

C++程式設計基礎

lesson 8

陳毅

本週概要

- 物件導向程式設計基本概念 – Part 3
 - 靜態成員
- 多載函數
- 物件導向程式設計基本概念 – Part 3
 - 繼承概念

物件導向程式設計基本概念 – Part 3

靜態成員(static member)

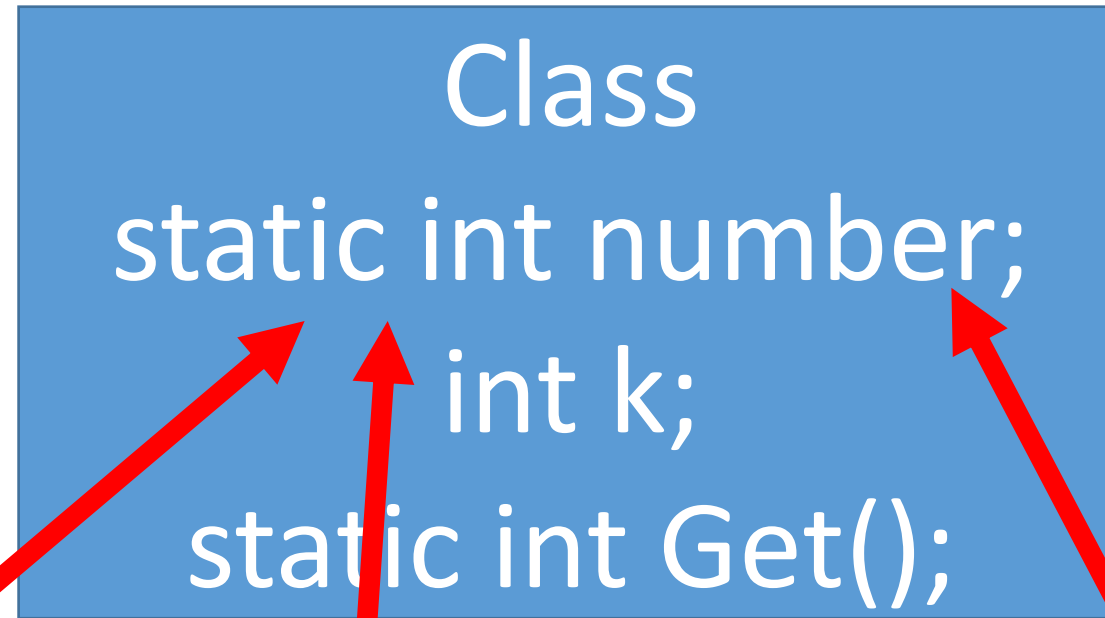
- 若某個資料成員被宣告為static，則其他同類別的物件皆可共享該靜態成員的資料。
- 若某個成員函數被宣告為static，則該static成員函數，不可呼叫非static的成員函數。
- 使用方法：宣告的型態前，加上「static」。

```
static 資料型態 變數名稱;  
static 函數型態 函數名稱(參數列){ 敘述區; }
```

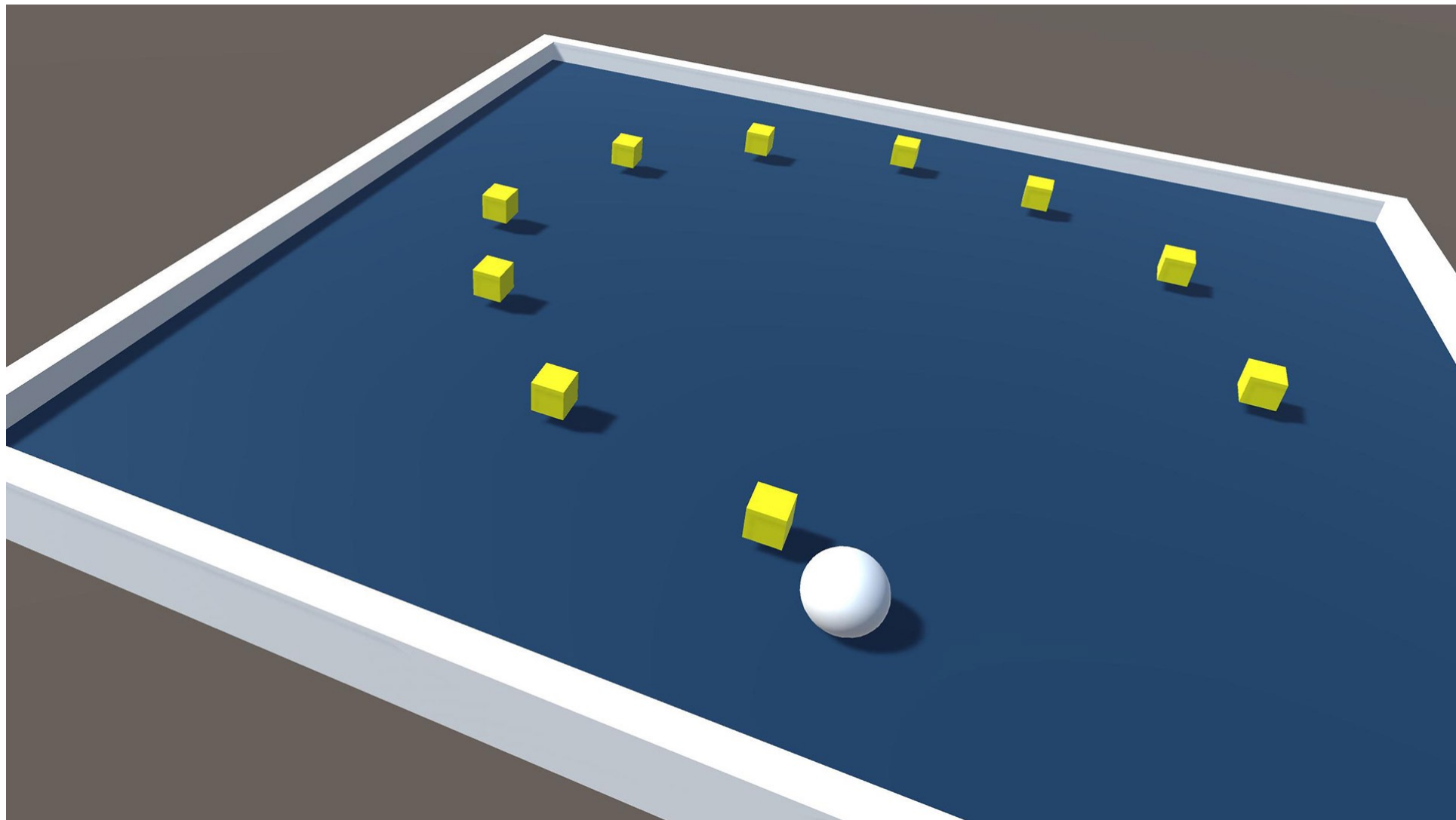
靜態成員(static member)

- 靜態的資料成員，必須要進行初始化，而且也只能被初始化一次。

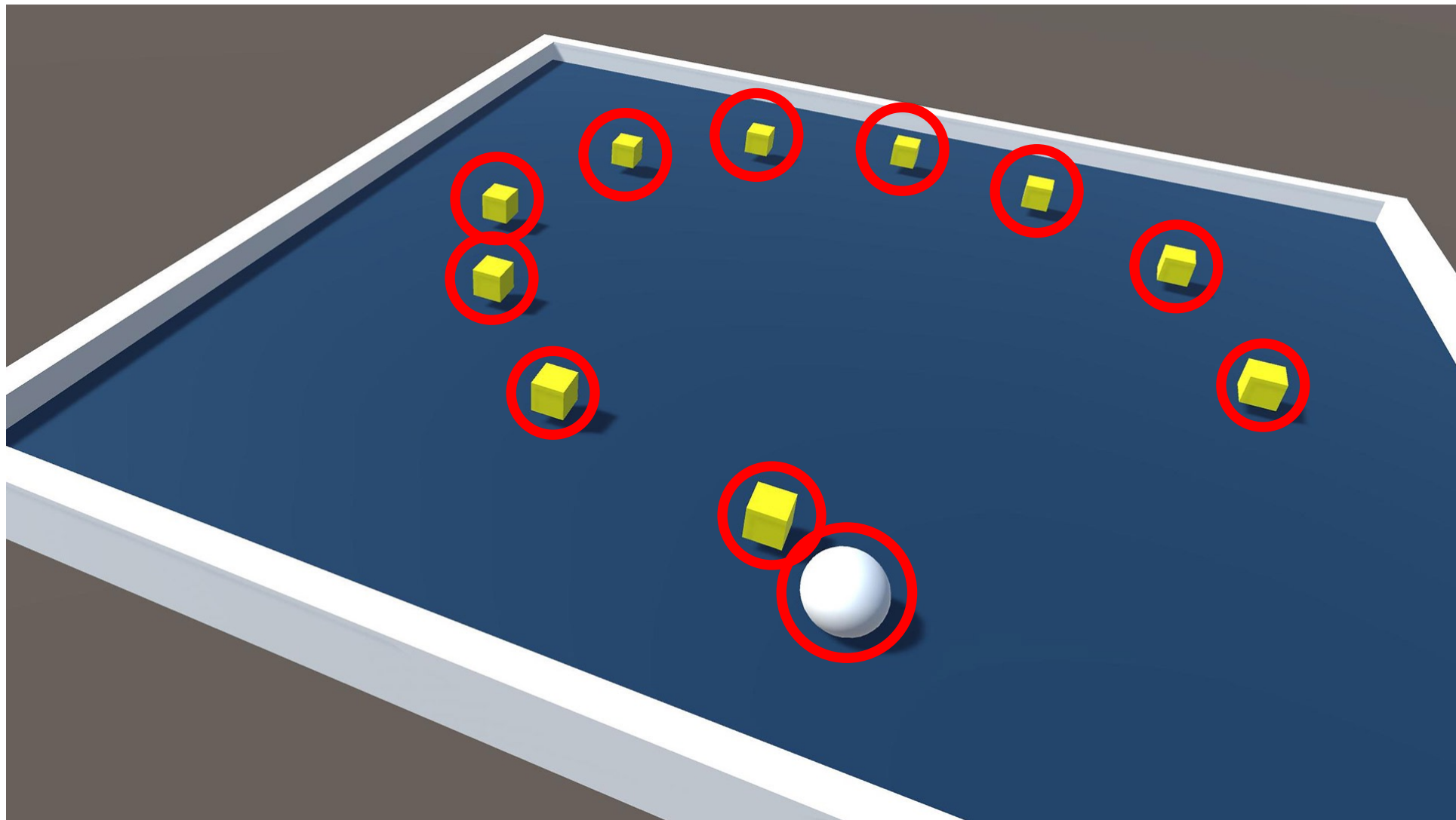
```
class Test{  
public:  
    Test(int k){ this->k = k; }  
    void Add1(){ number = number + 1; }  
    int Get(){ return number+k; }  
private:  
    static int number;  
    int k;  
};  
  
int Test::number = 0;
```



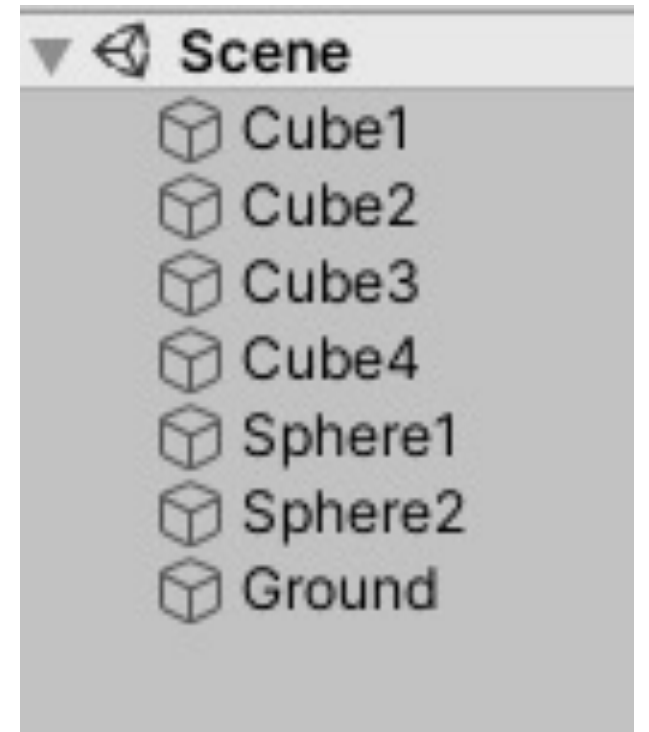
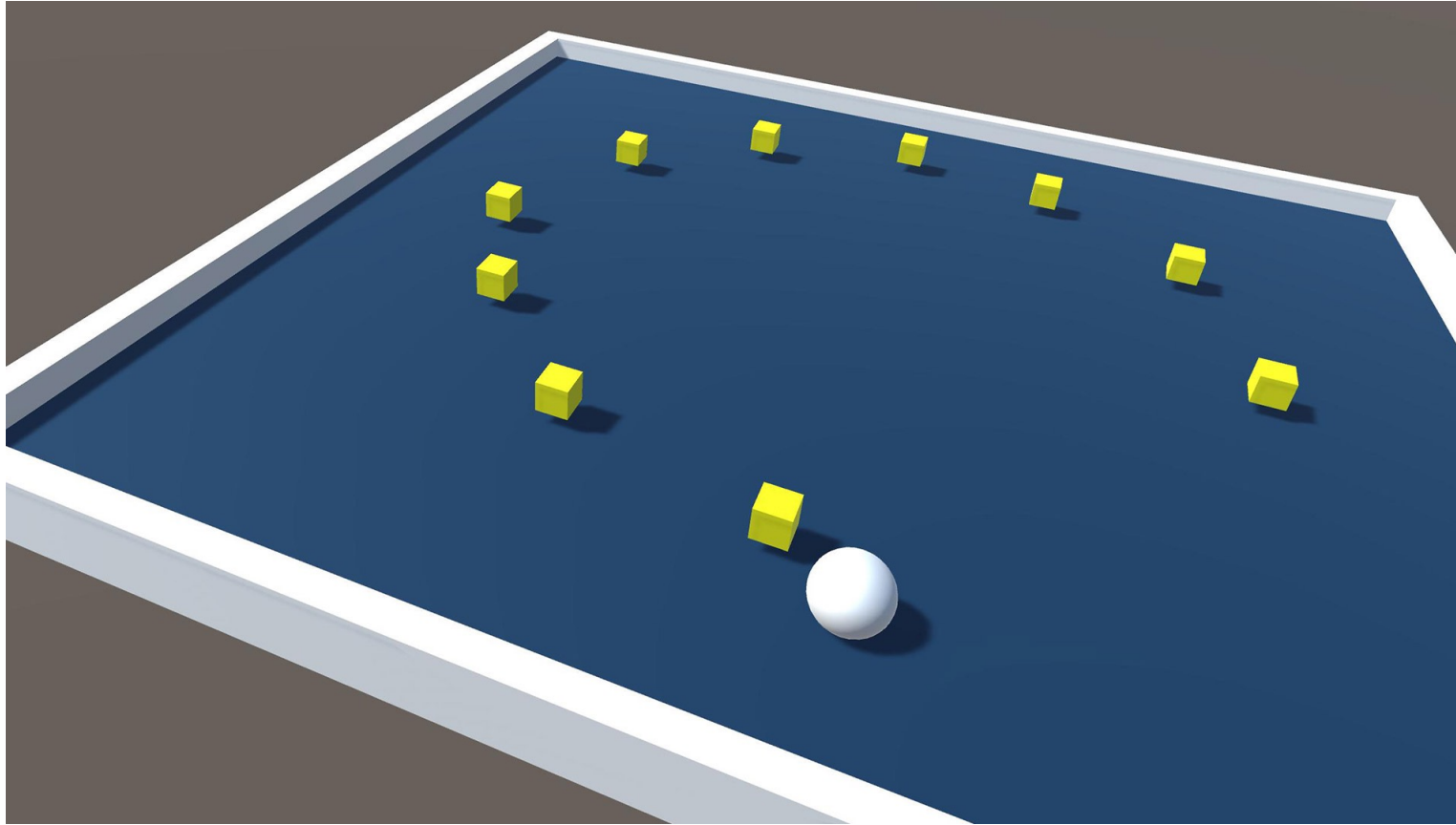
遊戲場景



遊戲場景



遊戲場景



遊戲場景

- 一個場景 (scene) 中會有許多的物體。
- 使用者可以在場景中新增物體及刪除物體。
- 每個物體 (object) 都會有自己的名稱與編號。
- 場景會記錄現在有多少個物體。
- 使用者可以查看場景中個別物體的資訊，或是查看整體的資訊。

遊戲場景

- 一個場景 (scene) 中會有許多的物體。
- 使用者可以在場景中新增物體及刪除物體。
- 每個物體 (object) 都會有自己的名稱與編號。
- 場景會記錄現在有多少個物體。
- 使用者可以查看場景中個別物體的資訊，或是查看整體的資訊。

多載函數

什麼是多載(overloading)？

- 四則運算也是函數的一種。
- $3+5$ ，實際上就是 *int operator+ (3, 5)*
- $6.3-1.8$ ，實際上就是 *double operator- (6.3, 1.8)*

什麼是多載(overloading)？

- 關於加法，加數與被加數有很多種組合。
- 整數 + 整數： $3+5$, $62+10$, ...
- 整數 + 小數： $7+1.3$, $6+3.14$, ...
- 小數 + 整數： $5.32+13$, $872.24+524$, ...
- 小數 + 小數： $34.43+235.73$, $1.548+745.231$, ...
- 那加法的函數雛形到底長什麼樣子呢？

什麼是多載(overloading)？

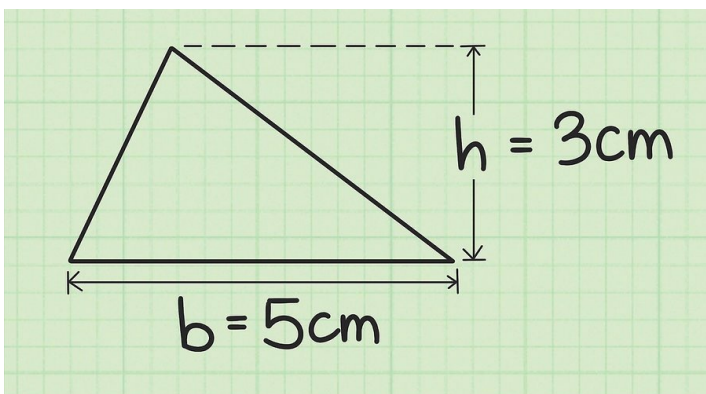
- Ans：加法的函數雛形其實有很多個！
- 整數 + 整數：int operator+ (int, int)
- 整數 + 小數：double operator+ (int, double)
- 小數 + 整數：double operator+ (double, int)
- 小數 + 小數：double operator+ (double, double)
- 同個函數名稱卻有不同的定義及功能，這就稱作多載(**overloading**)。

練習

- 想要計算一個三角形的面積有很多不同的算法，例如：

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

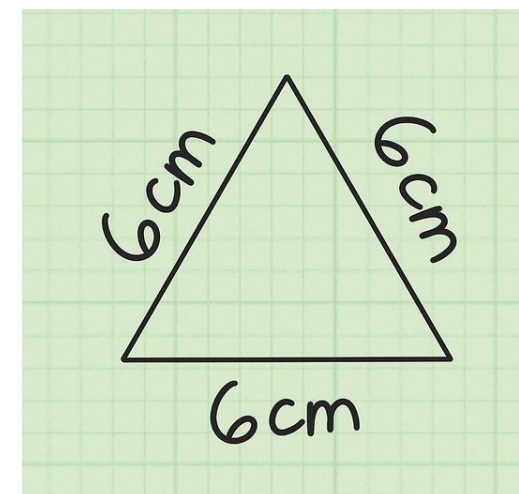
```
double area(int, int, int);
```



```
double area(int, int);
```

$$\text{Area} = \frac{bc}{2} \sin A$$

```
double area(int, int, double);
```



```
double area(int);
```


多載函數還能做到的事

- 若你定義了一個類別用來儲存複數，那要怎麼做複數四則運算呢？
 - 可自己定義屬於自己的四則運算(+, -, *, /)。

```
class Test{
public:
    Test(){ a = b = 0; }
    Test(int n, int m){ a = n; b = m; }
    Test operator+(Test);
    void display(){ cout << a << " " << b << endl; }
private:
    int a, b;
};


Test Test::operator+(Test obj){
    Test tmp;
    tmp.a = a + obj.a;
    tmp.b = b + obj.b;
    return tmp;
}
```


繼承

關於可愛的小動物們...



關於可愛的小動物們...

 Dog
<ul style="list-style-type: none">- name: string { set; get; }- weight: double { set; get; }- age: int { set; get; }
+ WangWang(): void

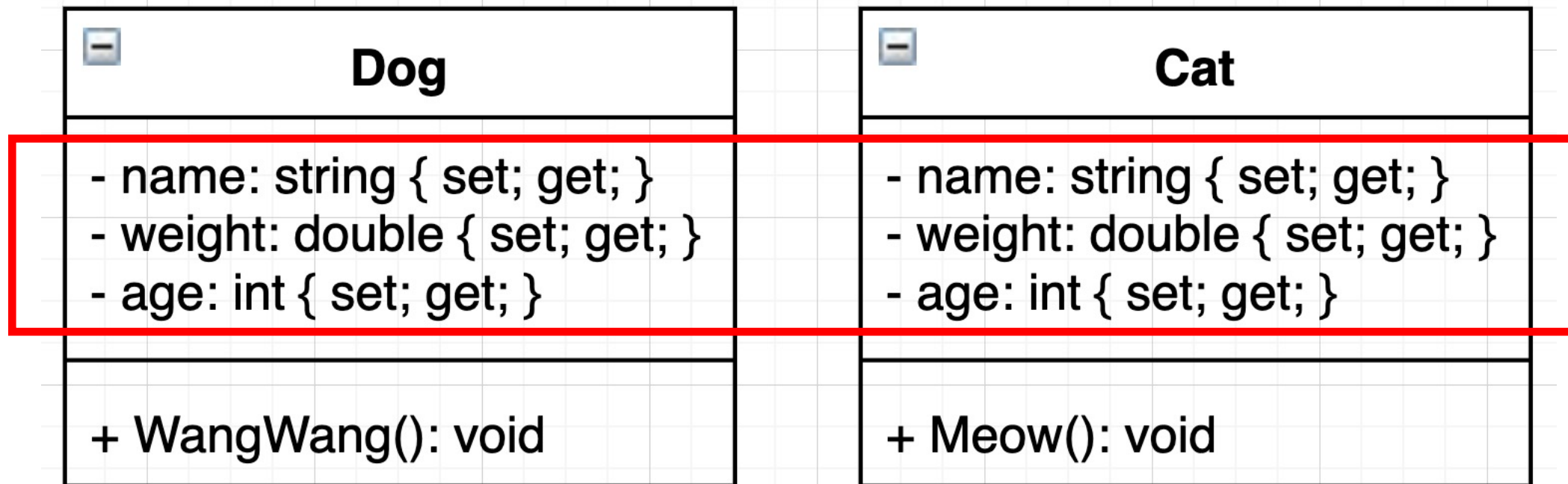
 Cat
<ul style="list-style-type: none">- name: string { set; get; }- weight: double { set; get; }- age: int { set; get; }
+ Meow(): void

關於可愛的小動物們...

```
class Dog{
public:
    void WangWang(){
        cout << "Wang! Wang!" << endl;
    }
    string GetName() { return name; }
    void SetName(string name) { this->name = name; }
    double GetWeight() { return weight; }
    void SetWeight(double weight) { this->weight = weight; }
    int GetAge() { return age; }
    void SetAge(int age) { this->age = age; }
private:
    string name;
    double weight;
    int age;
};
```


```
class Cat{
public:
    void Meow(){
        cout << "Meow~~~" << endl;
    }
    string GetName() { return name; }
    void SetName(string name) { this->name = name; }
    double GetWeight() { return weight; }
    void SetWeight(double weight) { this->weight = weight; }
    int GetAge() { return age; }
    void SetAge(int age) { this->age = age; }
private:
    string name;
    double weight;
    int age;
};
```

關於可愛的小動物們...

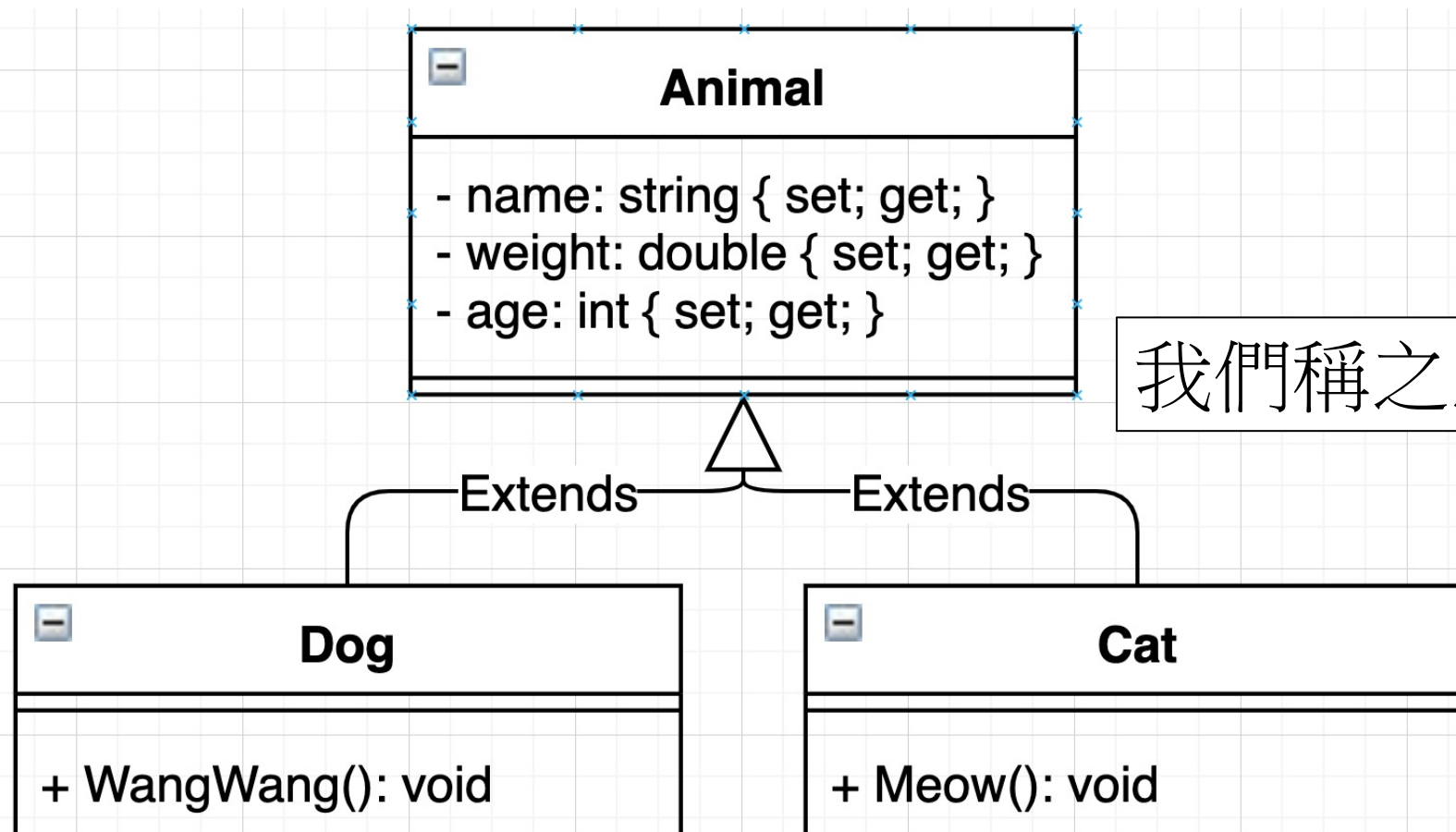


只要是動物，都有這些特徵

可愛小動物的特徵

 Animal
<ul style="list-style-type: none">- name: string { set; get; }- weight: double { set; get; }- age: int { set; get; }

把「動物」這個共同特徵抽離...



我們稱之為「繼承」


```
class Animal {
public:
    string GetName() { return name; }
    void SetName(string name) { this->name = name; }
    double GetWeight() { return weight; }
    void SetWeight(double weight) { this->weight = weight; }
    int GetAge() { return age; }
    void SetAge(int age) { this->age = age; }
private:
    string name;
    double weight;
    int age;
};
```

```
class Dog: public Animal{
public:
    void WangWang(){
        cout << "Wang! Wang!" << endl;
    }
};
```

```
class Cat: public Animal{
public:
    void Meow(){
        cout << "Meow~~~" << endl;
    }
};
```

繼承的用法

```
class 類別名稱: public 基底類別{  
public:  
    定義公用成員  
private:  
    定義私用成員  
};
```

繼承方式 訪問權限	Public	Protected	Private
Public	Public	Protected	Private
Protected	Protected	Protected	Private
Private			

補充資料

- Polymorphism – C++ reference:
<http://www.cplusplus.com/doc/tutorial/polymorphism/>

練習

- Zerojudge
 - c665 : 進制轉換
 - d092 : 算式也可以比大小！？
 - d625 : 踩地雷真好玩
 - a218 : 連猴子都會的小case
 - c638 : 天干地支
 - d098 : `stringstream`運用練習(C++)
 - a982 : 迷宮問題#1
- Reading: 課本 Ch11.5.3, Ch12