classes of objects
- Attributes = data members
- Behaviors = methods

OOP 概念
- 封裝 = hides inner details
- 繼承 = reused
- 多型

Operation Contracts 運算合約

purpose 目的 Assumptions 假設 Input 輸入 Output 輸出

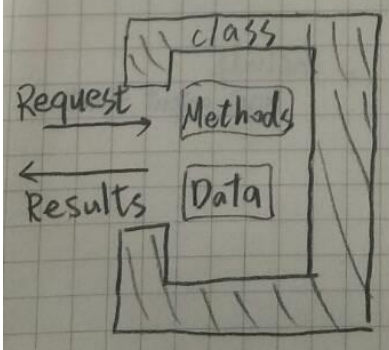
Modularity 模組化 ADT: Abtraction data type

highly cohesive modules desired 高內聚 → f 做一件事
Loosely Coupled modules desired 低耦合 → 彼此依賴少

Functional abstraction 功能性的抽象化
- 描述 (.h)
- 實作 (.cpp) → 資訊隱藏

Data abtraction 資料抽象化 → 資料結構

encapsulated 封裝



header file → a.h ⇒ #include "a.h"
.cpp

- Constructors
  - Initializer
- Destructors
  - int... → 自動刪除
  - each class has one destructor

class ColoredSphere : public Sphere ... { 繼承. 可覆載 (overriding)

: • 父類別 = Sphere ← defined methods, data members

} • 子 〃 = ColoredSphere

Overloading 多載

```
class A {
    public:
        Rational add(Rational); // same function name
        Rational add(long);
}
```

Private : only class instances
Protected : subclass instances
Public : any class instances

```
namespace a{
    int count=0;
}
```
// 自訂命名空間

```
using namespace a; // 使用命名空間
count++;
```

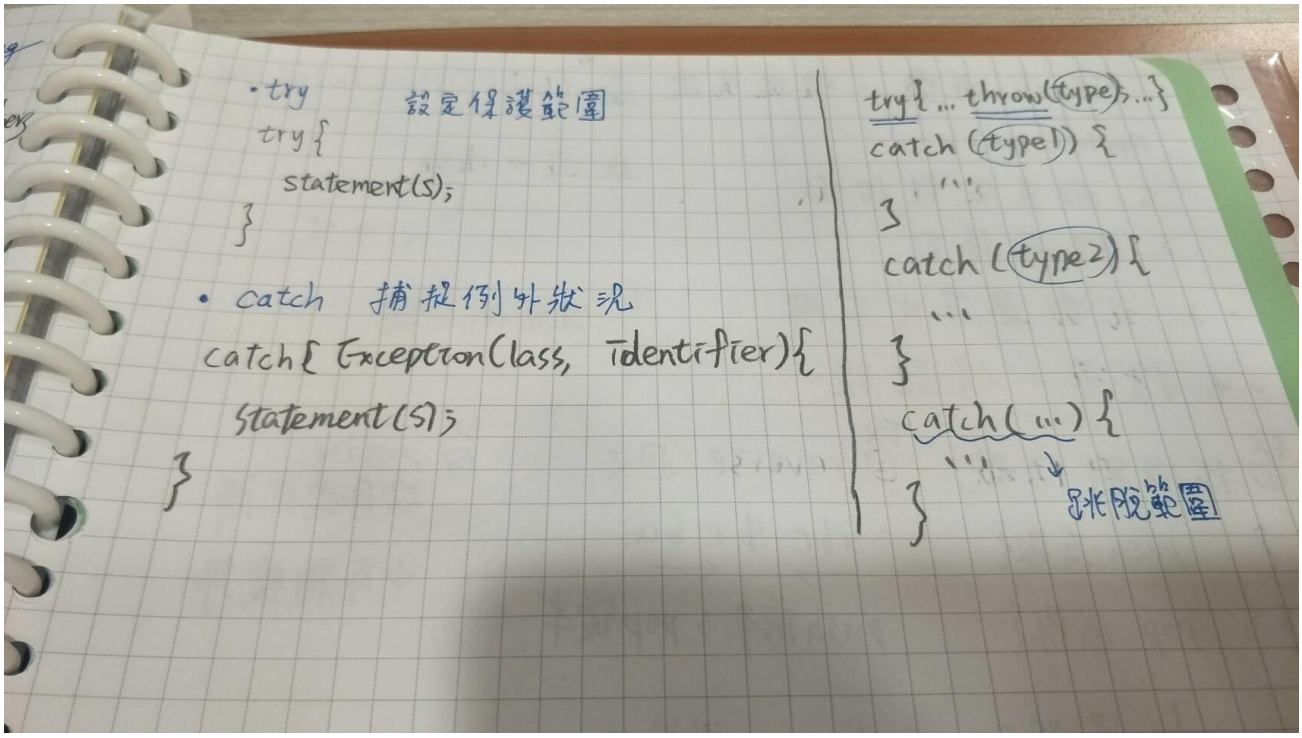
☆ C++ standard Library 標準函式庫
are declared in the std namespace → Include input, output function

→ using namespace std;

Ex: 原 std::cin >> a;

使用命名空間

```
using namespace std;
cin >> a;
```


• try 設定保護範圍
try {
statement(s);
}
• catch 捕捉例外狀況
catch(Exception(class, identifier){
statement(s);
}
try{... throw(type),...}
catch (type1) {
...
}
catch (type2) {
...
}
catch(...) {
...
}
跳脫範圍