|  |
| --- |
| **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**MusicParadise  
Object Design Documentation  
Versione 0.1**

**LOGO PROGETTO**

****

Data: 19/12/2017

**Coordinatore del progetto:**

|  |  |
| --- | --- |
| Nome | Matricola |
|  |  |
|  |  |

**Partecipanti:**

|  |  |
| --- | --- |
| Nome | Matricola |
| **Antonio Spera** | 0512103538 |
| **Alessandro De Riso** | 0512103802 |
| **Domenico Pannone** | 0512103730 |
| **Vincenzo Pandolfo** | 0512103694 |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Scritto da:** |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
|  |  |  |  |

Sommario

[2 INTRODUZIONE 4](#_Toc501485419)

[2.1 OBJECT DESIGN TRADE-OFF 4](#_Toc501485420)

[2.1.1 MODULARITA’ VS EFFICIENZA 4](#_Toc501485421)

[2.1.2 SICUREZZA VS EFFICIENZA 4](#_Toc501485422)

[2.1.3 PORTABILITA’ VS EFFICIENZA 4](#_Toc501485423)

[2.2 INTERFACCE DOCUMENTATION GUIDELINES 5](#_Toc501485424)

[2.2.1 FILE JAVA 5](#_Toc501485425)

[2.2.2 NAMING 5](#_Toc501485426)

[2.2.3 USO DEI COMMENTI 5](#_Toc501485427)

[2.2.4 ALTRE REGOLE DI STILE 6](#_Toc501485428)

[2.3 DEFINIZIONI, ACRONIMI E ABBREVIAZIONI 6](#_Toc501485429)

[2.4 RIFERIMENTI 6](#_Toc501485430)

[2.5 OVERVIEW 6](#_Toc501485431)

[3 PACKAGES 7](#_Toc501485432)

[4 INTERFACCE DELLE CLASSI 8](#_Toc501485433)

[4.1 CLASS DIAGRAM 8](#_Toc501485434)

[4.2 DESCRIZIONE DELLE CLASSI 8](#_Toc501485435)

[5 GLOSARRIO 9](#_Toc501485436)

# INTRODUZIONE

## OBJECT DESIGN TRADE-OFF

### MODULARITA’ VS EFFICIENZA

### SICUREZZA VS EFFICIENZA

Nel nostro sistema ogni richiesta viene validata attraverso l’uso della sessione e un controllo a livello di utenza. Questo ci permette di impedire un accesso non autorizzato. Per fare ciò ogni pagina servlet o jsp deve autenticare l’utente oppure verificare che l’utente sia stato autenticato in precedenza. Questa caratteristica però potrebbe far alzare il tempo di risposta del sistema soprattutto con carichi di lavoro molto alti. Eliminando i controlli per l’esistenza e validità della sessione e introdurre la precondizione di sessione esistente, può portare a rischi di sicurezza molto elevati, in quanto utenti maliziosi potrebbero richiamare servizi non consentiti provocando danni al sistema. Per questo motivo anche se a discapito dell’efficienza optiamo per questi controlli.

### PORTABILITA’ VS EFFICIENZA

## INTERFACCE DOCUMENTATION GUIDELINES

### FILE JAVA

Ogni file deve contenere:

* Commenti per comprendere al meglio il codice
* Dichiarazione dei package
* Sezione import
* Dichiarazione di classe o interfaccia
  + Attributi pubblici
  + Attributi privati
  + Attributi protetti
  + Costruttori
  + Metodi getter e setter
  + Altri metodi
* Classi interne

### NAMING

Vengono utilizzate convenzioni su nomi per rendere il programma più leggibile e comprensibile.

**Classi e interfacce**

I nomi delle classi possono essere composti da più parole (ad esempio **ProdottoBean),** ogni parola che compone il nome della classe ha l’iniziale maiuscola. I nomi devono essere semplici e descrittivi in modo da capire il ruolo della classe. Evitare abbreviazioni.

**Metodi**

I **metodi** devono essere verbi (composti anche da più parole) con iniziale maiuscola. Ad esempio: addProdotto().

**Costanti**

Rispettando le convenzioni suggerite da Sun, i nomi delle **costanti** vengono indicati da nomi con tutti i caratteri maiuscoli. Le parole vengono separate da **“\_”.** Ad esempio: **static final MAX\_COUNT = 10**

### USO DEI COMMENTI

E’ possibile utilizzare due diversi tipi di commenti:

* Commenti **Javadoc** (testo compreso tra /\*\* e \*/)
* Commenti per singole righe (delimitate da //)

L’utilizzo dei commenti **Javadoc** è suggerito prima della dichiarazione di:

* Classi e interfacce
* Costruttori
* Metodi
* Variabili di classe

I commenti compresi tra /\*\* e \*/ devono essere semplici e chiari, in modo da rendere leggibile la documentazione.

### ALTRE REGOLE DI STILE

Ulteriori regole sono le seguenti:

* Nomi di package, classi e metodi devono essere nomi descrittivi
* Si consiglia l’utilizzo di parti standard dei nomi in casi come:
  + Classi astratte (es: AbstractClass)
  + Design pattern (es: ProdottoModel)
  + Eccezioni (es: ProdottoNonTrovatoException)
* è consigliato l’uso di suffissi standard come “get”, “set”, “is”,”has”

## DEFINIZIONI, ACRONIMI E ABBREVIAZIONI

|  |  |
| --- | --- |
|  |  |
| RAD | Documento di analisi dei requisiti |
| SDD | System Design Document |
| MVC | Model Control View |

## RIFERIMENTI

RAD-MusicParadise.com

SDD-MusicParadise.com

## OVERVIEW

# PACKAGES

Le componenti base che costituiscono il sistema sono raggruppati in livelli. I tre livelli rappresentano la suddivisione che deriva dal modello si architettura selezionato per il sistema, l’**MVC: Model Control View.** Ciascun livello rappresenta un **package.**

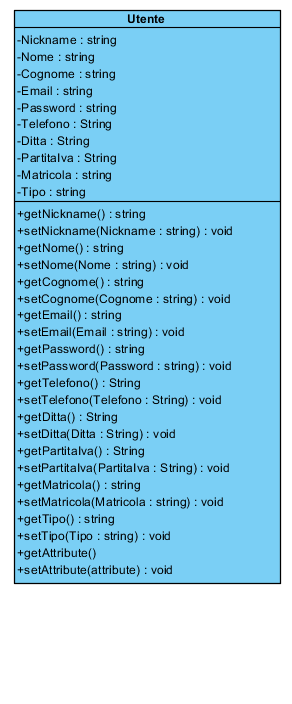
* **Model**
  + ProdottoModel
  + UtenteModel
  + OrdineModel
  + OffertaModel
  + FotoModel
* **Control**
  + RegistrazioneControl
  + AccediControl
  + RicercaProdottoControl
  + VisualizzaOrdiniCliente
  + AcquistoControl
  + LogOutControl
  + AggiungiPordottoCarrelloControl
  + RimuoviProdottoCarrelloControl
  + AggiungiProdottoCatalogo
  + RimuoviProdottoCatalogo
* **View**
  + Index
  + Catalogo
  + OrdiniCliente
  + Prodotto
  + profiloCliente
  + registrazione
  + Carrello

# INTERFACCE DELLE CLASSI

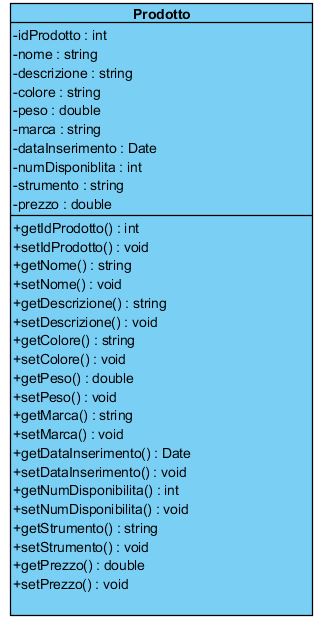
## CLASS DIAGRAM

## DESCRIZIONE DELLE CLASSI

### Utente



### Prodotto

**idProdotto** : Valore che identifica univocamente il prodotto nel sistema;

**nome**: Nome del prodotto;

**descrizione**: Descrizione dettagliata del prodotto;

**colore**: Colorazione in cui è disponibile il prodotto;

**peso**: Peso del prodotto;

**marca**: Marca del prodotto;

**dataInserimento**: Data in cui il prodotto è stato inserito nel catalogo;

**numDisponibilita**: Quantità disponibile delle unità di prodotto;

**strumento**: Famiglia degli strumenti di cui fa parte il prodotto;

**prezzo**: prezzo del prodotto;

# GLOSARRIO