

Liskov Substitution Principle in PHP (in Arabic)

Published on September 8, 2021



Osama Mustafa
PHP Back-End Developer

4 articles

✓ Following

مبدأ Liskov Substitution Principle هو المبدأ رقم (3) في المبادئ SOLID و هي مجموعة من المبادئ في المفهوم Object Oriented Design وهم كالتالي:

Single responsibility principle .1

Open/closed principle .2

Liskov substitution principle .3

Interface segregation principle .4

Dependency inversion principle .5

دي المبادي اللي اشرحت قبل كده

Interface Segregation Principle | <https://bit.ly/3no1573> •

Open Closed Principle | <https://bit.ly/3EbMnpC> •

أثناء المقالة هنستخدم الاختصار **LSP** عوضاً عن المصطلح الكامل (Liskov Substitution Principle)

لازم نقول التعريف بالإنجليزي الأول لمبدأ Liskov Substitution وبعدين نشرحه

*In a computer program, if S is a subtype of T,
then objects of type T may be replaced with
objects of type S without altering any of the
desirable properties of the program
(correctness, task performed, etc.)*

يعني ايه subtype أصلًا؟

لو عندي Class اسمها Dog يتعلّم Class Animal implements لـ Animal ، فهنا المبدأ Dog تعتبر subtype من المبدأ Animal

طيب لو عندي Class اسمها GermanShepherd (أكيد سمعت عن الكلاب الجير مين) بتورث من Class Animal

• كان الـ Class اللي هو GermanShepherd subtype من الـ Interface Animal اللي اسمه Animal interface implements للـ Dog اللي هو GermanShepherd الأب بتاع الـ Animal interface

• بطريقة مبسطة أكثر ... أنت مثلاً بتورث صفات من والدك، فأنت تعتبر subtype subclass من والدك ، ووالدك يعتبر subtype من جدك وهكذا

• بالمناسبة في متعدد في لغة PHP اسمها is_subclass_of() ي العمل check أو بتصوف هل الـ Object هو Class ثاني أو يعمل implements على Class الثاني ولا لا

• كده الصورة بدأت تتضح في ذهنك، عموماً sub-class subtype أو sub-class فالاتنين ليهم نفس المعنى تقريباً، ويمكن تصوّف صورة 1 علشان تفهم الموضوع أكثر

Image 1



طيب ايه هو مبدأ Liskov Substitution بشكل مبسط ؟

تعالي نشوف التعريف بشكل مبسط عن الشكل اللي فات بالإنجلش برضه وبعدين نشرحه

Every subclass/derived class should be substitutable for their base/parent class

دلوقي المبدأ ده بيقولك إن أي مكان هتستخدم فيه object من الـ Parent Class ، ينفع تشيل الـ object ده وتحط مكانه من الـ Child Class object

مثال بسيط: مش أنت دلوقي بيورث صفات من والدك، طيب لما والدك بيقى في موقف ما ومطلوب منه يأخذ قرار في حاجة معينة، لو أنت جيت مكانه في نفس الموقف فالمنفروض تتصرف نفس تصرفه

مبدأ LSP باختصار بيناقش فكرة إن إحنا لازم نعمل Inheritance بشكل سليم ، بمعنى أنه لما تيجي تعمل inherit من Parent Class، ينفع في اي وقت إن الـ Subclass تحل محل الـ Parent Class من غير ما يحصل أي مشكلة، والإ بيقى إحنا استخدمنا التوريث بشكل خاطئ

أنت لما تيجي تعمل Abstract Class أو لما تيجي تعمل Implementation لـ Interface ، ايه اللي يحصل؟ أنت بتاخذ الـ method من الـ Parent Class وتروج في الـ Child Class تعمل عليها ، override لحد هنا مفيش مشكلة

أحياناً أنت لما بتتجي تعمل override على الميثود فيتروج تغير الـ Behavior بتاعها في الـ Child Class ، فمبدأ LSP بيقولك بعد إتنك، فيه شوية حاجات لازم تمشي عليها لما تيجي تعمل override على الميثود، علشان الـ subtype يقدر يحل محل الـ Parent في اي وقت

طبعاً هيا القواعد اللي لازم نطبقها على الـ methods في الـ Child Class علشان نفضل محافظين على مبدأ LSP

Contravariance -1

The parameters of the overriding methods should be the same or more generic than the types of the parent's method parameters

يعني اي المصطلح ده : المصطلح ده معنی ده arguments سا بيجي معن meunod علىها، المصطلح ده بيقولك لما تجي تعمل override ينفع إنك تزود الد parameters range بتابع الد parameters ، بس لازم تقبلهم كلهم على الاقل يعني ، زي المثال الموجود في صورة 2

Image 2

في صورة 2 الميثود الموجود في الد Child Class هيa() range الميثود زوّدت الد parameters أو عدلت Widens لهم، بالنسبة الشكل ده int|float|string معناه إن الميثود ممكن تقبل integer أو float أو Union Types جديد انتصاف للإصدار 8 من لغة PHP واسمه syntax string وده

طبع تعالى نشوف مثال ثاني على موضوع الد Contravariance علشان تفهمه أكثر ، في صورة 3 ، دلوتنى

```
class Child extends Parent {
    public function process(int|float|string $value);
}
```

في صورة 2 الميثود الموجود في الد Child Class هيa() range الميثود زوّدت الد parameters أو عدلت Widens لهم، بالنسبة الشكل ده int|float|string معناه إن الميثود ممكن تقبل integer أو float أو Union Types جديد انتصاف للإصدار 8 من لغة PHP واسمه syntax string وده

طبع تعالى نشوف مثال ثاني على موضوع الد Contravariance علشان تفهمه أكثر ، في صورة 3 ، دلوتنى أنا عندي interface اسمه CatShelter (معناه مأوى للقطط)، وفيه ميثود اسمها() والمفروض أن أي Class هتعمل الد implements Interface ده فيها تروح تعمل على الميثود دا ، لحد هنا مفيش مشكلة

Image 3

مبدأ الد Contravariance بيقولك، إنت مسواح لك إنك تعمل Widen أو توسيع الد arguments أو كتب تانية، هتلافقه بيقولك إنك ممكن تخلي الد arguments بتعاتك في الد Child Class بشكل more generic أو less specific نفس المعنى، فكلهم لهم نفس المعنى

Image 4

في صورة 4، أنت لما جيت تعمل override على ميثود() takeIn() فبدل ما تبعـت لها object من Cat Class ، فـانت بـعـت لهاـ من object اللي هوـ يعتبر less specific أو شـكل أعلىـ منـ الدـ Catـ صحـ مـفيـشـ مشـكـلةـ

طبع تعالى نشوف صورة 5، هتلافق الميثود اللي هيa() takeIn() بتلخـدـ argument عـبارـةـ عنـ objectـ منـ MaineCoon Classـ (ماينـ كـونـ دـهـ نوعـ منـ القـطـ)، هناـ بـقـيـ فيهـ مشـكـلةـ، لأنـ المـفـروـضـ لـماـ تـجيـ تـعملـ overrideـ علىـ المـيثـودـ الليـ هيـa()ـ فـيمـسـوحـ لكـ إنـكـ تعـملـ لـ الدـ Widenـ argumentـ ، إـنـاـ هـاـ إـنـتـ عـدـلتـ Narrowـ أوـ تحـجـيمـ ليـهـ ، أوـ خـلـيـتـ شـكـلـ moreـ sp~cificـ وـدـهـ مـشـ صحـ أـوـ بـيـخـالـفـ مـبـداـ LSPـ وـيـاتـالـيـ الدـ Subclassـ مـشـ هـنـقـرـ تـحلـ محلـ الدـ Parent~Classـ فـيـ أيـ مكانـ!

Image 5

Covariance -2

The types of the returned values of the overridden method should be the same or more specific as the types returned by the

Covariance -2

The types of the returned values of the overridden method should be the same or more specific as the types returned by the same method in the parent class

المصطلح د بيفولك لما تجي تعمل return فمطلوب منك تعمل return زي الـ Parent بالضبط أو تعمل return لشكل بيقى عباره عن من الـ return اللي موجود في الـ Parent Class ، خلينا نشوف المثال الموجود في صورة

Image 6

Image 6

ف صورة 6 ، الـ method اللي هيا (`Image 6`) لما جت تعمل return ، رجعت `JpgImage` اللي هيا subtype من `Image` وده كده تمام، طب تعالى نشوف مثال تاني على الـ Covariance متعلق بموضوع الـ Union اللي انكلمنا عليه من شوية `Types`

Image 7

في صورة 7 ، هنا الـ method `process` اللي اسها `override` على الميثود الموجودة في الـ Parent فقط، هيا كده مش بتختلف مبدأ LSP ، هيا بتنقوله بضم انت عاوز تعمل return لـ `Parent Class` زي `Integer` زي `String` زي `Boolean` زي `Double` زي `Float` زي `Char` زي `Byte` زي `Short` زي `Long` زي `BigInteger` زي `BigDecimal` زي `BigInteger` زي `String` زي `Integer`

في صورة 7 ، هنا الـ method `process` اللي اسها `override` على الميثود الموجودة في الـ Parent فقط، هيا كده مش بتختلف مبدأ LSP ، هيا بتنقوله بضم انت عاوز عمل return لـ

في صورة 7، هنا الـ method process لما جت تعمل override على الميُثود الموجودة في الـ Parent فهى هترجع integer فقط، هيا كده مش بتختلف مبدأ LSP ، هيا بيقوله بس انت عاوز تعمل return لـ integer أو string ، وأنا اللي في استطاعتي إني أرجعلك integer بس !! فاك قالها خلاص تمام مفيش مشكلة

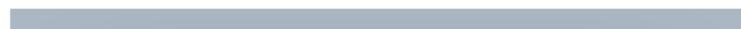
يبقى في القاعدة دي إحنا عملنا حاجة اسمها Narrow يعني تضييق أو تحجيم للـ Return Type، وساعتى هتراءها فى كتب أو مقالات تانية ، هتلقيه بيقولك إن الـ Return Types فى الـ more specific بيكون Covariance ، كده الاثنين ليهم نفس المعنى

Exceptions -3

*Always return exceptions of the same type or
more specialized*

لو فيه ميُثود في الـ Parent Class بتعمل Exception throw ، فلازم في الـ Child لما تيجي تعمل override على الميُثود دي، إنك تعمل throw نفس الـ Exception أو لـ More Specialized Exception ، خلينا نشووف مثال علشان المchorة توضح أكثر

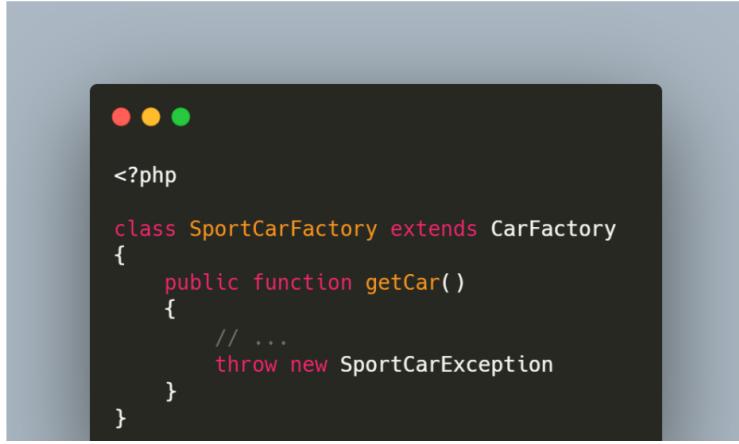
Image 8



VehicleException => CarException => SportCarException

في صورة 8، في ميثود getCar() في الـ Child Class إحنا عملنا throw لـ VehicleException اللي هيأ تعتبر الأب لـ CarException وده مش صحيح المفروض إن إحنا نعمل throw لنفس الـ Exception أو لشكل أصغر منه في التسلسل، زي المثال الموجود في صورة 9

Image 9



```
<?php

class SportCarFactory extends CarFactory
{
    public function getCar()
    {
        // ...
        throw new SportCarException
    }
}
```