1. Nêu ra các tính chất quan trọng của hướng đối tượng

Tính đóng gói (encapsulation): Ẩn dấu thông tin, chỉ truy cập được thông qua các phương thức get/set

Tính kế thừa (inheritance): Kế thừa các thuộc tính và phương thức của lớp cha. Tái sử dụng code

Tính đa hình (polymorphism): Biểu hiện qua các hình thái khác nhau.

Tính trừu tượng hóa (abstraction): Che giấu các chi tiết chỉ cung cấp giao diện bên ngoài, được thực hiện bằng lớp abstract và interface

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

Default: chỉ truy cập từ các lớp cùng packge.

Public: ko giới hạn

Private: truy cập trong cùng class qua get/set

Protected: truy cập từ các lớp, lớp con trong cùng package

1. Phân biệt class và instance

Class là một bản thiết kế mô tả các thuộc tính và chức năng của đối tượng.

Instance là 1 đối tượng được tạo nên từ class.

1. 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 | Interface |
| Khai báo | Tường minh | Mặc định |
| Biến | Hỗ trợ các loại biến | Chỉ hỗ trợ biến final và static  Mặc định các biến là hằng số |
| Phương thức | Hỗ trợ tất cả các phương thức | Chỉ chứa các phương thức abstract, default, private, static  Không có constructor |
| Kế thừa | Abstract có thể implement nhiều interface | Interface chỉ có thể extends interface  Hỗ trợ đa kế thừa |
| Access modifiers |  | Mặc định là public |
| Mục đích sử dụng | Làm lớp chung cho các lớp có liên quan | Định ra hành động có thể dùng chung cho các lớp |

Nếu cùng tên nhưng khác kiểu trả về: chương trình báo lỗi biên dịch

Nếu cùng tên cùng kiểu trả về: lớp kế thừa sẽ bị ghi đè

1. Thế nào là **Overriding**và **Overloading**

Overriding: Xảy ra khi lớp con ghi đè một phương thức của lớp cha để cung cấp cách triển khai khác.

Yêu cầu: Tên phương thức, kiểu trả về và danh sách tham số phải giống hệt phương thức trong lớp cha.

Overloading: Xảy ra khi có nhiều phương thức cùng tên trong cùng một lớp, nhưng chúng khác nhau về tham số (số lượng hoặc kiểu tham số).

Yêu cầu: Tên phương thức phải giống nhau, nhưng danh sách tham số phải khác nhau.

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

Private: không thể overriding vì nó chỉ cho phép truy cập trong lớp mà nó được khởi tạo.

Static: Ko thể overriding vì nó thuộc về lớp, ko phải đối tượngS

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

Phương thức final có thể được kế thừa tuy nhiên ko thể thay đổi nội dung.

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

this:

Dùng để tham chiếu đến đối tượng hiện tại của lớp.

Có thể dùng để gọi các phương thức hoặc thuộc tính của lớp hiện tại.

Có thể được dùng để gọi constructor khác trong cùng một lớp (sử dụng this()).

Super

Dùng để tham chiếu đến **lớp cha (superclass)** của lớp hiện tại.

Có thể dùng để gọi các phương thức hoặc thuộc tính của **lớp cha**.

Dùng để gọi **constructor** của lớp cha (sử dụng super()).