**Câu hỏi ôn tập câu 1 OOP**

1. **Lập trình hướng đối tượng là gì?**

* Là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải, xây dựng chương trình.
* Lập trình hướng đối tượng là phương pháp lập trình dựa trên kiến trúc lớp (class) và đối tượng (object).

1. **Các đặc điểm quan trọng của lập trình hướng đối tượng? Các ưu điểm của OOP**

*Các đặc điểm quan trọng của OOP:*

* Trừu tượng hóa – Abstraction Cách nhìn khái quát hóa về một tập các đối tượng có chung các đặc điểm được quan tâm (và bỏ qua những chi tiết không cần thiết).
* Đóng gói – Encapsulation Nhóm những gì có liên quan với nhau vào làm một, để sau này có thể dùng một cái tên để gọi đến. Vd: các hàm/ thủ tục đóng gói các câu lệnh, các đối tượng đóng gói dữ liệu của chúng và các thủ tục có liên quan.
* Thừa kế - Inheritance cho phép một lớp D có được các thuộc tính và thao tác của lớp C, như thể các thuộc tính và thao tác đó đã được định nghĩa tại lớp D. Cho phép cài đặt nhiều quan hệ giữa các đối tượng: Đặc biệt hóa – Tổng quát hóa
* Đa hình – Polymorphism Là cơ chế cho phép một tên thao tác hoặc thuộc tính có thể được định nghĩa tại nhiều lớp và có thể có nhiều cài đặt khác nhau tại mỗi lớp trong các lớp đó.

*Các ưu điểm của OOP:*

* Nguyên lý kế thừa: tránh lặp, tái sử dụng.
* Nguyên lý đóng gói – che dấu thông tin: chương trình an toàn không bị thay đổi bới những đoạn chương trình khác
* Dễ mở rộng, nâng cấp
* Mô phỏng thế giới thực tốt hơn.

1. **Lớp đối tượng là gì? Đối tượng là gì?**

*Lớp đối tượng:*

* Các đối tượng có các đặc tính tương tự nhau được gom chung thành lớp đối tượng. Một lớp đối tượng đặc trưng bằng các thuộc tính, và các hành động (hành vi, thao tác).
* Thuộc tính (Attribute): Một thành phần của đối tượng, có giá trị nhất định cho mỗi đối tượng tại mỗi thời điểm trong hệ thống.
* Thao tác (Operation): Thể hiện hành vi của một đối tượng tác động qua lại với các đối tượng khác hoặc với chính nó.

*Đối tượng:*

* Một đối tượng là 1 thực thể bao gồm thuộc tính và hành động

Ví dụ đối tượng: Một người

Một người có các thuộc tính: tên, tuổi, địa chỉ, màu mắt…

Các hành động: đi, nói, thở…

1. **Phân biệt các phạm vi truy xuất private, protected, public?**

|  |  |  |  |
| --- | --- | --- | --- |
| Từ khoá dẫn xuất  Phạm vi truy cập | Private | Protected | Public |
| Private | X | X | X |
| Protected | Private | Protected | Protected |
| Public | Private | Protected | Public |

* Thành phần private ở lớp cha thì không truy xuất được ở lớp con
* Kế thừa public: Lớp con kế thừa public từ lớp cha thì các thành phần protected của lớp cha trở thành protected của lớp con, các thành phần public của lớp cha trở thành public của lớp con.
* Kế thừa private: Lớp con kế thừa private từ lớp cha thì các thành phần protected va public của lớp cha trở thành private của lớp con.
* Kế thừa protected: Lớp con kế thừa protected từ lớp cha thì các thành phần protected va public của lớp cha trở thành protected của lớp con.

1. **Constructor là gì? Constructor mặc định? Phân biệt Constructor mặc định và Constructor khác. Destructor là gì?**

*Constructor:*

* Phương thức thiết lập hay còn gọi là constructor, là một loại phương thức đặc biệt dùng để khởi tạo thể hiện của lớp.
* Bất kỳ một đối tượng nào được khai báo đều phải sử dụng một hàm thiết lập để khởi tạo các giá trị thành phần của đối tượng.
* Hàm thiết lập được khai báo giống như một phương thức với tên phương thức trùng với tên lớp và không có giá trị trả về (kể cả void).
* Constructor phải có thuộc tính public
* Constructor có thể được khai báo chồng như các hàm C++ thông thường khác
* Constructor có thể được khai báo với các tham số có giá trị ngầm định

*Constructor mặc định:*

* Constructor mặc định (default constructor) là constructor được gọi khi thể hiện được khai báo mà không có đối số nào được cung cấp

*Phân biệt:*

* Constructor mặc định (default constructor) là constructor được gọi khi thể hiện được khai báo mà không có đối số nào được cung cấp. Ngược lại, nếu tham số được cung cấp tại khai báo thể hiện, trình biên dịch sẽ gọi constructor khác (overload).
* Đối với constructor mặc định, nếu ta không cung cấp bất kỳ constructor nào, C++ sẽ tự sinh constructor mặc định là một phương thức rỗng.
* Tuy nhiên, nếu ta không định nghĩa constructor mặc định nhưng lại có các constructor khác, trình biên dịch sẽ báo lỗi không tìm thấy constructor mặc định nếu ta không cung cấp tham số khi tạo thể hiện.

*Destructor:*

* Phương thức hủy bỏ hay còn gọi là destructor, được gọi ngay trước khi một đối tượng bị thu hồi.
* Destructor thường được dùng để thực hiện việc dọn dẹp cần thiết trước khi một đối tượng bị hủy.
* Một lớp chỉ có duy nhất một Destructor.
* Cú pháp:

+ Phương thức Destructor có tên trùng tên với tên lớp và có dấu ~ đặt trước

+ Không có giá trị trả về

* Được tự động gọi thực hiện khi đối tượng hết phạm vi sử dụng.
* Destructor phải có thuộc tính public.

1. **Kế thừa là gì? Đặc tính? Ưu điểm? Phạm vi truy xuất?**

*Kế thừa:*

* Kế thừa là một đặc điểm của ngôn ngữ dùng để biểu diễn mối quan hệ đặc biệt hóa – tổng quát hóa giữa các lớp. Các lớp được trừu tượng hóa và được tổ chức thành một sơ đồ phân cấp lớp.
* Sự kế thừa là một mức cao hơn của trừu tượng hóa, cung cấp một cơ chế gom chung các lớp có liên quan với nhau thành một mức khái quát hóa đặc trưng cho toàn bộ các lớp nói trên.
* Các lớp với các đặc điểm tương tự nhau có thể được tổ chức thành một sơ đồ phân cấp kế thừa (cây kế thừa).
* Quan hệ “là 1”: Kế thừa được sử dụng thông dụng nhất để biểu diễn quan hệ “là 1”.

Một sinh viên là một người

Một hình tròn là một hình ellipse

Một tam giác là một đa giác

*Đặc tính:*

* Cho phép định nghĩa lớp mới từ lớp đã có.

+ Lớp mới gọi là lớp con (subclass) hay lớp dẫn xuất (derived class)

+ Lớp đã có gọi là lớp cha (superclass) hay lớp cơ sở (base class).

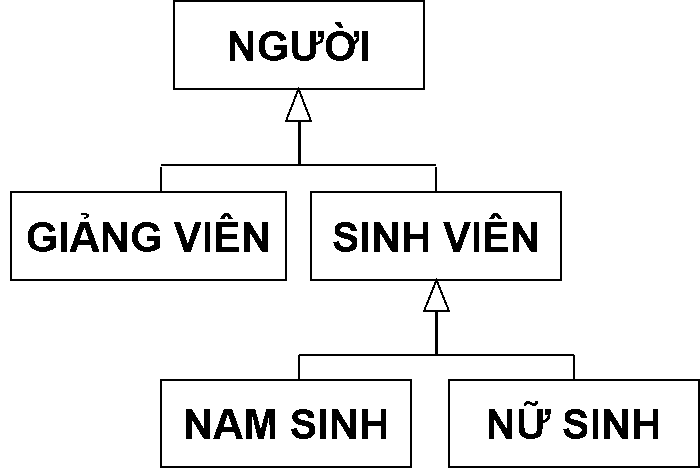
* Thừa kế cho phép:

+ Nhiều lớp có thể dẫn xuất từ một lớp cơ sở

+ Một lớp có thể là dẫn xuất của nhiều lớp cơ sở

*Ưu điểm:*

* Kế thừa cho phép xây dựng lớp mới từ lớp đã có.
* Kế thừa cho phép tổ chức các lớp chia sẻ mã chương trình chung, nhờ vậy có thể dễ dàng sửa chữa, nâng cấp hệ thống.
* Trong C++, kế thừa còn định nghĩa sự tương thích, nhờ đó ta có cơ chế chuyển kiểu tự động.



*Phạm vi truy vấn:*

|  |  |  |  |
| --- | --- | --- | --- |
| Từ khoá dẫn xuất  Phạm vi truy cập | Private | Protected | Public |
| Private | X | X | X |
| Protected | Private | Protected | Protected |
| Public | Private | Protected | Public |

* Thành phần private ở lớp cha thì không truy xuất được ở lớp con.
* Kế thừa public: Lớp con kế thừa public từ lớp cha thì các thành phần protected của lớp cha trở thành protected của lớp con, các thành phần public của lớp cha trở thành public của lớp con.
* Kế thừa private: Lớp con kế thừa private từ lớp cha thì các thành phần protected va public của lớp cha trở thành private của lớp con.
* Kế thừa protected: Lớp con kế thừa protected từ lớp cha thì các thành phần protected va public của lớp cha trở thành protected của lớp con.

1. **Đa hình là gì?**

* Đa hình là hiện tượng các đối tượng thuộc các lớp khác nhau có khả năng hiểu cùng một thông điệp theo các cách khác nhau.

Ví dụ: Nhận được cùng một thông điệp “nhảy”, một con kangaroo và một con cóc nhảy theo hai kiểu khác nhau: chúng cùng có hành vi “nhảy” nhưng các hành vi này có nội dung khác nhau.



* Tính đa hình xuất hiện khi có sự kế thừa giữa các lớp.
* Có những phương thức tổng quát cho mọi lớp dẫn xuất nên có mặt ở lớp cơ sở nhưng nội dung của nó chỉ được xác định ở các lớp dẫn xuất cụ thể.

Ví dụ, Phương thức tính diện tích của lớp hình, hình tam giác, tứ giác,…

1. **Hàm thuần ảo là gì? Lớp trừu tượng là gì?**

* Hàm thuần ảo (Phương thức ảo thuần tuý) có ý nghĩa cho việc tổ chức sơ đồ phân cấp các lớp, nó đóng vai trò chừa sẵn chỗ trống cho các lớp con điền vào với phiên bản phù hợp. Phương thức ảo thuần tuý là phương thức ảo không có nội dung, được khai báo với từ khóa virtual và được gán giá trị = 0
* Khi lớp có phương thức ảo thuần tuý, lớp trở thành lớp cơ sở trừu tượng. Lớp cơ sở trừu tượng không có đối tượng nào thuộc chính nó.

class Shape //Abstract

{

public :

//Pure virtual Function

virtual void draw() = 0;

}

Trong ví dụ trên, các hàm thành phần trong lớp Shape là phương thức ảo thuần tuý và lớp Shape là lớp cơ sở trừu tượng. Nó bảo đảm không thể tạo được đối tượng thuộc lớp Shape.

1. **Phân biệt khái niệm overload (tải chồng) và override (ghi đè)**

|  |  |  |
| --- | --- | --- |
|  | **Override** | **Overload** |
| Khái niệm | Là một tính năng cho phép một lớp con hoặc lớp con cung cấp một triển khai cụ thể của một phương thức đã được cung cấp bởi một trong các lớp siêu hoặc các lớp cha của nó. Nói cách khác, nếu lớp con cung cấp trình triển khai cụ thể của phương thức mà đã được cung cấp bởi một trong các lớp cha của nó, thì đó là ghi đè phương thức. | Nạp chồng phương thức đơn giản là có vài phương thức trùng tên nhưng khác nhau về đối số. Cài chồng phương thức cho phép ta tạo nhiều phiên bản của một phương thức, mỗi phiên bản chấp nhận một danh sách đối số khác nhau, nhằm tạo thuận lợi cho việc gọi phương thức. |
| Hành vi | Thay đổi hành vi hiện tại của phương thức. | Thêm hoặc mở rộng cho hành vi của phương thức. |
| Đa hình | Thể hiện tính đa hình tại run time. | Thể hiện tính đa hình tại compile time. |
| Danh sách tham số | Danh sách tham số phải giống nhau. | Danh sách tham số khác nhau (số lượng, thứ tự, kiểu dữ liệu) |
| Giá trị trả về | Kiểu trả về bắt buộc phải giống nhau. | Kiểu trả về có thể khác nhau. |
| Phạm vi | Xảy ra giữa 2 class có quan hệ kế thừa | Xảy ra trong phạm vi cùng 1 class. |

1. **Cho biết ý nghĩa và mục đích của các hàm get/set trong một lớp**

*Ý nghĩa:*

* Phương thức get/set chỉ có nhiệm vụ cho ta đọc/ghi giá trị cho các thành viên dữ liệu.

*Mục đích:*

* Ngoài việc bảo vệ các nguyên tắc đóng gói, ta cần kiểm tra xem giá trị mới cho thành viên dữ liệu có hợp lệ hay không.
* Sử dụng phương thức truy vấn cho phép ta thực hiện việc kiểm tra trước khi thực sự thay đổi giá trị của thành viên.
* Chỉ cho phép các dữ liệu có thể truy vấn hay thay đổi mới được truy cập đến.