جامعة صنعاء

كلية الحاسوب

نظم المعلومات

Solid Principal

إعداد : رقية قايد المثيل

مبادئ التصميم الثابتة لبناء أنظمة مستقرة ومرنة (SOLID)

إن الكود الخالي من الأخطاء أمر ضروري لبناء برامج مرنة ومستقرة، ولكن فعلياً هذا لا يكفي

فنحن نحتاج لمراعاة مبادئ التصميم وتعد" Solid "

أكثر مبادئ التصميم شهرةً، يمكن أن تساعدك في تجنب الثغرات الشائعة والتفكير في بنية تطبيقاتك بمستوى أعلى قبل البدء , نود توضيح بعض المصطلحات المستخدمة في هذا المقال , وهم اثنان :

صف وتعني المصطلح البرمجي كلاس Class

تابع وتعني المصطلح البرمجي دالة function

ما هي مبادئ التصميم الثابتة؟

هي خمسة مبادئ تصميم برامج تُمكنك من كتابة تعليمات برمجية فعالة Solidمبادئ تصميم

من المهم معرفة مبادئ OOP مثل الوراثة والتغليف

هو اختصار مساعد للذاكرة حيث يمثل كل حرف مبدأ لتصميم البرامج،

s (Single responsibility principle)

أي يمتلك الصف مسؤولية إفرداية فقط

O(Open-closed principle)

يجب أن تكون البنى البرمجية مفتوحة إلى التمديد ومغلقة بالنسبة للتعديل

L(Liskov substitution principle)

يجب على الكائنات الموجودة أثناء عمل البرنامج أن تكون قابلة للاستبدال بكائنات أخرى وارثة لها دون التأثير على صحة البرنامج

I (Interface segregation principle)

إن وجود عدد من الواجهات الخاصة بالزبون أفضل كثيراً من وجود واجهة وحيدة عمومية الأهداف

.

D (Dependency inversion principle)

يجب الاعتماد على التجريدات وليس التحقيقات"، عكس التبعية هي طريقة تساعد على اتباع هذا المفهوم

تؤدي مبادئ SOLID

إلى بنية برمجية أكثر مرونة واستقراراً مما يُسهل صيانتها وتوسيعها، وتصبح أقل عرضة للانهيار

مبدأ المسؤولية الفردية S

مبدأ المسؤولية الفردية هو أول مبدأ تصميم ل SOLID

ويمثله الحرف "S".

ينص على أنه في أي تطبيق مُصمم، يجب أن يرتبط كل صف

بمهمة خاصة به. واستخدام هذه المهمة يتم عند وجود سبب

لاستخدام هذا الكود عندما نتعامل الصف مع اكثر من مهمة .

المبدأ مفتوح – مغلق O

وينص هذا المبدأ على أن الصفوف ووحدات الأكواد الأخرى يجب أن تكون مفتوحة للتوسع ولكن يجب إغلاقها للتعديل

مبدأ الاستبدال L

هو المبدأ الثالث ويمثله حرف L

مبدأ فصل الواجهة I

مبدأ فصل الواجهة هو SOLID الذي يمثله الحرف I
مبدأ تصميم

مبدأ عكس التبعية

مبدأ عكس التبعية هو SOLID الذي يمثله حرف D
المبدأ الخامس والأخير لتصميم

الهدف من مبدأ انعكاس التبعية هو تفادي الكود المرتبط بشدة بكود آخر، لأنه يكسر التطبيق بسهولة

ينص المبدأ على ما يلي :

يجب ألا تعتمد النماذج عالية المستوى على النماذج " منخفضة المستوى. يجب أن يعتمد كلاهما على مبدأ " التجريد ".

بمعنى آخر، تحتاج إلى فصل النماذج عالية المستوى عن المستوى المنخفض

عادةً ما تقوم الصفوف العالية بتغليف منطقي معقد بينما تتضمن الصفوف ذات المستوى المنخفض بيانات أو أدوات مساعدة.