# Introduzione

Una volta redatti i documenti relativi all’analisi dei requisiti e all’identificazione degli obiettivi di design passiamo alla realizzazione di un ulteriore documento, L’Object Design Document.

Tale documento integrerà tutte le informazioni già evidenziate nei precedenti documenti in modo chiaro e preciso, servirà come base per l’implementazione e conterrà:

* Le specifiche legate alle interfacce per ogni sottosistema;
* Le librerie per le classi;
* I design pattern al fine di risolvere problemi e proteggere le classi da futuri cambiamenti.

## Object Design Trade-offs

In questa sezione verranno definiti i trade-offs relativi al sistema “Youth Club”

* **Prestazioni vs. Costi,** considerando il budget a disposizione, è stato di utilizzare componenti open source, con l’obiettivo di minimizzare i costi in rapporto alla qualità delle funzione rilasciate;
* **Interfaccia vs Tempo di risposta,** per garantire tempi di risposta sufficientemente brevi è stato deciso di focalizzare l’attenzione sulla realizzazione di una parte di back-end il più lineare possibile in modo da ridurre i tempi di risposta del sistema;
* **Interfaccia vs. Usabilità,** grazie all’utilizzo delle componenti scelte si garantisce un design gradevole e facilmente comprensibile; e conseguentemente un’interfaccia facilmente usabile;
* **Costi vs. Mantenimento,** grazie all’utilizzo delle componenti e i linguaggi d’implementazione scelti, il sistema risulterà essere facilmente mantenibile a costi ridotti.

## Linee guida per l’implementazione

Le linee guide per lo sviluppo del progetto software sono dettate da “Java™ Platform, Standard Edition 7 API Specification” documento web contenete regole ben precise sullo standard da seguire durante l’implementazione.

Linee guida:

le linee guida sono molto importanti in quanto vanno a stabilire i Naming Convention dei componenti utilizzati.

**Nome variabili:**

I nomi delle variabili devono essere brevi e significativi, possono essere formati da lettere e cifre la cosa importante che il primo elemento deve essere una lettera minuscola.

Nel caso in cui il nome della variabile è formato da più parole, le successive alla prima devono iniziare con la maiuscola (Ex: primoNome).

Le variabili locali saranno dichiarate all’interno dei metodi, quindi saranno create solo quando il metodo verrà invocato e cancellate alla terminazione del metodo. (NB le variabili devono essere inizializzate prima del utilizzo)

**Inserire nostro esempio**

Le costanti dove il valore messo al loro interno non può essere modificato per definire il loro identificativo può essere usato “\_” per separare le parole che lo compongono (EX PRIMO\_NOME)

**Inserire nostro esempio**

**Nome metodi:**

I metodi definisco comportamenti i comportamenti degli oggetti.

I nome dei metodi inizieranno con la *lettera minuscola* e, qualora formati da più parole le successive dovranno iniziare con la maiuscola “nomeMetodo()”.

Il nome del metodo deve avere un significato cioè va a specificare l’azione che viene effettuata.

Ai metodi viene aggiunto un commento in JavaDoc, il quale verrà posizionato prima della dichiarazione del metodo e serve a descriverne lo scopo. Nel commento sono anche presenti delle informazioni riguardanti i paramenti, il valore di ritorno e le possibili eccezioni che potrebbero essere lanciate.

**Nome classi:**

I nome delle classi dovranno iniziare con la lettera maiuscola come tutte le parole che seguiranno. (EX NomeClasse)

Il nome della classe deve essere intuitivo e comprensibile. Ogni file sorgente(.class) ha al suo interno una sola classe l’introduzione alla classe che conterrà lo scopo di quest’ultima

/\*

\*scopo della classe

\*/

La classe ha una struttura da seguire:

* Dichiarazione classe pubblica
* Dichiarazione delle costanti
* Dichiarazione delle variabili di classe e d’istanza
* Costruttore
* Metodi e i relativi commenti

**Inserire screen**

**Nomi package:**

I nome dei pacchetti dovranno essere scritti in minuscolo concatenando un insieme di sostantivi separati dal punto. In generale, un nome comincia con il dominio di primo livello dell’organizzazione che lo produce, seguito dal dominio e da altri eventuali sottodomini, scrivendo infine un nome specifico per quel particolare package. NB non sono ammessi caratteri particolari.

## Definizioni, acronomi e abbreviazioni

**Acronomi:**

RAD: Requirements Analysis Document

SDD: System Design Document  
ODD: Object Design Document  
DAO: Data Access Object

# Riferimeti

Bla bla bla bla

**COTS (Components Off The Shelf)**

in questa sezione definiamo tutte le componenti hardware e software disponibili sul mercato che andremmo ad usare nel nostro progetto