UIT Trường Đại học Công nghệ Thông tin | Khoa Khoa học và Kỹ thuật Thông tin

BAN HỌC TẬP

BUỔI 2

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

TÍNH KẾ THỪA

INHERITANCE

TRAINER | ĐỖ NHẬT TÂN
KHDL2021

QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Giữa các lớp đối tượng có những loại quan hệ

- Quan hệ một một (1-1)
- Quan hệ một nhiều (1-n)
- Quan hệ nhiều nhiều (n-n)
- Quan hệ đặc biệt hóa, tổng quát hóa

QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Quan hệ một-một (1-1)

- Quan hệ một-một: Một đối tượng thuộc lớp này quan hệ với một đối tượng thuộc lớp kia và một đối tượng thuộc lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
- Kí hiệu:

QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Quan hệ một-một (1-1)

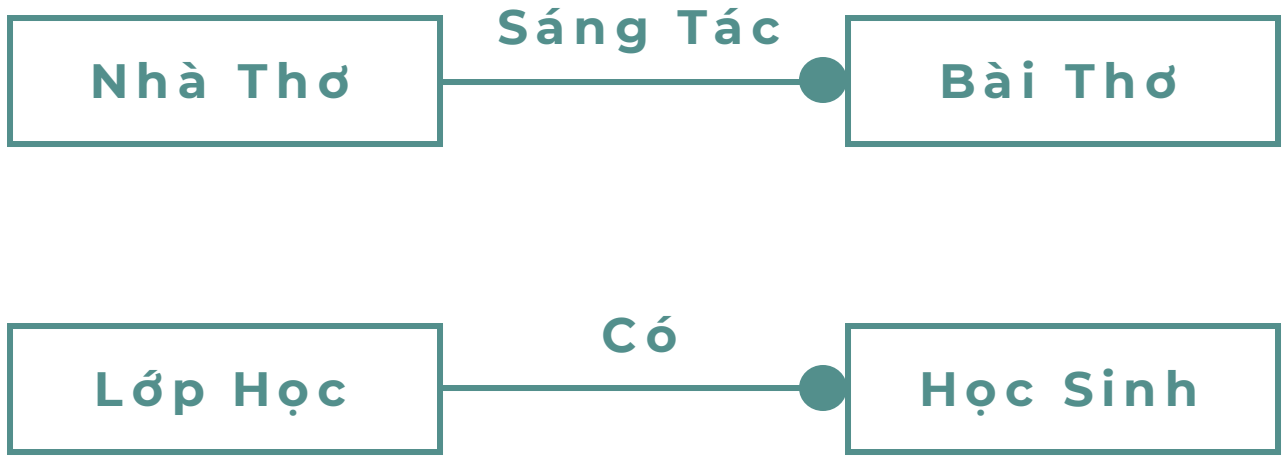
QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Quan hệ một-nhiều (1-n)

- Quan hệ một-nhiều: Một đối tượng thuộc lớp này quan hệ với nhiều đối tượng thuộc lớp kia và một đối tượng lớp kia có quan hệ duy nhất với một đối tượng thuộc lớp này.
- Kí hiệu:

QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Quan hệ một-nhều (1-n)



QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Quan hệ nhiều-nhiều (n-n)

- Quan hệ nhiều-nhiều: Một đối tượng lớp này có quan hệ với nhiều đối tượng lớp kia và một đối tượng lớp kia cũng có quan hệ với nhiều đối tượng lớp này.

- Kí hiệu:

QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Quan hệ nhiều-nhiều (n-n)

QUAN HỆ GIỮA CÁC LỚP ĐỐI TƯỢNG

Quan hệ Đặc biệt hóa – Tổng quát hóa

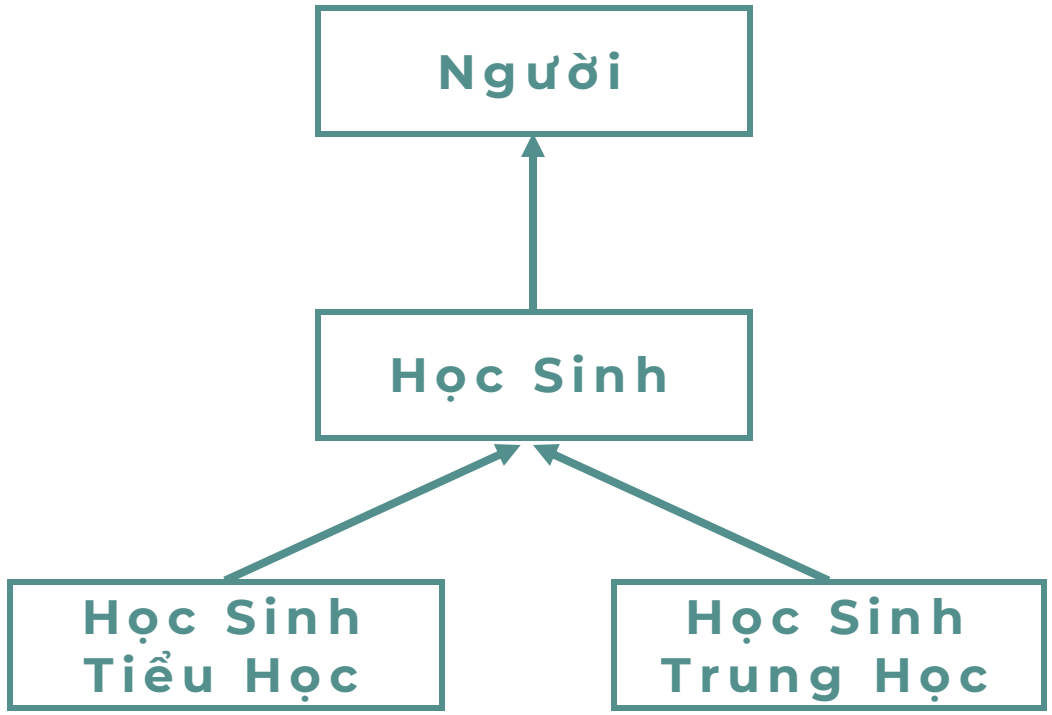
- Quan hệ đặc biệt hóa – tổng quát hóa: Lớp đối tượng này là trường hợp đặc biệt của lớp đối tượng kia và lớp đối tượng kia là trường hợp tổng quát của lớp đối tượng này.
- Kí hiệu:

KẾ THỪA

KẾ THỪA

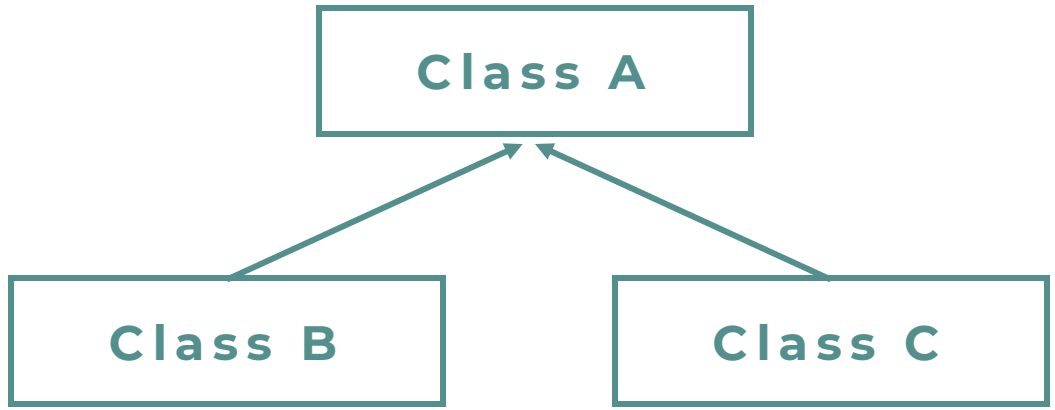
- Biểu diễn mối quan hệ đặc biệt hóa – tổng quát hóa.
- Các lớp được trừu tượng hóa và được tổ chức thành một sơ đồ phân cấp lớp.
- Kế thừa là mức cao hơn của trừu tượng hóa, cung cấp cơ chế gom chung các lớp có liên quan thành một mức khái quát hóa đặc trưng cho toàn bộ các lớp nói trên.

KẾ THỪA



ĐẶC TÍNH

- Cho phép định nghĩa lớp mới từ lớp đã có
  - Lớp đã có gọi là Lớp Cơ Sở (Base Class) hoặc Lớp Cha (Superclass).
  - Lớp mới gọi là Lớp Dẫn Xuất (Derived Class) hoặc Lớp Con (Subclass).



CÚ PHÁP

```
class BaseClass {
    // Thành Phần của Lớp Cơ Sở
};

class DerivedClass : public/protected/private BaseClass {
    // Thành Phần bổ sung của Lớp Dẫn Xuất
};
```

Kiểu kế thừa / dẫn xuất

Kế Thừa

KIỂU DẪN XUẤT

Kiểu Dẫn Xuất / Kiểu Kế Thừa

Thành phần Lớp Con	Private	Protected	Public
Private	–	–	–
Protected	Private	Protected	Protected
Public	Private	Protected	Public

Kiểu Dẫn Xuất / Kiểu Kế Thừa

Thành phần Lớp Con	Private	Protected	Public
Private	**public** trong lớp dẫn xuất. Có thể truy cập bằng các hàm riêng không tĩnh, hàm **friend** và các hàm không thành viên	**Protected** trong lớp dẫn xuất. Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend**.	**Private** trong lớp dẫn xuất Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend**.
Protected	**Protected** trong lớp dẫn xuất Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend**.	**Protected** trong lớp dẫn xuất. Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend**.	**Private** trong lớp dẫn xuất. Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend.**
Public	Giấu trong lớp dẫn xuất. Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend** thông qua các hàm thành viên **public** và **protected** của lớp cơ sở.	Giấu trong lớp dẫn xuất. Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend** thông qua các hàm thành viên **public** và **protected** của lớp cơ sở.	Giấu trong lớp dẫn xuất. Có thể truy cập trực tiếp bởi các hàm thành viên không tĩnh, các hàm **friend** thông qua các hàm thành viên **public** và **protected** của lớp cơ sở.

Kế Thừa

CÚ PHÁP

```
class Person {
protected:
    string name;
    int age;
};

class Staff : public Person {
protected:
    float salary;
};
```

LỢI ÍCH

- Xây dựng lớp mới từ lớp đã có.
- Chia sẻ mã chương trình chung ⇒ Dễ sửa chữa, nâng cấp.
- Cơ chế chuyển kiểu tự động trong C++.
- Dễ dàng bổ sung hoặc định nghĩa lại các thành phần.

Không sử dụng Kế Thừa

```
1 class Person {
2 private:
3    string name;
4    int age;
5 };
6
7 class Staff {
8 private:
9     string name;
10    int age;
11    float salary;
12 };
```

Sử dụng Kế Thừa

```
1 class Person {
2 private:
3     string name;
3     int age;
4 };
5
6 class Staff : public Person {
7 private:
8    float salary;
9 };
```

Không sử dụng Kế Thừa

```
1 class Person {
2 private:
3    string name;
4    int age;
5 };
6
7 class Staff {
8 private:
9     string name;
10    int age;
11    float salary;
12 };
```

Sử dụng Kế Thừa

```
1 class Person {
2 protected:
3     string name;
3     int age;
4 };
5
6 class Staff : public Person {
7 private:
8    float salary;
9 };
```

UIT Trường Đại học Công nghệ Thông tin | Khoa Khoa học và Kỹ thuật Thông tin

BAN HỌC TẬP

BUỔI 2

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

TÍNH ĐA HÌNH

POLYMORPHISM

TRAINER

Đặng Trung Hậu
MMCL2021

ĐA HÌNH (Polymorphism)

Đa Hình

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,...

KHÁI NIỆM

Tính đa hình/đa xạ (polymorphism) trong OOP cho phép các đối tượng khác nhau nhưng chung một lớp cơ sở thực hiện một hành động (phương thức) theo những cách khác nhau.

VD: Chó, mèo, chuột cùng thuộc lớp cơ sở là thú thực hiện cùng một hành động là kêu nhưng mỗi con phát ra một âm thanh khác nhau.

PHÂN LOẠI

Tính đa hình chủ yếu được chia làm 2 loại:

- **Compile time Polymorphism:** gồm có nạp chồng hàm (Function Overloading) và nạp chồng toán tử (Operator Overloading). Tức là các hàm/toán tử cùng tên nhưng nhận nhiều đối tượng khác nhau.
- Ví dụ: `choAn(Cho c)` và `choAn(Meo m)` là 2 hàm cùng tên nhưng nhận 2 đối tượng khác nhau.

PHÂN LOẠI

- **Runtime Polymorphism**: gồm có Virtual function được sử dụng để ghi đè và định nghĩa đúng chức năng của phương thức.
- Ví dụ: lớp **Thu** có phương thức **keu()** và lớp **Cho** là lớp con của lớp **Thu**. Khi đó chúng ta phải định nghĩa lại phương thức **keu()** cho lớp **Cho** và ghi đè lên phương thức ở lớp **Thu**.

OVERLOADING/OVERRIDING

- **OVERLOADING:** Một kỹ thuật cho phép trong một class có thể có nhiều phương thức trùng tên nhưng khác nhau về danh tham số.
- Đặc điểm: phương thức A overload phương thức B thì A và B chỉ được phép trùng tên, phải khác nhau về danh sách tham số (thứ tự, kiểu dữ liệu,..) và kiểu dữ liệu trả về.

OVERLOADING/OVERRIDING

- **OVERRIDNG:** Một kỹ thuật sử dụng trong trường hợp lớp con kế thừa từ lớp cha và muốn định nghĩa lại một phương thức đã có mặt ở lớp cha.
- Đặc điểm: phương thức A overide phương thức B thì A và B phải cùng tên, danh sách tham số, kiểu dữ liệu trả về.

PHƯƠNG THỨC ẢO

PHƯƠNG THỨC ẢO

Khái niệm

- Là cách thể hiện tính đa hình trong C++.
- Các phương thức có tính đa hình phải được định nghĩa là một phương thức ảo.

PHƯƠNG THỨC ẢO

Cú Pháp

- Ta quy định một hàm thành phần là Phương thức ảo bằng cách thêm từ khóa virtual vào trước khai báo hàm.

```
virtual <Kiểu_Dữ_Liệu> <Tên_Phương_Thức>();
```

Đa Hình

PHƯƠNG THỨC ẢO

Lưu ý

- Phương thức ảo chỉ hoạt động thông qua con trỏ.
- Phương thức ảo chỉ hoạt động nếu các 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).

Đa Hình

```
1   class Animal
2   {
3   public:
4       virtual void Voice()
5       {
6           cout << "\nGrr Grr";
7       }
8   };
```

Đa Hình

```
1   class Dog : public Animal
2   {
3   public:
4       void Voice()
5       {
6           cout << "\nGau Gau Gau";
7       }
8   };
```

Đa Hình

```
1  class Cat : public Animal
2  {
3  public:
4      void Voice()
5      {
6          cout << "\nMeow Meow" ;
7      }
8  };
```

PHƯƠNG THỨC THUẦN ẢO

Đa Hình

PHƯƠNG THỨC THUẦN ẢO

Khái Niệm

- Là phương thức ảo và không có định nghĩa bên trong.

Cú Pháp

```
virtual <Kiểu_Dữ_Liệu> <Tên_Phương_Thức>() = 0;
```

LỚP TRỪU TƯỢNG

Khái niệm

- Là lớp chỉ được dùng làm cơ sở cho các lớp khác.
- Bắt buộc phải có ít nhất một phương thức thuần ảo.

LỚP TRỪU TƯỢNG

Đặc Điểm

- Dùng để định nghĩa một số khái niệm tổng quát, chung cho các lớp khác.
- Không có đối tượng nào thuộc Lớp trừu tượng.

MỐI LIÊN HỆ GIỮA PHƯƠNG THỨC THUẦN ẢO VÀ LỚP TRỪU TƯỢNG

- Các phương thức thuần ảo thường được chứa trong lớp trừu tượng.
- Theo quan điểm chung, lớp trừu tượng không nhất thiết phải chứa phương thức thuần ảo.
- Theo quan điểm C++, lớp trừu tượng phải chứa ít nhất một phương thức thuần ảo.

PHƯƠNG THỨC THUẦN ẢO

Lưu ý

- Lớp dẫn xuất từ lớp trừu tượng phải định nghĩa lại tất cả các phương thức thuần ảo của lớp trừu tượng.
  - Hoặc tiếp tục là phương thức thuần ảo.
  - Hoặc được cài đặt chi tiết trong lớp dẫn xuất.
- Có ý nghĩa trong việc tổ chức phân cấp các lớp: phương thức thuần ảo trong lớp cơ sở sẽ là phương thức chung cho các lớp dẫn xuất thừa kế và cài đặt.

CẢM ƠN CÁC BẠN ĐÃ LẮNG NGHE !

bit.ly/DD_OOP_2