**OOP REVIEW**

**1. 4 tính chất của hướng đối tượng**

- Tính đóng gói (Encapsulation) :

Là cách để che dấu những tính chất xử lý bên trong của đối tượng, những đối tượng khác không thể tác động trực tiếp làm thay đổi trạng thái mà chỉ có thể tác động thông qua các method public của đối tượng.

- Tính kế thừa (Inheritance) :

Là kỹ thuật cho phép kế thừa lại những tính năng mà một đối tượng khác đã có, giúp tránh việc code lặp dư thừa mà chỉ xử lý công việc tương tự.

- Tính đa hình (Polymorphism):

Là một đối tượng thuộc các lớp khác nhau có thể hiểu cùng một thông điệp theo cách khác nhau.

\*Trong code để thể hiện tính đa hình có 2 cách:

• Method Overloading : là cách nạp chồng các method có cùng tên nhưng khác tham số

• Method Overriding:  Đây là một phương pháp được ghi đè lại các method ảo của một lớp cha nào đó

- Tính trừu tượng (abstraction) :

Tính trừu tượng là một tiến trình ẩn các chi tiết trình triển khai và chỉ hiển thị tính năng tới người dùng. Tính trừu tượng cho phép bạn 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.

Trong Java, chúng là sử dụng abstract class và abstract interface để có tính trừu tượng.

**2. Lợi ích của việc sử dụng 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.

**3. Khi nào ta sử dụng Abstract class**

Chúng ta sử dụng Abstract class khi định nghĩa một đối tượng có những chức năng A,B,C trong đó tính năng A,B chắc chắn sẽ thực thi theo cách nào đó, còn tính năng C phải tùy thuộc vào đối tượng cụ thể là gì, như đối tượng Dog, Cat tuy chúng đều có thể phát ra âm thanh nhưng âm thanh là khác nhau. Vì vậy method Speak() là abstract method để chỉ ra rằng tính năng này còn dang dở chưa rõ thực thi, các lớp extend phải hoàn thành nốt tính năng này, còn những tính năng đã hoàn thành vẫn sử dụng như bình thường đây là những tính năng chung.

**4. Khi nào ta sử dụng Interface**

Chúng ta sử dụng Interface khi muốn xây dựng được bộ khung mẫu mà các lớp phải follow theo hoặc khi ta muốn chức năng được dùng chung bởi các implement class mà không quan tâm chúng là gì.

**5. Override là gì (đưa ra ví dụ)**

- Là phương thức đã xuất hiện ở lớp cha và xuất hiện tiếp ở lớp con.

- Khi đối tượng thuộc lớp con gọi phương thức thì sẽ chọn lựa và chạy theo phương thức trong lớp con.

- Nếu lớp con không có phương thức đó thì mới lên kiếm ở lớp cha để chạy

- Ghi đè là hình thức đa hình (polymorphism) trong quá trình thực thi (Runtime)

Ví dụ : Có một hàm ở lớp cha và bạn kế thừa từ lớp đó, nhưng ở lớp dẫn xuất bạn muốn khai triển hàm đó theo một cách khác dựa trên hàm có sẵn từ lớp cha để phục vụ cho mục đích gì đó, khi đó bạn sẽ sử dụng override(hay còn gọi là ghi đè) để định nghĩa lại hàm đó từ lớp cha sao cho phù hợp với mục đích của bạn ở lớp dẫn xuất.

**6. Overload là gì (đưa ra ví dụ)**

Đây là khả năng cho phép một lớp có nhiều thuộc tính, phương thức cùng tên nhưng với các tham số khác nhau về loại cũng như về số lượng. Khi được gọi, dựa vào tham số truyền vào, phương thức tương ứng sẽ được thực hiện.

Ví dụ : class Calculation {

void sum(int a, int b) {

System.out.println(a + b);

}

void sum(int a, int b, int c) {

System.out.println(a + b + c); }

}

**7. Các access modifier trong java, khi dùng nó thì quyền truy cập sẽ như thế nào?**

Có bốn loại access modifier trong Java đó là: public, protected, default và private. Tổng quan mỗi loại access modifier có ý nghĩa như sau:

• public: tất cả các class khác trong hay ngoài package và nội bộ class đều có thể truy cập đến các biến, các phương thức public của class này.

• protected: chỉ những class ở cùng package hoặc những class extends từ class này và nội bộ class mới có thể truy cập đến các biến, phương thức public của class này

• default: chỉ những class ở cùng package với class này và nội bộ class thì mới có thể truy cập đến các biến, các phương thức public của class này.

• private: chỉ những class nằm bên trong class này mới có quyền truy cập đến các biến, phương thức public của class này.