Phill-CPP-0802

Inheritance 繼承

abstraction 階層化 → inheritance → efficient → 軟體工程

Window → Warning Window

→ Yes, no window

→ Fatal window

父類別 → base class

子類別 → derived class

primitive object → 空

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

class Vehicle{
  public:
    string name = "vehicle";
    void launch(){
      cout << "start your engine!" <<endl;
    }
};

class Car : public Vehicle{
    public:
      string brand = "Tesla";
      string model= "Model-S";
};


int main()
{
    Car myCar;
    myCar.launch();
    cout << "my car is "<< myCar.brand << ",  "<< myCar.model <<endl;

    return 0;
}
```

衍生不同 class

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

class Vehicle{
  public:
    string name = "vehicle";
    void launch(){
      cout << "start your engine!" <<endl;
    }
};

class Car : public Vehicle{
    public:
      string brand = "Tesla";
      string model= "Model-S";
};

class Rocket : public Vehicle{
  public:
    string brand="SpaceX";
    string model="Falcon-9";
};


int main()
{
    Car myCar;
    Rocket myRoc;
    myCar.launch();
    myRoc.launch();

    cout << "my car is "<< myCar.brand << ",  "<< myCar.model <<endl;
    cout << "my rocket is "<< myRoc.brand << ",  "<< myRoc.model <<endl;

    return 0;
}
```

multilevel inheritance

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

class myClass{
  public:
    void myFunction(){
      cout << "function"<<endl;
    }
};
```

```
class myChild : public myClass{

};

class myGrandChild: public myChild{

};

int main()
{
    myChild zzz;
    myGrandChild abc;
    zzz.myFunction();
    abc.myFunction();
    return 0;
}
```

在繼承權限控管

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

class Person{
  public:
    int bonus=100;
  protected:
    int salary=100;
  private:
    int deposit=100;
};

class Programmer : public Person{
  public:
    void setSalary(int s){
      salary= s;
    }

    int getSalary(){
      return salary;
    }

    // int getDeposit(){
    //  return deposit;
    // }
};


int main()
{
    Programmer p;
    p.bonus=1000;

    cout << "bonus->" << p.bonus<<endl;

    p.setSalary(1000);

    cout << "salary -> " << p.getSalary() << endl;

   // cout << p.getDeposit();
    return 0;
}
```

- encapsulation
  - public → 都可以, 內外父子
  - private → 父子不行 只有自己
  - protected → 父子可以 外人不行