Programmazione Orientata agli Oggetti

Qualità del codice:
Java Base Library
Documentazione
Package

Concetti introdotti

- 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 classi importanti per nome
  - Sapere come trovare e usare altre classi
- Importante:
  - Ci serve conoscere solo l'interfaccia, non l'implementazione

Leggere la documentazione

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

Leggere la documentazione

- *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

l'*interfaccia* della classe

Leggere la documentazione

- *La documentazione non include*
  - Campi privati
    (tutti i campi dovrebbero essere privati)
  - Metodi privati
  - Il corpo dei metodi e dei costruttori

l'*implementazione* della classe

Produrre la documentazione

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

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 la documentazione

- Un’idea semplice, ma molto efficace
  - La documentazione viene generata dai commenti immersi nel codice
  - Basta usare una semplice sintassi
  - 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://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/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  Michael Kolling and David J. Barnes
 * (traduzione e adattamento di
 *  Paolo Merialdo, Valter Crescenzi)
 * @see Stanza
 * @see Gioco
 * @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 il nome identificatore dell'attrezzo
  */
  public String getNome(){
     return this.nome;
  }

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

Creare la 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
  - Fornire un dominio di protezione (cfr modificatori di accesso)

Importazioni

- 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 pacchetto 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 il package e scrivendo direttamente il nome della classe

```
import java.util.*;
…
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 diadia;
public class Stanza {
 …
}
```

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

- Se invece scrivessimo:

```
package diadia;
class Stanza {
 …
}
```

la classe **`Stanza`** non potrebbe essere usata fuori dal package **`diadia`**

Package

- Il nome di un package possiede struttura gerarchica
- Tale struttura deve trovare corrispondenza 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
- Supponiamo di mettere tutto il nostro codice nella directory c:\src
- La versione 0.90 di **`diadia`** è nel package:

  `projects.diadia.diadia_0_90`

  quindi le classi Java sono nella directory

  `projects/diadia/diadia_0_90`
- Per compilare una classe (ad esempio la classe **`Stanza.java`**) del package:
  - dalla radice del package, cioè dalla directory che contiene la directory **`projects`** (supponiamo **`C:\src`**)

    ```
    javac projects/diadia/diadia_0_90/Stanza.java
    ```
  - oppure, dalla directory in cui si trovano le classe da compilare

    ```
    javac -classpath "C:\src\" Stanza.java
    ```

    (supponendo che **`projects`** 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 **`projects`**)

    ```
    java projects.diadia.diadia_0_90.Gioco
    ```
  - oppure da una directory qualunque

    ```
    java -classpath "C:\src" projects.diadia.diadia_0_90.Gioco
    ```

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