

Linguaggi

*Corso M-Z - Laurea in Ingegneria Informatica
A.A. 2007-2008*

Alessandro Longheu

<http://www.dit.unict.it/users/alongheu>

alessandro.longheu@dit.unict.it

- lezione 18 -

Package in Java

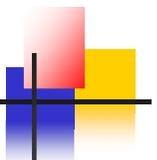
1

A. Longheu – Linguaggi M-Z – Ing. Inf. 2007-2008

Il concetto di Package

- Una applicazione è spesso composta di molte classi (eventualmente correlate)
- Un package è un gruppo di classi che costituiscono una unità concettuale.
 - un package può comprendere parecchie classi
 - anche definite in file separati
- Una dichiarazione di package ha la forma:
package < nomepackage > ;
- Se presente, deve essere all'inizio di un file.

2



Esempio

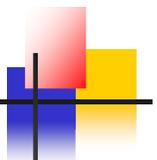
```
package pippo;  
public class Counter {  
    ...  
}
```

File Counter.java

```
package pippo;  
public class Esempio4 {  
    public static void main(String args[]) {  
        ...  
    }  
}
```

File Esempio4.java

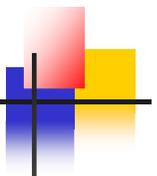
3



Package e File System

- Esiste una corrispondenza biunivoca fra
 - nome del package
 - posizione nel file system delle classi del package
- Un package di nome pippo richiede che tutte le sue classi si trovino in una cartella (directory) di nome pippo
- Per compilare una classe Counter che fa parte di un package pippo occorre:
 - porsi nella cartella superiore a pippo e invocare il compilatore con il percorso completo della classe:
javac pippo/Counter.java
- Per eseguire una classe Esempio4 che fa parte di un package pippo occorre:
 - porsi nella cartella superiore a pippo
 - invocare l'interprete con il nome assoluto della classe:
java pippo.Esempio4

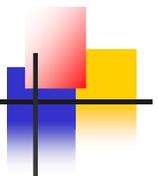
4



Package di default

- Se una classe non dichiara di appartenere ad alcun package, è automaticamente assegnata al package di default
- Per convenzione, questo package fa riferimento alla cartella (directory) corrente
 - è l'approccio finora usato in tutti gli esempi
 - si possono compilare ed eseguire i file nella cartella in cui si trovano, senza premettere percorsi o nomi assoluti
- Sono possibili nomi di package strutturati, come:
 - `java.awt.print` `pippo.pluto.papero`

5



Package di default

- **ATTENZIONE:** questo non significa che un package ne contenga un altro!
 - `pippo.pluto.papero` è un package indipendente da un eventuale package `pippo.pluto`
 - Le classi di tali package hanno perciò un nome assoluto strutturato:
 - `java.awt.print.Book` `pippo.pluto.papero.Counter`
- Difetti
 - Ogni volta che si usa una classe, Java richiede che venga denotata con il suo nome assoluto
 - Questo è chiaramente scomodo se il nome è lungo e la classe è usata frequentemente.

6

Importazione di nomi

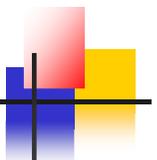
- Si introduce il concetto di importazione di nome, per evitare di dover riscrivere più volte il nome assoluto di una classe:
import java.awt.print.Book;
- Da questo momento, è possibile scrivere semplicemente *Book* invece del nome completo *java.awt.print.Book*
- Per importare tutti i nomi pubblici di un package, si scrive *import java.awt.print.*;*
- Attenzione: l'import è ben diversa dalla #include del C
 - in C, il pre-processor gestisce la #include copiando il contenuto del file specificato nella posizione della #include stessa
 - in Java non esiste alcun pre-processor, e non si copia assolutamente nulla si stabilisce solo una "scorciatoia" per scrivere un nome corto al posto di uno lungo. \

7

Package e visibilità

- Oltre a pubblico / privato, in Java esiste un terzo tipo di visibilità: la visibilità package
- E l'impostazione predefinita per classi e metodi non esiste quindi una parola chiave per indicarla
- Significa che dati e metodi sono accessibili solo per le altre classi dello stesso package in qualunque file siano definite
- Altre classi, definite in altri package, non possono accedere a dati e metodi di questo package qualificati a "visibilità package", esattamente come se fossero privati.
- A differenza del C, il file rimane solo un contenitore fisico, non definisce più uno scope di visibilità.
- Non è quindi possibile, né sensato, pensare di definire una classe visibile in un solo file: la visibilità si esprime solo con riferimento ai package.

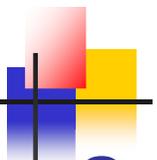
8



Package java.lang

- Il nucleo centrale del linguaggio Java è definito nel package java.lang
- E sempre importato automaticamente: la frase import java.lang.* è sottintesa
- java.lang definisce i tipi primitivi e buona parte della classi di sistema
- Molte altre classi standard sono però definite altrove: ci sono più di 50 package
 - java.awt, java.util, java.io, java.text, javax.swing, ...

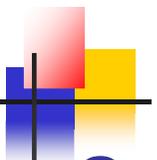
9



Distribuzione di un'applicazione con jar

- dopo aver scritto un'applicazione Java bisogna distribuirla, ma un'applicazione Java è fatta di molte classi: Come distribuirle insieme in modo efficiente?
- Zipparle insieme è una prima soluzione, ma non si hanno informazioni su come è organizzata l'applicazione
- JDK introduce un formato evoluto, il jar
 - fisicamente un file zip che si apre con winzip
 - contiene informazioni aggiuntive nel file MANIFEST
 - grazie ad esso si può eseguire una applicazione java senza decomprimerla (basta fare doppio clic sul file .jar)

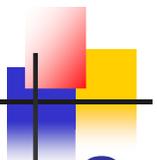
10



Distribuzione di un'applicazione con jar

- Per creare un file jar base
- *jar cf nomearchivio.jar classi*
- classi è l'elenco delle classi da includere (file .class)
- la forma *.class permette di aggiungere tutte le classi di una cartella
- l'elenco può contenere file di cartelle diverse
- questo file jar non può essere direttamente eseguito perché manca l'indicazione della classe che contiene il main

11



Distribuzione di un'applicazione con jar

- Per creare un file jar completo
- *jar cmf info.txt nomearchivio.jar classi*
- infot.txt è un file testo che contiene la riga
- *Main-Class: nomeclasseMain*
- questa riga viene aggiunta al file MANIFEST
- Per eseguire un file jar
- *java -jar nomefile.jar*

12