Phạm Ngọc Hải

- Nêu ra các tính chất quan trọng của hướng đối tượng  
- Access modifier trong java có những loại nào ? Nêu đặc điểm của từng loại  
- Phân biệt class và instance  
- Phân biệt **Abstract**và **Interface**, Nêu trường hợp sử dụng cụ thể. Nếu 2 interface hoặc 1 abstract và 1 interface có 1 function cùng tên, có thể cùng hoặc khác kiểu trả về cùng được kế thừa bởi một class, chuyện gì sẽ xảy ra?  
- Thế nào là **Overriding**và **Overloading**  
- Một function có access modifier là private or static có thể overriding được không?  
- Một phương thức final có thể kế thừa được không ?  
- Phân biệt hai từ khóa **This**và **Super**

**Trả lời**

Nêu ra các tính chất quan trọng của hướng đối tượng:

+ Kế thừa: Cho phép lớp con kế thừa thuộc tính và phương thức của lớp cha

+ Trừa tượng: Ẩn dấu cơ chế, các thức hoạt động, chi tiết bên trong. Chỉ cho phép người dùng biết đến chức năng của đối tượng.

+ Đa hình: Một phương thức có thể thực hiện theo nhiều cách khác nhau, thường thông qua việc ghi đè (overriding) hoặc nạp chồng (overloading)

+ Đóng gói: Ẩn dấu, bảo vệ dữ liệu bên trong một lớp, tránh truy cập trái phép, và chỉ có thể truy cập qua các phương thức công khai:

Access modifier trong java có những loại nào ? Nêu đặc điểm của từng loại

-Có 4 loại Access modifier: Public, private, protected, default

+Public: Có thể truy cập ở bất kì lớp nào

+Private: Chỉ có thể truy cập bên trong chính lớp đó

+Protected: Có thể truy cập các lớp bên trong gói, truy cập ngoài gói phải thông qua tính kế thừa

+Default: Có thể truy cập các lớp bên trong gói

Phân biệt class và instance

Class: Mô tả hành vi và trạng thái của đối tượng. Không chiếm không gian bộ nhớ

Instance: Là một đối tượng được tạo ra từ class. Chiếm không gian bộ nhớ

Phân biệt **Abstract**và **Interface**, Nêu trường hợp sử dụng cụ thể. Nếu 2 interface hoặc 1 abstract và 1 interface có 1 function cùng tên, có thể cùng hoặc khác kiểu trả về cùng được kế thừa bởi một class, chuyện gì sẽ xảy ra?

**Abstract Class**:

Là một lớp không thể được khởi tạo.

Có thể chứa cả phương thức trừu tượng (abstract method) và phương thức cụ thể (concrete method).

Có thể có biến thành viên(private, public,protectd,default), getter, setter và constructor.

Một lớp chỉ có thể kế thừa một abstract class.

Sử dụng khi các lớp con có quan hệ chặt chẽ với nhau(đặc điểm hành vi), cùng sử dụng phương thức chung đã định nghĩa sẵn ở abstract class .

**Interface**:

Biến là hằng số (static final).

Không có constructor

Một lớp có thể implements nhiều interface.

Sử dụng khi bạn muốn định nghĩa một tập hợp các phương thức mà các lớp không liên quan có thể thực hiện.

Nếu 2 interface hoặc 1 abstract và 1 interface có 1 function cùng tên: Vẫn có thể kế thừa trong cùng một lớp

+Cùng tên, kiểu trả về: Không vấn đề:

**+Cùng tên, khác kiểu trả về:** Lỗi biên dịch

Thế nào là **Overriding**và **Overloading**

**Overriding**:

Định nghĩa lại một phương thức trong lớp con mà đã được định nghĩa trong lớp cha.

Được sử dụng khi muốn thay đổi hành vi của phương thức trong lớp cha.

Phương thức trong lớp con phải có cùng tên, kiểu trả về và tham số với phương thức trong lớp cha.

Xảy ra ở các lớp con

**Overloading**:

Định nghĩa nhiều phương thức cùng tên nhưng khác tham số (số lượng tham số hoặc kiểu tham số) trong cùng một lớp.

Xảy ra trên cùng 1 lớp

- Một function có access modifier là private or static có thể overriding được không?

Không, một phương thức có **access modifier** là **private** hoặc **static** không thể bị **overriding**

Một phương thức final có thể kế thừa được không ?

**Một phương thức final không thể bị kế thừa**

Phân biệt hai từ khóa **This**và **Super**

Từ khóa **this**

this tham chiếu đến đối tượng hiện tại, được dùng để truy cập biến và phương thức của lớp hiện tại.

super tham chiếu đến lớp cha từ lớp con, được dùng để truy cập các thành viên của lớp cha hoặc gọi constructor của lớp cha