Bài 16: OOP

Nguyễn Hoàng Anh

Tính đóng gói

```cpp
#include <iostream>
#include <string>
using namespace std;
class SinhVien{
    public:
        string name;
        int id;
        SinhVien(string newName){
            static int ID = 1;
            id = ID;
            ID++;

            name = newName;
        }
};
int main(int argc, char const *argv[])
{
    SinhVien sv1("Hoang"), sv2("Tuan");

    sv1.name = "Anh@123";
    sv1.id = 2;
    sv1.display();
    return 0;
}
```

object 1 có thể khởi tạo tên chứa ký tự đặc biệt và ID có thể giống object 2. Vậy làm sao để giải quyết vấn đề này?

Tính đóng gói

- **Tính đóng gói** (Encapsulation) là ẩn đi các property "mật" khỏi người dùng. Và để làm được điều này, ta sẽ khai báo các property ở quyền truy cập **private/protected** (tức là không thể truy cập trực tiếp tới các property này thông qua object bên ngoài).
- Trong trường hợp ta muốn đọc hoặc ghi các property này, thì ta sẽ truy cập gián tiếp thông qua các method ở quyền truy cập public.

Tính đóng gói

```cpp
#include <iostream>
#include <string>
using namespace std;

class SinhVien{
    private:
        string name;
        int id;

    public:
        SinhVien(){
            static int ID = 1;
            id = ID;
            ID++;
        }

        void setName(string newName){   // setter method
            // kiểm tra điều kiện
            name = newName;
        }

        string getName(){   // getter method
            return name;
        }
```

Tính kế thừa

```
class SinhVien
{
    public:
        string ten;
        int id;
        string chuyenNganh;
};

class HocSinh
{
    public:
        string ten;
        int id;
        string lop;
};

class GiaoVien
{
    public:
        string ten;
        int id;
        string chuyenMon;
};
```

Liệu có thể gộp những property giống nhau ở 3 class để rút gọn source code hay không?

Tính kế thừa

- Tính kế thừa (Inheritance) là khả năng sử dụng lại các property và method của một class trong một class khác và có thể mở rộng thêm tính năng. Ta chia chúng làm 2 loại là class cha và class con.
- Để kế thừa từ class khác, ta dùng ký tự `":"`.
- Có 3 kiểu kế thừa là public, private và protected. Những property và method được kế thừa từ class cha sẽ nằm ở quyền truy cập của class con tương ứng với kiểu kế thừa.

Kế thừa public

- Các member public của class cha vẫn sẽ là public trong class con.
- Các member protected của class cha vẫn sẽ là protected trong class con.
- Các member private của class cha không thể truy cập trực tiếp từ class con nhưng có thể được truy cập gián tiếp qua các phương thức public hoặc protected của class cha.

Kế thừa public

```
#include <iostream>
#include <string>

using namespace std;

class DoiTuong{
    // private:
    //    string ten;
    //    int id;

    protected:
        string ten;
        int id;

    public:
        DoiTuong(){
            static int ID = 1;
            id = ID;
            ID++;
        }

        void setName(string _ten){
            // check chuỗi nhập vào
            ten =  ten;
```

Kế thừa protected

- Các member public, protected của class cha sẽ là **protected** trong class con.
- Các member private của class cha không thể truy cập trực tiếp từ class con nhưng có thể được truy cập gián tiếp qua các phương thức public hoặc protected của class cha.

  public → protected

  protected → protected

Kế thừa protected

```cpp
#include <iostream>
#include <string>

using namespace std;

class DoiTuong{
    protected:
        string ten;
        int id;

    public:
        DoiTuong(){
            static int ID = 1;
            id = ID;
            ID++;
        }

        void setName(string _ten){
            // check chu ỗi nhập vào
            ten = _ten;
        }

        void display(){
            cout << "ten: " << ten << endl;
            cout << "id: " << id << endl;
```

Kế thừa private

- Các member public, protected của class cha sẽ trở thành **private** trong class con.
- Các member private của class cha không thể truy cập trực tiếp từ class con nhưng có thể được truy cập gián tiếp qua các phương thức public hoặc protected của class cha.

public → private

protected → private

Kế thừa private

```cpp
#include <iostream>
#include <string>
using namespace std;

class DoiTuong{
    protected:
        string ten;
        int id;

    public:
        DoiTuong(){
            static int ID = 1;
            id = ID;
            ID++;
        }

        void setName(string _ten){
            // check chuỗi nhập vào
            ten = _ten;
        }

        void display(){
            cout << "ten: " << ten << endl;
            cout << "id: " << id << endl;
        }
```

Tính đa hình

- Tính đa hình (Polymorphism) có nghĩa là "nhiều dạng" và nó xảy ra khi chúng ta có nhiều class có liên quan với nhau thông qua tính kế thừa.
- Tính đa hình có thể được chia thành hai loại chính:
  - **Đa hình tại thời điểm biên dịch (**Compile-time Polymorphism**)**.
  - **Đa hình tại thời điểm chạy (**Run-time Polymorphism**)**.

Tính trừu tượng

- Tính trừu tượng đề cập đến việc ẩn đi các chi tiết cụ thể của một đối tượng và chỉ hiển thị những gì cần thiết để sử dụng đối tượng đó. Và để làm được điều này, ta sẽ khai báo các method ở quyền truy cập **private/protected.**

Tính trừu tượng

$$ax^2 + bx + c = 0$$

Tính $\Delta = b^2 - 4ac$

- Nếu $\Delta < 0 \Rightarrow$ vô nghiệm
- Nếu $\Delta = 0 \Rightarrow$ nghiệm chung x = …
- Nếu $\Delta > 0 \Rightarrow$ x1 = …

  x2 = …

Tính trừu tượng

```cpp
#include <iostream>
#include <string>
#include <cmath>

using namespace std;

class GiaiPhuongTrinh {
    private:   // a,b,c,x1,x2,delta: tính đóng gói
        double a;
        double b;
        double c;
        double x1;
        double x2;
        double delta;
        void tinhNghiem(){    // tính trừu tượng
            delta = b*b - 4*a*c;
            if (delta < 0){
                delta = -1;
            }
            else if (delta == 0){
                x1 = x2 = -b/ (2*a);
            }
            else if (delta > 0){
                x1 = (-b + sqrt(delta))/(2*a);
                x2 = (-b - sqrt(delta))/(2*a);
```