Polymorphism – Đa hình

CSC10003 – Phương pháp lập trình hướng đối tượng

Nội dung thực hành lần này sẽ tập trung phân tích và hướng dẫn một số vấn đề về tính đa hình trong lập trình hướng đối tượng

1. Tính chất đa hình trong lập trình hướng đối tượng

a. Liên kết tĩnh, liên kết động và phương thức ảo

- Trong quan hệ kế thừa, bên cạnh việc tạo trực tiếp một đối tượng cụ thể, ta có thể sử dụng biến con trỏ để lưu trữ đối tượng cần tạo. Cơ chế trong quan hệ kế thừa cho phép tạo một **biến con trỏ có kiểu dữ liệu là lớp cha, trỏ đến đối tượng kiểu lớp con.**
  - Xem ví dụ sau, ta có quan hệ giữa heo, mèo và động vật là quan hệ kế thừa, trong đó con heo, con mèo kế thừa từ lớp động vật.

```
class DongVat
{
	//Các thuộc tính...
public:
	void TiengKeu()
	{
		cout << "Chua biet dong vat gi!!" << endl;
	}
};

class ConHeo : public DongVat
{
	//Các thuộc tính riêng của con heo
public:
	void TiengKeu()
	{
		cout << "Ut Ut!!" << endl;
	}
};

class ConMeo : public DongVat
{
	//Các thuộc tính riêng của con mèo
public:
	void TiengKeu()
	{
		cout << "Meo meo!!" << endl;
	}
};
```

```
void main()
{
	ConMeo conmeo1;
	DongVat* conmeo2;
	//Con trỏ có kiểu của lớp cha ĐƯỢC
	//tham chiếu đến đối tượng lớp con
	conmeo2 = &conmeo1;
	conmeo2->TiengKeu();

	//Con trỏ có kiểu của lớp cha ĐƯỢC
	//tham chiếu đến đối tượng lớp con
	DongVat* conheo1 = new ConHeo();
	conheo1->TiengKeu();

	delete conheo1;
}
```

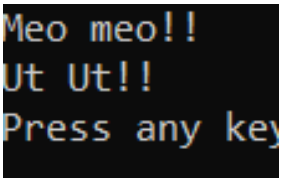
```
Chua biet dong vat gi!!
Chua biet dong vat gi!!
Press any key to continue . . .
```

- Dựa trên tính chất cho phép trỏ đến những lớp con trong quan hệ kế thừa, ta có thể sử dụng nạp chồng hàm (function overloading) để cài đặt mã nguồn thao tác trên đối tượng của lớp cha vẫn có thể chạy được cho đối tượng của lớp con, tuy nhiên cần kết hợp với tính đa hình.
  - Lưu ý : chỉ cho phép biến con trỏ có kiểu dữ liệu lớp cha trỏ đến đối tượng lớp con chứ không được phép làm ngược lại : **biến con trỏ có kiểu dữ liệu lớp con không được phép trỏ đến đối tượng lớp cha**
- Trong ví dụ trên, ta có thể thấy khi biến con trỏ mang kiểu dữ liệu của lớp cha (lớp Động Vật) trỏ đến đối tượng lớp con (con mèo, con heo), khi gọi phương thức nạp chồng (tiếng kêu) thì đều gọi lại phương thức này ở lớp cha !!! Đây chính là **liên kết tĩnh**.
- Để có thể gọi được đúng phương thức của đối tượng mà con trỏ đang trỏ đến, cần phải cài đặt **cơ chế liên kết động cho phương thức ở lớp cha**, bằng cách thêm từ khóa **virtual**
  - Phương thức nào được cài đặt liên kết động bằng cách thêm từ khóa virtual được gọi là **phương thức ảo hay hàm ảo**
- Phương thức ảo hay hàm ảo mang **tính chất đa hình** :
  - Sử dụng con trỏ có kiểu thuộc lớp cha :
    - Trỏ đến đối tượng thuộc lớp con nào thì gọi hàm hay phương thức của lớp con đó tương ứng
- Để cài đặt một phương thức hay thuộc tính mang tính đa hình :
  - Cần phải có quan hệ kế thừa, cần phải sử dụng biến con trỏ với kiểu ở lớp cha, trỏ đến đối tượng lớp con
  - Xác định phương thức nào cần đa hình, thêm từ khóa virtual cho phương thức đó ở lớp cha
  - Lớp con cài đặt nạp chồng phương thức đó (không thêm virtual ở phương thức của lớp con, có thể thêm nếu kế thừa nhiều tầng !!!)

```
class DongVat
{
	//Các thuộc tính...
public:
	virtual void TiengKeu() //Phương thức ảo
	{
		cout << "Chua biet dong vat gi!!" << endl;
	}
};

void main()
{
	ConMeo conmeo1;
	DongVat* conmeo2;
	//Con trỏ có kiểu của lớp cha ĐƯỢC
	//tham chiếu đến đối tượng lớp con
	conmeo2 = &conmeo1;
	conmeo2->TiengKeu(); //Đa hình

	//Con trỏ có kiểu của lớp cha ĐƯỢC
	//tham chiếu đến đối tượng lớp con
	DongVat* conheo1 = new ConHeo();
	conheo1->TiengKeu(); //Đa hình
	delete conheo1;
}
```



- Cài đặt tính kế thừa kết hợp với đa hình giúp chương trình viết ngắn hơn, mang tính tổng quát đồng thời có khả năng mở rộng, tùy biến linh hoạt
- Xem xét ví dụ về khả năng mở rộng và viết mã nguồn ngắn hơn :
  - Giả sử ta cần cài đạt hàm để phát ra tiếng kêu của bất kỳ động vật nào truyền vào
    - Nếu không cài đặt đa hình : cần cài thêm xử lý riêng cho mỗi lớp con thêm mới vào do chưa biết cụ thể động vật truyền vào là động vật gì.
    - Nếu có cài đặt đa hình : không cần cài xử lý riêng cho mỗi lớp con, lớp con khi thêm mới đã tự cài đặt sẵn nạp chồng phương thức đa hình.
  - Minh họa mã nguồn cho cả 2 cách cài đặt :

```
//KHÔNG sử dụng tính đa hình
void PhatRaTiengKeu(DongVat dv, string loaidv)
{
	if (loaidv == "ConHeo")
	{
		//Xử lý cho trường hợp con heo
	}
	else if (loaidv == "ConMeo")
	{
		//Xử lý cho trường hợp con mèo
	}
	//Nếu có thêm con khác:
	//Phải cài lớp con, phải viết thêm xử lý cho mỗi lớp con
	//else if (loaidv == "ConVit")
	//.... con gà, con chó, .....
}

//CÓ sử dụng tính đa hình (cài đặt virtual)
void PhatRaTiengKeu(DongVat *dv)
{
	//Tính đa hình tổng quát
	//Thêm bất kỳ lớp con nào cũng không cần viết xử lý thêm
	dv->TiengKeu();
}
```

b. *Phương thức thuần ảo, hàm thuần ảo (pure virtual) và lớp trừu tượng (abstract class)*

- Khi cài đặt kế thừa và đa hình, thường có xu hướng đưa những cái chung lên ở lớp cha và cài đặt xử lý linh hoạt ở lớp con. Do đó, lớp cha thường sẽ không có ý nghĩa ứng dụng trong thực tế, vậy không nên tạo ra đối tượng của lớp cha.
  - Lớp con heo, con mèo đều kế thừa từ lớp động vật. Ta không thể tạo đối tượng cho lớp động vật do lớp này mang ý nghĩa trừu tượng (chưa xác định được động vật này tên gọi là gì, có tiếng kêu như thế nào ?) mà phải tạo đối tượng cụ thể là con heo hay con mèo.
  - Cách xử lý tốt hơn: ở lớp cha phương thức ảo chỉ có khai báo, không có phần cài đặt
- **Các phương thức ảo chỉ có khai báo và không có phần cài đặt được gọi là phương thức thuần ảo (hay hàm thuần ảo – pure virtual method), cài đặt bằng việc gán virtual = 0**
- **Lớp nào có chứa phương thức thuần ảo được gọi là lớp trừu tượng.**
- **Các lớp con kế thừa từ lớp trừu tượng bắt buộc phải cài lại toàn bộ các phương thức thuần ảo có khai báo ở lớp cha**
- **Quy ước UML : các phương thức đa hình đều được viết chữ in nghiêng**

```
class DongVat //Lớp cha chính là lớp trừu tượng do có 1 phương
thức thuần ảo
{
	//Các thuộc tính...
public:
	//Khai báo phương thức thuần ảo
	virtual void TiengKeu() = 0;
};

class ConHeo : public DongVat
{
public:
	//Lớp con kế thừa bắt buộc phải cài lại phương thức thuần ảo
	void TiengKeu()
	{
		cout << "Ut Ut!!" << endl;
	}
};

class ConMeo : public DongVat
{
	//Các thuộc tính riêng của con mèo
public:
	//Lớp con kế thừa bắt buộc phải cài lại phương thức thuần ảo
	void TiengKeu()
	{
		cout << "Meo meo!!" << endl;
	}
```

c. *Phương thức hủy ảo (hàm hủy ảo – virtual destructor)*

- Do sử dụng tính chất đa hình, cần phải sử dụng con trỏ có kiểu thuộc lớp cha, trỏ đến đối tượng thuộc lớp con nào thì gọi hàm hay phương thức của lớp con đó tương ứng.
- Việc khai báo con trỏ ở lớp cha trỏ đến đối tượng ở lớp con, khi được khai báo, phương thức tạo lập ở lớp cha sẽ được gọi trước, sau đó đến phương thức tạo lập ở lớp con.
- Tuy nhiên, phương thức hủy chỉ được gọi theo đối tượng của lớp cha, do biến con trỏ ban đầu tạo lập là biến con trỏ trỏ đến kiểu dữ liệu của lớp cha
  - **Để gọi được phương thức hủy của lớp con kế thừa, cần cài đặt phương thức hủy ở lớp cha là phương thức ảo, được gọi là phương thức hủy ảo.**
- Trong trường hợp lớp con có biến kiểu con trỏ, nếu không cài đặt phương thức hủy ảo ở lớp cha, lớp con không được hủy, dẫn đến việc không gọi delete con trỏ được, chương trình sẽ bị **Memory Leak!!!**
- Ví dụ minh họa sau:

```
class DongVat
{
	//Các thuộc tính...
public:
	//Cài đặt phương thức hủy ảo để được gọi ở lớp con
	virtual ~DongVat()
	{
		cout << "Dong vat bi huy!!" << endl;
	}
};
class ConHeo : public DongVat
{
	int *cannang;
public:
	~ConHeo()
	{
		delete cannang;
		cout << "Con heo bi huy!!" << endl;
	}
};

void main()
{
	//Constructor của lớp cha được gọi trước
	//Constructor của lớp con được gọi sau
	DongVat* conheo1 = new ConHeo();
	conheo1->TiengKeu(); //Đa hình

	//Chỉ phương thức hủy của lớp cha được gọi
	//Nguyên nhân: biến con trỏ khai báo ban đầu là kiểu lớp cha
	//Cần cài đặt hủy ảo ở lớp cha để gọi hủy ở lớp con trước,
rồi đến hủy lớp cha
	delete conheo1;
}
```

d. Bài tập thực hành

Bài tập 1: Dựa trên bài tập 1 của tuần 7 – kế thừa, sinh viên hãy cài đặt lại bài tập sử dụng kế thừa và đa hình với thông tin bổ sung sau:

Biết rằng trường tuyển thêm nhân sự về vị trí về thực tập sinh về lĩnh vực nghiên cứu, trong đó thực tập sinh sẽ được định hướng theo con đường của nghiên cứu viên để tham gia vào các dự án nghiên cứu trong tương lai. Do chưa có kinh nghiệm, thực tập sinh không được phép tham gia vào các dự án nghiên cứu hiện tại của các nghiên cứu viên mà chỉ được tham gia vào dự án nghiên cứu dự bị dành riêng cho thực tập. Khi tuyển vào, thực tập sinh sẽ có thời hạn thực tập được tính bằng

tháng và được phát lương, với lương thực tập sinh mỗi tháng sẽ bằng 10% tổng số dự án nghiên cứu dự bị * 10000. Dự án nghiên cứu dự bị có mã dự án bắt đầu bằng "R" và mã dự án có ràng buộc tương tự như mã dự án của nghiên cứu viên và chuyên viên.
Dựa trên thông tin bổ sung, sinh viên khai báo danh sách con trỏ các nhân sự hiện có của trường đại học (kiểu vector), áp dụng đa hình cài đặt thêm các thao tác:

- Tính tổng lương cần trả cho tất cả nhân sự của trường đại học
- Nhập và xuất thông tin tùy theo loại nhân sự tương ứng
- Xuất ra lương của nhân sự có lương cao nhất trường

Sinh viên vẽ sơ đồ lớp UML cho bài tập này.

Bài tập 2: Viết chương trình C++ áp dụng kế thừa và đa hình cài đặt mã nguồn, vẽ sơ đồ UML dựa trên mô tả của yêu cầu sau:

Bạn đang xây dựng một ứng dụng nghe nhạc trong đó có đầy đủ những tính năng cơ bản như chơi nhạc, lưu trữ nhạc thành playlist và hiển thị lyric nhạc. Để có được dữ liệu về các bài hát, bạn sẽ đăng ký lấy từ nguồn của những trang nghe nhạc uy tín hiện nay. Các trang nghe nhạc cho phép lấy thông tin về tên bài hát, lời bài hát, tên ca sỹ, thể loại nhạc, năm sáng tác và lượt nghe hiện tại trên trang nghe nhạc đó. Nhằm để đảm bảo quyền lợi của ca sỹ, những năm gần đây, các ca sỹ hợp tác ký kết độc quyền ra mắt nhạc mới chỉ trên một số trang nghe nhạc nhất định, đưa vào yếu tố bản quyền nhằm nâng cao giá trị của sản phẩm âm nhạc. Do đó, bài hát khi lấy về được phân loại ra gồm bài hát phổ thông và bài hát có bản quyền. Bài hát có bản quyền bên cạnh những thông tin mà trang nghe nhạc cung cấp, còn được biết thêm về giá trị bản quyền, yêu cầu bất kỳ người nào nghe nhạc phải đều trả số tiền tương ứng ghi trong giá trị bản quyền. Ngoài ra, các trang nghe nhạc còn cung cấp thêm thông tin về thể loại nhạc với 3 thể loại tương ứng: nhạc Việt Nam, nhạc Âu Mỹ và nhạc Hàn Quốc. Hệ thống sẽ có một danh sách quản lý tất cả các bài hát hiện tại, cho phép nhập và xuất thông tin bài hát qua console, đồng thời hiển thị bảng xếp hạng top 5 bài hát dựa trên lượt nghe tương ứng.
Để dễ quản lý và phục vụ mục đích cá nhân hóa, bạn xây dựng thêm hệ thống tài khoản để quản lý nhạc và để giải quyết các vấn đề liên quan đến thanh toán, lợi nhuận. Hệ thống cho phép đăng nhập vào tài khoản thông qua tên đăng nhập và mật khẩu. Mật khẩu cần phải có ít nhất 8 ký tự và không được chứa ký tự khoảng cách. Nếu chưa có tài khoản, người dùng có thể tạo một tài khoản thường hoặc tài khoản vip. Tài khoản vip khi nghe nhạc bản quyền sẽ được giảm 50% số tiền giá trị bản quyền, đồng thời khi đăng ký cho phép người dùng chọn lựa thời hạn duy trì (tính bằng tháng) và có mức phí duy trì cho mỗi tháng. Ngoài ra, tài khoản vip còn có chức năng gợi ý bài hát có số lượt nghe nhiều nhất dựa trên thể loại nhạc mà người dùng yêu cầu. Khi đến hạn, tài khoản vip yêu cầu người dùng gia hạn, hệ thống sẽ hỗ trợ gia hạn bằng cách nhập số tiền gia hạn tương ứng và tính toán thời gian gia hạn dựa trên số tiền nhập và mức phí duy trì.
Người dùng khi sử dụng tài khoản nghe nhạc có thể lưu trữ lại các bài nhạc mà mình muốn nghe thành một playlist. Hệ thống cần hỗ trợ thêm và xóa bài nhạc vào playlist, đồng thời developer có thể nhập và xuất thông tin của bài nhạc, của playlist ra màn hình console tương ứng. Đây là toàn bộ mô tả của một hệ thống ứng dụng nghe nhạc cần thiết.