OOP

a. Các tính chất quan trọng của hướng đối tượng (OOP):

Encapsulation (Đóng gói): Bảo vệ dữ liệu bằng cách ẩn các thông tin bên trong và chỉ cho phép truy cập thông qua các phương thức công khai.

Inheritance (Kế thừa): Cho phép một lớp con kế thừa các thuộc tính và phương thức của lớp cha, giúp tái sử dụng mã nguồn.

Polymorphism (Đa hình): Cho phép một đối tượng có thể có nhiều hình thái khác nhau, phương thức có thể có nhiều cách thực hiện khác nhau tùy theo ngữ cảnh.

Abstraction (Trừu tượng hóa): Chỉ tập trung vào các chi tiết quan trọng, giấu đi các chi tiết phức tạp, giúp người dùng dễ hiểu hơn khi làm việc với hệ thống.

Ví dụ: Một lớp "Animal" có thuộc tính chung là move() và các lớp con như "Bird" sẽ fly(), "Fish" sẽ swim(), nhưng tất cả đều kế thừa phương thức move() từ lớp "Animal".

b. Các loại access modifier trong Java:

Private: Chỉ cho phép truy cập từ bên trong lớp hiện tại.

Default (Package-private): Nếu không khai báo rõ ràng, các phương thức hoặc biến sẽ được truy cập trong cùng một package.

Protected: Cho phép truy cập từ các lớp cùng package và các lớp con (kể cả khi nằm ở package khác).

Public: Có thể truy cập từ bất cứ đâu trong chương trình.

c. Phân biệt Class và Instance:

Class: Là bản thiết kế (blueprint) cho các đối tượng, định nghĩa thuộc tính và hành vi chung cho đối tượng.

Instance: Là đối tượng cụ thể của một class, là một phiên bản cụ thể được tạo ra từ class đó.

Ví dụ: Class "Car" định nghĩa thuộc tính như màu sắc, số bánh xe. Một instance của class "Car" có thể là "car1" với màu đỏ, 4 bánh xe.

d. Phân biệt Abstract và Interface:

Abstract Class: Lớp trừu tượng có thể chứa các phương thức có thân hàm và các phương thức trừu tượng (chưa có thân hàm). Một lớp con chỉ có thể kế thừa từ một abstract class.

Interface: Chỉ định nghĩa phương thức (không có thân hàm) và hằng số, lớp có thể triển khai nhiều interface cùng lúc.

Trường hợp sử dụng:

Sử dụng abstract khi muốn chia sẻ hành vi chung giữa các lớp con nhưng có thể tùy chỉnh chi tiết.

Sử dụng interface khi muốn định nghĩa các phương thức mà các lớp triển khai phải thực hiện, mà không quan tâm đến cách thực hiện.

Nếu 2 interface hoặc 1 abstract và 1 interface có cùng function tên nhưng khác kiểu trả về:

Nếu cùng kiểu trả về, class có thể kế thừa và ghi đè phương thức đó.

Nếu khác kiểu trả về, sẽ xảy ra lỗi biên dịch (compile-time error).

e. Phân biệt Overriding và Overloading:

Overriding: Ghi đè một phương thức của lớp cha trong lớp con để cung cấp một cách thực hiện khác.

Overloading: Một phương thức có cùng tên nhưng khác tham số (số lượng hoặc kiểu dữ liệu tham số).

Ví dụ:

Overloading: add(int a, int b) và add(double a, double b)

Overriding: Phương thức toString() của lớp Object được ghi đè trong các lớp con để trả về chuỗi mô tả cụ thể của đối tượng.

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

Private: Không thể overriding vì phương thức này không được truy cập từ lớp con.

Static: Không thể overriding vì nó là phương thức tĩnh, thuộc về class chứ không phải đối tượng.

g. Một phương thức final có thể kế thừa được không?

Không, một phương thức khai báo là final không thể bị ghi đè (overridden) bởi lớp con.

h. Phân biệt từ khóa this và super:

This: Tham chiếu đến đối tượng hiện tại của class.

Super: Tham chiếu đến lớp cha của đối tượng hiện tại, dùng để gọi phương thức hoặc truy cập biến của lớp cha.

Ví dụ:

this() dùng để gọi constructor khác của cùng class.

super() dùng để gọi constructor của lớp cha.