Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate

Laboratorio Interdisciplinare B

Documentazione di Progetto

Angela Locoro

Dipartimento di Scienze Teoriche e Applicate

Angela.Locoro@uninsubria.it

Documentazione di Progetto

1. Manuale Utente
2. Manuale Tecnico
3. JavaDoc

Documentazione di Progetto

- Manuale Utente
  - Manuale di alto livello del funzionamento del programma
  - Come usare il programma (quale ambiente usare, come installare l'ambiente, come installare il programma, come eseguirlo, …)
  - Aggiungere screenshot per semplificare la comprensione del funzionamento / uso del programma
  - Utilizzare un livello di astrazione e un linguaggio adeguati ad un pubblico non esperto e non tecnico

Documentazione di Progetto

- Manuale Utente – Struttura
  - Frontespizio (Titolo, Autori, Data e Versione documento)
  - Indice
  - Installazione
    - Requisiti di sistema
    - Setup ambiente
    - Installazione programma
  - Esecuzione ed Uso
    - Setup e Lancio del programma
    - Uso delle Funzionalità
    - Data set di test
  - Limiti della soluzione sviluppata
  - Sitografia / Bibliografia
    - *es. QtSpim Simulator, Online: www.xyz.com*
    - *es. D. Tosi, "Come fare un progetto software", nome rivista, anno*

Documentazione di Progetto

- Manuale Tecnico
  - dettagliare dal punto di vista tecnico la struttura dell'applicazione
  - Le scelte progettuali (diagrammi UML e ER)
  - le scelte architetturali
  - le strutture dati utilizzate
  - le scelte algoritmiche
  - il formato dei file e la loro gestione (es. path)
  - l'uso eventuale di pattern, riportando parti di codice significativo
  - la JavaDoc è parte del manuale tecnico (il codice deve quindi essere commentato opportunamente)
  - utilizzare un livello di astrazione e un linguaggio adeguati a esperti tecnici del dominio

Documentazione di Progetto

- Manuale Tecnico - Struttura
  - Si segua la struttura del Manuale Utente

Documentazione di Progetto

- Manuale Tecnico – Progettazione

  - **Documentazione del sistema client/server con UML**. UML deve far vedere la struttura statica del sistema ad un livello di astrazione maggiore del codice sorgente (Class diagram). UML deve far vedere la struttura dinamica del sistema, catturando le principali interazioni fra classi attraverso Sequence Diagrams, Interaction Diagram e State diagrams. NB: non è necessario usare tutti questi diagrammi, a patto che l'insieme prodotto sia sufficiente a rappresentare chiaramente il comportamento del sistema (sia dal punto di vista strutturale che comportamentale)

  - **Documentazione del database.** Documento di analisi dei requisiti ristrutturato e documentazione associata allo schema ER (ristrutturato e non), con eventuale specifica di vincoli in linguaggio naturale.

Documentazione di Progetto

- JavaDoc [1]
  - è uno strumento che permette di documentare i file sorgenti di un programma usando le informazioni riportate al loro interno
  - il programmatore inserisce i commenti al codice in un formato predefinito
  - i commenti vengono riconosciuti dal programma javadoc che li converte in un formato di consultazione (html, pdf, ...)
  - un commento Javadoc è un testo HTML racchiuso tra i tag `/**` e `*/`
  - Javadoc è pensato solo per descrivere le funzionalità di Package, Classi, interfacce, Metodi (non può essere usato per commentare pezzi di codice)

[1] *https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html*

Documentazione di Progetto

Esempio Javadoc

```
/**
 * Questo &egrave; un commento <em>Javadoc</em>.
 * Gli spazi e gli asterischi a inizio riga
 * <strong>sono</strong> sempre ignorati.
 */
```

L'effetto è il seguente:

Questo è un commento *Javadoc*. Gli spazi e gli asterischi a inizio riga **sono** sempre ignorati.

Documentazione di Progetto

- Formato generale di un tag: @**name comment**

  dove nome specifica quale tipo di informazione si sta dando e

  il commento è l'informazione.

  Esempio: @**author William Shakespeare**

- Ogni tag deve essere su una riga nuova.

- I commenti possono estendersi su più righe, ma non ci devono essere righe vuote

Documentazione di Progetto

- JavaDoc (cont.)
  - la documentazione deve quindi comprendere la descrizione di ogni package, classe, interfaccia, metodo e attributo
  - Il commento è sempre posto subito prima della dichiarazione della classe, interfaccia, metodo, attributo
  - Può contenere tag HTML per aiutare la formattazione (es. tag di struttura come `<strong>, <em>`, ecc)

- JavaDoc - Classi
  - deve descrivere lo scopo della classe
  - al termine della descrizione si mette il tag `@author Nome Cognome` per indicare l'autore del codice
  - Se ci sono più autori, si mettono più tag su righe separate, uno di seguito all'altro

Documentazione di Progetto

- JavaDoc - Classi
  - deve descrivere lo scopo della classe
  - al termine della descrizione si mette il tag `@author Nome Cognome` per indicare l'autore del codice
  - Se ci sono più autori, si mettono più tag su righe separate, uno di seguito all'altro

Tag minimo per le classi

```
/** ...
  *
  * @author William Shakespeare
  * @author Christopher "Kit" Marlowe
  */
  public class Drama {...}
```

Documentazione di Progetto

- JavaDoc – Attributi
  - deve descrivere a cosa serve l'attributo
  - solo il primo paragrafo verrà riportato nella descrizione generale. Gli altri andranno nella descrizione puntuale
  - un nuovo paragrafo inizia con il tag `<p>` a inizio linea
  - usare i tag `<code>` e `</code>` per racchiudere il nome degli attributi
    - vengono visualizzati nel font usato per il codice

- JavaDoc – Metodi
  - per ogni metodo usare sempre i tag a inizio linea:
    - `@param <nome parametro>` per fornire una breve descrizione del parametro (se più parametri, rispettare l'ordine con cui sono dichiarati)
    - `@return` per fornire una breve descrizione di ciò che il metodo ritorna
    - `@throws <nome eccezione>` per fornire una descrizione delle circostanze che determinano il sollevamento dell'eccezione

Documentazione di Progetto

- JavaDoc – Package
  - deve riportare la documentazione generale del package
  - usare i tag `@see` per riferirsi alle classi, metodi, attributi in esso contenuti

- Per generare la documentazione si usa il comando `javadoc` che fa parte della distribuzione di Java Development Kit (JDK)

Aspetti Organizzativi del Progetto

- Gli aspetti organizzativi di un progetto sono relativi a:
  - Ruoli delle persone
  - Struttura del progetto e project management

Aspetti Organizzativi del Progetto

ACTIVITIES	TASKS	DESCRIPTION (ISSUES / LIMITATIONS, ETC.)
Project Management	T1	Monitor design issues and schelule
	T2	Monitor development issues and schelule
	T3	Monitor doc issues and schelule
	T4	Monitor quality / testing issues and schelule
Design Management	T1	Design DB
	T1.1	Design Tables
	T1.2	Restructure DB
	T2	Design System
	T2.1	Design Architecture
	T2.1.1.	Design Class Diagrams
	…	
	T2.2	Desing Systems Dynamics
	T2.1.1.	Design Sequence Diagrams
Development Management	T1	Develop Client
	T1.1	Develop Class X
	…	…
	T2	Develop Server
	T2.1.	Develop Class Y
	…	…
Documentation & Quality Management	T1	Write Documentation
	T2	Check documentation quality
	T3	Define Software testing