Phill-CPP-0705

public, private

private → friend function

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

class Cirle{
  private:
    double radius=10;

    friend void read_radius(Cirle& c);

  public:
    double compute_area(double r){
      if(r>=0){
        radius = r;
      }
      cout << "Inside class ->" << radius <<endl;
      return 3.14159*radius*radius;
    }
};

void read_radius(Cirle& c){
  cout <<"I am friend access private member->" << c.radius <<endl;
}

int main()
{
    Cirle obj;
    cout << obj.compute_area(100) << endl;
    read_radius(obj);
    return 0;
}
```

Data encapsulation

- 控管權限

實作 member functions

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

class Person{
  public:
    int id;
    string name;

    void showName(){
      cout << "my name is " << name <<endl;
    }

    void printName(); //不要做在這裡
};

//implement
void Person::printName(){
  cout << "my name printName -> "<< name <<endl;
}

int main()
{
    Person p1;

    p1.name="Phill";
    p1.showName();
    p1.printName();
    return 0;
}
```

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

class Person{
  public:
    int id;
    string name;

    void showName(){
      cout << "my name is " << name <<endl;
    }

    void printName(); //不要做在這裡
    void changeName(string newName);
};

//implement
void Person::printName(){
  cout << "my name printName -> "<< name <<endl;
}
```

```
void Person::changeName(string newName){
  name = newName;
}

int main()
{
    Person p1;

    p1.name="Phill";
    p1.changeName("Phill Liu");
    p1.printName();
    return 0;
}
```

Constructor

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

class Person{
  public:
    int id;
    string name;

    void showName(){
      cout << "my name is " << name <<endl;
    }

    Person(string initName, int initID){
      //constructor , initalization
      name = initName;
      id = initID;
    }



    void printName(); //不要做在這裡
    void changeName(string newName);
};

//implement
void Person::printName(){
  cout << "my name printName -> "<< name <<endl;
  cout << "id=" << id <<endl;
}

void Person::changeName(string newName){
  name = newName;
```

```
}

int main()
{
    Person p1("Phill123", 1);
    Person p2("Phill", 2);
    Person p3("Phill Liu", 3);


    //p1.changeName("Phill Liu");
    p2.printName();
    p3.printName();
    return 0;
}
```

練習

- Car 類別
- brand, model, year
- constructor
- (Tesla model-S 2022), (BMW, X5, 1999)

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

class Car {

public:
  string brand;
  string model;
  int year;
  void showInfo(){
    cout << "Car Infomation -->" << brand <<" " << model <<" " << year <<endl;
  }
    Car(string initbrand , string initmodel, int inityear){
      brand=initbrand;
      model=initmodel;
      year=inityear;
    }

};

int main()
{

    Car p1("xxxx","xxxx",123);
    Car p2("Tesla","model-S",2022);
```

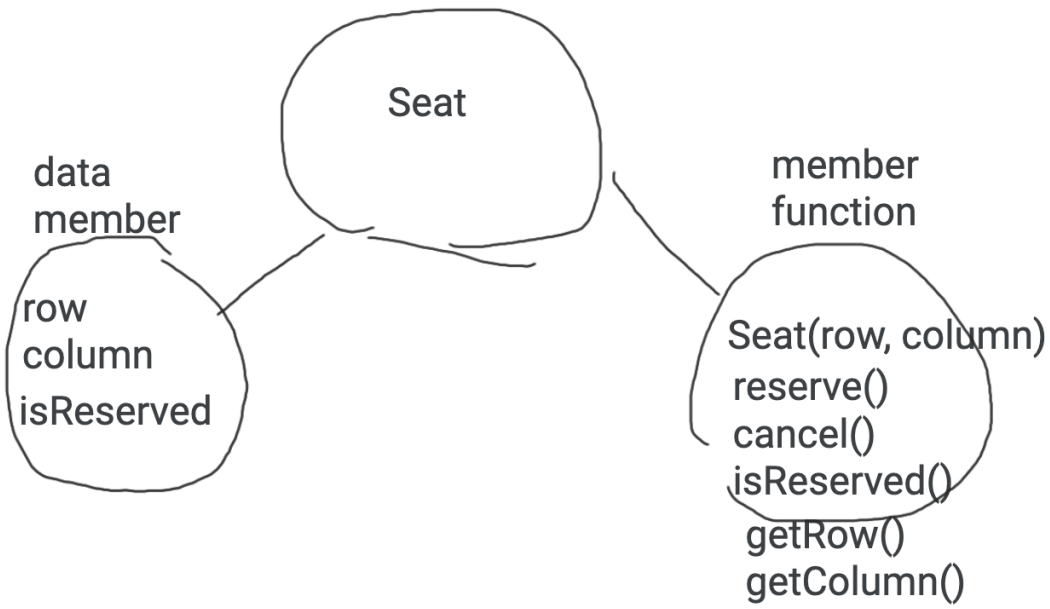
```
    Car p3("BMW","X5",1999);


    p2.showInfo();
    p3.showInfo();
    return 0;
}
```

Assignment

- 電影院座位系統
- OO 角度去設計
- class → Seat → member (data, functions)
- 思考一下跟 原先的設計有何不同 優缺點

→ Person




```
#include <iostream>
using namespace std;
class Seat{
```

```
  private:
    int row;
    int column;
    bool isReserved;

  public:
    Seat(int r, int c){
      row=r;
      column= c;
      isReserved=false;
    }

    void reserve(){
      isReserved=true;
    }

    void cancel(){
      isReserved=false;
    }

    bool isReservedXXX(){
      return isReserved;
    }

    int getRow(){
      return row;
    }


    int getColumn(){
      return column;
    }


};

Seat::Seat(int r, int c){
      row=r;
      column= c;
      isReserved=false;
 }
int main()
{
    Seat s[3]={
      Seat(1,1),
      Seat(2,2),
      Seat(3,3)
    };
    // for(int i=1; i<10; i++)
    //   for(int j=1; j<10; j++)
    //       Seat s[i*j]= Seat(i,j);

    cout << "Row: " <<s[0].getRow()<<", Column:" <<s[0].getColumn() <<endl;
```

```
    cout << "reserved:" <<s[0].isReservedXXX() <<endl;

    // s.reserve();
    // cout << "reserved:" <<s.isReservedXXX() <<endl;

    // s.cancel();
    // cout << "reserved:" <<s.isReservedXXX() <<endl;

    return 0;
}
```

- Setter → reserve(), cancel() → public function → control private data
- Getter → isReserved(), getRow() → public function → retrieve private data

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