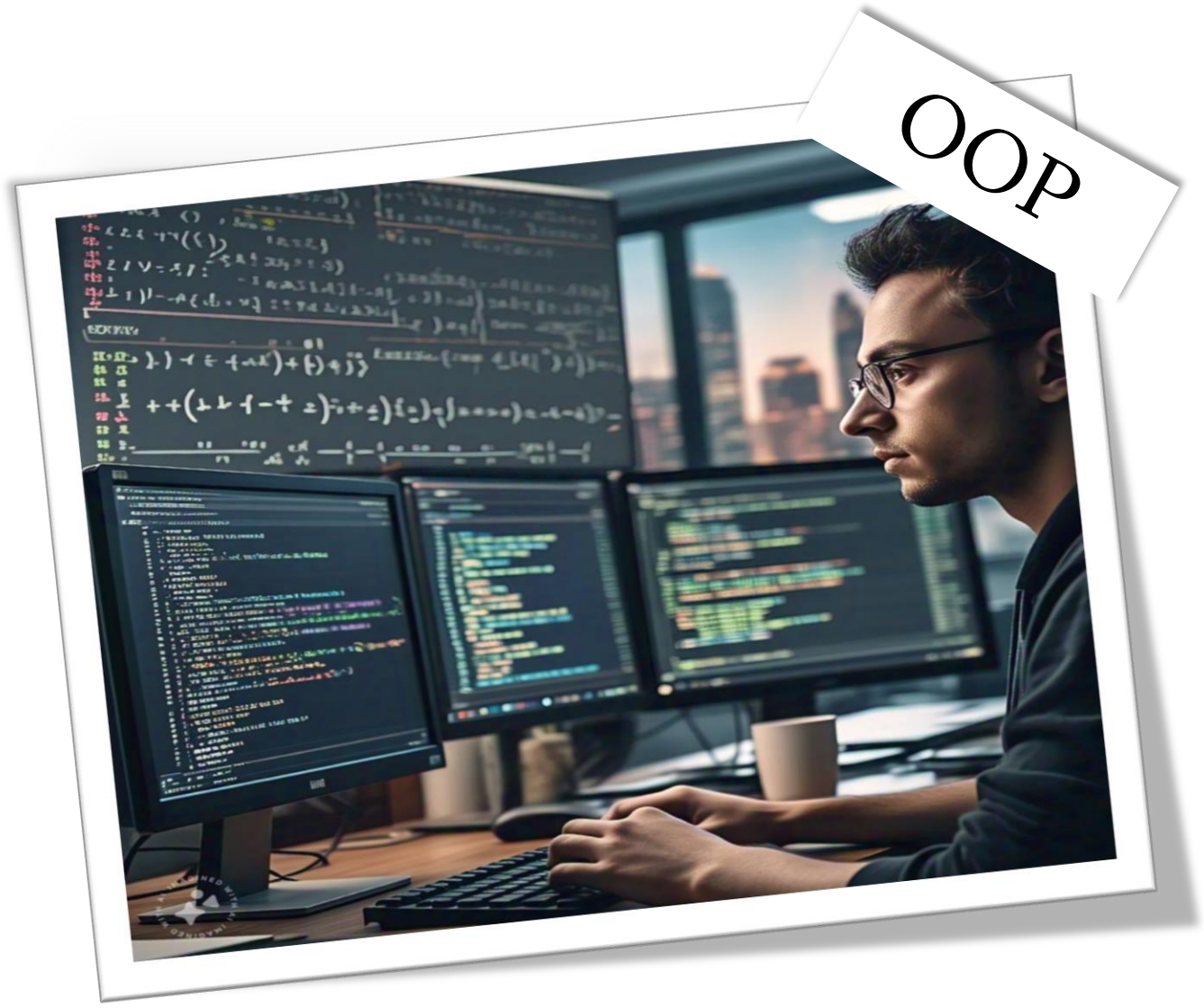
الجمهورية اليمنية

وزاره التعليم العالي

كليه العلوم والهندسة



إشراف الدكتور/إبراهيم محمد الشامي

الطالبة/شيماء فاروق المقطري

المقدمة

البرمجة الكائنية (OOP-Object-Oriented Programming) هي نمط برمجي يعتمد على مفهوم الكائنات (Object) حيث تعتبر الكائنات هي الوحدة الأساسية في تصميم البرامج يتيح هذا النمط للمطورين تنظيم الكود بشكل افضل مما يسهل عملية الصيانة والتوسع.

تستخدم OOP في العديد من لغات البرمجة بما في ذلك PHP مما يجعلها واحده من أساليب البرمجة الأكثر شيوعاً .

تاريخ ظهور الOOP في PHP:

بدا ظهور ال OOP من الإصدار ٥ الذي تم اطلاقه في يوليو ٢٠٠٤حيث سأستعرض تفاصيل تاريخيه موجزه عن كيفيه ظهوره ...

الإصدار (PHP4): لم يكن يدعم OOP بشكل كامل ،كانت هناك بعض الميزات الأساسية لمن لم تكن هناك كائنات حقيقيه كما هو الحال في لغات البرمجة الكائنية الأخرى .

الإصدار (PHP5): قدم دعماً كاملاً بما في ذلك :

- الفئات (Classes) والكائنات (Objects).
- الوراثة (Inheritance).
- التجريد (Abstraction).
- التعددية الشكلية (Polymorphism).
- التغليف (Encapsulation).
- الواجهات (Interfaces) والفئات المجردة (Abstract Classes).

حيث قدمت OOP في PHP5 تحسينات كثيره في كيفيه كتابه الكود وإدارة التطبيقات الكبيرة مما يساعد المطورين على بناء تطبيقات اكثر تعقيداً وقابليه للصيانة .

****

وظائف واهميه OOPفي PHP:

١. إعادة استخدام الكود:
يعزز إعادة استخدام الكود من خلال الوراثة والتغليف مما يسهل تطوير التطبيقات الكبيرة والمعقدة
٢. تحسين الصيانة:

يوفر OOP هيكله جيده تجعل من السهل تعديل الكود وإصلاح الأخطاء دون تأثير على أجزاء أخرى من البرنامج .

٣. تسهيل الفهم :
يسمح OOP بتمثيل الأشياء في العالم الحقيقي مما يجعل الكود اكثر وضوحاً وسهوله في الفهم .
٤. تسهيل التعاون :

مع OOP يمكن للفرق العمل على أجزاء مختلفة من التطبيق بشكل متوازي مما يعزز التعاون بين المطورين .

٥. تعزيز الأمان :

من خلال استخدام التغليف يمكن حمايه البيانات الحساسة ومنع الوصول غير المصرح به .

ومن خلال هذه الأهيم يتضح لنا ان الOPP هو نمط برمجي قوي يساعد المطورين في PHP بناء تطبيقات مرنه وسهله الصيانة، من خلال فهم استخدام المفاهيم الأساسية مثل الكائنات ،الفئات ،الوراثه،والتعدديه الشكليه،يمكن تطوير تطبيقات قويه ومعقده بشكل اكبر ولكنه يعد جزءاً ،PHPلا يقتصر فقط على OOP كفاءه اساسياً من العديد من لغات البرمجة الحديثة ما يجعله مهاره لاي مطور.

****

لكتابه كود OOP جيد في PHP تتطلب اتباع مجموعه من الممارسات الجيدة التي تعزز من جوده الكود وسهوله الصيانة والمرونة ،

اليك بعض افضل الممارسات :

❖ **استخدام التسميه الواضحة :**

تسميه الفئات والطرق :يجب ان تعكس أسماء الفئات والطرق وظيفتها بوضوح واستخدام

❖ **تطبيق مبادى SOLID:**

- **S: single Responsibility Principle:**

يجب ان تكون كل فئه مسؤوله عن مهمه واحده

- **O:open /closed Principle:**

يجب ان تكون الفئات مفتوحه للإضافة لكن مغلقه للتعديل

- **L:Liskov Substitution Principle:**

يجب ان تكون الفئات الفرعية قابله للاستبدال مع الفئات الأصلية

- **I:Interface segregation Principle:**

يجب ان يتم تقسيم الواجهات الكبيرة الى واجهات صغيره

- **D:dependency inversion principle:**

يجب ان تعتمد الفئات على الواجهات وليس على الفئات المحددة

❖ **استخدام التجريد :**

استخدام الفئات المجردة والواجهات لتعريف السلوكيات المشتركة بين الفئات المختلفة .

❖ **التغليف:**

اجعل الخصائص خاصه او محميه واستخدم طرقاً عامه للوصول اليها يساعد ذلك في حمايه البيانات .

❖ **تجنب التكرار :**

حاول تجنب تكرار الكود من خلال استخدام الفئات والطرق .

❖ **استخدم الوراثة بحذر :**

اعتمد الوراثة فقط عندما يكون ذلك منطقياً واستخدم التركيب(composition) كبديل عندما يكون ذلك مناسباً .

❖ **كتابه اختبارات وحدات :**

اكتب اختبارات وحدات (Unit Tests) لتحقق من صحه الكود وسلوكه ،يساعد ذلك في ضمان عدم كسر الكود عند اجراء تغييرات .

❖ **توثيق الكود:**

استخدم التعليقات والتوثيق المناسب لشرح الكود خاصه للطرق المعقدة او غير الواضحة .

❖ **استخدام الفئات المجردة والواجهات بشكل مناسب :**

استخدم الفئات المجردة والواجهات لتعزيز التعددية الشكلية وتوفير هيكل مشترك للفئات .

❖ **تبسيط الكود:**

حاول الحفاظ على كود بسيطاً وواضحاً ،تجنب التعقيد غير الضروري.

ان اتباع هذه الممارسات يساعد في كتابه الكود oop جيد في PHP يعزز من جوده وقابليه صيانته من خلال التركيز على التنظيم التجريد والتوثيق يمكنك بناء تطبيقات قويه ومرنه تسهل على المطورين الاخرين فهمها وتعديلها .

****

بعد ان تعرفنا على تاريخ الOOP وكذلك الأهمية والفائدة منه وماهي الممارسات التي يجب الاعتماد عليها الان سوف نتعرف على المفاهيم الأساسية لل OOP بشي من التفصيل :

المفاهيم السياسية OOP في PHP:

١) الكائنات (Objects):

الكائن هو وحده تمثل شيئاً في العالم الحقيقي ،يحتوي على بيانات (خصائص)وسلوكيات (طرق).

مثال:

```php
<?php

Class Flower{
    public $color;
    public $type;
    public function __construct($color,$type){
        $this->color=$color;
        $this->type=$type;

    }
}
$rose=new Flower();
```

٢) الفئات (classes):

الفئه هي قالب يستخدم لأنشاء كائنات، تحدد الفئه الخصائص والطرق التي ستملكها الكائنات التئ تُنشأ منها

مثال:

```php
<?php

Class Flower{
    public $color;
    public $type;
    public function __construct($color,$type){
        $this->color=$color;
        $this->type=$type;

    }
}
$rose=new Flower();
```

٣) الوراثة (Inheritance):

تسمح بإنشاء فئه جديده تعتمد على فئه موجوده مما يعزز إعادة استخدام الكود .

مثال:

```php
Class Flower{
    public $color;
    public $type;
    public function __construct($color,$type){
        $this->color=$color;
        $this->type=$type;

    }
}
Class jury extends Flower{
public $price;
}
```

٤) التعددية الشكلية (Polymorphism):

تتيح استخدام كائنات مختلفة بنفس الطريقة ،جيت يمكن ان تتصرف الكائنات بشكل مختلف بناء على نوعها.

مثال:

```
Class Flower{
    public $color;
    public $type;
    public function Type (){
        return "The all FLOWER";
    }
}
Class jury extends Flower{
public $price;
public function Type (){
    return "The JURY";
}
}
```

٥) التجريد(Abstraction):

يسمح لك بتعريف واجهات (interfaces) اوفئات مجرده (Abstract Classes) تحتوي على طرق بدون تنفيذ .

مثال:

```
Class Flower{
    abstract public function price();
}
Class jury extends Flower{
    public function price(){

return"the price is ";

    }
}
```

٦) الواجهات (Interfaces):

تحدد مجموعه من الطرق التي يجب ان تنفذها الفئات ،لا تحتوي الواجهات على تنفيذ للطرق .

مثال:

```
15
16  interface sunflower{
17  public function start();
18
19  }
20
21  class jury implements sunflower{
22
23      public function start(){
24  return"the nice flower";
25
26      }
27
28  }
```

٧) التغليف (Encapsulation):

يعني حمايه البيانات داخل الكائن مما يجعل الخصائص خاصه (private) او محميه (protected) ولا يمكن الوصول اليها مباشره من خارج الكائن .

مثال :

```
Class Flower{
    private $price;
    public function price($amount){
        $this->price+=$amount;
    }
}
```

٨) الخصائص (properties):

الخصائص هي المتغيرات التي تستخدم لتخزين البيانات المتعلقة بالكائن يمكن ان تكون الخصائص عامه (public) او خاصه (private) او محميه (protected)

مثال:

```
}
$rose= new jury();
$rose->start();
```

٩) الطرق(Methods):

الطرق هي الدوال الموجودة داخل الفئه وتحدد السلوكيات المرتبطة بالكائن يمكن ان تأخذ طرق معلمات وتعيد قيماً.

مثال:

```
    public function price(){

return"the price is ";

    }
}
```