Phill-CPP-0628

class —> object (instance實例)

- members
  - data members
  - member functions

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

class Person{
public:
  int id;
  string name;
  void showName(){
    cout << "my name is "<< name <<endl;
  }
};

int main()
{
    Person phill;

    phill.name = "Phill";
    phill.showName();
    return 0;
}
```

成員→ data, function

權限控管

- public
- private
- ~~protected~~

public

- 對所有人開放 存取權 使用權
- 跨class使用

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

class Circle{
  public:
    double radius=3;

    double compute_region(){
      return radius*radius*3.14;
    }
};

int main()
{
    Circle o;

    o.radius= 10;

    cout << "radius = " << o.radius <<endl;
    cout << "region =" << o.compute_region() << endl;
    return 0;
}
```

private

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

class Circle{
  private:
    double radius=3;
   public:
    double compute_region(){
      return radius*radius*3.14;
    }
};

int main()
{
    Circle o;

    //o.radius= 10;
```

```
    //cout << "radius = " << o.radius <<endl;
    cout << "region =" << o.compute_region() << endl;
    return 0;
}
```

setter

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

class Circle{
  private:
    double radius=3;
   public:
    double compute_region(){
      return radius*radius*3.14;
    }
    //setter
    void setRadius(double setting){
        radius = setting;
    }
};

int main()
{
    Circle o;

    o.setRadius(10);

    //cout << "radius = " << o.radius <<endl;
    cout << "region =" << o.compute_region() << endl;
    return 0;
}
```

練習

建立一個 class 銀行帳戶

- 存錢 deposit()
- 提錢 withdraw()
- 查詢餘額 queryBalance()

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

```
class BankAccount {
public:

    string owner;
    double balance;



    // 存款方法
    void deposit(double amount) {
        balance += amount;
        cout << "存款成功, 餘額為 " << balance << endl;
    }

    // 提款方法
    void withdraw(double amount) {
        if (amount > balance) {
            cout << "超過餘額" << endl;
            return;
        }
        balance -= amount;
        cout << "成功, 餘額為 " << balance << endl;
    }

    // 查詢帳戶資訊方法
    void queryBalance() {
        cout << "帳號名稱 " << owner << endl;
        cout << "餘額為 " << balance << endl;
    }
};

int main() {
    BankAccount a;
    a.owner="Phill";
    a.balance=3000000;
    a.queryBalance();
    a.deposit(100);
    a.queryBalance();
    return 0;
}
```

abstraction 抽象化

- 規劃時候, 把物件的細節隱藏, 只公開規格, 彼此隱藏複雜度, 整體軟體工程複雜度會降低