**SOLID**

1. **Single Responsibility Principle(SRP) -** Nguyên lý Đơn Trách Nhiệm

Một class chỉ nên giữ một trách nhiệm duy nhất

Ta cần thiết kế các module sao cho đơn giản, một module chỉ có 1 chức năng duy nhất

public class Customer

{

public int CustomerId { get; set; }

public string CustomerName { get; set; }

public string Address { get; set; }

}

public virtual IQueryable<TEntity> GetAll()

{

return dbset;

}

public virtual TEntity GetByID(object id)

{

return dbset.Find(id);

}

public virtual void Insert(TEntity entity)

{

dbset.Add(entity);

}

public virtual void Delete(TEntity entity)

{

//Deleting by passing entity

if (context.Entry(entity).State == EntityState.Detached)

dbset.Attach(entity);

//remove item

dbset.Remove(entity);

}

1. **Open/Closed Principle -** Nguyên lý Đóng Mở

Có thể thoải mái mở rộng 1 module, nhưng hạn chế sửa đổi bên trong module đó (open for extension but closed for modification).

1 module cần đáp ứng 2 điều kiện sau:

* **Dễ mở rộng**: Có thể dễ dàng nâng cấp, mở rộng, thêm tính năng mới cho một module khi có yêu cầu.
* **Khó sửa đổi**: Hạn chế hoặc cấm việc sửa đổi source code của module sẵn có.

1. **Liskov Substitution Principle -** Nguyên lý Thay Thế **Liskov**

Trong một chương trình, các object của class con có thể thay thế class cha mà không làm thay đổi tính đúng đắn của chương trình

1. **Interface Segregation Principle – Nguyên lý phân tách interface**

Thay vì dùng 1 interface lớn, ta nên tách thành nhiều interface nhỏ, với nhiều mục đích cụ thể

1. **Dependency Inversion Principle**

1. Các module cấp cao không nên phụ thuộc vào các module cấp thấp. Cả 2 nên phụ thuộc vào abstraction.

2. Interface (abstraction) không nên phụ thuộc vào chi tiết, mà ngược lại. (Các class giao tiếp với nhau thông qua interface, không phải thông qua implementation.)

* KT Abstract