7. Classes and Data Encapsulation in Java

與 C++之類別相同，Java 之類別同樣支援對類別成員存取權限之設定與資料封裝，對於受保護的 private 資料成員之初值設定必需使用建構函式。

Java 與 C++之類別二者原則上大致相同，惟在程式規則上與格式上稍有不同，故本章主要以與 C++之類別比較的方式來介紹 Java 之類別。

7.1 Access Control for Class Member in Java

Java 類別(class)成員(資料或函式) 之存取權限，分成 private、public、protected、與 package 四類。

private & public:

same as C++。

no specifier(package):

類別本身及其同一 package 之程式均可直接存取。

A package member acts as public member within the same package, but private to other packages.

protected:

類別本身、其衍生類別或其同一 package 之程式均可對其直接存取，其他部分(其他 package，且非其衍生類別)之程式則不可。

A protected member is accessible to the class itself, its subclasses and classes in the same package.

公開程度；public -> protected -> package -> private

程式範例: oop_ex61.java (vs oop_ex_b.cpp)

注意要點:

1. 與 C++的指定格式不同，每一個成員(資料或函式)均需以關鍵字明確宣告其存取權限(無指定者實為指定其為 package)。

2. 同一檔案中定義的類別，屬於同一 package。
3. Java 之資料成員於定義時可指定其初值，或動態記憶體配置，此與 C++不同。
4. Java 之物件，均需以 new 產生。

7.2 Constructors (建構函式) in Java

與 C++相同，Java 之類別亦保留與類別同名，無傳回值的函式，作為建構函式，其遵守 Java 一般函式的規則，與 C++同樣可透過函式負載(overloading)的功能，提供多個建構函式供選擇使用，但不支援函式預設初值。

Java 之物件均需以 new 指令產生，故所有 Java 之建構函式均是使用 new 並以其後的參數呼叫。

C++：

```
User u1;                    //呼叫 default constructor 產生物件 u1;
User* u2 = new User(); //呼叫 default constructor 產生物件，並以 u2 指標定位
```

Java：

```
User u1;                    //產生參考 u1(未呼叫建構函式產生物件);
User u2 = new User();   //呼叫 default constructor 產生物件，並以 u2 參考定位
```

7.2.1 Default Constructor (預設建構函式)

與 C++相同，如一類別沒有定義任何建構函式的話，編譯器會自動為您產生一個預設建構函式，有定義其他建構函式則不會。

與 C++不同的是，這個預設建構函式，會將 reference type 變數初始化為 null，primitive type 變數初始化為 0。

程式範例: oop_ex62.java (vs oop_ex57.cpp)

7.2.2 Copy Constructors (拷貝建構函式)

與 C++相同，我們可為一類別定義拷貝建構函式。由於 Java 之參考近似於 C++之指標(位址值，非陣列或物件之實體)，故對於 reference type 之成員，需做記憶體配置(new)，之後逐員拷貝。

程式範例: oop_ex63.java (vs oop_ex58.cpp)

7.2.3 Copy Assignment Operator (拷貝指定運算子)

與 C++不同，Java 並沒有運算子負載(operator overloading)之功能，故沒有拷貝指定運算子之定義。

再一次重複要注意的是，對於兩同類別之參考 ref1 與 ref2，以下 statement 是將 ref1 也參照到 ref2 所參照到的物件，而非 ref1 與 ref2 所各自所參照之物件間的拷貝。

```
classX ref1 = new classX(….);
classX ref2 = new classX(….);
ref1 = ref2;
```

如欲以「=」複製物件，Java 提供了 Object Cloning 之機制，即一物件可依照本身的狀態複製產生一資料相同的新物件，並傳回新物件的位址，如下。

```
ref1  =  (classX) ref2.clone( );
```

其效果有如以下 C++使用拷貝指定運算子之結果，但過程並不同。

obj1 = obj2; (注意：classX obj1 = obj2; 是呼叫拷貝建構函式)

有關 Object Cloning 之機制，之後會特別介紹。

7.3 Garbage Collection in Java

Java 對於記憶體之管理是使用 Garbage Collection 之機制，故沒有也不需解構函式(Destructor)。