

ماذا نقصد ب SOLID Principle !!

SOLID Principle : هي مجموعه من المبادئ التي نحتاجها اثناء تصميم ال Software الخاص بنا و نطبق هذه المبادئ قبل البدء في ال Design Patterns بمعني انه يجب مراعاة هذه المبادئ الخاصه بال SOLID Principle قبل استخدام ال Design Patterns   
  
ما هو ال Design Patterns : هو عباره عن مجموعه من القواعد لحل مشكلات معينه بأفضل طريقه لها .  
" سوف نتحدث عنها عند الانتهاء من ال SOLID ‘ان شاء الله’ " .

اسم SOLID هو اختصار مساعد للذاكرة حيث يمثل كل حرف مبدأ لتصميم البرامج، كما يلي:

(Single responsibility principle) : S وهذا المبدأ اختصار لأول حرف وهو حرف (S) في كلمة (SOLID) ، والذي يشير إلى مبدأ المسؤولية المنفردة، والذي يعني أن كل جزء من الكود البرمجي يجب أن يكون له مهمة محددة وواحدة للقيام بها، حيث أن عندما يتعامل الكلاس مع أكثر من مهمة واحدة فإن أي تغييرات ممكن إجراؤها على الوظائف قد تؤثر على الكلاسات الأخرى.

(Open-closed principle) : Oوهذا المبدأ اختصار لثاني حرف وهو حرف (O) في كلمة (SOLID)، والذي يعني أن الكلاس يجب أن يكون مرن وقابل للإضافة وتعديل المحتوى دون الحاجة إلى تعديل أي محتوى موجود مسبقاً.

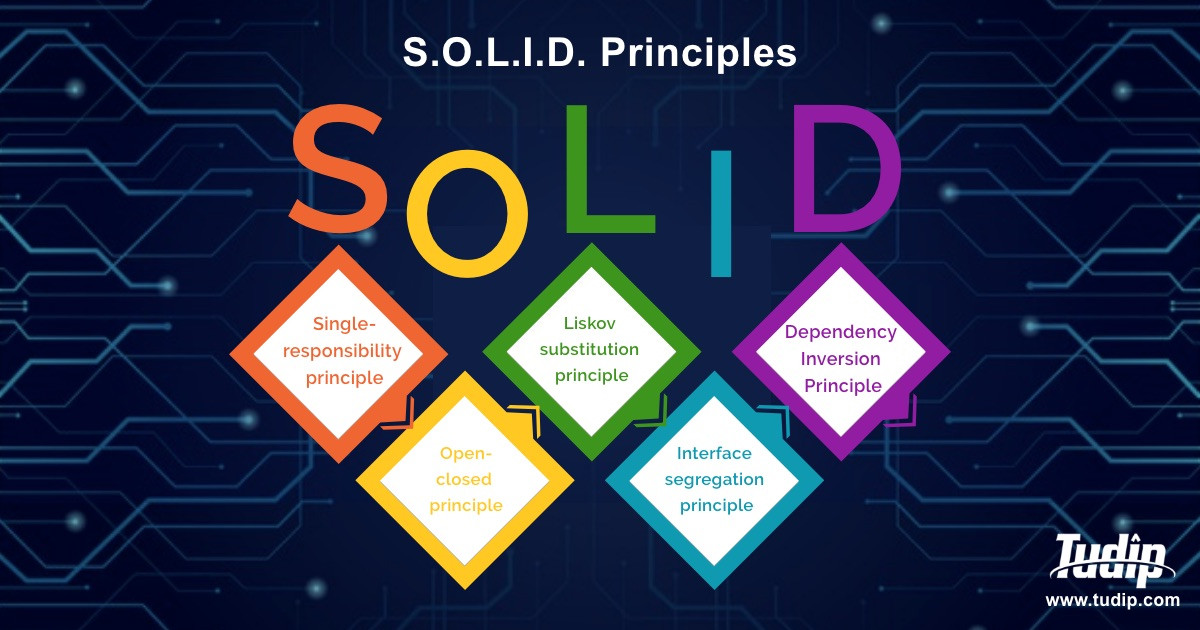
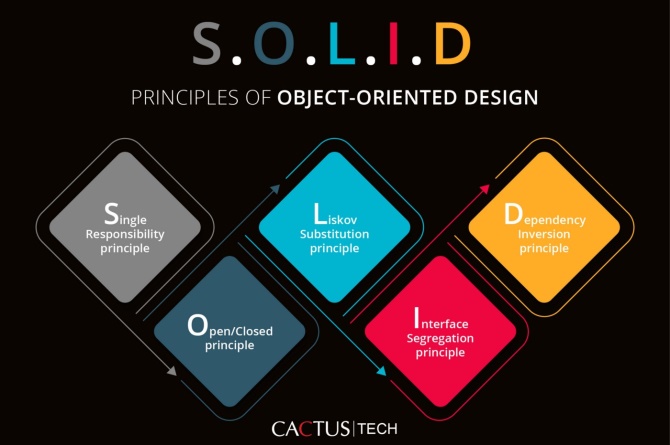
(Liskov substitution principle) : L وينص هذا المبدأ على أنه إذا كان التطبيق يستخدم (Object) معين من كلاس رئيسي أو أب (Base Class or Parent Class)، فإنه يجب أن يكون قادر على استخدام أي (Object) من الكلاسات المشتقة (Child) (Class) من الـ Parent) Class ) وذلك دون أن يعلم التطبيق بذلك، ويظل يعمل بكفاءة عالية.

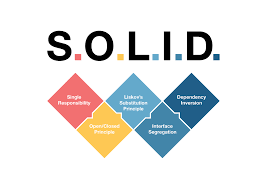
(Interface segregation principle) : I مبدأ فصل الواجهة هو مبدأ تصميم (SOLID) الرابع الذي يمثله الحرف “I” في الاختصار، وهذا المبدأ بسيط حيث أنه ينص على أنه لا يجب إلزام أي كائن على أخذ دالة لن يستخدمها أبداً.

( Dependency inversion principle) : D مبدأ عكس التبعية هو المبدأ الخامس والأخير لتصميم (SOLID) الذي يمثله حرف “D” والذي قدمه (Robert C Martin)، والهدف من هذا المبدأ هو تفادي الكود المرتبط بشدة بكود آخر، لأنه يكسر التطبيق بسهولة.

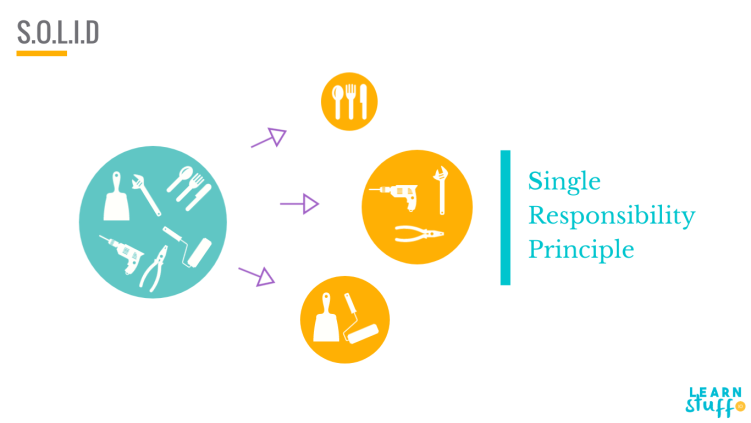
## فوائد استخدام مبادئ SOLID:

* الحصول على كود سهل الفهم.
* الحصول على كود قابل للتعديل.
* الحصول على كود قابل للاختبار.



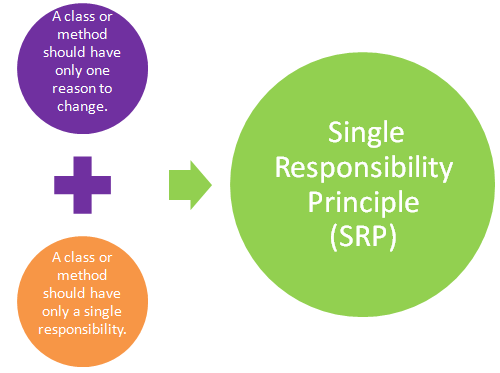


هنتكلم النهارده عن اول مبدأ من هذه المبادء الخمس و هو مبدأ   
 Single Responsibility Principle ( S.R.P)



Single Responsibility Principle ( S.R.P)

"Do one thing and do it well"

‘Uncle Bob’ , Robert C. Martin   
" There Should never be more than one reason for a class to change "  


في هذا المبدأ نقوم بجعل كل Class يقوم بعمل وظيفه واحده فقط و ينفذها بشكل جيد و ايضا لا ينبغي ان يكون هناك اكثر من سبب للتعديل علي ال Class الخاص بي .

هناخد مثال يوضح الفكره كلها بكل بساطه دلوقتي !  
مثال :

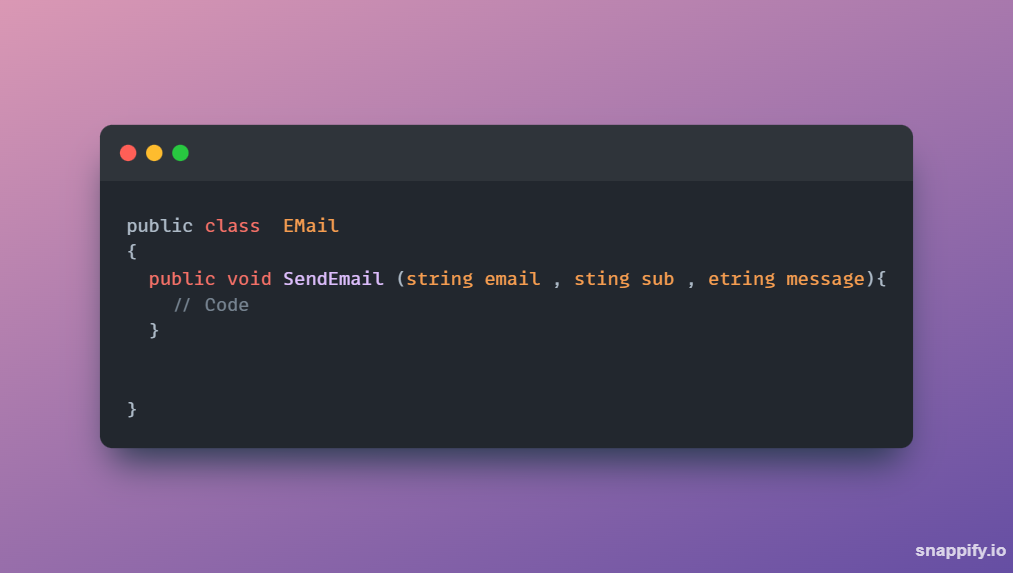


في هذا المثال قمت بإنشاء Class خاص بال Registration للموظفين و قمت بإنشاء بداخله function خاصه بإضافة الموظفين و أخرى لإرسال ال E-Mail و في كل منهم الكود الخاص بالعمليه التي يقوم بها   
لنقوم بتحليل هذا ال Class و نرى هل هوا يقوم بتطبيق مبدأ ال SRP ام لا !

اولا : هذا ال Class يقوم بأكثر من عمليه و هي اضافة الموظفين و العمليه الاخرى و هي ارسال Emails للمموظفين لا مانع من عمليات لها علاقه بال Registration الخاص بالموظف لكن هل ارسال ايميل من ضمن هذه العمليه ! بالطبع لا .

ثانيا : اذا اردت ان اقوم بالتعديل علي ال Email حينها سوف اقوم بفتح ال Class للتعديل علي الكود الموجود في ال Email او اي تعديل اريده و ايضا ال Function الاخرى و هي الخاصه بتسجيل بيانات الموظف اذا اردت ان اقوم بالتعديل علي هذه البيانات حينها سوف اقوم بالدخول علي ال Class مره اخرى للتعديل  
 هنا يوجد اكتر من سبب للتعديل علي هذا ال Class فهو لا يطبق مبدأ ال SRP .

الحل !   
 



هنا قمنا بتطبيق مبدأ SRP علي المثال السابق   
 كل Class يقوم بفعل وظيفه واحده فقط الاول يقوم بتسجيل بيانات الوظ و الثاني يقوم بإرسال ال Emails علي سبيل المثال اذا اردنا ان نقوم بالتعديل علي كود ال Email لاي سبب حينها لن نقوم بالتعديل داخل ال Employee ولكن التعديل سوف يكون فقط داخل ال Email

و هنا لا يوجد سبب للتعديل فقط الا اذا اردت التعديل علي ال Email و ايضا في تسجيل بيانات الموظف لا يوجد اي سبب يستدعي التعديل علي هذا ال Class الا اذا اردت التعديل علي بيانات الموظف فقط .

في المثال التالي اخبروني اذا كنت قمت باتباع مبدأ ال SRP ام لا !  
واذا كانت الاجابه لا فكيف نقوم بتطبيق هذا المبدأ عليه !

