

**INGEGNERIA DEL SOFTWARE 2016-2017**

**Documento di Object Design 0.1**

**Fiorazon**

**15/12/2016**

**Sommario**

[**1.** **Introduction** 3](#_Toc469562348)

[**I.** **Object design trade-offs** 3](#_Toc469562349)

[**II.** **Interface documentation guidelines** 3](#_Toc469562350)

[**III.** **Definitions, acronyms, and abbreviations** 3](#_Toc469562351)

[**IV.** **References** 3](#_Toc469562352)

[**2.** **Packages** 3](#_Toc469562353)

[**3.** **Class Interfaces** 5](#_Toc469562354)

[**4.** **Team** 5](#_Toc469562355)

[**5.** **Storia delle versioni** 5](#_Toc469562356)

### **Introduction**

The first section of the ODD is an introduction to the document. It describes the general trade-offs made by developers (e.g., buy vs. build, memory space vs. response time), guidelines and conventions (e.g., naming conventions, boundary cases, exception handling mechanisms), and an overview of the document.

Interface documentation guidelines and coding conventions are the single most important factor that can improve communication between developers during object design. These include a list of rules that developers should use when designing and naming interfaces. These are examples of such conventions:

Classes are named with singular nouns.

Methods are named with verb phrases, fields, and parameters with noun phrases.

Error status is returned via an exception, not a return value.

Collections and containers have an elements() method returning an Enumeration.

Enumerations returned by elements() methods are robust to element removals.

Such conventions help developers design interfaces consistently, even if many developers contribute to the interface specification. Moreover, making these conventions explicit before object design makes it easier for developers to follow them. In general, these conventions should not evolve during the project.

### **Object design trade-offs**

### **Interface documentation guidelines**

### **Definitions, acronyms, and abbreviations**

### **References**

### **Packages**

Individuiamo 4 packages in cui dividere il codice, uno per ogni sotto-sistema:   
-carrelloPackage  
-prodottiPackage  
-utentiPackage  
-acquistoPackage

Il package carrelloPackage contiene i file “Carrello.jsp”, “ControlloCarrello.java”, “CarrelloManager.java”.   
Esso contiene tutte quelle pagine .jsp e .java responsabili dell’interazione con il carrello.  
Interagisce con l’acquistoPackage dal file “ControlloCarrello.java” verso “Pagamento.jsp” (Operazione di acquisto del carrello) e con prodottiPackage tramite “ControlloCarrello.java” chiamato da “Catalogo.jsp” (Operazione di aggiunta prodotto al carrello).

Il package prodottiPackage contiene i file “AggiungiProdotto.jsp”, “ControlloProdotti.java”, “ProdottiManager.java”, “Prodotto.java”, “Catalogo.jsp”, “CancellaProdotto.jsp”, “ModificaProdotto.jsp”, “Risultati.jsp”.   
Esso contiene tutte quelle pagine .jsp e .java responsabili delle operazioni riguardanti i prodotti.  
Interagisce con carrelloPackage tramite “Catalogo.jsp” verso “ControlloCarrello.java” (Operazione di aggiunta prodotto al carrello). Interagisce con utentiPackage tramite “AggiungiProdotto.jsp” da “PaginaAmministratore.jsp” (Operazione di aggiunta di un nuovo prodotto) e tramite “Catalogo.jsp” da “PaginaAmministratore.jsp”(Operazioni di cancellazione di un prodotto e di modifica di un prodotto).

Il package utentiPackage contiene i file “PaginaAmministratore.jsp”, “PaginaUtente.jsp”, “ControlloUtenti.java”, “UtentiManager.java”, “PaginaUtente.jsp”, “PaginaLogin.html”, “PaginaRegistrazione.jsp”.  
Esso contiene tutte quelle pagine .jsp e .java responsabili di quelle operazioni riguardanti gli utenti e l’amministratore.  
Interagisce con acquistoPackage tramite “PaginaAmministratore.jsp” verso “OrdiniAmministratore.jsp” (Operazione di avanzamento dello stato di spedizione di un ordine), tramite “PaginaUtente.jsp” verso “OrdiniAmministratore.jsp” (Operazione di Conferma dell’arrivo di un ordine). Interagisce con prodottoPackage tramite “PaginaAmministratore.jsp” verso “AggiungiProdotto.jsp” (Operazione di aggiunta di un nuovo prodotto), verso “Catalogo.jsp” (Operazione di cancellazione di un prodotto e di modifica di un prodotto).

Il package acquistoPackage contiene i file “OrdiniAmministratore.jsp”, “ControlloOrdini.java”, “Ordine.java”, “OrdiniManager.java”, “Pagamento.jsp”.   
Esso contiene quei file responsabili di eseguire l’acquisto e di gestire la vita di un ordine.  
Interagisce con utentiPackage tramte “OrdiniAmministratore.jsp” da “PaginaAmministratore.jsp” (Operazione di avanzamento dello stato di spedizione di un ordine) e da “PaginaUtente.jsp” (Operazione di Conferma dell’arrivo di un ordine). Interagisce con carrelloPackage tramite “Pagamento.jsp” da “ControlloCarrello.java” (operazione di acquisto del carrello).

### **Class Interfaces**

Nel package carrelloPackage si individuano 2 classi:

ControlloCarrello.java  
Questa classe è il control object responsabile delle operazioni di gestione del carrello

+controlloAcquistoCarrello()   
Questo metodo permette di acquistare il carrello

+aggiungiProdottoCarrello()   
Questo metodo permette di aggiungere un prodotto al carrello

+rimuoviProdottoCarrello()   
Questo metodo permette di rimuovere un prodotto dal carrello

CarrelloManager.java  
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloCarrello.java effettuando le operazioni di persistenza dei dati relative al carrello.

+controlloAcquistoCarrello()  
Questo metodo permette di acquistare il carrello

+aggiungiProdottoCarrello()  
Questo metodo permette di aggiungere un prodotto al carrello

+rimuoviProdottoCarrello()  
Questo metodo permette di rimuovere un prodotto dal carrello

Nel package prodottiPackage si individuano 2 classi:

ControlloProdotti.java  
Questa classe è il control object responsabile delle operazioni sui prodotti

+aggiungiProdotto()   
Questo metodo permette all’amministratore di aggiungere un nuovo prodotto

+caricaProdotti()   
Questo metodo permette di ricercare tutti i prodotti  
+cancellaProdotto()   
Questo metodo permette all’amministratore di eliminare un prodotto esistente

+modificaProdotto()   
Questo metodo permette all’amministratore di modificare un prodotto esistente

+ricercaProdotti()   
Questo metodo permette di effettuare una ricerca sui prodotti

ProdottiManager.java  
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloProdotti.java effettuando le operazioni di persistenza dei dati relative ai prodotti

+aggiungiProdotto()   
Questo metodo permette all’amministratore di aggiungere un nuovo prodotto

+caricaProdotti()   
Questo metodo permette di ricercare tutti i prodotti  
+cancellaProdotto()   
Questo metodo permette all’amministratore di eliminare un prodotto esistente

+modificaProdotto()   
Questo metodo permette all’amministratore di modificare un prodotto esistente

+ricercaProdotti()   
Questo metodo permette di effettuare una ricerca sui prodotti

Nel package utentiPackage si individuano 2 classi:

ControlloUtenti.java   
Questa classe è il control object responsabile delle operazioni sugli utenti

+controlloCancella()   
Questo metodo permette ad un utente di cancellarsi

+controlloLoginAmministratore()  
Questo metodo permette l’autenticazione dell’amministratore

+controlloLoginUtente()   
Questo metodo permette l’autenticazione dell’utente

+controlloLogoutUtente()   
Questo metodo permette il logout dell’utente

+controlloLogoutAmministratore()   
Questo metodo permette il logout dell’amministratore

+controlloModificaDati()   
Questo metodo permette ad un utente di modificare i propri dati personali

+controlloRegistrazione()   
Questo metodo permette ad un nuovo utente di registrarsi

UtentiManager.java   
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloUtenti.java effettuando le operazioni di persistenza dei dati relative agli utenti

+controlloCancella ()   
Questo metodo permette ad un utente di cancellarsi

+controllologinAmministratore()   
Questo metodo permette l’autenticazione dell’amministratore

+controlloLoginUtente ()   
Questo metodo permette l’autenticazione dell’utente

+controlloModificaDati()  
Questo metodo permette ad un utente di modificare i propri dati personali

+controlloRegistrazione()  
Questo metodo permette ad un nuovo utente di registrarsi

Nel package acquistoPackage si individuano 2 classi:

ControlloAcquisto.java   
Questa classe è il control object responsabile delle operazioni sugli ordini

+caricamentoOrdini()   
Questo metodo permette all’amministratore di ricercare tutti gli ordini

+controlloSpedizione()   
Questo metodo permette di far avanzare lo stato di spedizione di un ordine

+controlloPagamento()   
Questo metodo permette di effettuare l’acquisto del carrello

AcquistoManager.java   
Questa classe è il model object responsabile delle operazioni di accesso al database della classe ControlloAcquisto.java effettuando le operazioni di persistenza dei dati relative agli ordini

+caricamentoOrdini()  
Questo metodo permette all’amministratore di ricercare tutti gli ordini

+controlloSpedizione()  
Questo metodo permette di far avanzare lo stato di spedizione di un ordine

+controlloPagamento()  
Questo metodo permette di effettuare l’acquisto del carrello

### **Team**

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Giuseppe Siani | 0512102958 |
| Carmine D’ Alessandro | 0512103402 |
| Aleandro Giuseppe Libano | 0512102976 |

### **Storia delle versioni**

|  |  |  |  |
| --- | --- | --- | --- |
| DATA | VERSIONE | DESCRIZIONE | AUTORE |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |