

Programmazione Orientata agli Oggetti

Qualità del codice:
Java Base Library
Documentazione
Package

Sommario

- Java Base Libraries
- Consultare la documentazione
- Produrre la documentazione delle proprie classi
- Package

Java Base Libraries

- Migliaia di classi
- Decine di migliaia di metodi
- Molte classi utili che ci semplificano drasticamente la vita
- Un programmatore competente deve essere in grado di lavorare con le librerie

Java Base Libraries

- Un programmatore competente dovrebbe saper costruire le proprie classi, ma anche sapere quando è inutile scriverne di nuove
- Per poter usare efficacemente una libreria, bisogna:
 - Conoscere alcune sue importanti classi per nome
 - Sapere come trovare e usare altre classi
- Importante:
 - Serve conoscere solo l'interfaccia, non l'implementazione

Documentazione di Librerie (1)

- La documentazione delle librerie Java è in formato HTML
 - **javadoc**
- Si può leggere agevolmente con un browser
- Class API: *Application Programmers' Interface*
 - Descrizione delle interfacce per tutte le classi della libreria

Documentazione di Librerie (2)

- *La documentazione include*
 - Il nome della classe
 - Una descrizione generale della classe
 - Una lista dei costruttori e dei metodi
 - Valori di ritorno e parametri per costruttori e metodi
 - Una descrizione dello scopo di ciascun costruttore e di ciascun metodo
- Come si usa la classe: tutto quello che serve al
programmatore-utilizzatore

Documentazione di Librerie (3)

- *La documentazione non include*
 - Campi privati
(tutti i campi dovrebbero essere privati)
 - Metodi privati
 - Il corpo dei metodi e dei costruttori
- Dettagli sull'implementazione: non servono al *programmatore-utilizzatore*
- Anzi: potrebbe risultare controproducente doverli necessariamente conoscere

Produrre Documentazione (1)

- Le classi che progettiamo dovrebbero essere documentate come le classi della libreria
- Altri programmatori devono essere in grado di usare le nostre classi senza conoscere l'implementazione di dettaglio

Elementi della Documentazione

- *La documentazione di una classe dovrebbe includere*
 - Il nome della classe
 - Un commento che descriva lo scopo e caratteristiche generali della classe
 - Un numero di versione
 - Il nome degli autori
 - Riferimenti ad altre classi
 - Documentazione per ciascun costruttore e per ciascun metodo

Documentare Costruttori e Metodi

- *La documentazione di ciascun costruttore / metodo dovrebbe includere*
 - Nome e tipo di ciascun parametro
 - Una breve descrizione di ciascun parametro
 - Una descrizione dello scopo e della funzione del costruttore/metodo
 - Il nome del metodo
 - Il tipo di ritorno
 - Una descrizione del valore ritornato

Produrre Documentazione (2)

- Un'idea tanto semplice quanto efficace
 - Documentazione in formato ipertestuale
 - pagine web
 - La documentazione viene generata dai commenti immersi direttamente nel codice
 - Basta usare una semplice sintassi pensata allo scopo
 - e l'utility **javadoc**

Commenti di Documentazione

- Tutti i comandi javadoc si trovano solo entro commenti `/** ... */`
- Speciali marcatori (immersi nei commenti) permettono di definire aspetti specifici della documentazione
- Per l'elenco completo dei marcatori javadoc si consulti la documentazione:

<http://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/javadoc.html#javadoctags>

Documentazione delle Classi

- *Forma generale*

```
/**  
 * Nome-classe: commento che descrive  
 * scopo e caratteristiche generali della classe  
 *  
 * @author nome-autore  
 * @see riferimento ad altra classe  
 * @see riferimento ad altra classe  
 * @version versione  
 */  
public class Nome-classe {
```

Esempio

```
/**
 * Una semplice classe che modella un attrezzo.
 * Gli attrezzi possono trovarsi all'interno delle stanze
 * del labirinto.
 * Ogni attrezzo ha un nome ed un peso.
 *
 * @author    docente di P00
 * @see Stanza
 * @version 0.9
 */
public class Attrezzo {
    ...
}
```

Documentazione dei Costruttori

- *Forma generale*

```
/**  
 * Commento che descrive scopo e caratteristiche  
 * generali del costruttore  
 *  
 * @param nome-parametro breve descrizione  
 */  
Nome-classe(...) {
```

Esempio

```
/**
 * Crea un attrezzo
 * @param nome il nome che identifica l'attrezzo
 * @param peso il peso dell'attrezzo
 */
public Attrezzo(String nome, int peso) {
    this.peso = peso;
    this.nome = nome;
}
```


Documentazione dei Metodi

- *Forma generale*

```
/**  
 * Commento che descrive scopo e caratteristiche  
 * generali del metodo  
 *  
 * @param nome-parametro breve descrizione  
 * @return valore di ritorno, breve descrizione  
 */  
public type nome-metodo(...) {
```

Esempio

```
/**
 * restituisce il nome identificatore dell'attrezzo
 * @return identificatore dell'attrezzo
 */
public String getNome(){
    return this.nome;
}
```

```
/**
 * restituisce il peso dell'attrezzo
 * @return peso dell'attrezzo
 */
public int getPeso(){
    return this.peso;
}
```

Generare Documentazione

- Si usa il tool javadoc
- Per i dettagli

javadoc *.java

Package

- Le classi sono raggruppate in **package**
- Questo raggruppamento consente di:
 - Mantenere assieme classi concettualmente e logicamente correlate
 - Creare spazi di nomi che evitino conflitti
 - Definire un dominio di protezione (cfr modificatori di accesso)

Import

- Una classe può usare tutte le classi dello stesso package e tutte le classi *pubbliche* di altri package
- Si può accedere alle classi pubbliche di un altro package in due modi
 - Usando il nome completamente qualificato di una classe, cioè anteponendo il nome del pacchetto alla classe:

```
java.util.Scanner s =  
    new java.util.Scanner(input);
```

- Importando la classe e scrivendone direttamente il nome

```
import java.util.Scanner;  
...  
Scanner s = new Scanner(input);
```

Package

- È bene organizzare il proprio codice organizzando le classi in package
- Le classi che appartengono ad un package devono dichiarare la propria appartenenza al package tramite la dichiarazione

`package nome-package;`

- La dichiarazione di appartenenza ad un package deve comparire all'inizio del file
- Una classe può appartenere al più ad un package

Package, Convenzioni sui Nomi

- Il nome di un package deve essere univoco.
- A tal fine **di solito** il nome del package comprende il nome del dominio Internet dell'organizzazione, scritto in ordine inverso
`package it.uniroma3.diadia;`

Package e Classi Pubbliche

- Una classe può essere usata al di fuori del package solo se è dichiarata **pubblica**

- Esempio:

```
package it.uniroma3.diadia;  
public class Stanza {  
    ...  
}
```

la classe **Stanza** può essere usata al di fuori del package **it.uniroma3.diadia** (importandola)

- Se invece scrivessimo:

```
package it.uniroma3.diadia;  
class Stanza {  
    ...  
}
```

la classe **Stanza** non potrebbe essere usata fuori dal package **it.uniroma3.diadia**

Package

- Il nome di un package possiede una struttura gerarchica
- Tale struttura deve trovare corrispondenza diretta nel file system
- Ad esempio le classi del package

`it.uniroma3.diadia`

devono essere memorizzate nella cartella

`it/uniroma3/diadia`

Package: Compilazione ed Esecuzione

- Per compilare le classi di un package si deve far riferimento alla gerarchia fisica
- Con un buon IDE (ad es. Eclipse: cfr. *source folders*) il processo è trasparente
- Da riga di comando questo può essere un po' articolato:
 - Supponiamo di mettere tutto il nostro codice nella directory `C:\src`
 - La versione *base* di **diadia** è nel package:
`it.uniroma3.diadia`
quindi le classi Java sono nella directory
`src/it/uniroma3/diadia`
 - Per **compilare** una classe (ad esempio la classe `Stanza.java`) del package:
 - dalla radice del package, cioè dalla directory che contiene la directory `it` (supponiamo `C:\src`)
`javac it/uniroma3/diadia/Stanza.java`
 - oppure, dalla directory in cui si trovano le classi da compilare
`javac -classpath "C:\src\" Stanza.java`
(supponendo che la directory `it` sia nella directory `src` del volume C:)
 - Per **eseguire** una classe di un package si deve far riferimento alla gerarchia logica
 - dalla radice del package (cioè dalla directory che contiene la directory `it`)
`java it.uniroma3.diadia.DiaDia`
 - oppure da una qualunque directory
`java -classpath "C:\src" it.uniroma3.diadia.DiaDia`

Ricapitolazione

- Java offre un insieme estremamente vasto e ricco di librerie
- Le librerie sono documentate in un formato standard
- Nella definizione delle nostre classi è possibile creare automaticamente documentazione standard
 - Usando opportunamente i commenti
- Le librerie (e le applicazioni) sono organizzati in package
 - Creazione di uno spazio univoco dei nomi
 - Raggruppamento delle classi

Esercizio

- Mettere tutte le classi dello studio di caso (versione base) in un package diadia
- Compilare ed eseguire il programma