能減少

Interface => 當多個類別(Class)之間有共同的方法（function），但方法實做的方式有差異，可以將這些共用「方法」寫成「介面（Interface）」，讓其他的「子類別（Class）」去實做這個介面

Interface => 類似一個模板

規範子類／子介面一定要具備某些東西，包含變數與function

可以被多重 繼承(extends)，子介面會擁有父母們的所有規範。但父母們之間的function不能重名。

interface裡只能有public function，且都是空的。

可以有 constant ，但不能被override

implement 的時候，必須 override所有的fucntion

不能直接 new 出來，要先有 class 去implement 它.

trait =>

PHP5.4 才出現

用來減少重複程式碼的狀況產生，通常是許多不同的 class 間 共通的某些 function，便於直接使用而不用考慮繼承問題。例如 "人" 跟 "狗" 都可以 "叫"，但兩者沒有繼承關係

使用上，直接在 class 裡 use 即可

class 可以Override跟Overload

Abstract 抽象方法跟實體方法都可以Override跟Overload

interface 引入多個子interface的method可以同名，但傳入設置必須一模一樣才能

Override，且interface 可以互相繼承

trait method 可以Override但property不行，引入多個子trait的method不可以同名，同名必須用use指定執行哪個或更名

use Trait1,Trait2,Trait3{

Trait1::say insteadof Trait2,Trait3;

Trait2::run insteadof Trait1,Trait3;

Trait3::run as trait3run;

Trait1::say as trait1say;

Trait1::say as private trait1say\_;

}

類別（Class） 抽象類別（Abstract Class） 介面（Interface） 特徵（Trait）

宣告屬性（attribute） ✓ ✓ ✖ ✓

常數（const） ✓ ✓ ✓ ✖

實例化（new class） ✓ ✖ ✖ ✖

抽象方法（abstract function） ✖ ✓ ✓ ✓

實作方法內容（functoin()） ✓ ✓ ✖ ✓

類別是否可繼承多個 ✖ ✖ ✓ ✓

覆寫(override) ✓ ✓ ✓ ✓(property不行)

branch命名格式：

feature/issue單號或功能編號

bug/issue單號或功能編號

分支命名不可有中文字，且一律採用小寫英文命名

程式撰寫規則

使用底線連接英文單字 => $i\_am\_a\_property

不可有表示型態的前綴字 => $str\_user\_name

函式名稱使用小駝峰的命名方式且括弧必須於特定位置

function doSomething()

{

....

}

函式盡量以動詞開頭

運算元(+ - \* / =)前後都必須有空格

換行運算元應為最後一個字元且第二行要縮排

public $i\_am\_a\_property =

'Hello World';

常數名稱全大寫，並以底線分隔=> const APPLICATION\_NAME

盡量避免縮寫，常見除外

一般檔案備註

/\*\*

\* 訂單管理　Controller

\* php version 7.3

\*

\* @category Controller

\* @package App\Http\Controllers

\* @author Cyril <cyril\_wang@company.com>

\* @copyrignt 2019 My Company

\* @version GIT: @1.0.0@

\*/

api檔案備註

/\*\*

\* Log紀錄處理 API

\*

\* @category Helper

\* @package App\Helpers

\* @author Cyril <cyril\_wang@company.com>

\* @copyrignt 2019 My Company

\* @version GIT: @1.0.0@

\* @api

\*/

類別備註

/\*\*

\* 類別說明主旨

\*

\* 詳細說明: <description>

\*

\* @category Helper

\* @package App\Helpers

\* @author Cyril <cyril\_wang@company.com>

\*/

屬性與變數備註

/\*\*

\* @var int 產生圖片寬

\*/

private static $image\_width = 120;

常數備註

/\*\*

\* 驗證碼產生字元組合(純數字)

\*/

const TYPE\_NUMBER = '0123456789';

methods與functions註解

/\*\*

\* 審核霸王金贈點

\*

\* @param int $dispense\_id 贈點紀錄 ID

\* @param string $ip 後台管理者連線 IP 位址

\* @return bool 審核處理結果

\* @throws OverlordPointAccountInsufficientBalanceException

\*/

public function approveDispense(int $dispense\_id, string $ip)

{

...

}

class註解

/\*\*

\* 取得訂單

\*

\* @return object class:Order 訂單 Model

\*/

private static function getLatestOrder()

{

return Order::orderBy('order\_date', 'desc')->limit(1)->get();

}