**Object Design Document**

Sommario

[**1.** **Introduction** 2](#_Toc27488462)

[**1.1.** **Object design trade-offs** 2](#_Toc27488463)

[**1.2.** **Interface documentation guidelines** 2](#_Toc27488464)

[**1.3.** **Definitions, acronyms, and abbrevations** 2](#_Toc27488465)

[**1.4.** **References** 3](#_Toc27488466)

[**2.** **Packages** 3](#_Toc27488467)

[**3.** **Class interfaces** 3](#_Toc27488468)

# **Introduction**

## **Object design trade-offs**

• **Memory space vs. Response time:**

poiché il nostro sistema EmmeShop di progetto visto che si tratta di un e-commerce dovrà gestire una grande mole di dati per cui abbiamo bisogno di un DBMS che memorizzi tutti i dati, dal carrello dell’utente, ai prodotti e categorie dei relativi negozi. Quindi questo implica scritture e letture in un DB però avendo un tempo di risposta contenuto nei limiti dettati dai requisiti non funzionali nel RAD.

• **Functionality vs. Usability**

Il nostro software incorpora funzionalità che soddisfano le specifiche dei requisiti. Con MySQL come componente del database, è un [Relational database management system](https://it.wikipedia.org/wiki/Relational_database_management_system) (RDBMS) composto da un [client](https://it.wikipedia.org/wiki/Client) a [riga di comando](https://it.wikipedia.org/wiki/Riga_di_comando) e un [server](https://it.wikipedia.org/wiki/Server). Invece per la logica di controllo e business utilizziamo le servlet basato sul linguaggio Java e JSP, Css e JavaScript come front-end. L'usabilità del sistema è pienamente raggiunta.

• **Portability vs. Readability**

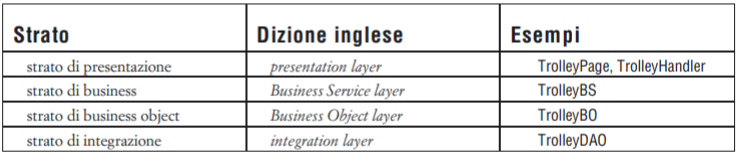
Il nostro sistema sacrifica la leggibilità del codice, per far fronte ad un requisito non funzionale la portabilità del sistema attraverso vari dispositivi, che siano schermi dei pc, tablet, o smartphone. Usando opportuni linguaggio di formattazione delle pagine Web “Css”, tramite le Media Queries permette di strutturare il contenuto di una pagina Html e JSP adattandola alla dimensione dello schermo. Inoltre usando Java Script siamo riusciti a realizzare un sito multilingua.

• **Rapid Development vs. Robustness**

Il Sistema sacrifica lo sviluppo di software robusto, per garantire un sistema funzionante nel più breve tempo richiesto, per poi garantire in futuro aggiornamenti per rendere il software robusto e per implementare altre funzionalità, o rendere più dettagliate quelle presistenti.

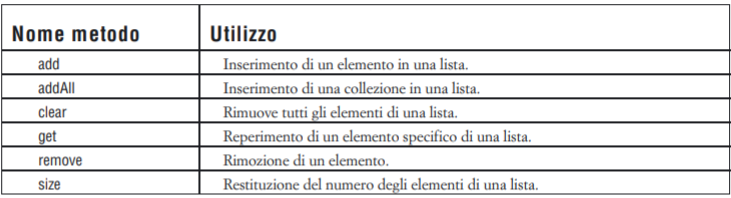
## **Interface documentation guidelines**

• Le classi sono nominate con nomi singolari, relativamente brevi e significativi. Non usare verbi, aggiungere un suffisso relativo allo strato di appartenenza, usare suffisso Exception per le classi di eccezioni.



• I metodi sono denominati con frasi verbali, campi e parametri con frasi di nomi, brevi ed intuitivi. Non ripetere nomi della classe nel nome del metodo.

•usare una convenzione chiara nella selezione dei nomi dei metodi (add, addAll, get, set, size, clear, remove)



• selezionare nomi significativi per attributi e variabili e parametri, brevi.

• considerare l’utilizzo delle lettere i, j, k per le variabili di ciclo

• Lo stato dell'errore viene restituito tramite un'eccezione, non un valore di ritorno.

• utilizzare il livello di accesso più ristretto possibile



• massimizzare la coesione tra le classi e minimizzare accoppiamento.

• eseguire controllo del valore dei parametri passati ad un metodo appena possibile.

## **Definitions, acronyms, and abbrevations**

## **References**

* Problem Statement
* Requirements Analysis Document
* System Design Document
* [http://java.sun.com](http://java.sun.com/)

# **Packages**

# **Class interfaces**