#Reflecton bài 7 : clean code ngày 02/02/2021

\*Những điều mới

-Clean code có tác dụng gì :

+Giúp cộng tác nhóm

+Dễ hiểu code và khiến code đơn giản dễ đọc hơn

+Dễ thay đổi dễ bảo trì dễ cải tiến

+Không có code trùng lặp

+Thể hiện ý tưởng thiết kế

+Trực tiếp giải quyết được vấn đề một cách tốt nhất

-SOLID là 5 nguyên tắc thiết kế hướng đối tượng :

+Single responsibility priciple (Một lớp chỉ chịu trách nhiệm về một nhiệm vụ thể nào đó mà thôi )

.Khi một class có nhiều chức năng nó sẽ trở nên cồng kềnh và khó đọc khó maintain

VD:

class Employee

{

string position;

function developSoftware(){};

function testSoftware(){};

function saleSoftware(){};

}

+Open/Closed principle (Không thể sửa đổi class có sẵn nhưng có thể mở rộng bằng kế thừa )

.Muốn thêm chức năng cho chương trình nên viết class mới và mở rộng class cũ chứ không nên sửa đổi .Việc mở rộng chức năng ta cần thêm code mới việc đấy sẽ làm code cồng kềnh và khó quản lý , hơn vì vậy nên tách phần dễ thay đổi ra khỏi phần khó thay đổi để đảm bảo không ảnh hưởng đến phần còn lại

VD:

abstract class Connection()

{

public abstract function doConnect();

}

class SqlServer extends Connection

{

public function doConnect()

{

//connect with SqlServer

}

}

class MySql extends Connection

{

public function doConnect()

{

//connect with MySql

}

}

+Liskov substitution principe ( các đối tượng (instance) kiểu class con có thể thay thế các đối tượng kiểu class cha mà không gây ra lỗi)

+ Interface segregation principle (Thay vì dùng interface lớn thì ta tách ra thành các interface nhỏ, với nhiều mục đích cụ thể )

.Có một interface lớn và khoảng 100 methods .Việc implements sẽ rất vất vả vì các class impliment interface sẽ bắt buộc thực thi toàn bộ các method của interface .Vì thế nên khi tách các interface thành các interface nhỏ bao gồm các mothod liên quan đến nhau ,và implement và quản lý dễ hơn

VD:

interface Animal {

void eat();

}

interface RunnableAnimal extends Animal {

void run();

}

interface FlyableAnimal extends Animal {

void fly();

}

+ Dependency inversion principle (Interface và abstraction không nen phụ thuộc vào chi tiết và ngược lại các class giao tiếp với nhau thông qua interface và abstraction không phải thông qua implementation )

.Những thành phần trong 1 chương trình chỉ nên phụ thuộc vào những cái trừu tượng (abstraction)

-**SOLID** là 5 nguyên tắc cơ bản trong việc thiết kế phần mềm. Nó giúp chúng ta tổ chức sắp xếp các function, method, class một cách chính xác hơn. Làm sao để kết nối các thành phần với nhau.Công dụng :

+Rõ ràng dễ hiểu ,dễ thay đổi ,dễ tái sử dụng .