OOP Review

1. 4 tính chất của OOP( Lợi ích và ví dụ)

- Tính đóng gói: Có thể gói dữ liệu và mã chương thành một cục gọi là lớp (class) để dễ quản lí. Ngoài ra, các lớp liên quan đến nhau có thể được gom chung lại thành package

+ Tất cả các trường (field) của lớp có có chế độ chỉ đọc (read-only) hoặc chỉ ghi (write-only), tức là chỉ có hàm getter hoặc setter.

+ Một lớp có thể có toàn bộ điều khiển thông qua những gì được lưu giữ trong các trường (field) của nó.

+ Người sử dụng của class không biết cách các class lưu trữ dữ liệu. Một class có thể thay đổi kiểu dữ liệu của một trường và người dùng class không cần sự thay đổi trong code.

- Tính trừu tượng:

+ Cho phép các lập trình viên loại bỏ tính chất phức tạp của đối tượng bằng cách chỉ đưa ra các thuộc tính và phương thức cần thiết của đối tượng trong lập trình, cải thiện khả năng bảo trì của hệ thống.

+ Giúp chúng ta tập trung vào những cốt lõi cần thiết của đối tượng thay vì quan tâm đến cách nó thực hiện.

+ Cung cấp nhiều tính năng mở rộng khi sử dụng kết hợp với tính đa hình và kế thừa trong lập trình hướng đối tượng.

- Tính thừa kế:

+ Kế thừa cho phép chúng ta xây dựng lớp mới từ một lớp đã có, có thể sử dụng lại các thuộc tính, phương thức, hay dựa trên đặc điểm của mối quan hệ đặc biệt hóa, tổng quát hóa.

+ Tổ chức các lớp dựa trên chia sẻ mã chương trình chung, dễ dàng chỉnh sửa, nâng cấp hệ thống khi cần

- Tính đa hình: Tạo điều kiện cho các lập trình viên gia tăng khả năng tái sử dụng những đoạn mã nguồn được viết một cách tổng quát và có thể thay đổi cách ứng xử một cách linh hoạt tùy theo loại đối tượng.

- Ví dụ cho 4 tính chất của OOP:

Tạo 2 lớp Cat và Dog kế thừa từ Animal. Khi khởi tạo chúng sẽ có tên. Chúng override lại phương thức say\_hello để chào hỏi theo cách riêng của chúng. Thể hiện tính đóng gói (đóng gói biến tên và phương thức say\_hello với nhau) và tính thừa kế (Cat và Dog mang đặc điểm chung là có say\_hello từ Animal).

Tạo lớp Zoo để quản lí nhiều Animal, có (1) phương thức add, remove để thêm, bớt các Animal (các đối tượng của các lớp thừa kế từ Animal), (2) phương thức say\_hello\_all để gọi say\_hello của tất cả đối tượng nó quản lí. <- Thể hiện tính đa hình, Zoo gọi chỉ gọi một phương thức say\_hello, nhưng tùy con vật mà lời chào hỏi sẽ khác nhau.

2. Overriding là gì ? (ví dụ)

- Ghi đè (Overriding): là class con định nghĩa lại một method của lớp cha nhưng phải cùng tên và cùng số lượng tham số với method của lớp cha.

Ví dụ:

class Animal {

public int canFly(){

return false;

}

}

class Bird extends Animal{

// overriding

public int canFly(){

return true; }}

3. Overloading là gì ?(ví dụ)

- Nạp chồng (Overloading): là class con định nghĩa lại method của lớp cha nhưng khác số lượng tham số truyền vào so với lớp cha.

Ví dụ:

class {

public int sum(a, b){

return a + b;

}

// overloading

public int sum(a, b, c){

return a + b + c;

}

}.

4. Lợi ích của việc sử dụng lập trình hướng đối tượng

- Dễ dàng quản lý code khi có sự thay đổi chương trình.

- Dễ mở rộng dự án.

- Tiết kiệm được tài nguyên đáng kể cho hệ thống.

- Có tính bảo mật cao.

- Có tính tái sử dụng cao.

5. Các access modifier và quyên truy cập khi sử dụng nó

Private: chỉ được truy cập trong phạm vi lớp.

Default: Nếu không khai báo modifier nào, thì nó chính là trường hợp mặc định. Default là chỉ được phép truy cập trong cùng package.

Protected: được truy cập bên trong package và bên ngoài package nhưng phải kế thừa. Protected có thể được áp dụng cho biến, phương thức, constructor. Nó không thể áp dụng cho lớp.

Public: được truy cập ở mọi nơi.

6. Khi nào dùng Abstract ?

Sử dụng abstract class khi các class được extends có nhiều điểm chung và có mối quan hệ với nhau.

Ví dụ Byte, Double, Float… “ đều là” Number.

Tạo một class abstract khi bạn đang cung cấp các hướng dẫn cho một class cụ thể.

7. Khi nào sử dụng Interface ?

* Tạo Interface khi chúng ta cung cấp các hành vi bổ sung cho class cụ thể và những hành vì này không bắt buộc đối với clas đó.