Linguaggi

Corso di Laurea Ingegneria Informatica (M-Z)
A.A. 2006-2007

Alessandro Longheu
http://www.diit.unict.it/users/alongheu
alessandro.longheu@diit.unict.it

Documentazione in Java

Javadoc Terminology

- API documentation (API docs) or API specifications (API specs)
  - Descrizione delle API, intended primarily for programmers writing in Java.
  - Possono essere generate automaticamente con lo strumento Javadoc o "by hand".
- Commenti per documentare il codice (**doc comments**)
  - I commenti speciali devono essere delimitati da /** ... */.
- **javadoc**
  - Lo strumento incluso in JDK per generare la documentazione di API attraverso I "doc comments"
- Lo strumento Javadoc può generare 4 tipi di file sorgenti:
  - Source code files – contiene i commenti alle classi, interfacce, costruttori, attributi e metodi
  - Package comment files
  - Overview comment files – commenti su un insieme di package
  - Miscellaneous unprocessed files - include immagini, esempi di codice sorgente, applets, file HTML files, etc.

Produzione dei commenti

- Un doc comment è scritto in HTML e deve precedere la classe, gli attributi, i metodi. Esso è composto da una descrizione, seguita da alcuni blocchi preceduti da tag (@param, @return, @see), ad esempio:

```
/**
 * Returns an Image object that can then be painted on the screen.
 * The url argument must specify an absolute {@link URL}. The name
 * argument is a specifier that is relative to the url argument…
 *
 * @param url an absolute URL giving the base location of the image
 * @param name the location of the image, relative to the url argument
 * @return the image at the specified URL
 * @see Image
 */
public Image getImage(URL url, String name) {
    try {          return getImage(new URL(url, name));  }
catch (MalformedURLException e) {      return null;      } }
```


Produzione dei commenti

- HTML Risultante:

getImage

public Image **getImage**(URL url, String name)

Returns an Image object that can then be painted on the screen. The url argument must specify an absolute URL. The name argument is a specifier that is relative to the url argument…

Parameters:

url - an absolute URL giving the base location of the image

name - the location of the image, relative to the url argument

Returns:

the image at the specified URL

See Also:

Image

Descrizione dei commenti di metodi overloaded

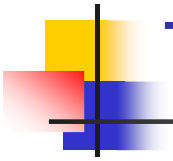
- È possibile scrivere una prima frase che consente di distinguere il metodo overloaded, ad esempio:

```
/**
 * Costruttore.
 */
foo() { ...

/**
 * Costruttore che specifica il numero di .....
 */ foo(int n) { ...
```

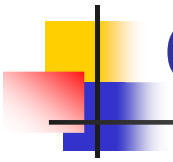

Riuso di commenti dei metodi

- È possibile evitare la riscrittura di "doc comments" utilizzando il concetto di ereditarietà
- Qusto funziona in tre casi:
  - Quando un metodo in una classe sovrascrive il metodo della superclasse
  - Quando un metodo in una interfaccia sovrascrive il metodo della super-interfaccia
  - Quando un metodo in una classe implementa un metodo di una interfaccia



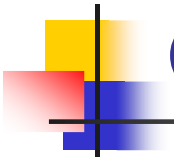
Tag per la documentazione

- I Tags sono di due tipi :
  - **Block tags**
    - Possono essere posti solo nella sezione che segue la descrizione principale.
    - I Block tags hanno la seguente forma: @tag.
  - **Inline tags**
    - Possono essere posti in qualsiasi posto nella descrizione principale e nei commenti per block tags.
    - Gli Inline tags hanno la seguente forma: {@tag}.



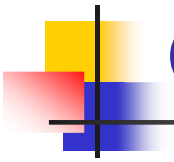
Ordine dei tag

- @author (solo per classi e interface)
- @version (solo per classi e interface)
- @param (solo per metodi e costruttori)
- @return (solo per metodi)
- @exception (@throws è un sinonimo introdotto in Javadoc 1.2)
- @see
- @since
- @serial (or @serialField or @serialData)
- @deprecated



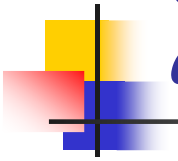
@author *name-text*

- Aggiunge un "Author" entry con la specifica di un name-text per generare documenti dove compare l'opzione -author.
- Un doc comment può contenere più volte il tag @author.



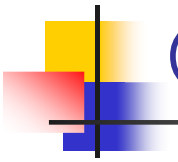
@version *version-text*

- Aggiunge un "Version" heading con la specifica del version-text per generare documenti dove compare l'opzione -version.
- Questo tag ha la funzione di numerare le versioni del software di cui questo codice è parte
  - Il version-text non ha una particolare struttura interna.



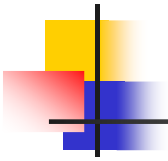
@param *parameter-name description*

- Aggiunge un parametro alla sezione "Parameters".
- La descrizione può essere continuata nella linea successiva.
- Questo tag è utilizzabile solo per metodi e costruttori.



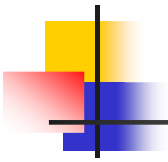
@return *description*

- Aggiunge una sezione "Returns" con il description text.
- Questo testo dovrebbe descrivere il tipo di ritorno e i valori permessi come valori di ritorno.
- Questo tag è utilizzabile solo per i metodi.



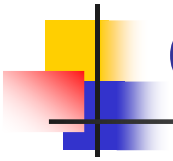
@exception *class-name description*

- a.k.a. @throws class-name description
- Aggiunge una sezione "Throws" nella documenazione generata con il class-name e il description text.
  - Il class-name è il nome dell'exception che può essere lanciata dal metodo.
- Questo tag è utilizzabile solo per metodi e costruttori.
- Possono esserci più @throws tags
- Se un @throws tag non esiste per una eccezione nella clausola throws, lo strumento Javadoc automaticamente aggiunge una exception all' output HTML (con nessuna descrizione) cosi come quelle con un @throws tag.



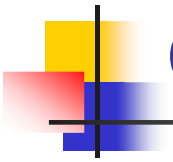
@see *reference*

- Aggiunge una intestazione "See Also" con un link o un text entry che si riferisce ad una reference.
- Un "doc comment" può contenere qualsiasi numero di @see tags, che sono tutti raggruppati sotto la stessa intestazione.
- Il @see tag ha tre varianti:
  - @see "string"
  - @see label
  - @see package.class#member label
- Valido in qualsiasi commento: introduzione, package, classi, interface, costruttori, metod. o attributi.
- Per inserire un in-line link è possibile usare {@link}



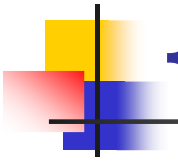
@since *since-text*

- Aggiunge una sezione " Since " nella documenazione generata con il since-text.
- Il testo non ha una particolare struttura.
- Valido in qualsiasi commento: introduzione, package, classi, interface, costruttori, metod. o attributi.
- Questo tag specifica che tale modifica è apportata da un particolare release software specificata dal since-text.
- Per esempio: @since 1.4



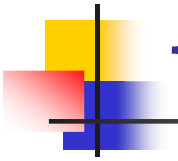
@deprecated *deprecated-text*

- Aggiunge un commento indicante che questa API non dovrebbe essere più utilizzata
- Il tool Javadoc sposta il deprecated-text in testa alla descrizione principale, ponendola in italico e facendola precedere dalla scritta di avvertimento in grassetto "Deprecated".
- Valido in qualsiasi commento: introduzione, package, classi, interface, costruttori, metod. o attributi.
- La prima frase del deprecated-text dovrebbe almeno dire che l'uso dell' API è deprecated e che cosa la sostituisce.
- Le altre frasi possono spiegare perchè è deprecated.



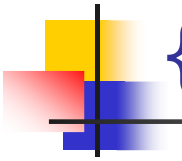
{@docRoot}

- Rappresenta il cammino relativo per generare la radice del documento (destination).
- É utile quando si vuole includere un file da riferire dalle pagine generate



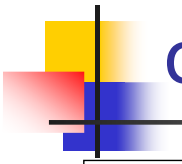
{@link *package.class#member label*}

- Inserisce un in-line link con testo visibilie uguale a label che punta alla documentazione per lo specificico package, class o componente.
- Questo tag è valido in tutto i documenti di commento: overview, package, class, interface, constructor, method e field, incluso la parte di testo di qualsiasi tag (come @return, @param e @deprecated).
- Per esempio
- {@link #getComponentAt(int, int) getComponentAt} .
- È un commento che si riferisce al metodo getComponentAt(int, int)



{@value}

- Quando si usa un commento a un attributo statico mostra il valore della costante.
- Questi valori sono i valori mostrati nel Constant Field Values page.
- Questo tag è valido solo per i commenti degli attributi.



Esempio dei commenti di una classe

```
/**
 * A class representing a window on the screen.
 * For example:
 * <pre>
 * Window win = new Window(parent);
 * win.show();
 * </pre>
 *
 * @author Sami Shaio
 * @version %I%, %G%
 * @see java.awt.BaseWindow
 * @see java.awt.Button
 */
 class Window extends BaseWindow { ... }

/**
 * The X-coordinate of the component.
 *
 * @see #getLocation()
 */
 int x = 1263732;
```

Esempio dei commenti di una classe

```
/**
 * Returns the character at the specified index. An index
 * ranges from <code>0</code> to <code>length() -
   1</code>.
 *
 * @param index the index of the desired character.
 * @return the desired character.
 * @exception StringIndexOutOfRangeException
 * if the index is not in the range <code>0</code>
 * to <code>length()-1</code>.
 * @see java.lang.Character#charValue()
 */
 public char charAt(int index) { ... }
```


Come utilizzare Javadoc?

javadoc –d directory_for_html sourcefiles

O

javadoc –d directory_for_html packages