Module XII
套件、IMPORT與類別路徑

1. Java原始檔案格式
2. 套件(Package)
3. 套件編譯與執行
4. import(引用)套件
5. 靜態引用套件(static import)
6. 類別路徑(classpath)

Java原始檔案格式

- 在Java檔案中可能會出現三個稱為編譯單元(compilation units)的元素，這些元素皆非必要，但，如果有這些元素，則一定要依以下順序出現：
  - 1. package　宣告
  - 2. import　引用敘述
  - 3. class　類別
- 例如：
  - 1. package myPackageName;
  - 2. import yourPackageName1.*;
  - 3. import yourPackageName2.*;
  - 4. class MyClass {…}

套件 (Package)

- Java提供套件(Package)的機制，它就像一個管理容器，可以將所定義的名稱區隔管理在package底下，而不會有類別名稱相互衝突的情況發生
- Java的package被設計為與檔案系統結構相對應，而以檔案管理的觀點著手，將性質相似之類別集合在一起
  - 譬如java.sql表示名稱java的目錄底下有一個子目錄名叫sql，並且其內存放的都是Java資料庫連線相關的類別檔
- 套件宣告：
  - 宣告於原始檔案的第一行
  - package 套件名稱(myPackageName);
  - package com.ibm;

套件編譯與執行

- 所有屬於 myPackageName 類別庫的.class檔案都必須儲存在myPackageName資料夾下，若不使用package宣告，Java預設會將類別檔置於目前工作環境所在的目錄中
- 因為source檔(.java檔)與.class檔不一定要放在同一個目錄下，所以使用package宣告時，必須在編譯時透過 –d option，指定類別檔要置於哪個目錄之中

編譯： javac **–d .** HelloWorld.java

【註：「.」指編譯後的class檔置於目前的目錄位置】

執行： java packageName.HelloWorld

【註：要在原來的目錄下執行】

import (引用) 套件

- 關鍵字 (keyword)中的 import 可用來引入API中的功能，或是自行定義的套件(package)
- Java會自動引用的兩個套件：

  - java.lang.*：常用的類別，如String類別已置於此套件中
  - 目前工作環境所在的package
- 若使用上述之外的其它套件，則必須用 **import** 引用敘述

  - 如：**import java.xxx.*;**
    - 註：**不包含其子目錄的類別**

      如import java.xxx.yyy.*; 是引用不同的套件

import (引用) 套件或特定別

- 引用套件中**所有類別**：

  - **import** java.sql.*;

    Date date = new Date(…);

- 引用套件中**特定的類別**：

  - **import** java.sql.**Date**;

    Date date = new Date(…);

- **如不使用import敘述，則必須使用類別長名稱**

  - java.**sql.Date** date = new java.**sql.Date**(…);

  - java.**util.Date** date = new java.**util.Date**(…);

範例：TestImport.java

靜態 (引用) 套件

- 靜態引用套件 【JDK 5加入的功能】
  - 靜態引用可導入類別內的所有static fields與static methods，亦即使用這些static members無需再指定其類別名稱
  - 使用 * 星號字元可導入類別內所有靜態成員
- 如：
  - import static java.lang.Math.*;

    ……

    r = sin(PI * 2);　//等於 r = Math.sin(PI * 2);
- 避免過度使用static import功能，否則容易造成混淆而不利於維護

範例：TestStaticImport.java

類別路徑 (classpath) (1/2)

- classpath可以讓Java應用程式在編譯和執行時，可以找到要用的相關類別
- 根據JDK文件說明，Java以下面三類classpath順序，依序找尋所需的class

1. Bootstrap classes(Core classes)：
    - Java2 Platform核心類別函式庫
    - 現有，已置於%JAVA_HOME**%\jre\lib\rt.jar**檔案
    - JDK預設會自動載入，不必額外再作設定
2. Extension classes
    - Java2 Platform擴充的類別函式庫
    - 指的是%JAVA_HOME%**\jre\lib\ext目錄**下的jar檔或zip檔，third-party的類別函式庫可以放在這目錄下
    - JDK預設會自動載入此目錄內所有的jar檔或zip檔，不必額外再作設定

類別路徑 (classpath) (2/2)

3. Users classes

・是使用者自己寫的類別函式庫(third-party的類別函式庫也可)

・使用者必需額外作設定，JDK才會載入類別

・是指我們在環境變數classpath設定路徑下的classes或jar檔

如：在作業系統的環境變數，預先新增classpath變數：

可能為 .;C:\myLib\xxx.jar;C:\myLib\yyy.jar;C:\myClass;

JDK 6可以一次簡化為 **C:\myLib***

或如：在命令列中：

set classpath = %classpath%;C:\myLib2\zzz.jar;C:\myClass2;

或如：在編譯及執行時：

javac –classpath “%classpath%;C:\myLib2\zzz.jar;C:\myClass2;“ HelloWorld.java

java –classpath “%classpath%;C:\myLib2\zzz.jar;C:\myClass2;” HelloWorld

(或-cp)

章節整理

- Java的import觀念與C的前置處理器概念不同，因為Java是動態編譯(JIT)，當需要該類別時才去讀取，而非C是一開始就直接載入類別函式庫
- Package概念就如檔案系統的資料夾結構管理，通常我們在開發時，會以package做為分工與功能區別的標準
- 雖然現有強大IDE協助編譯，但我們還是要瞭解.class檔產生的相關位置與編譯觀念，有助我們學習上釐清
- 培養Java各種常用類別的套件名稱，如java.util、java.io、java.sql等，將有助於API的資料查找