Università degli Studi di Salerno

Corso di Gestione Progetti Software

A.A. 2012/2013

Alfonso Murolo

Giulio Franco

Linda di Geronimo

Object Design Document

Versione 0.1

Storia delle revisioni

Versione	Data	Autori	Descrizione
0.1			Prima bozza

Riferimenti

Indice

1 Introduzione

1.1 Tradeoff nel Design degli Oggetti

1.2 Linee guida per la documentazione delle interfacce

Nell'implementazione del sistema, i programmatori dovranno attenersi alle linee guida di seguito definite.

1.2.1 Classi e Interfacce Java

Nella scrittura di codice per le classi Java, ci si atterrà allo standard [CCJPL] nella sua interezza, con particolare attenzione a:

1. Convenzioni sui nomi (cap. 9 dello standard);
2. Struttura dei file (cap. 3 dello standard);
3. Accesso alle variabili (sez. 10.1 dello standard)
4. Commenti speciali (sez. 10.5.4 dello standard)
5. Utilizzo degli spazi bianchi (cap. 8 dello standard)

Si praticano, inoltre, le seguenti restrizioni e variazioni:

1. I commenti iniziali (sez. 3.1.2 dello standard) sono obbligatori, e devono avere la forma:

```
/*
 * NomeClasse
 * Breve descrizione della classe
 *
 * Progetto @silo
 * Copyleft
 * Nome del responsabile <email del responsabile>
 */
```

2. Tutte le variabili dovrebbero essere private (`private`). Ogni variabile non privata deve essere preceduta da un commento, che spiega la ragione per cui tale variabile non è privata. Tutte le variabili che non vengono mai modificate dovrebbero essere dichiarate come costanti (`final`).
3. L'uso dell'operatore condizionale ternario è scoraggiato.
4. Si devono usare gli spazi per gestire l'indentazione.
   L'unità di indentazione è 4 spazi.
   Le tabulazioni sono di 8 spazi.

1.2.2 Base di dati

Le tabelle della base di dati dovrebbero rispettare la terza forma normale di Codd (3NF). Ove ciò non si verifichi, tale fatto deve essere riportato e motivato nella documentazione della base di dati.

Le scelte di gestione nel trattamento dell'integrità referenziale devono essere riportate e motivate nella documentazione della base di dati.

I nomi utilizzati all'interno della base di dati devono seguire le seguenti convenzioni:

1.2.3 Pagine Java lato Server (JSP)

Le pagine JSP devono, quando eseguite, produrre, in ogni circostanza, una documento conforme allo standard HTML versione 4.01 “strict” ([HTML41]).

Le parti Java delle pagine devono aderire alle convenzioni per la codifica in Java, con le seguenti puntualizzazioni:

1. Il tag di apertura (`<%`) è seguito immediatamente dalla fine della riga;
2. Il tag di chiusura (`%>`) si trova all'inizio di una riga;
3. È possibile emendare alle due regole precedenti, se il corpo del codice Java consiste in una singola istruzione:

```
<!-- Accettabile -->
<% for (String par : paragraphs) {%>
<p class='item'><% out.print(par); %></p>
<% } %>

<!-- Non accettabile -->
<p class='item'><% List<String> paragraphs = getParagraphs();
out.print(paragraphs.get(i++));%></p>
```

Il codice HTML, Javascript e CSS prodotto deve attenersi alle relative convenzioni.

1.2.4 Pagine HTML

Le pagine HTML, statiche e dinamiche, devono essere totalmente aderenti allo standard HTML versione 4.01 “strict” ([HTML41]).

Inoltre, il codice HTML statico deve utilizzare l'indentazione, per facilitare la lettura, secondo le seguenti regole:

1. Un'indentazione consiste in 4 spazi;
2. Ogni tag deve avere un'indentazione maggiore o uguale del tag che lo contiene
   2.1. Non si dovrebbero superare gli 8 livelli di indentazione;
   2.2. Qualora si superassero gli 8 livelli, è possibile azzerare il livello di indentazione, portandolo a 0 anziché incrementarlo;
3. Sebbene l'uso dell'indentazione sia arbitrario, ogni tag di chiusura deve avere lo stesso livello di indentazione del corrispondente tag di apertura;

```
<!-- Accettabile -->
<div><span><nl>
    <li>Uno</li>
    <li>
        Due
    </li>
</nl></span></div>

<!-- Non accettabile -->
<div><span>
    <nl>
    <li>Uno</li>
    <li>
        Due
    </li>
```

```
    </nl></span>
</div>
```

4. I tag di commento devono seguire le stesse regole che si applicano ai tag normali;
5. Gli script e i fogli di stile incorporati, se non si completano in una sola riga di testo, devono partire dal livello 0 di indentazione.

I fogli di stile e gli script incorporati nelle pagine devono rispettare le stesse linee guida che si applicano ai documenti dedicati.

Le sezioni dinamiche (prodotte tramite l'oggetto `out`) dei documenti HTML generati da pagine JSP dovrebbero rispettare le stesse convenzioni dello HTML statico.

1.2.5 Script Javascript

Gli script che svolgono funzioni distinte dal mero rendering della pagina dovrebbero essere collocati in file dedicati.

Il codice Javascript deve seguire le stesse convenzioni per il layout e i nomi del codice Java.

I documenti Javascript devono essere iniziati da un commento analogo a quello presente nei file Java.

Le funzioni Javascript devono essere documentate in modo analogo ai metodi Java.

Gli oggetti Javascript devono essere preceduti da un commento in stile Javadoc, che segue il seguente formato:

```
/**
 * Descrizione breve
 * Eventuale ulteriore descrizione
 * Specifica degli argomenti del costruttore (@param)
 *
 * Metodo nomeMetodo1
 *     Descrizione breve
 *     Eventuale ulteriore descrizione
 *     Specifica degli argomenti (@param)
 *     Specifica dei risultati (@return)
 *
 * Metodo nomeMetodo2
 *     Descrizione breve
 *     Eventuale ulteriore descrizione
 *     Specifica degli argomenti (@param)
 *     Specifica dei risultati (@return)
 *
 * …
 */
function ClasseX(a, b, c) {
```

1.2.6 Fogli di stile CSS

Tutti gli stili non inline devono essere collocati in fogli di stile separati.

Ogni foglio di stile deve essere iniziato da un commento analogo a quello presente nei file Java.

Ogni regola CSS deve essere formattata come segue:

1. I selettori della regola si trovano a livello 0 di indentazione, uno per riga;
2. L'ultimo selettore della regola è seguito da parentesi graffa aperta (`{`);

3. Le proprietà che costituiscono la regola sono listate una per riga e sono indentate rispetto ai selettori;
4. La regola è terminata da una parentesi graffa chiusa (`}`), collocata da sola su una riga;

Le proprietà e le regole poco chiare dovrebbero essere precedute da un commento esplicativo.

Le regole possono essere divise in blocchi concettualmente legati, preceduti da commenti in stile Javadoc, che ne spiegano lo scopo, e seguiti da 2 righe bianche.

1.3 Definizioni, acronimi e abbreviazioni

2 Pacchetti

3 Interfacce delle classi

3.1 Presentation

3.2 Application

3.3 Storage

3.4 Beans

3.5 Exceptions

4 Documentazione del riuso