الاســم: نوال عبد الجليل سيف

عثمان.

رقـم القيد: 21_16_0042

نظـم معلومات/ عام

SOLID PRINCIPAL.

المبادئ الخمسة لمفهوم SOLID لتطوير البرمجيات

SOLID Principle:

تُعد (SOLID Principle) مجموعة من المبادئ لبناء (Software) والتي تتبع أفضل الحلول البرمجية والتي يطلق عليها (Best Practices) ؛ حيث أنها تساعد المستخدم في تجنب الثغرات الشائعة والتفكير في بنية تطبيقاته بمستوى أعلى، وتكمن أهمية هذه المبادئ في الحصول على كود خالي من الأخطاء، ويعد هذا الأمر ضروري لبناء برامج مرنة ومستقرة.

وكلمة (SOLID) هي اختصار لخمسة مبادئ خاصة في عملية التصميم والتي تهدف إلى جعل تصاميم البرمجيات أكثر تفهماً ومرونة وفيما يلي هذه المبادئ الخمسة، ولاستخدام هذه المبادئ يجب أن يكون المستخدم مُلماً بمفاهيم البرمجة الموجهة للكائنات (OOP).

المبادئ الخمسة الخاصة بمفهوم SOLID:

مبادئ تصميم (SOLID) هي خمسة مبادئ تصميم خاصة بالبرامج تُمكنك من كتابة تعليمات برمجية فعالة، وفيما يلي هذه المبادئ:

١. مبدأ المسؤولية الواحدة أو الفردية – Single Responsibility Principle:

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.

٢. المبدأ مفتوح – مغلق – Principle Open Closed:

وهذا 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.

بمعنى انك يجب أن يتم تصميم الكلاسات بطريقة ما تمكنك من إضافة ميزات جديدة او تعديل سلوك المكونات الحالية؛ بحيث لا تطلب تعديل كبير في الكود البرمجي الحالي.

٣. مبدأ الاستبدال والتعويض – Liskov Substitution

Principle:

مبدأ الاستبدال والتعويض (Liskov) وهو المبدأ الثالث لـ (SOLID)، ويمثله الحرف (L)، وهذا المبدأ نسبة إلى (Barbara Liskov) هي مَن طور هذا المبدأ في عام ١٩٨٧، وينص هذا المبدأ على ما يلي:

"إذا كان (S) فرع تابع ل (T) فإن كائنات (T) يجب أن يتم استبدالها بكائنات (S) دون وقوع أي شيء غير مرغوب به في البرنامج".

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 Class (Parent)، وذلك دون أن يعلم التطبيق بذلك، ويظل يعمل بكفاءة عالية.

٤. مبدأ فصل الواجهة – Interface Segregation Principle:

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.

٥. مبدأ عكس التبعية -Dependency Inversion Principle:

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.