javadoc: documentazione di applicazioni, classi, package

//commento

/*
commento
*/

/**
commento javadoc
*/

pagine HTML che descrivono classi, interfacce, costruttori, metodi e campi con indicatori di visibilità public o protected

i commenti javadoc devono precedere immediatamente la dichiarazione dell'entità (classe, interfaccia, metodo, costruttore, campo) a cui si riferiscono

/**
Sezione descrittiva
Sezione tag
*/

Esempio

/**
* Sezione descrittiva del commento doc
*
* @tag Commento per il tag
*/

È possibile utilizzare tag HTML all'interno del commento

Se il metodo mioMetodo1() in una certa classe o interfaccia non ha alcun commento doc, javadoc utilizza il commento doc di mioMetodo2(), che mioMetodo1() implementa o di cui effettua l'override.

Tag:

- @author [nome]
  Aggiunge “Author:” seguito dal nome specificato.
  Ogni commento doc può contenere molteplici tag @author, presentati in ordine cronologico.

- @version [versione]
  Aggiunge “Version:” seguito dalla versione specificata.

- @param [nome del parametro] [descrizione]
  Aggiunge il parametro specificato e la sua descrizione alla sezione “Parameters:” del metodo corrente. Il commento doc riferito ad un certo costruttore o ad un certo metodo deve obbligatoriamente presentare un tag @param per ognuno dei parametri attesi, nell’ordine in cui l’implementazione del costruttore o del metodo specifica i parametri stessi.

- @return [descrizione]
  Aggiunge “Returns:” seguito dalla descrizione specificata. Indica il tipo restituito e la gamma di valori possibili.
  Può essere inserito solamente all’interno del codice sorgente Java, dove deve essere obbligatoriamente usato per ogni metodo, a meno che questo non sia un costruttore oppure non presenti alcun valore di ritorno (void).

- @throws [nome completo della classe] [descrizione]
  Aggiunge “Throws:” seguito dal nome della classe specificata (che costituisce l’eccezione) e dalla sua descrizione. Il commento doc riferito ad un certo costruttore o ad un certo metodo deve presentare un tag @exception per ognuna delle eccezioni che compaiono nella sua clausola throws, presentate in ordine alfabetico.

- @see [riferimento]
  Aggiunge “See Also:” seguito dal riferimento indicato. Può essere il nome di una classe (completo del package a cui appartiene) alla cui documentazione ci si collega

- @since [versione]
  Utilizzato per specificare da che momento l’entità di riferimento (classe, interfaccia, metodo, costruttore, campo) è stata inserita nell’API.

javadoc [opzioni] [package] [file sorgenti] [@mieifile]

opzioni (alcune):

- public, mostra nella documentazione solamente entità public;
- protected, mostra nella documentazione solamente entità protected e public (questa è l'opzione di default);
- package, mostra nella documentazione solamente entità package, protected e public;
- private, che mostra nella documentazione tutte le classi, i metodi e i campi;
- sourcepath [locazione dei sorgenti], che specifica il percorso per raggiungere i file sorgenti Java quando, dopo il comando javadoc, è presente l'argomento [package];
- classpath [locazione delle classi (classpath)], che specifica il percorso per raggiungere le classi Java;
- d [nome della directory], che specifica, in modo assoluto oppure relativo, la directory di destinazione nella quale javadoc salva i file HTML generati;
- version, che permette di processare il tag @version nella documentazione generata;
- author, che permette di processare il tag @author nella documentazione generata;

package: un elenco di nomi di package, separati da spazi. E' necessario indicare nello specifico ognuno dei package che si intende documentare, qualificandolo con il nome completo e senza la possibilità di utilizzare wildcard come * (asterisco);

file sorgenti: un elenco di nomi di file sorgenti Java, separati da spazi, ognuno dei quali può iniziare con un path e contenere wildcard come * (asterisco);

@mieifile: un tag che specifica uno o più file i quali contengono nomi di package e nomi di file sorgenti Java, in qualunque ordine, uno per ogni riga. Javadoc tratta il contenuto dei file indicati come se ogni singola riga si trovasse direttamente su linea di comando.

Javadoc è in grado di produrre output a partire da quattro diverse possibili fonti:

- Il codice sorgente Java
- file di commento ai package. Deve avere nome package.html (uguale per tutti i package). Deve essere posizionato nella directory del package, la medesima che contiene i file sorgenti Java. package.html è scritto in HTML, e non contiene /** e */, e nemmeno l'asterisco in testa ad ogni riga.;
- file di commento generale, Ha solitamente nome overview.html ed è di norma posizionato nella più "alta" directory interna al percorso contenente i file sorgenti. È scritto in HTML, con le stesse caratteristiche appena viste per package.html;
- file diversi non processati che si ritiene opportuno includere nella documentazione, e rendere quindi raggiungibili dalle pagine HTML generate da javadoc: è il caso, ad esempio, di file con immagini, di file sorgenti Java (.java) con esempi, di file Java compilati (.class) o anche di file HTML il cui contenuto potrebbe essere troppo esteso per un normale commento doc. I file in questione devono essere inseriti in una directory chiamata doc-files, la quale può essere subdirectory di una qualunque directory di package. E' prevista l'esistenza di una directory doc-files per ogni package, e i nomi dei file devono essere esplicitamente indicati all'interno dei commenti doc di riferimento. Javadoc non esamina il contenuto di doc-files, limitandosi a copiarlo nella corretta directory di destinazione.

Esempio 4

```
/**
* Classe di esempio.
* @author M. M.
* @version 1.0
*/
public class MiaClasse {
/**
* Variabile di tipo int
*/
public int miaVariabile01;
/**
* Variabile che contiene la rappresentazione della classe
*/
private String miaVariabile02 = "public class MiaClasse";
/**
* Costruttore che inizializza miaVariabile01
* @param mioParametro il valore per inizializzare miaVariabile01
*/
public MiaClasse(int mioParametro) {
miaVariabile01 = mioParametro;
}
/**
* Restituzione del valore di miaVariabile01
* @return il valore di miaVariabile01
* @see AltraClasse
*/
public int getMiaVariabile01() {
return miaVariabile01;
}
/**
*Rappresentazione della classe
* @return il valore di miaVariabile02
* @see java.awt.Button
*/
public String toString() {
return miaVariabile02;
}
}
```

Caso di documentazione di una o più classi:

nella cartella in cui i sorgenti si trovano:

javadoc -d docum *.java

Caso di documentazione di un package:

prima parte: installazione del package it.unipg.liste

C:\>echo %CLASSPATH%
.;C:\jdk142\lib;C:\jdk142

le classi del package vanno installate nel cammino it\unipg\liste a partire da una delle cartelle del CLASSPATH

esempio:
C:\jdk142\it\unipg\liste

Nella cartella si trovano le classi:
ApplicazioneLista.java
Elemento.java
Lista.java
Pila.java

Il comando javadoc va invocato dalla cartella che contiene il cammino delle cartelle del package
C:\jdk142>javadoc -d docum it.unipg.liste