第十章

物件導向思維

本章綱要

- 使用 string 類別處理字串(§10.2)
- 開發具有物件參數的函式(§10.3)
- 儲存和處理物件陣列(§10.4)
- 區別實體與靜態的變數和函式(§10.5)
- 定義常數函式以防止資料項目意外的被更改(§10.6)
- 探索程序性模式與物件導向模式之差異(§10.7)
- 設計 BMI 的類別(§10.7)
- 開發用於建構合成關係的類別 (§10.8)
- 設計堆疊類別(§10.9)
- 依照類別設計指引來設計類別(§10.10)

C++ 的 string 類別

建立字串

- create an empty string: no-arg
  - string newString;
- create a string from a string literal
  - string newString(stringLiteral);

範例：字串的附加

您可以使用一些多載函式，將新的內容附加於字串。例如下述程式碼：

```
string s1("Welcome");
s1.append(" to C++"); // appends " to C++" to s1
cout << s1 << endl; // s1 now becomes Welcome to C++
string s2("Welcome");
s2.append(" to C and C++", 0, 5); // appends " to C" to s2
cout << s2 << endl; // s2 now becomes Welcome to C
string s3("Welcome");
s3.append(" to C and C++", 5); // appends " to C" to s3
cout << s3 << endl; // s3 now becomes Welcome to C
string s4("Welcome");
s4.append(4, 'G'); // appends "GGGG" to s4
cout << s4 << endl; // s4 now becomes WelcomeGGGG
```

範例：指定一字串

您可以使用一些多載函式，將新的內容指定給字串。例如下述程式碼：

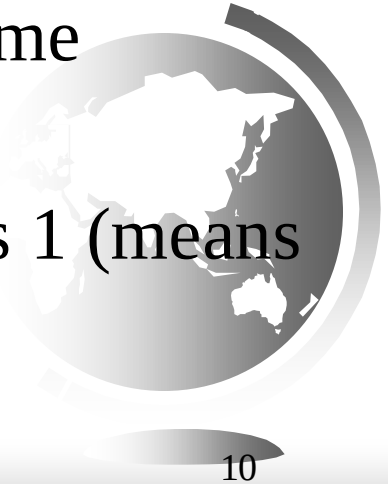
```
string s1("Welcome");
s1.assign("Dallas"); // assigns "Dallas" to s1
cout << s1 << endl; // s1 now becomes Dallas
string s2("Welcome");
s2.assign("Dallas, Texas", 0, 5); // assigns "Dalla" to s2
cout << s2 << endl; // s2 now becomes Dalla
string s3("Welcome");
s3.assign("Dallas, Texas", 5); // assigns "Dalla" to s3
cout << s3 << endl; // s3 now becomes Dalla
string s4("Welcome");
s4.assign(4, 'G'); // assigns "GGGG" to s4
cout << s4 << endl; // s4 now becomes GGGG
```

範例：at、clear、erase 以及 empty 函式

- 使用 at(index) 函式來存取在特定索引的字
- clear() 函式用以刪除字串
- erase(index, n) 函式用以消掉字串某些字元
- empty() 函式用以測試字串是否是空字串。

例如下述程式碼:

```
string s1("Welcome");
cout << s1.at(3) << endl; // s1.at(3) returns c
cout << s1.erase(2, 3) << endl; // s1 is now Weme
s1.clear(); // s1 is now empty
cout << s1.empty() << endl; // s1.empty returns 1 (means true)
```



範例： length、size、capacity 以及 c_str() 函式

```
string s1("Welcome");
cout << s1.length() << endl; // Length is 7
cout << s1.size() << endl; // Size is 7
cout << s1.capacity() << endl; // Capacity is 7
s1.erase(1, 2);
cout << s1.length() << endl; // Length is now 5
cout << s1.size() << endl; // Size is now 5
cout << s1.capacity() << endl; // Capacity is still 7
```

字串比較

經常在程式需要比較兩個字串的內容。此時可以使用 compare 函式。此函式將回傳大於 0、等於 0 或小於 0，分別表示字串大於、等於或小於另一字串。例如下述程式碼：

```
string s1("Welcome");
string s2("Welcomg");
cout << s1.compare(s2) << endl; // returns -1
cout << s2.compare(s1) << endl; // returns 1
cout << s1.compare("Welcome") << endl; // returns 0
```

取得子字串

您可透過 at 函式取得字串中的單一字元。亦可經由 substr 函式，取得某字串裡的子字串。例如下述程式碼：

```
string s1("Welcome");
cout << s1.substr(0, 1) << endl; // returns W
cout << s1.substr(3) << endl; // returns come
cout << s1.substr(3, 3) << endl; // returns com
```

在字串中搜尋

您可以使用 find 函式找尋字串中的子字串或是字元。例如下述程式碼:

```
string s1("Welcome to HTML");
cout << s1.find("co") << endl; // returns 3
cout << s1.find("co", 6) << endl; // returns -1
cout << s1.find('o') << endl; // returns 4
cout << s1.find('o', 6) << endl; // returns 9
```

插入字串與取代字串

您可以使用 insert 和 replace 函式分別在字串中插入和取代一子字串。例如下述程式碼:

```
string s1("Welcome to HTML");
s1.insert(11, "C++ and ");
cout << s1 << endl; // s1 becomes Welcome to C++ and HTML
string s2("AA");
s2.insert(1, 4, 'B');
cout << s2 << endl; // s2 becomes to ABBBBA
string s3("Welcome to HTML");
s3.replace(11, 4, "C++");
cout << s3 << endl; // returns Welcome to C++
```

字串運算子

運算子	說明
[]	利用陣列索引運算子擷取字元
=	拷貝一字串的內容於另一字串
+	連結兩個字串於一新字串
+=	附加一字串的內容於另一字串
<<	插入一字串於串流中
>>	從串流中擷取字元於一字串，當字元是白色空白或是結束字元則結束之
==、!=、<、<=、>、>=	比較字串大小的關係運算子

字串運算子

- string s1 = "ABC"; // 指派字串
- cout << s1[1];   // 輸出 'B'
- string s3 = s1 + "DEFG";
- cout << s3 << endl; // ABCDEFG
- s1 += "ABC";
- cout << s1 << endl; // ABCABC
- s1 = "ABC";
- s2 = "ABE";
- cout << (s1 == s2) << endl; // 0 (false)
- cout << (s1 != s2) << endl; // 1 (true)
- cout << (s1 > s2) << endl;   // 0 (false)
- cout << (s1 < s2) << endl;   // 1 (true)
- cout << (s1 >= s2) << endl; // 0 (false)
- cout << (s1 <= s2) << endl; // 1 (true)

將數值轉換為字串

stringstream 是 C++ 標準函式庫 <sstream> 中提供的一個類別，它允許**將數值與字串之間互相轉換**，並且可以像輸出(cout) 或輸入 (cin) 一樣操作字串。

以下是其範例:

```
1  stringstream ss;
2  ss << 3.1415;
3  string s = ss.str();
```

分割字串

您經常會從字串中擷取一些字(word)。假設這些字是以白色空白隔開。可使用前一小節所討論的 stringstream 類別來完成此項工作。範例程式 10.1 為從一字串擷取一些字，並加以顯示於不同行。

分割字串

```
1  #include <iostream>
2  #include <sstream>
3  #include <string>
4  using namespace std;
5
6  int main()
7  {
8      string text("Programming is fun");
9      stringstream ss(text);
10
11     cout << "The words in the text are " << endl;
12     string word;
13     while (!ss.eof())
14     {
15         ss >> word;
16         cout << word << endl;
17     }
18
19     return 0;
20 }
```

讀取字串

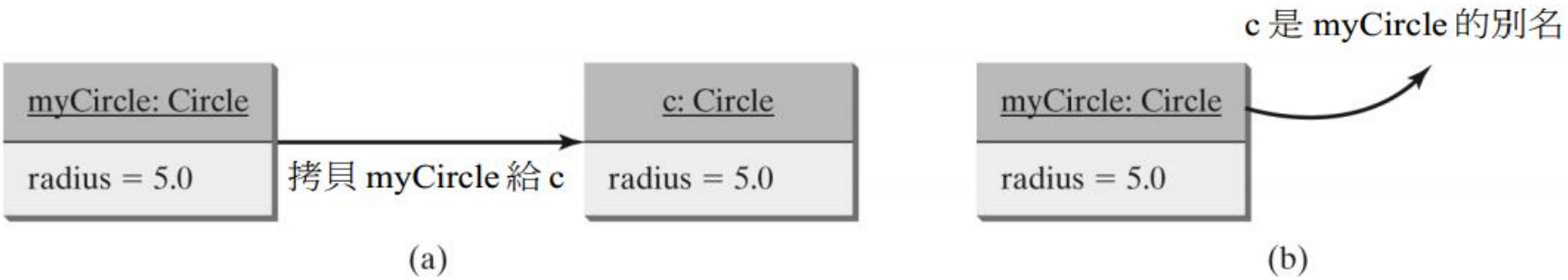
```
string city;
cout << "Enter a city: ";
cin >> city; // Read to array city
cout << "You entered " << city << endl;
```

```
string city;
cout << "Enter a city: ";
getline(cin, city, '\n'); // Same as getline(cin, city)
cout << "You entered " << city << endl;
```

傳送物件給函式

我們可以利用傳值(by value)或傳參考(by reference)的方式傳送物件給函式。

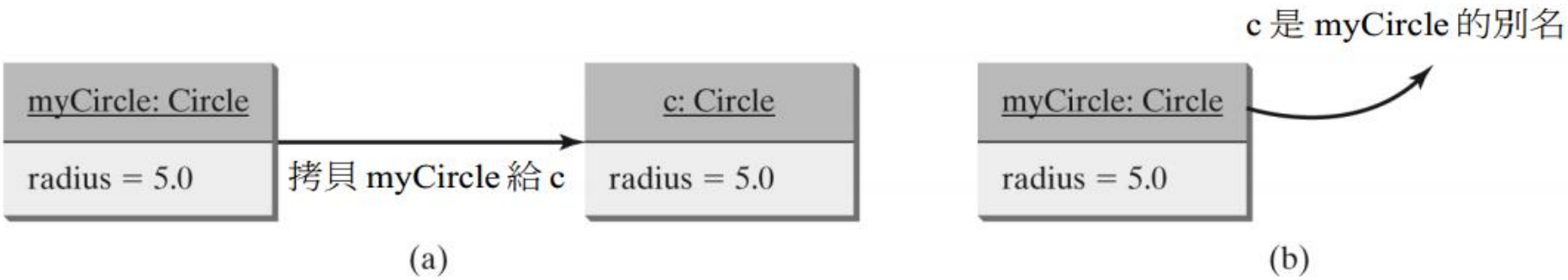
這兩種中以傳參考較有效率。



傳送物件給函式

我們可以利用傳值(by value)或傳參考(by reference)的方式傳送物件給函式。

這兩種中以傳參考較有效率。



傳送物件給函式-傳值(by value)

```cpp
#include <iostream>
#include "CircleWithPrivateDataFields.h"
using namespace std;

void printCircle(Circle c)
{
    cout << "The area of the circle of "
        << c.getRadius() << " is " << c.getArea() << endl;
}

int main()
{
    Circle myCircle(5.0);
    printCircle(myCircle);

    return 0;
}
```

```cpp
#include "CircleWithPrivateDataFields.h"

// Construct a default circle object
Circle::Circle()
{
    radius = 1;
}

// Construct a circle object
Circle::Circle(double newRadius)
{
    radius = newRadius;
}

// Return the area of this circle
double Circle::getArea()
{
    return radius * radius * 3.14159;
}

// Return the radius of this circle
double Circle::getRadius()
{
    return radius;
}

// Set a new radius
void Circle::setRadius(double newRadius)
{
    radius = (newRadius >= 0) ? newRadius : 0;

    /*三元運算子語法：
       (條件) ? 值1 : 值2;
       如果 條件 為 true，則回傳 值1。
       如果 條件 為 false，則回傳 值2。
       */

}
```

```cpp
#ifndef CIRCLE_H
#define CIRCLE_H

class Circle
{
public:
    Circle();
    Circle(double);
    double getArea();
    double getRadius();
    void setRadius(double);

private:
    double radius;
};

#endif
```

傳送物件給函式-傳參考(by reference)

```cpp
#include <iostream>
#include "CircleWithPrivateDataFields.h"
using namespace std;

void printCircle(Circle c)
{
    cout << "The area of the circle of "
        << c.getRadius() << " is " << c.getArea() << endl;
}

int main()
{
    Circle myCircle(5.0);
    printCircle(myCircle);

    return 0;
}
```

```cpp
#include "CircleWithPrivateDataFields.h"

// Construct a default circle object
Circle::Circle()
{
    radius = 1;
}

// Construct a circle object
Circle::Circle(double newRadius)
{
    radius = newRadius;
}

// Return the area of this circle
double Circle::getArea()
{
    return radius * radius * 3.14159;
}

// Return the radius of this circle
double Circle::getRadius()
{
    return radius;
}

// Set a new radius
void Circle::setRadius(double newRadius)
{
    radius = (newRadius >= 0) ? newRadius : 0;

    /*三元運算子語法：
       (條件) ? 值1 : 值2;
       如果 條件 為 true，則回傳 值1。
       如果 條件 為 false，則回傳 值2。
       */

}
```

```cpp
#ifndef CIRCLE_H
#define CIRCLE_H

class Circle
{
public:
    Circle();
    Circle(double);
    double getArea();
    double getRadius();
    void setRadius(double);

private:
    double radius;
};

#endif
```

物件陣列

Circle circleArray[10];//宣告有10個Circle物件的陣列，且會呼叫無參數建構函式

Circle circleArray[3] = {Circle(3), Circle(4), Circle(5)};

EX: TotalArea

實例與靜態成員

Circle circle1;

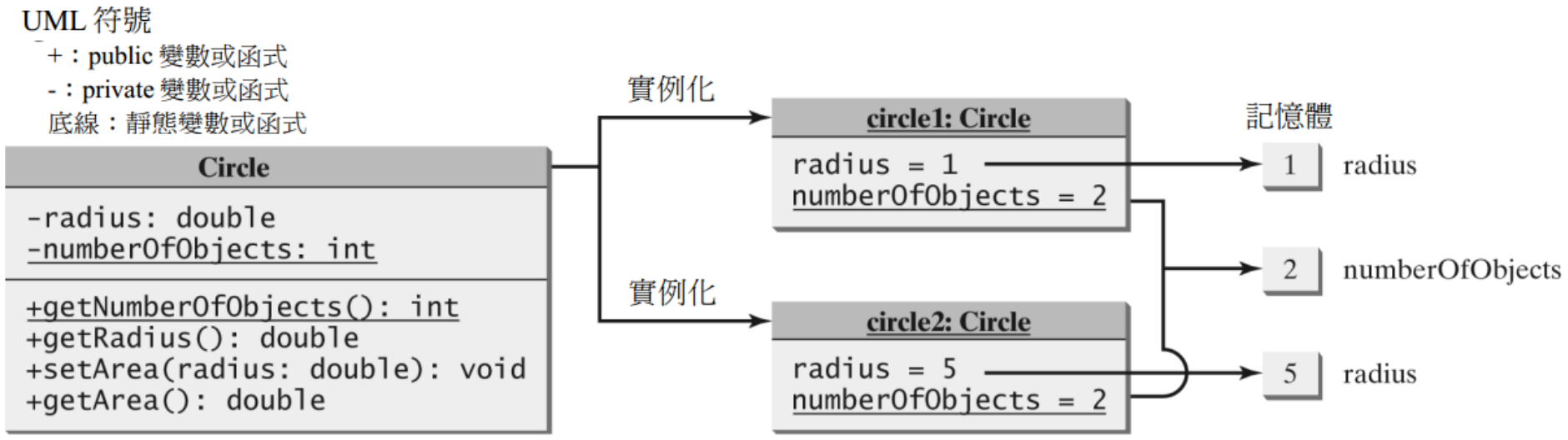
Circle circle2(5.0)

circle1的radius跟circle2的radius是互相獨立的，而且儲存在不同記憶體空間。

實例與靜態成員

若想要類別中所有的實例共享資料，就可以使用靜態變數(static variable)，其值是存在共同的記憶體位置。

靜態函數(static function)不需要使用類別實例來呼叫。



提示

- 使 用 ClassName::functionName(arguments) 來呼叫靜態函式，
- 以ClassName::staticVariable 來存取靜態變數。
- 這會改善可讀性，因為讀者可以很容易的看出getNumberOfObjects()為靜態函式。

提示

- 如何決定變數或函式應為實例或靜態?
  - ✓ 與指定類別的實例相依的話，則應宣告為實例變數與函式。
  - ✓ 與特定類別的實例不相依的話，則應宣告為靜態變數與函式。
- 例如每一圓形有它自己的半徑，所以半徑是相依於指定的圓形，因此 radius 是 Circle 類別的實例變數。
- 因為 getArea 函式是相依於指定的圓形，所以此函式是實例函式，
- 因為 numberOfObjects、getNumberOfObjects 不相依於任何指定的圓形，所以此函式應該宣告為靜態。

常數的成員函式

我們常使用 const 關鍵字以指定常數參數，告訴編譯器此參數不可以在函式中被更改。您也可以使用 const 關鍵字指定常數的成員函式(簡稱常數函式)，告訴編譯器此函式不可以更改物件的資料項目。如何告知，只要在函式宣告的標頭後加上 const 即可。

保護「不該被修改件物狀態」的函式，意外更改物件狀態

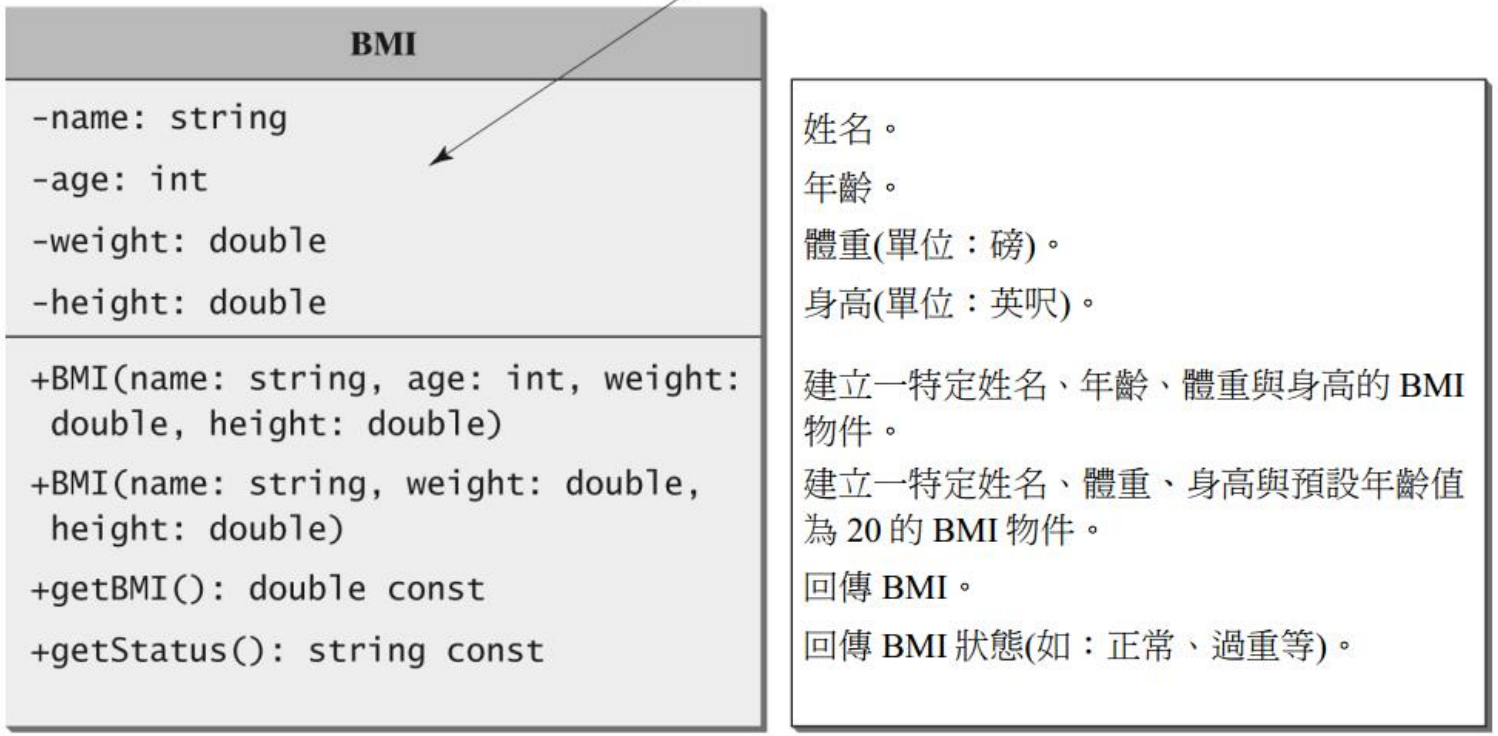
物件導向思維

本書介紹了使用迴圈、函式與陣列的基本程式設計技巧來解決問題，這些技巧為物件導向程式設計奠定了紮實的基礎。類別對於建立重複使用的軟體提供了彈性化與模組化的功能。

此節將使用物件導向方法改善在第 3 章的問題解法，透過這個物件導向方法，您可以清楚得知程序導向與物件導向程式設計的不同之處，並感受到使用物件與類別來開發重複使用程式碼的好處。

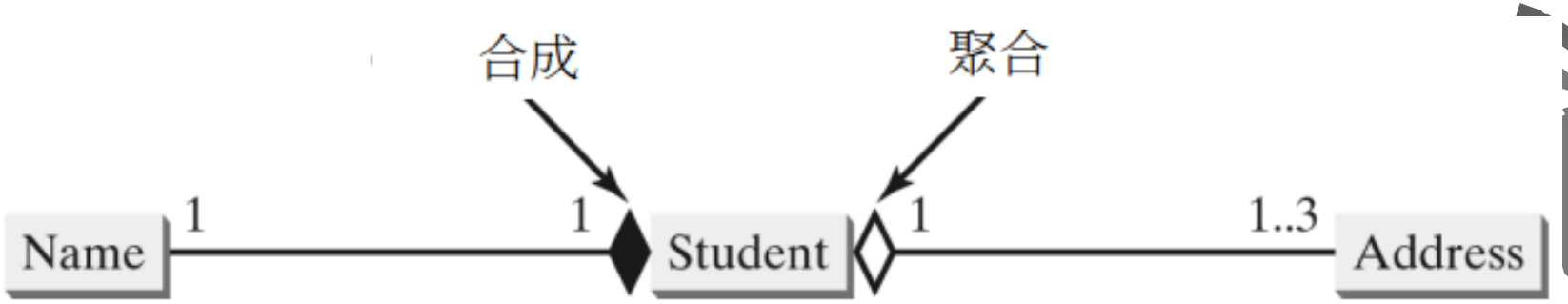
範例：BMI 類別

get 函式是處理類別所提供的資料項目，為了簡潔起見，在 UML 圖中加以省略



物件合成

合成是聚合(aggregation)關係中的特例。聚合關係是建構 has-a 的關係且是代表兩個物件間所有權的關係。擁有所有權的物件稱為聚合物件(aggregating object)且其類別稱為聚合類別(aggregating class)，被擁有的物件稱為被聚合物件(aggregated object)且其類別稱為被聚合類別(aggregated class)。



Class 的表示

聚合關係通常是被表示為聚合類別的資料成員，例如圖 10.12 的關係可以表示如下程式碼：

```
class Name
{
   ...
}
```

Aggregated class

```
class Student
{
private:
   Name name;
   Address address;

   ...
}
```

Aggregating class

```
class Address
{
   ...
}
```

Aggregated class

註釋

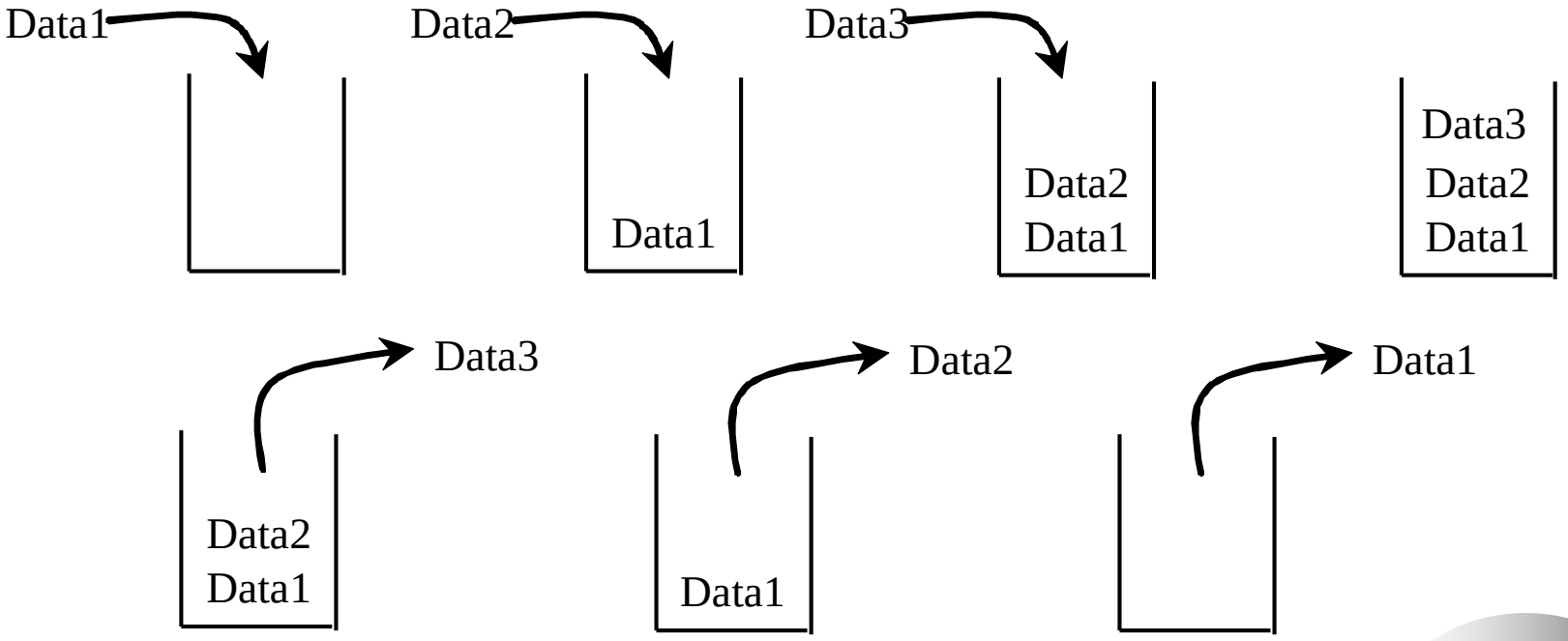
因為聚合與合成關係都是以相同方式來使用類別，我們無法區分此兩種關係，所以統稱為合成。

個案研究：StackOfIntegers 類別

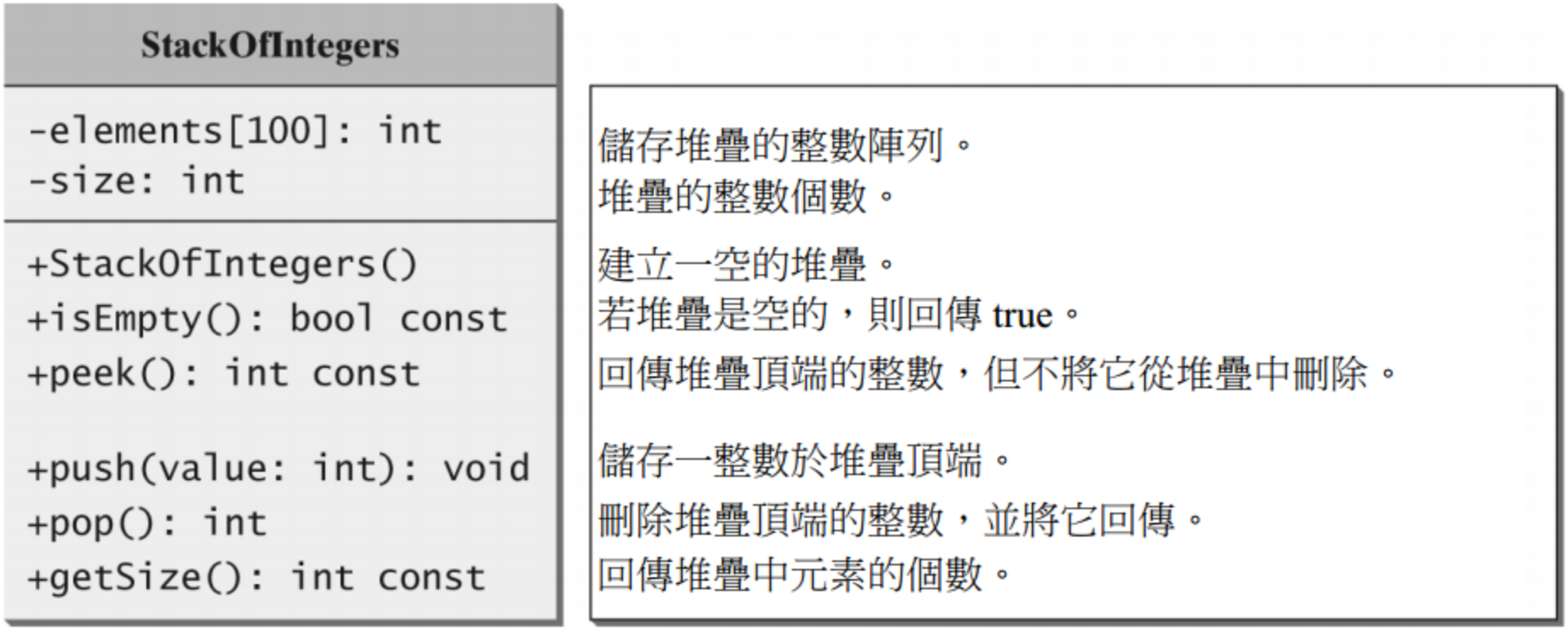
還記得堆疊是一個資料結構，其採用後進先出(last in first out；LIFO)的法則來存取資料。

堆疊有許多應用，例如：編譯器使用堆疊來處理函式的呼叫，當某函式被呼叫時，它的參數與區域變數將會被推(push)到堆疊中，當此函式又在呼叫另外函式時，新函式的參數與區域變數又會再被推到堆疊中，當此函式完成它的工作後就會回傳值到它的呼叫者(caller)，一些相關的空間就會從堆疊中釋放出來。

個案研究：StackOfIntegers 類別

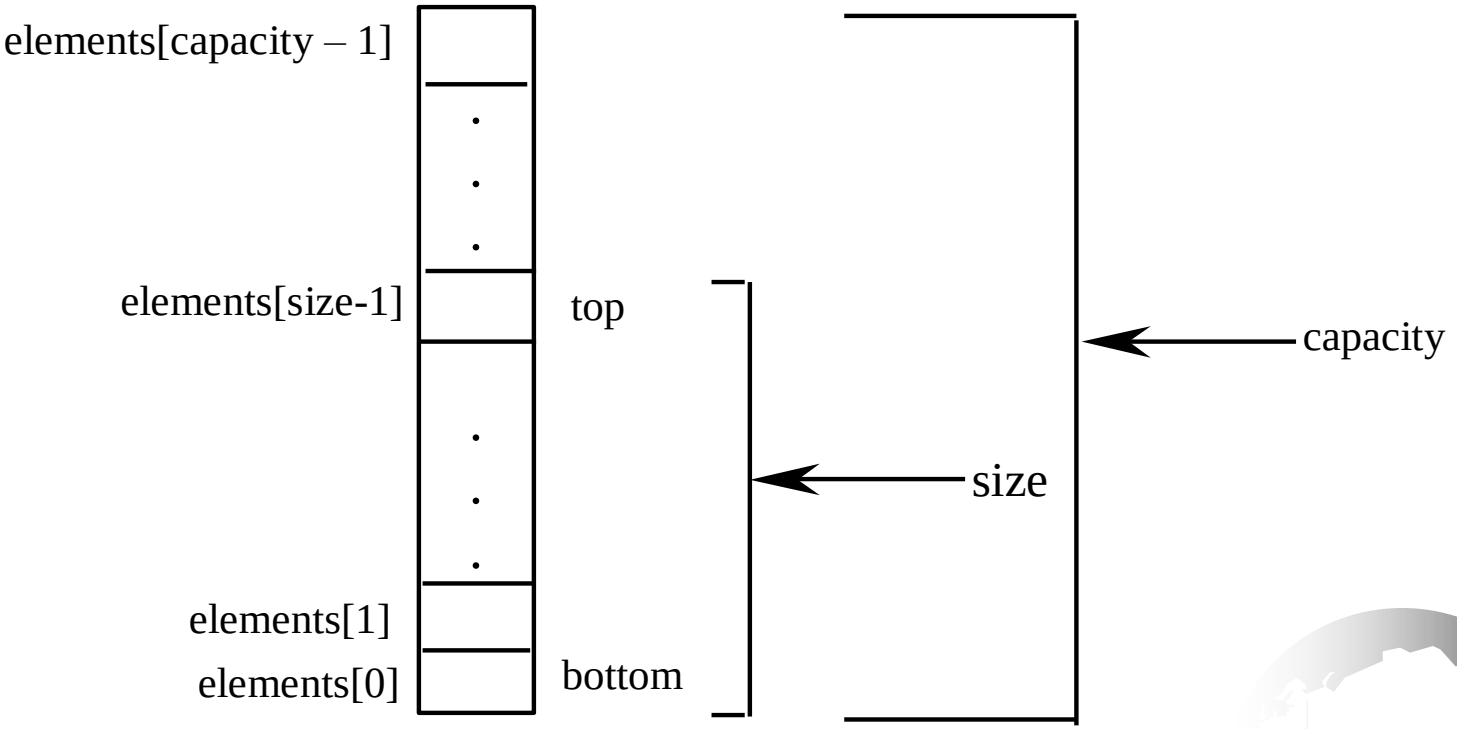


個案研究：StackOfIntegers 類別



StackOfIntegers.h | StackOfIntegers.cpp | TestStackOfIntegers | Run

實作



類別設計指引：內聚性(cohesion)

- 一個類別應該描述一個單一個體(entity)，且所有類別操作邏輯上必須符合內聚性的目的。

- 具有許多責任的單一個體是可以被分解為許多類別，而這些類別各自具有各自的責任。

類別設計指引：一致性(consistency)

- 依據標準 Java 程式設計風格，以及命名規則(naming convention)，來對於類別、資料成員與函式進行命名，比較流行的風格就是在建構函式之前做資料成員的宣告，在函式之前先做建構函式的定義。

類別設計指引：封裝性(encapsulation)

- 類別應該要使用 private 的關鍵字來隱藏資料，避免讓使用者直接存取，這樣可以使得類別易於維護。

- 如果您想要讓資料成員具有可讀性，則只要提供 get 函式；如果您想要讓資料成員具有可修改性，則只要提供 set 函式。類別也可以隱藏函式，不讓使用端試圖更改之，此類的函式也應定義為 private。

類別設計指引：清晰性(clarity)

- 使用者在許多不同的組合、順序與環境下皆能夠使用該類別，因此您應該設計一個沒有限制的類別讓使用者使用它、設計的資料成員可以讓使用者以任何順序來設定它，或是與其他值相結合，以及設計的函式在任何順序下獨立運作。

- 例如在範例程式 9.13 的 Loan 類別，包含 setLoanAmount、setNumberOfYears與 setAnnualInterestRate 函式。

類別設計指引：完整性(completeness)

- 類別是被設計來給許多使用者使用，為了能夠被廣泛的應用，類別透過屬性與函式應該提供各種不同方式來客製化，例如，string 類別包含了超過 20 個函式來對於各種不同的應用。

類別設計指引：實體與靜態比較

- 實體變數與實體函式是依據特定類別的物件。若是變數被類別所產生的所有物件共同使用的話，則稱為靜態變數。

- 例如，在範例程式 10.9 的 Circle 類別中的 numberOfObjects 資料成員是被所有 Circle 類別所產生的物件所共享的，因此宣告為靜態變數。若是一個函式與類別所產生的物件沒有關係，此函式將被定義為靜態函式，例如在 Circle 類別中的 getNumberOfObjects 函式與任何的物件沒有關係，因此被定義為靜態函式。