

# Java程式設計進階 套件管理

鄭安翔

ansel\_cheng@hotmail.com

# 課程大綱

- 1) 套件 **Package**
- 2) 專案開發及部署
- 3) 靜態引入

# 套件(package)的使用

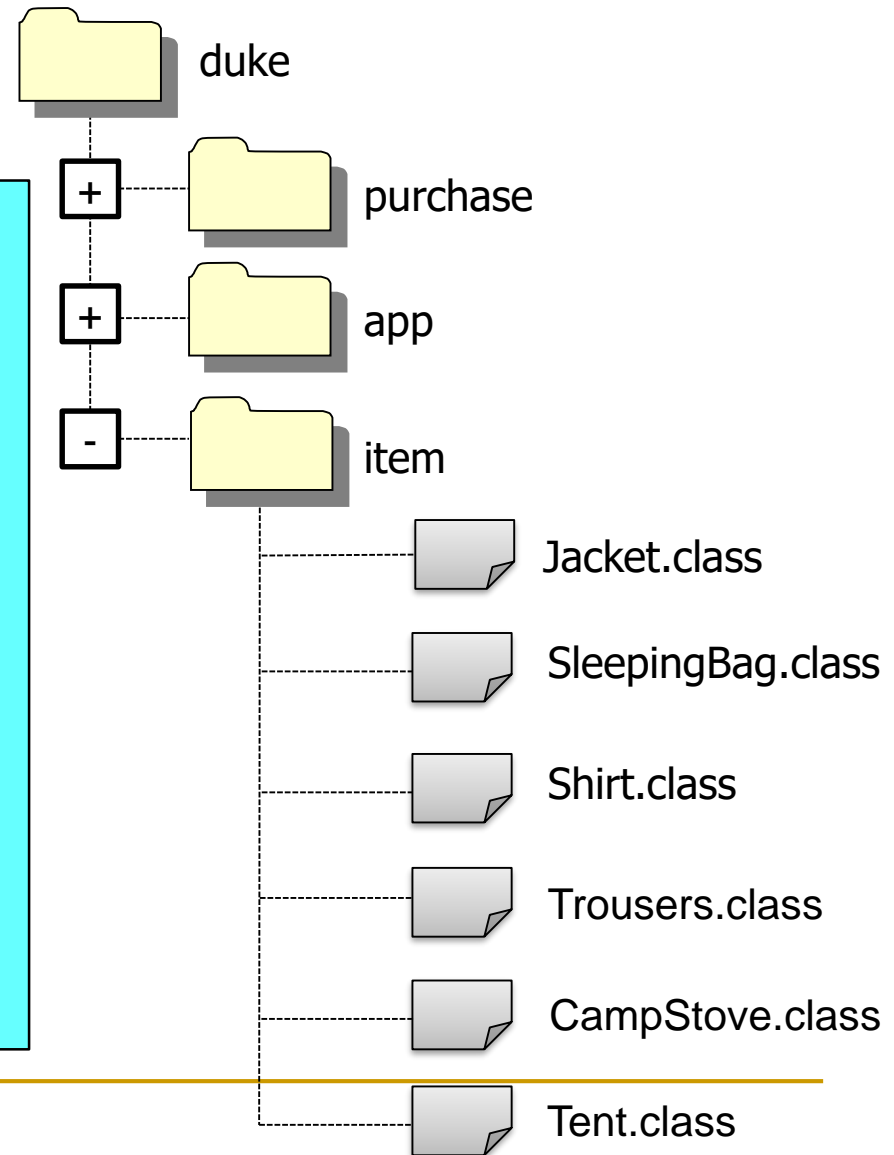
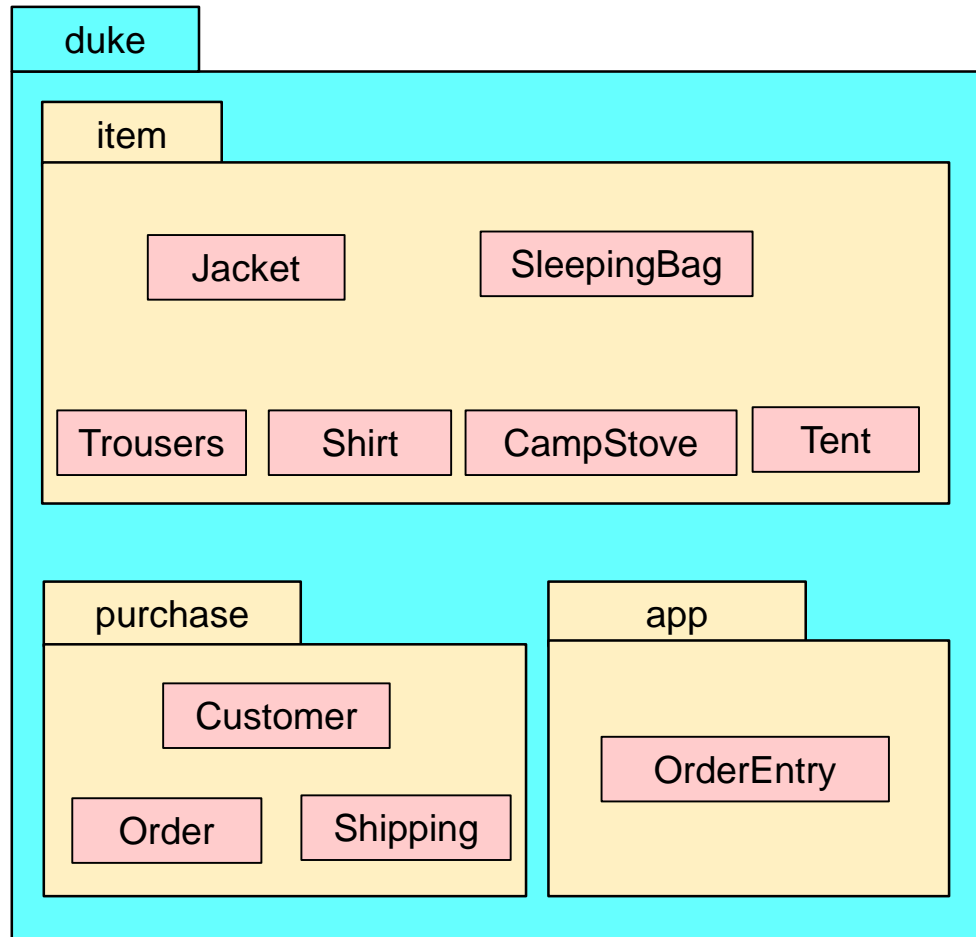
## ■ 目的

- 將許多不同的類別檔案實際分門別類放在一起
- 使類別名稱不相衝突
- 提供package(default)的存取保護

## ■ 實體型態

- 同電腦之檔案目錄系統

# Package 套件設計



# Package陳述式

## ■ Package 宣告

`package` <package名稱>[.<subPackage名稱>]\*;

- 一個 Java 原始檔中只能有一個套件的宣告
- 套件宣告一定要在第一行
- 沒有套件宣告的類別則屬於預設套件 (default package), 就是目前目錄

## ■ Package下編譯與執行

- 切換至application根目錄
- 編譯 `javac <package 路徑>\類別名稱.java`
- 執行 `java <package 路徑>.<類別名稱>`

# Package 類別存取

- 同package之下的類別
  - 可直接用短名存取 (類別名稱本身)
- 使用其他package下的類別
  - 以類別的完整長名存取 (套件名稱~... . 類別名稱)
  - 引入套件後，可以類別的短名存取
    - 下列套件不需 import
      - java.lang.\*
      - default package(目前工作目錄)

# 引入套件/類別

## ■ 引入套件/類別語法

- **import** 敘述告知 JVM：程式中使用短名的類別可能的全名為何

- 引入套件中指定類別

```
import <package>[.<subPackage>]*.<class>;
```

- 引入套件中所有類別

```
import <package>[.<subPackage>]*.*;
```

# 類別的完整長名存取

- OrderEntry1 類別存取 duke.purchase.Order 及 duke.item.Shirt 類別

```
01 package duke.purchase;  
02 public class Order {  
03     .....  
04     .....  
05 }
```

```
01 package duke.item;  
02 public class Shirt {  
03     .....  
04     .....  
05 }
```

```
01 package duke.app;  
02 public class OrderEntry1 {  
03     public static void main(String[] args) {  
04         duke.purchase.Order order = new duke.purchase.Order();  
05         duke.item.Shirt shirt = new duke.item.Shirt();  
06         order.addShirt(shirt);  
07         .....  
08     }  
09 }
```



# 引入套件以類別短名存取

- OrderEntry2 類別存取 duke.purchase.Order 及 duke.item.Shirt 類別

```
01 package duke.purchase;  
02 public class Order {  
03     .....  
04     .....  
05 }
```

```
01 package duke.item;  
02 public class Shirt {  
03     .....  
04     .....  
05 }
```

```
01 package duke.app;  
02 import duke.purchase.*;  
03 import duke.item.Shirt;  
04 public class OrderEntry2 {  
05     public static void main(String[] args) {  
06         Order order = new Order();  
07         Shirt shirt = new Shirt();  
08         order.addShirt(shirt);  
09         .....  
10     }  
11 }
```

# import 注意事項

- package、import、class 出現順序
  - 套件宣告
  - 套件/類別引入
  - 類別定義

```
01 package duke.purchase;  
02 import java.util.ArrayList;  
03 import java.io.*;  
04 public class Order {  
05     .....  
06 }
```

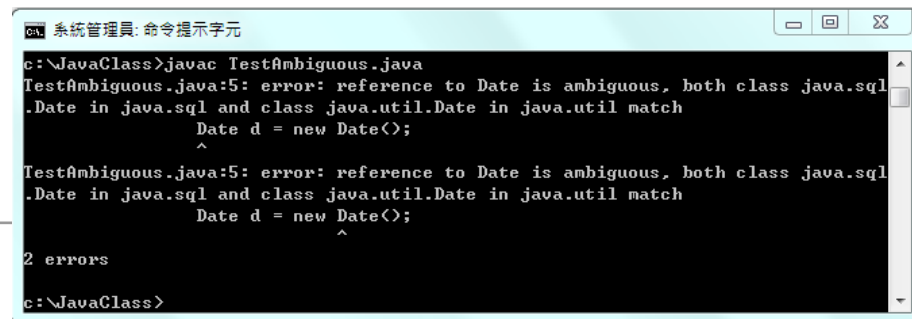
# import 注意事項

## ■ 避免名稱混淆

- 同一個短名，在類別路徑 **ClassPath** 中可找到兩個不同可能的長名時

- 發生編譯錯誤
- 仍需用類別長名來存取
- 類別引入優先於套件引入

```
01 import java.util.Date;
02 import java.sql.*;
03 public class TestAmbiguous {
04     public static void main(String[] args){
05         java.util.Date d = new java.util.Date();
06     }
07 }
```



The screenshot shows a Windows command prompt window titled "系統管理員: 命令提示字元". The user has run the command `c:\JavaClass>javac TestAmbiguous.java`. The output shows two identical error messages: `TestAmbiguous.java:5: error: reference to Date is ambiguous, both class java.sql.Date in java.sql and class java.util.Date in java.util match`. The error points to the line `Date d = new Date();` in the code. The prompt ends with `2 errors` and `c:\JavaClass>`.

```
c:\JavaClass>javac TestAmbiguous.java
TestAmbiguous.java:5: error: reference to Date is ambiguous, both class java.sql
.Date in java.sql and class java.util.Date in java.util match
        Date d = new Date();
        ^
TestAmbiguous.java:5: error: reference to Date is ambiguous, both class java.sql
.Date in java.sql and class java.util.Date in java.util match
        Date d = new Date();
        ^
2 errors
c:\JavaClass>
```

# 預設存取權限

## ■ 預設存取權限

- 類別、屬性、方法及建構子前不加權限修飾字
- 其存取權限為同一個套件的類別才可存取，其他套件的類別不可存取

```
01 package duke.purchase;
02 public class Order {
03     //1:9:00-18:00, 2:18:00-22:00
04     int deliveryTime = 1;
05 }
```

```
01 package duke.purchase
02 public class Shipping {
03     public void addOrder(Order o){
04         switch(o.deliveryTime){
05             }
06     }
07 }
```

同套件類別可存取

```
01 package duke.app;
02 import duke.purchase.*;
03 public class OrderEntry3 {
04     public static void main(String[] args) {
05         Order order = new Order();
06         switch(order.deliveryTime){
07             }
08     }
09 }
```

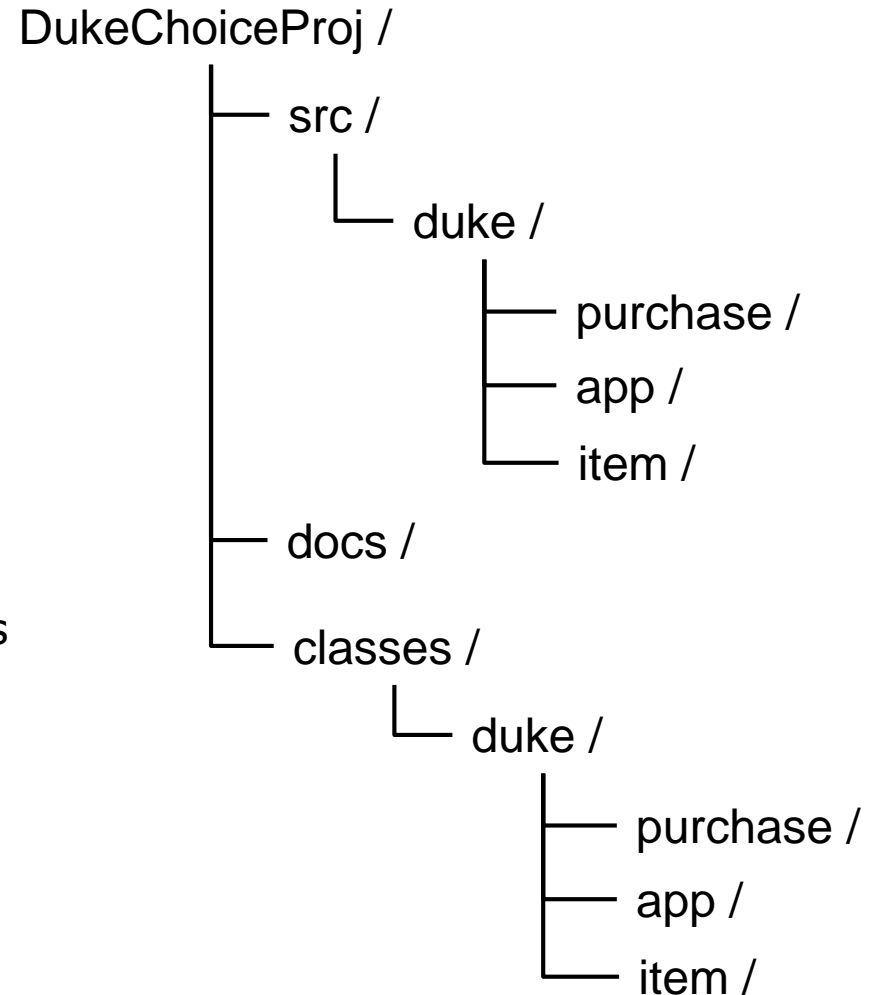
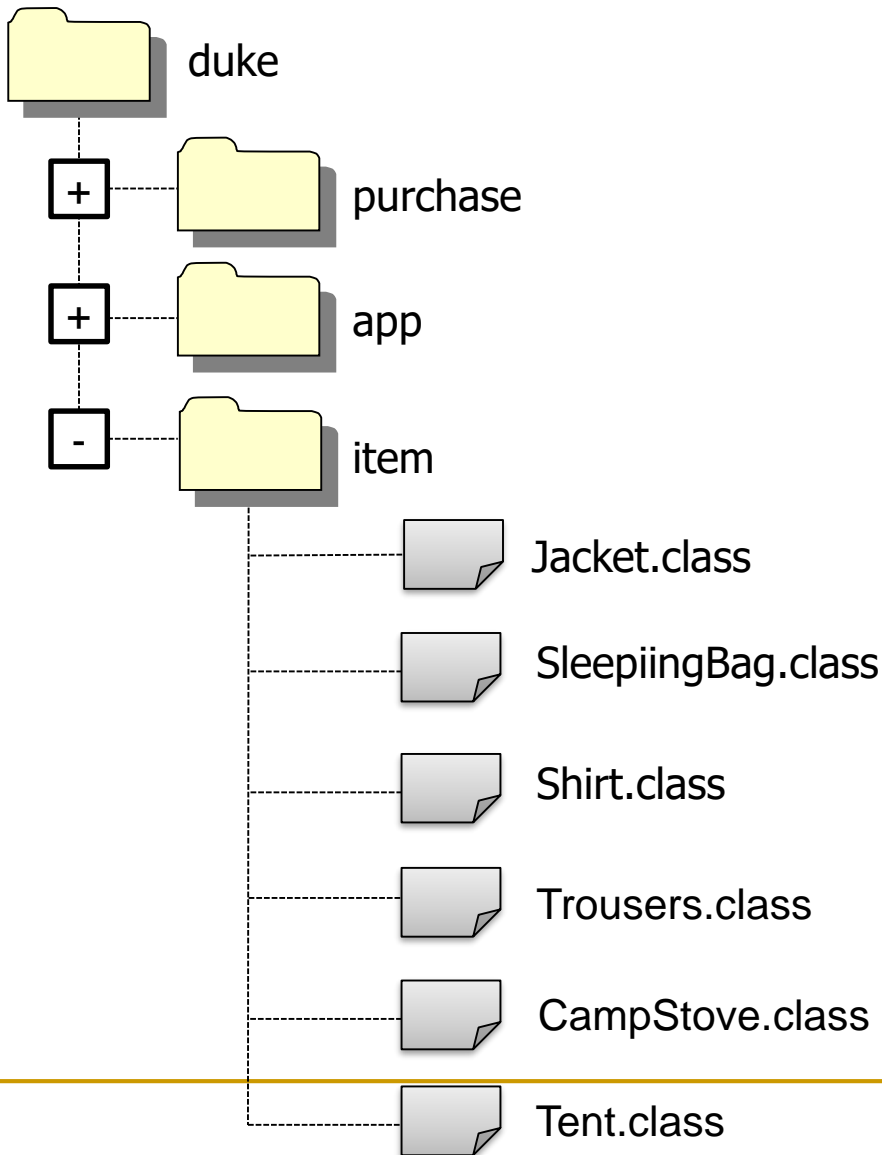
不同套件類別不可存取

---

# 課程大綱

- 1) 套件 Package
- 2) 專案開發及部署
- 3) 靜態引入

# 開發時期的目錄架構



# 開發時期的目錄架構

- 管理原始碼與位元碼檔案：**src**、**classes**資料夾
  - **src**資料夾中存放**Java**原始檔檔案(.java)
  - 編譯好的位元碼檔案，通常指定存放至**classes**資料夾
  - 編譯時使用**-d**選項指定類別檔置放目錄

```
cd DukeChoiceProj/src
javac -d ../classes duke/app/OrderEntry.java
```
  - 在**src**目錄下執行**duke.app.OrderEntry**

```
java -cp ../classes duke.app.OrderEntry
```

# 佈署

## ■ Java Archive (JAR)

- 將 `package` 下的多個 `java classes` 壓縮成一個 `.jar` 檔
  - 以便於攜帶與部署。
- 包裝與壓製工具
  - JDK 在 `<JAVA_HOME>\bin` 目錄下提供了 `jar.exe`
- 壓製格式：預設使用 **ZIP** 壓縮格式



# 佈署

- 製作可執行Executable Java Archive (JAR)

- 撰寫暫存檔，內容指定主類別

**Main-Class: MyPackage.MyClass**

- 使用jar工具，製作包含META-INF/MANIFEST.MF清單檔的Java Archive檔

**jar cvmf tempfile MyProgram.jar MyApp**

- 執行 jar 程式 (部分作業系統可以點選jar檔兩次執行)

**java -jar /path/ MyProgram.jar**

# JAR工具應用

## ■ 常見的 jar 工具用法、功能及命令：

<b>-c</b>	建立新的歸檔（歸檔指的就是 jar 檔）
<b>-t</b>	列出歸檔的目錄
<b>-x</b>	從歸檔擷取以命名的(或全部)檔案
<b>-u</b>	更新現有的歸檔
<b>-v</b>	在標準輸出中產生 verbose 輸出
<b>-f</b>	指定歸檔檔名
<b>-m</b>	包含來自指定 manifest 檔案的顯示資訊
<b>-O</b>	不使用 ZIP 壓縮
<b>-M</b>	不建立項目的 manifest 檔案
<b>-i</b>	為指定的 jar 檔產生索引資訊
<b>-C</b>	變更為指定的目錄並包含下列檔案

# JAR工具應用

## ■ META-INF 目錄

- 說明部署時的相關信息，如安全性和版本信息。
- JavaSE 平台識別並解釋 **META-INF** 目錄中的文件，以便部署與配置應用程式，其文件包含：
  - ✓ **MANIFEST.MF**：清單檔
  - ✓ **INDEX.LIST**：索引資訊檔
  - ✓ **xxx.SF**：**JAR** 文件的簽名文件
- **MANIFEST.MF** 文件檔常用的設定值：

Manifest-Version: 1.0	Manifest 的版本
Ant-Version: Apache Ant 1.6.1	Ant 的版本
Created-By: Sean Cheng	創始人
Main-Class: com.MyForm	主程式
Class-Path: .	類別路徑

# 課程大綱

- 1) 套件 **Package**
- 2) 專案開發及部署
- 3) **靜態引入**

# 靜態引入 static import

## ■ 靜態引入

### □ Java SE 5.0 新增功能

- 不須指名類別名稱，直接使用其類別成員

### □ 語法

- 引入類別的特定靜態(static)成員(屬性及方法)
- 使用 “\*” 字元，引入類別下所有的靜態成員(屬性及方法)

```
import static java.lang.Integer.*;
import static java.lang.System.out;
public class ManyHelloWorld {
    public static void main(String[ ] args) {
        int number = parseInt(args[0]);
        for (int i=0; i<number; i++)
            out.println("Hello! World!");
    }
}
```

out是System類別的靜態屬性

直接使用out

引入Integer類別所有的靜態成員

parseInt(String s)是Integer類別的靜態方法  
直接使用parseInt("...")

# 靜態引入 static import

```
01 import static java.lang.Math.PI;
02 public class Circle {
03     private double radius;
04
05     public Circle(double radius){
06         this.radius = radius;
07     }
08
09     public double area() {
10         return PI * radius * radius;
11     }
12
13     public double circumference() {
14         return 2 * PI * radius;
15     }
16 }
```

用途：可以少打一些重複的程式碼

注意事項：

- ▣ 過度使用時，會使程式碼難以理解及維護
- ▣ 容易有名稱衝突的問題