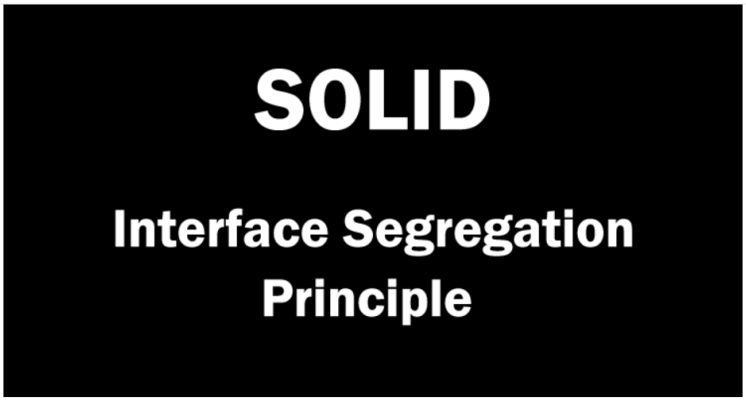


Interface Segregation Principle in PHP (in Arabic)

Published on August 31, 2021

Osama Mustafa
PHP Back-End Developer
4 articles ✓ Following

مبدأ Interface Segregation هو المبدأ رقم (4) من خمس مبادئ مشهورة في الـ Object Oriented Design و هما Principles

1. Single responsibility principle
2. Open/closed principle
3. Liskov substitution principle
4. Interface segregation principle
5. Dependency inversion principle

دي المبادئ اللي اتشرحت قبل كده

- Open Closed Principle | **https://bit.ly/3EbMnpC**
- Liskov Substitution Principle | **https://bit.ly/2XeCebf**

كلمة segregation معناها بالعربي "فصل"، بفصل كذا حاجة عن بعض، المهم دلوقتي عندنا interface اسمه Worker ، الموظف اللي عندنا في الشركة في الاغلب هيكون بيكتب كود، وبياخد بريك، وبيحضر اجتماعات، وبيتعامل مع العملاء، وبيقبض مرتبه في آخر الشهر ولحد هنا مفيش مشكلة ... فعملنا interface وحطينا فيه الـ methods زي ما حضراتكم شايفين في الصورة

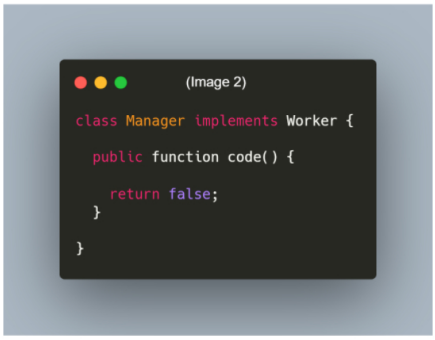
(Image 1)

```php
interface Worker {
  public function takeBreak()
  public function code()
  public function callToClient()
  public function attendMeetings()
  public function getPaid()
}
```

دلوقتي أنا هعمل Class اسمه Manager ، هتقولي وايه المشكلة؟ المدير بيحضر اجتماعات وبياخد بريك وبيقبض

مرتبه في آخر الشهر وبيتعامل مع العملاء ... بس مش بيكتب كود بايده!! ده راجل مسئول عن البيزنس وملوش دعوة بالأكواد

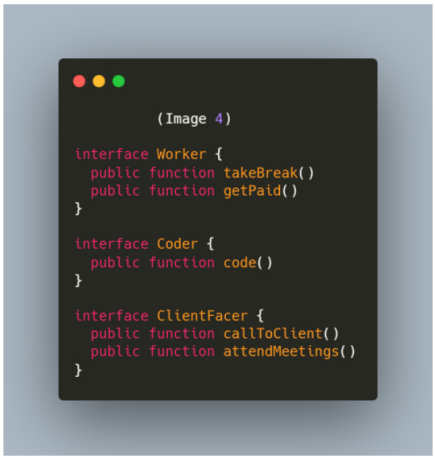
فهتضطر تروح جوه Class الـ Manager وتعمل implement للميثود اللي اسمها ()code في حين إن أنت مش محتاجها أصلاً!



فأنت كبرت دماغك وقلت عادي هنمشي الموضوع وخلاص، وحبيت تعمل Class تاني اسمه Developer ، الراجل الديفلوبر بياخد بريك وبيكتب كود، لكن ملوش تعامل مباشر مع العملاء ولا بيحضر معاهم الاجتماعات، هوا بياخد تعليماته من المدير وخلاص، فأنت هتضطر تعمل implement للميثود ()callToClient في حين إن أنت برضه مش محتاجها ونفس الفكرة مع ميثود ()attendMettings

بدأت تلاحظ المشكلة بالظبط وتفهم إن أنا بقى عندي interface مليان methods ، ومُضطر كل ما اجي اعمل Class جديدة ، اروح اعمل implementation لميثود ملهاش أي لازمة و هتتضاف تحصيل حاصل

حل المشكلة إني اعمل split أو فصل للـ interface بتاعي لـ interfaces أصغر، وكل واحد منهم بيؤدي functionality أو وظيفة محددة ، فبدل ما يبقى عندي interface واحد بس، لا أنا هخليهم تلاتة وليكن مثلاً Coder، Worker، ClientFacer



كده المشكلة اتحلت، واقدر اعمل الـ implementation في الـ Classes بتاعتي من غير ما أضطر إني أكتب كود أنا مش محتاجه، يعني Class بتاعت الـ Developer هتعمل implement للـ interfaces اللي هيا Worker, Coder فقط لا غير.

والـ Class بتاعت الـ Manager هتعمل implement للـ interfaces اللي هيا Worker, ClientFacer فقط لا غير.

وده رابط المصدر اللي موجود فيه الشرح باللغة الانجليزية

https://www.hashbangcode.com/article/solid-principles-php

ما كان من توفيق فمن الله وحده، وما كان من خطأ فمني

Report this

Published by

Osama Mustafa
PHP Back-End Developer
Published • 3w

4 articles ✓ Following

مقالة مُبسطة عن مبدأ Interface Segregation Principle رقم 4 في الـ SOLID Principles وهو المبدأ

Like Comment Share 37 • 2 comments

Reactions

+25

2 Comments

Most relevant

Add a comment...

Mahmoud Khairy • 1st
Full-stack Developer PHP
3w

المبدا ده يعتبر اسهل مبدا وتقدر تكتشفه بسهوله
وبجد شرحه حلو مع اني قريت كتير
وياريت تشرح المبدا رقم 3 لانه اصعب واحد بالنسبالي ومش فاهمه كويس
See translation

Like · 1 | Reply · 1 Reply

Osama Mustafa • 1st
PHP Back-End Developer
3w

حاضر بإذن الله .. هيكون فيه بوست عن المبدأ التالت
See translation

Like · 1 | Reply

Osama Mustafa
PHP Back-End Developer
✓ Following

More from Osama Mustafa

Dependency Inversion Principle in PHP (in Arabic)
Osama Mustafa on LinkedIn

Open Closed Principle in PHP (in Arabic)
Osama Mustafa on LinkedIn

Liskov Substitution Principle in PHP (in Arabic)
Osama Mustafa on LinkedIn