12/12/2022

Abdo Gamal Ahmed

no campany

Design pattern

Intro



Table of Contents

[0-start with: 2](#_Toc122749892)

[1 – liskov substitution principle : 2](#_Toc122749893)

[1.1 rectangle/square ex on substitution principle by code : 3](#_Toc122749894)

[1.2- detect LSP violation: 5](#_Toc122749895)

[1.3 – some important notes : 7](#_Toc122749896)

[2 – interface segregation principle : 8](#_Toc122749897)

# 0-start with:

هنبداء ف تالت مبداء من solid principle .



# 1 – liskov substitution principle :

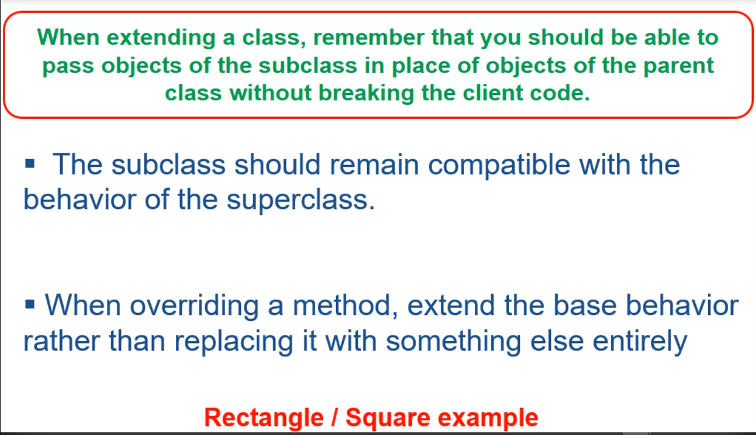
احنا زمان كنا بنعمل inheritance بين 2 class لما تكون العلاقه هى is a . لما اكون قادر قول ان ال class دة is a كذا . مش مجرد الاقى شويه حجات مشتركه اجرى اعمل inheritance . ف عمنا liskov زود شرط كمان على موضوع ال is a .

**قلك لازم ال child class دة يكون substitution replacement for parent class :** يعنى عشان نعمل inheritance لو عندك كود قيدم بياخد object من ال parent class ف لو بعت object من ال child class بدال ال parent class مش هيحصل اى مشكله على مستوى ال logic او ال run . ولو حصل مشكله لما تبعت ال child class كدة انت مش ماشى على ال liskov principle .

فعشان نكون متبعين liskov principle هتتاكد من الحجات دى .

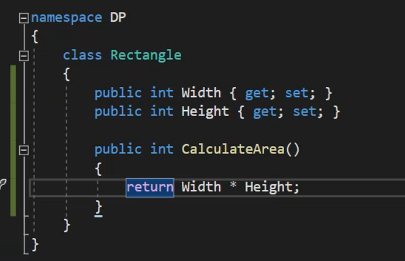
1. الchild class لازم يكون compatible مع ال parent . يعنى ال child class هيكون ليه logic مش مختلف عن ال parent
2. لما تعمل override method مينفعش تغير ال action انت مجرد بتغير ال implementation مختلف بنفس الlogic . يعنى ف معظم الاحيان وانت تعمل override method هتلاقى نفسك جوة ال override method دى بتنده على ال base method وبعدين بتكمل عليها ب implementation من عندك .

* **وتعاله ناخد مثال سريع** : افرض مثلا عند emp class ,accountant class بحسب المرتب بتاع ال emp ف لما اجى احسب ال salary accountant هخل ال accountant يعمل inherit للemp واخلى ال accountant يعمل override لل function الى بتحسب ال salary . و كلوجك مفورض حساب ال salary بالنسبه للaccountant تكون عباره عن حسبت ال salary بتاع ال emp بس هنزود عليه حاجة او ننقص منه حاجة . ف المفروض جوة الfunc الى بتحسب الsalary بتاع ال accountant انادم الداله الى بتحسب الsalary بتاع ال emp وبعد كدة اعمل extend ازود ال behavior الى انا عاوزة . مش ان اروح اغير ال implementation بالكامل بتاع الفنكشن الجديدة .
* لو مفهمش المثال تعاله تاخد مثال بالكود .

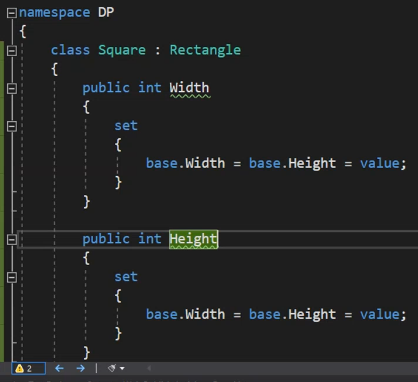
دة اختصار الشرح الى انا شرحته فوق .

## 1.1 rectangle/square ex on substitution principle by code :

هشرحلك دلوقتى مشكله قبلناها ف ال c++ وعرفنا نحلها :

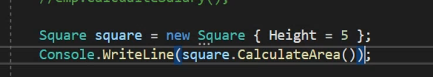
هنعمل دلوقتى class rectangle وsquare class .ونخلى ال squareيورث من ال rectangle ودة صح كلوجك على فكرة يعنى اقدر اقول ان ال square هو rectangle is a بس طوله زى عرضه . ف غرض المثال دة يثبتلك ان مش اى حاجة ينطبق عليها قاعدة is a ينفع اعمل inherit فيها وتعاله نشوف ازاى .

1. عملنا كلاس rectangle
2. عملنا كمان class square وخلناه يورث من ال rectangle. بس هنا ف مشكله ان كدة كلاس square شايف المتغرات الى فكلاس rectangle ويقدر يتعامل معاهم هم الاتنين ودة غلط عشان المربع ملهوش طول وارتفاع هو ليه side بس .

* ف الحل ف ال c++ كنا بنخلى المتغيارت الى ورثتها من المستطيل دى private ف كدة مكنتش بتبان للناس من برة الكلاس بس الحل دة شغال ف ال c ومش موجود ف الc# .
* الحل دلوقتى ف ال c# ان اعمل override للproperty الى ورثتها .وغير الbehavior بتاعها يعنى اخلى ال طول بتاع base class بيساوى الارتفاع بتاع base class .

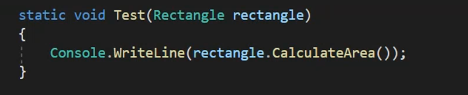
ف كدة لما اعمل object من المربع واحط الطول او الارتفاع بيساوى اى قيمه

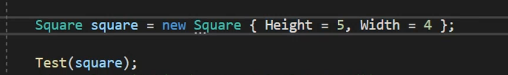
كدة هو هيخلى الطول والارتفاع بتاع الbase بقيمه متساويه.

1. لو جينا ف ال main وندينا داله calc هنلاقيها طلعت قيمه مظبطه الى هى 25 وعلى فكرة داله ال calc دى بتستخدام الطول والعرض الى ف ال base عشان انت

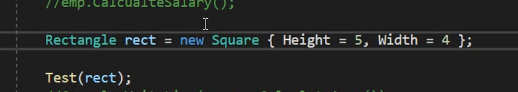
مكتبتش كلمه virtual عند الbase

ولا كتبتoverride عند الchild .

1. فين المشكله طيب او المكشله هتظهر امتى؟ لو كان ف function انا عملها من زمان قبل معمل square class ودة شكل الفنكشن .
2. ف الفنكشنtest دى عادى ابعتلها object من square ومفروض تطلع قيمه صح! القيمه الصح لينا انا كصاحب function test هى انها تضرب الطول ف العرض . ف انا كصاحب فنكشن دى هعمل object من square وهخلى الطول ب5 والعرض ب 4 وابعته . ف هكون متوقع الاجابه تكون 20 بس هلاقى الى طلع هو 16 . **والسبب ان الى عمل class square غير ال behavior بتاع الproperty . ف** كدة القيمه طلعت غلط من وجه نظر الtest function .



1. وطبعا ينفع تخلى ال reference بتاعك من انوع عادى وهتظهر نفس المشكله الى فوق .



1. تانى هقلك المشكله حصلت ليه: انت غيرت الbehavior بتاع ال property وخليت لما اعمل set للطول بعمل set للعراض كمان. ف انت دلوقتى function test مستنيه منك تبعت rectangle او اى بديل بس يكون بديل صح .
2. الخلاصه : عشان اخلىsquare يورث من ال rectangle اول سوال تسئله

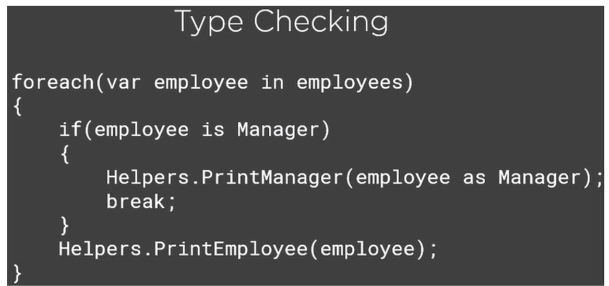
* هل ال square is a rectangle ؟ ف الاحابه ايوة
* تانى واهم حاجة هى هل ال square هو substitution لل rectangle ؟ لو الاجابه ايوة اعمل inherit لو الاجابه لا لا خلاص مش هتعمل inheritance . طيب مال هنعمل ايه لو مش هعمل inheritance اعمل كل واحد كلاس لوحدة بس كدة انت هتكرر نفس الفنكشن الموجوة ف rectangle موجودة ف square .

- ف حل تانى انك متعملش كلاس square وتخلى ف bool isSquare جوة ال rectangle class ولو ال isSquare دى بتساوى true خلى ال width =height .

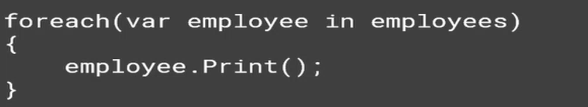
## 1.2- detect LSP violation:

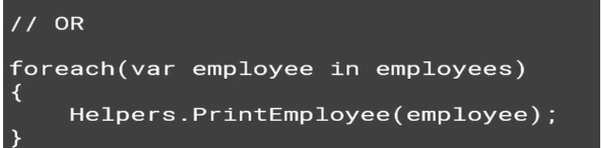
ازاى اعرف ان انا **مش** ماشى على ال LSP بطريقه صح .

1. لو لقيت نفسك جوة الكود بتعمل check بستخدام is or as type .

ودة مثال ان معايه list of emp وبلف عليهم وجيت ف النص كدة بقله او ال emp دة هو manger اعمل حاجة مختلفه او حاجه خاصه. **دة معناه ان مش ف كل الحلات ال manger هو substation لل emp** عشان جيت ف حاله معنيه وعملت condition .والمشكله هنا ان هتلاقى ال foreach بتاعتك كبرت خالص من كتر ال condition الى هتكون فيها .

* طيب حل المشكله ان اخلى ال manger يعمل override لل function print ويعمل ال implementation بتاعه . وجوة ال foreach ننادم على print . ف كدة حسب ال object هينادى ال print .

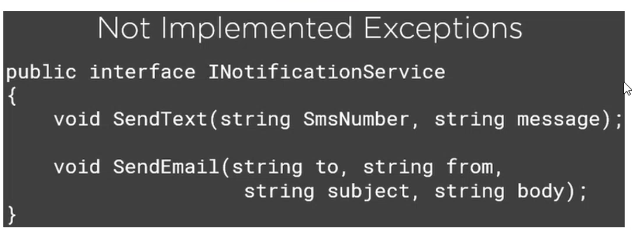


* او ممكن تستخدام داله جوة كلاس تانى زى الاسكرن الى هنا دى .

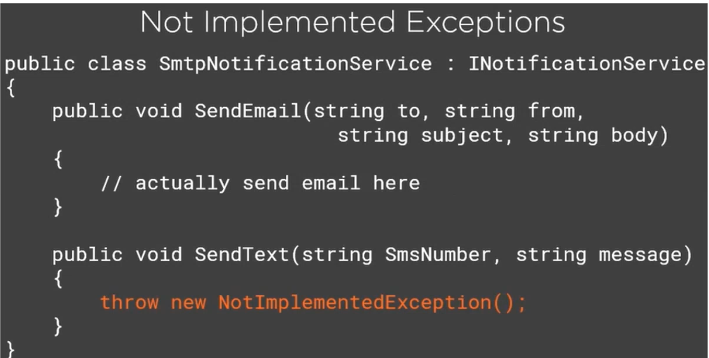
**ف خلاصه المشكله دى** : امتى على مبداء "tell do not ask " يعنى لو لقيت نفسك بتسئل سوال علىtype for object .او حتى لو كنت بتعمل check لقيمه reface جيبلك هل هو بيساوى null ولا لا ف دة برضو لازم اللمبه الحمرة تنور عندك عشان دة معناه ان برضو انت معتمد على value معينه وعشان كدة ف design pattern الل nullable .

1. تانى حاجة مفروض تخلى المبه الحمرة تنور عندك هو انك تكون عامل : throw not implemented exception .

تعاله ناخد مثال عشان تفهم افرض ان انا عامل interface inotifcationService وده اى حد عاوز يبعت notification يعمله implement . وفيه 2method واحدة عشان لو هبعت ميل والتنيه عشان لو هبعت text .



طيب افرض ان انا مهتم ب ان ابعت ميل بس ومش محتاج ال function التانيه هعمل ايه؟ خلاص هعمل implement للinterface بس مش هكتب الكود لكل ال method هكتب كود الى محتاجه بس والى مش محتاجها هكتب جوها throw not implemented Exaptation عشان اضحك على ال compiler ومعملش implement لكل الmethod بالشكل دة يعنى.

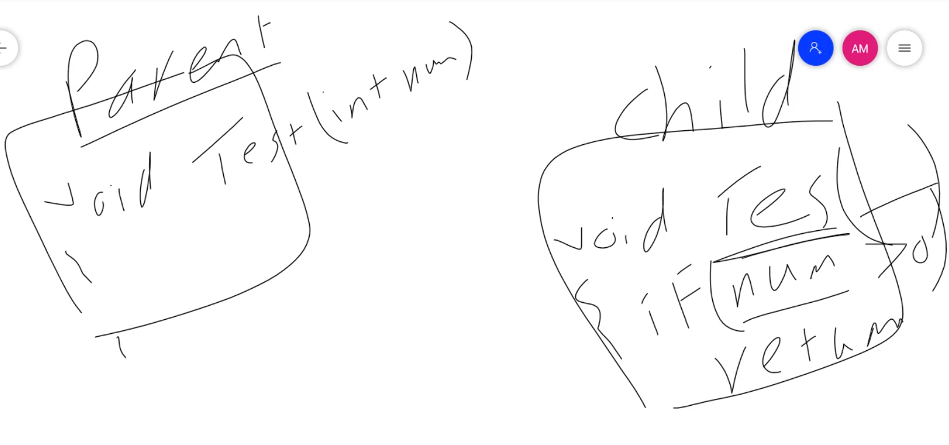


المشكله بقا ان لو انا كود قديم وبقول ان هتعامل مع اى كلاس عامل implement ل inotifcationService وجوة منى بنادم على داله ال sendtext الى مش معملها implement ف الكود هيضرب لو اتبعتلى objet من الكلاس بتاعك . ف بالتالى child دة مش بديل للbase وانت مش ماشى على LS principle .

## 1.3 – some important notes :

شويه حجات عشان تطمن ان ال child هو substitution للbase .

1. **A subclass should not strengthen pre-condition**  : يعنى انا عندى 2 class الاول هو ال parent والتانى وارث منه واسمه child. . ف ال parent جواه داله اسمها test ف انا روحت عملتها overload جوه ال child وعملت شرح ان لوالقيمه الى اتبعت لل test دى موجب هعمل return . ف انت كدة يعتبر غيرت ف ال action وهنشوف المشكله الى هتحصل .

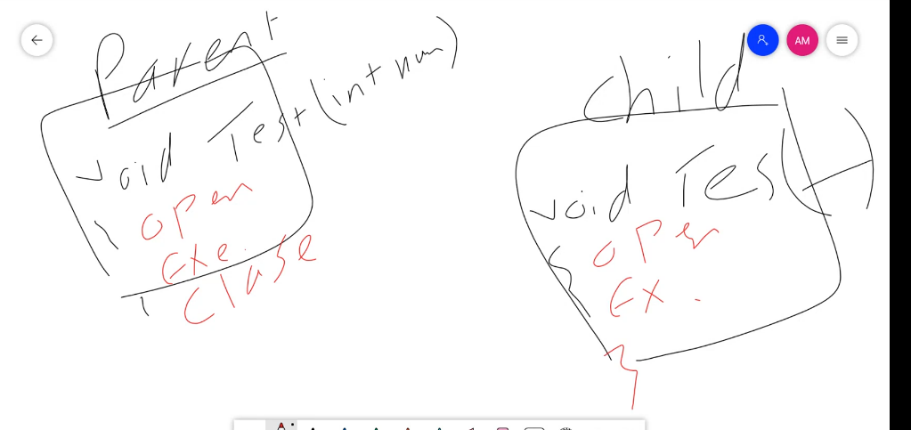


ف انا كنت كاتب function زمان بتستقبل object من كلاس parent وجوها بنده على function test وبتعامل معها . ف انت لو عملت object من ال child وبعته للfunction الى عمها زمان دى مش هتشتغل صح او الناتج الى متوقعه هيكون غلط . ودة بسبب انك عملت strengthen .

1. **A subclass should not weaken post-condition :**

نفس المثال الى فات عندى 2 class الاول هو ال parent والتانى هو ال child ال child عمل override لداله جوة ال parent . وخلى بدال م الداله بتاعه ال parent كانت بتفتح ال connection وتعمل execute وتقفل ال connection خلها تفتح وتعمل execute بس ومقفلش الكونكشن .

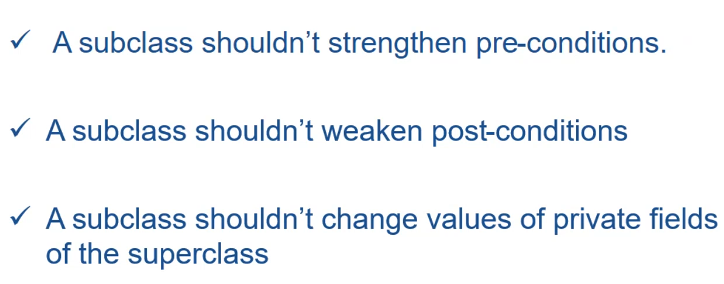
* ف لو انا كنت عامل داله زمان برضو قبل معمل ال child . وبعتلها object من ال child كدة ال connection مش هتتقفل فهتحصل مشاكل عشان الكود الى ف داله القديمه مش بيقفل الconnection ف مثلا لو نداهت على function بتفتح الكونكشن تانى هتحصل مشكله .



ف خلاصه النقط الى فاتت:

مينفعش تزود شرط على الحاجة الى بتخدها وكمان مينفعش تقلل من الstate الى مفروض تطلع بيها من ال function .

* ف لو متوقع من ال parent يطلع من ال function بتلت حجات يبقا لازم يبقا لازم ال child كمان يطلع بتلت حجات .

1. مينفعش تعدل ف protected member بتوع ال parent من ال child class لازم بدون استخدام الدوال الى هو موفرهالك .

ودول نفس التلت نقط الى فاته .

نروح لل principle الى بعده .

# 2 – interface segregation principle :



المبداء دة بيقلك من الاخر ان ال client مش مجبر implementation ل function هو مش عوزها يعنى ابعد عن ال fate interfaces .

Fat interfaces : ال interfaces fat هو عبارة عن ال interfaces او ال abstraction class الى جوها مجموعه كبيرة من ال function وفمعظم الاحيان ال function دول مش بيكونه relate ببعض اصلا .

* ف هنرفع شعار ال client ميعملش implement لل حجات الى محتاجه بس . ف معنى كدة ان كل ميزيد ال func الى جوة ال interface هيزيد احتمال ان client يعمل implement لحاجة هو مش عوزها .
* وخلى بالك عادى ممكن يبقا ف interface فيه شويه دوال كتير بس بيخدمه على نفس الفكرة دة وارد .

**تعاله ناخد مثال :**