****

**UNIVERSITÀ DEGLI STUDI DI SALERNO**

**Ingegneria del Software**

**ANNO ACCADEMICO 2015/2016**

**LOGO**

****

**OBJECT DESIGN**

***Versione 1.0***

**Top Manager:**

|  |  |
| --- | --- |
|  | **Nome** |
| **Prof. De Lucia Andrea** | |

**Partecipanti:**

|  |  |
| --- | --- |
| **Partecipante** | **Matricola** |
| Carfora Andrea | 0512102638 |
| Grappone Renato | 0512102536 |

**Revision History:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autore** |
| 24/10/2015 | 1.0 | Object Design | Carfora Andrea  Grappone Renato |

**Indice**

### **Introduzione**

* 1. **Obiettivi design Trade-Offs**
  2. **Linee Guida per l’interfaccia**
  3. **Definizioni,Acronimi e Abbreviazioni**
  4. **Riferimenti**

1. **Calss Interface**
2. **Glossario**

# Introduzione

1.1. **Compromessi dell'object design**

Nella fase di Object design sorge la necessità di prendere delle decisioni. Alcuni dei trade-off relativi al progetto SDI sono:

a) ***Comprensibilità vs Costi***: La comprensibilità del codice è un aspetto molto importante, soprattutto durante la fase di testing. Ogni classe e metodo deve essere facilmente interpretabile anche da chi non ha collaborato al progetto. Una appropriata documentazione del codice sorgente aumentala comprensibilità dello stesso. Ovviamente questa caratteristica comporterà un incremento del tempo necessario per lo sviluppo del progetto.

b) ***Buy vs Build***: Il compromesso adottato è quello espresso dall'affermazione “*If you can get 80% of the functionality you need from a packaged solution, then buy it. Otherwise, build it*”. Naturalmente, il sistema SDI può essere suddiviso in moduli. Se riusciamo a trovare qualche modulo che include la gran parte delle funzionalità di cui abbiamo bisogno, allora possiamo utilizzarlo, in modo tale da risparmiare tempo e risorse. Quando viene riutilizzato un modulo esistente, quindi, si ha il vantaggio di avere una soluzione pronta per l'uso. Tuttavia, se lo sviluppatore non sa come tale modulo funziona (perché il codice sorgente non è reso disponibile), integrarlo nel sistema potrebbe risultare molto difficile, richiedendo costi maggiori di quelli attesi.

c) ***Funzionalità vs Tempo di consegna***: Le funzionalità costituiscono un aspetto fondamentale del sistema. Tuttavia, la data di consegna è fissata e, pertanto sorge la necessità di trovare un buon bilanciamento. Ciò comporta che prima della consegna, occorre realizzare tutte le funzionalità di base del sistema.

1.2. **Linee guida per la documentazione delle interfacce**

Al fine di rendere più uniforme la comunicazione tra i membri del team di sviluppo, e rendere i

programmi più comprensibili, adottiamo specifiche linee guida e convenzioni relative alla documentazione delle interfacce.

Per maggiori dettagli sugli standard di codifica, si rimanda al documento di specifica delle linee guida per il codice Java.

***Denominazione***:

Le convenzioni sono molto utili nella comprensione del codice sorgente in quanto possono fornire informazioni anche attraverso la semplice funzione di identificatore, ad esempio, se è una costante, un package, o una classe.

Questa sezione fornisce una panoramica sulle convenzione di denominazione. Esse sono descritte singolarmente nella tabella.

|  |  |  |
| --- | --- | --- |
| **Identifier Type** | **Rules for Naming** | **Examples** |
| **Classi** | I nomi delle classi dovrebbero essere sostantivi scritti secondo il sistema Camel. I nomi delle classi che modellano un eccezione devono terminare per “ Exception”. I nomi delle classi devono essere semplici e descrittivi e per quanto possibile evitare acronimi e abbreviazioni. | Class: AccountManager  Exception: EmailException |
| **Metodi** | I metodi devono essere verbi, e devono seguire il sistema Camel, in cui la prima lettera di ogni parola è in maiuscolo ad eccezione della prima. | setNomeProdotto();  setDescrizione(); |
| **Variabili** | I nomi delle variabili devono essere brevi ma significative. Seguono il sistema camel in cui la prima lettera è minuscola e ,in caso di parole composte, la lettera che delinea l’inizio della successiva deve essere maiuscola. | codice,  nomeProdotto,  descrizione |
| **Costanti** | I nomi delle costanti sono maiuscoli con le parole separate da underscore. | ACCOUNT\_INSERT |
| **Parametri** | I nomi dei parametri devono seguire le convenzioni definite per le variabili |  |
| **Package** | I nomi dei package seguono il sistema Camel | Exception  Entità |

1.3. **Definizioni, acronimi e abbreviazioni *Definizioni:***

* ***Trade-off:*** Situazione in cui si pone una scelta di compromessa da due obiettivi ugualmente desiderabili, ma in conflitto tra di loro.
* ***Package:*** Insieme di classi java.
* ***JavaDoc:*** Strumento che estrae dai commenti di un programma una documentazione  dettagliata del codice.
* ***Acronimi:***
* **RAD:** Requirement Analysis Document.
* **SDD:** System Design Document.
* **ODD:** Object Design Document.
* ***Abbreviazioni:***

**SI:** Sportswear Identification.

1.4. **Referenze**

-RAD; -SDD;

- Linee guida codice Java.

**2. PACKAGE**

Per il raggruppamento delle classi del nostro progetto si è cercato di mantenere la struttura identificata all'interno del SDD, cioè i tre layer principali, con l'aggiunta di altri. Per la parte Client del sistema i package sono i seguenti:

|  |
| --- |
| **PACKAGE** |
| Entità |
| Exceptions |
| Managers |
| DbMom |
| Servlet |

Per la parte Web, invece, i package sono i seguenti:

|  |
| --- |
| GestioneLoginAdmin |
| GestioneLoginCliente |
| GestioneBilancioAdmin |
| GestioneClientiAdmin |
| GestioneMagazzinoAdmin |
| GestioneUtenteAdmin |

2.1. **Descrizioni dei package**

**2.1.1. SI Client**

|  |  |
| --- | --- |
| **Nome package** | **Descrizione** |
| Entità | Package che contiene le entità chiavi del sistema |
| Exceptions | Package che contiene le eccezioni che si possono verificare all’interno del sistema |
| Managers | Package che contiene i gestori delle operazioni che possono essere effettuate su un oggetto contenuto in Entità. |
| dBMom | Package che contiene gli oggetti che gestiscono la comunicazione con il database. |

**2.1.2. SI Web**

|  |  |
| --- | --- |
| **Nome package** | **Descrizione** |
| GestioneLoginAdmin | Package che contiene la gestione del login da parte dell’admin |
| GestioneLoginCliente | Package che contiene la gestione del login da parte del cliente |
| GestioneBilancioAdmin | Package che contiene le classi che implementano le azioni sulla gestione del bilancio dell’admin. |
| GestioneClientiAdmin | Package che contiene le classi che implementano le azioni sulla gestione clienti dell’admin |
| GestioneMagazzinoAdmin | Package che contiene le classi che implementano le azioni sulla gestione del magazzino da parte dell’admin ( Visualizza prodotti, categorie, etc ) |
| GestioneUtenteAdmin | Package che contiene le classi che implementano le azioni sulla gestione utenti da parte dell’admin. |
| Exceptions | Package che contiene le eccezioni che possono verificarsi nel sistema |

/\* FATTO

# Overall Strategy and Approach

## Testing Strategy

<Project Name> System Testing will include testing of all functionalities that are in scope (Refer Functional Scope Section) identified. System testing activities will include the testing of new functionalities, modified functionalities, screen level validations, work flows, functionality access, testing of internal & external interfaces.

## System Testing Entrance Criteria

In order to start system testing, certain requirement must be met for testing readiness. The readiness can be classified into:

## Testing Types

### Usability Testing

User interface attributes, cosmetic presentation and content will be tested for accuracy and general usability. The goal of Usability Testing is to ensure that the User Interface is comfortable to use and provides the user with consistent and appropriate access and navigation through the functions of the application (e.g., access keys, consistent tab order, readable fonts etc.)

### Functional Testing

The objective of this test is to ensure that each element of the component meets the functional requirements of the business as outlined in the:

* Business / Functional Requirements
* Business rules or conditions
* Other functional documents produced during the course of the project i.e. resolution to issues/change requests/feedback

## Suspension Criteria and Resumption Requirements

This section will specify the criteria that will be used to suspend all or a portion of the testing activities on the items associated with this test plan. /\* FATTO