**LÝ THUYẾT OOP**

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

* Lập trình hướng đối tượng là phương pháp lập trình lấy đối tượng làm nền tảng để xây dựng thuật giải xây dựng chương trình.

1. ***Lớp, đối tượng là gì? Phân biệt lớp và đối tượng.***

* Một đối tượng là một thực thể bao gồm thuộc tính và phương thức.
* 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 phương thức.
* Thuộc tính: một thành phần của đối tượng, có giá trị nhất định cho một đối tượng tại mỗi thời điểm //trong hệ thống.
* Phương thức: thể hiện hành vi của một đối tượng tác động qua lại với các đối tượng khác hoặc với chính nó.

|  |  |
| --- | --- |
| **Lớp** | **Đối Tượng** |
| Là một template chung cho tất cả các đối tượng | Là một thể hiện của lớp |
| Chỉ được khai báo một lần | Có thể có nhiều đối tượng thuộc lớp |
| Khi khai báo thì không được cấp phát vùng nhớ | Khi khai báo thì được cấp phát vùng nhớ |
| Là một nhóm đối tượng giống nhau | Là những đối tượng cụ thể có thật |

1. ***Trình bày các đặc điểm quan trọng của Lập trình hướng đối tượng.***

* **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.

1. ***Trình bày ý nghĩa và mục đích của hàm get/set trong một lớp.***

* Set: Vì dữ liệu được khai báo trong nhãn private chỉ được truy xuất trong phạm vi lớp đó, nếu muốn thay đổi dữ liệu đó ở ngoài phạm vi lớp thì phải dùng hàm set, ngoài ra còn để kiểm soát thông tin nhập vào biến đó có phù hợp với đề bài hay không.
* Get: Tương tự như set, muốn lấy dữ liệu được khai báo trong nhãn private thì chỉ được truy xuất trong phạm vi lớp, nên nếu muốn truy xuất dữ liệu đó ở ngoài phạm vi lớp phải dùng hàm get để lấy dữ liệu ra.

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

* **Private:** mọi thành phần được khai báo trong private chỉ được truy xuất bên trong phạm vi lớp và hàm bạn, lớp bạn.
* **Protected:** mọi thành phần được khai báo trong protected chỉ được truy xuất bên trong phạm vi lớp và có thể truy cập từ lớp dẫn xuất, hàm bạn, lớp bạn.
* **Public:** mọi thành phần được khai báo trong public đều được truy xuất ở bất kỳ hàm nào.

1. ***Constructor – destructor là gì? Phân biệt constructor mặc định – constructor khác. Phân biệt constructor – destructor.***

**Constructor:**

* Định nghĩa: là một loại phương thức đặc biệt dùng để khởi tạo thể hiện (đối tượng) của lớp.
* Công dụng: dùng để thiết lập khởi tạo các giá trị thành phần của đối tượng.
* Có 3 phương thức thiết lập: Thiết lập mặc định (không có tham số truyền vào), thiết lập sao chép (tham số truyền vào là một đối tượng thuộc về class), thiết lập có tham số đầu vào.
* Đặc điểm:
* Có tên trùng với tên lớp.
* Không có kiểu trả về.
* Có thể không có hoặc có tham số đầu vào.
* Trong một lớp có thể có một hay nhiều constructor.
* Constructor phải có thuộc tính public.
* Constructor mặc định: được gọi khi đối tượng được khai báo mà không có đối số nào được cung cấp.

**Destructor**

* Định nghĩa: Destructor là hàm hủy bỏ được gọi ngay trước khi một đối tượng bị thu hồi, dùng để dọn dẹp cần thiết trước khi một đối tượng bị hủy.
* Destructor phải được khai báo ở nhãn public và chỉ có duy nhất 1 destructor trong class.
* Công dụng: được tự động gọi khi đối tượng hết phạm vi sử dụng.
* Đặc điểm của destructor:
  + Có tên trùng với tên class với nhưng có dấu ~ đặt trước.
  + Không có kiểu trả về.
  + Không có tham số đầu vào.
  + Trong một lớp có duy nhất 1 destructor.

**Phân biệt constructor mặc định và các constructor khác:**

|  |  |
| --- | --- |
| Constructor mặc định | Constructor sao chép hoặc constructor có tham số đầu vào |
| * Là hàm dùng để khởi tạo giá trị cho đối tượng * Được khai báo như một phương thức có tên trùng tên lớp * Không có kiểu dữ liệu trả về * Khai báo ở nhãn public | |
| Không có tham số truyền vào hoặc tất cả tham số điều có giá trị mặc nhiên. | Có một hoặc nhiều tham số đầu vào. |
| Khi khởi tạo đối tượng thì không cần truyền tham số vào | Khi khai báo đối tượng thì phải truyền tham số nếu không trình biên dịch sẽ dùng constructor mặc định. |
| Khởi tạo các đối tượng có cùng dữ liệu được khai báo trong constructor mặc định | Khởi tạo những đối tượng khác dữ liệu tùy thuộc vào tham số mà ta truyền vào |
| Chỉ có 1 constructor mặc định | Có thể có nhiều constructor có tham số |

\*\* Nếu không khai báo constructor thì trình biên dịch sẽ tự động sinh constructor mặc định, nhưng một khi đã khai báo các constructor khác rồi mà không khai báo constructor mặc định thì chương trình sẽ báo lỗi.

**Phân biệt construcor và destructor :**

|  |  |
| --- | --- |
| Constructor | Destructor |
| * Không có kiểu dữ liệu trả về * Khai báo ở nhãn public * Được gọi tự động | |
| Tên trùng tên lớp | Tên trùng tên lớp nhưng thêm ~ ở trước |
| Khởi tạo giá trị cho đối tượng | Hủy đối tượng |
| Có nhiều constructor | Chỉ có 1 destructor |

1. ***Khái niệm về sự kế thừa và những ưu điểm của nó. Cho ví dụ minh họa.***

* Kế thừa là một tính chất đặc trưng của lập trình hướng đối tượng. 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. Nó có nghĩa là một class thừa hưởng lại tất cả các thuộc tính, phương thức của class mà nó kế thừa.
* 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).
* Ưu điểm của kế thừa:
* Cho phép xây dựng 1 lớp mới từ lớp đã có.
* Cho phép chia sẻ các thông tin chung nhằm tái sử dụng và đồng thời giúp ta dễ dàng nâng cấp, dễ dàng bảo trì.
* Định nghĩa sự tương thích giữa các lớp, nhờ đó ta có thể chuyển kiểu tự động.

**Ví dụ:** Giả sử ta có lớp TamGiac chứa thông tin toạ độ của 3 điểm A, B, C. Ta biết rằng tam giác cân là 1 trường hợp đặc biệt của tam giác (ngược lại tam giác là trường hợp tổng quát của tam giác cân). Từ đó ta có thể cho lớp TamGiacCan kế thừa lại lớp TamGiac để có thể sử dụng lại các thông tin như toạ độ 3 điểm A, B, C mà không cần phải khai báo.

1. ***Phân biệt phạm vi kế thừa.***

|  |  |  |  |
| --- | --- | --- | --- |
| Từ khóa  Phạm vi dẫn xuất  truy xuất | 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 và 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. ***Phương thức ảo – thuần ảo là gì? Khái niệm lớp cơ sở trừu tượng?***

* Phương thức ảo là cách thể hiện tính đa hình trong C++. Các phương thức ở lớp cơ sở có tính đa hình phải được định nghĩa là một phương thức ảo.
* Phương thức ảo chỉ hoạt động thông qua con trỏ.
* Muốn hàm trở thành phương thức ảo có 2 cách:
* Thêm từ khóa virtual vào trước khai báo hàm. Ví dụ: virtual void Nhap();
* Phương thức tương ứng ở lớp cơ sở đã là phương thức ảo.
* Phương thức ảo chỉ hoạt động nếu phương thức ở lớp cơ sở và lớp con có nghi thức giao tiếp giống hệt nhau.
* Nếu ở lớp con không định nghĩa lại phương thức ảo thì sẽ gọi phương thức ở lớp cơ sở (gần nhất có định nghĩa)

1. ***Trình bày khái niệm đa hình.***

* Tính đa hình là hiện tượng các đối tượng thuộc các lớp khác nhau có thể hiểu cùng 1 thông điệp theo các cách khác nhau.

**Ví dụ:** có 3 con vật: chó, mèo, lợn. Khi ta bảo “kêu” thì con chó sẽ kêu gâu gâu, con mèo sẽ kêu meo meo và con heo sẽ kêu ẹt ẹt. Cả 3 con vật có thể hiểu cùng một thông điệp là “kêu” nhưng thực hiện theo các cách khác nhau.

1. ***Phân biệt khái niệm override – overload.***

|  | **Override** | **Overload** |
| --- | --- | --- |
| Khái niệm | là một tính năng cho phép một lớp con cung cấp một triển khai cụ thể của phương thức đã được cung cấp bởi một trong các lớp cha của nó. | Nạp chồng phương thức các phương thức trùng tên nhưng khác nhau về đối số. Cài chồng phương thức cho phép ta tạo nhiều phiên bản, 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 có thể khác nhau. |
| Quyền truy cập | Phương thức ghi đè ở lớp con phải có quyền truy cập bằng hoặc lớn hơn phương thức được ghi đè ở lớp cha. | Các phương thức nạp chồng có thể có quyền truy cập khác nhau. |
| 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. |