Università degli Studi di Salerno

Corso di Ingegneria del Software
Classe 1 Resto 0
Corso di Laurea in Informatica
A.A. 2022/23

Quiad
Object Design Document

Versione 2.0
02/12/2022

Partecipanti al progetto e scriventi

Nome	Matricola
Di Pasquale Valerio	0512110638
Troisi Vito	0512109807

Revision History

Data	Versione	Descrizione	Autore
30/11/2022	1.0	Prima stesura ODD e Indice	D.P.V. T.V.
02/12/2022	2.0	Sezione introduttiva e vincoli in OCL	D.P.V. T.V.

Indice

Overview

Nel presente documento, sarà esplicitata la progettazione di dettaglio del sistema Quiad, per la gestione del proprio albero genealogico.

Lo Object Design Document vuole presentare i trade-offs di progettazione e, per quanto concerne la *slice* del sistema che sarà implementata, la specifica delle interfacce dei moduli e l'organizzazione delle classi dei medesimi. Seguono la presente sezione introduttiva, alcune linee guida utili per la comprensione delle scelte di design e di specifica delle interfacce suddette.

Trade-offs di Object Design

Linee guida per la documentazione delle interfacce

Le interfacce delle classi dei moduli che saranno soggetti ad implementazione, saranno specificate tenendo conto delle seguenti prassi:

- I nomi di attributi, metodi e parametri saranno esplicitati in camel-case.
- I nomi dei metodi getter/setter legati ad un attributo seguiranno la forma "get...()" e "set...()", imponendovi il nome dell'attributo.
- Precondizioni, postcondizioni ed invarianti di classe saranno esplicitati mediante OCL (Object Constraint Language).

Definizioni, acronimi, abbreviazioni

Una lista alfabetizzata di definizioni ed acronimi utili per la lettura della presente:

- RAD: Requirements Analysis Document.
- RMD: Rationale Management Document, i.e. il documento in cui è riportato il razionale dietro alcune scelte fatte in fase di progettazione.
- OCL: Object Constraint Language, il quale sarà impiegato per esplicitare i contratti di ogni metodo, e per ciascuna classe.
- ODD: Object Design Document, il presente documento.

- SDD: System Design Document, i.e. il documento di progettazione architetturale del sistema.

Riferimenti

Al fine di garantire una migliore comprensione dello ODD, si invita ad osservare il Requirements Analysis Document, in particolar modo le sezioni 3.5.3 e 3.5.4, rispettivamente legate alla modellazione ad oggetti ed alla modellazione dinamica.

Si invita inoltre a consultare il SDD per analizzare le scelte legate alla progettazione di sistema, quale la gestione dei dati persistenti, esplicitata alla sezione 2.4.

Interfacce delle classi

```
context UtenteBase inv:
     tree->include(getUserNode()) Λ
     getUserNode() != null

context UtenteBase::addNode(n) pre:
     n != null Λ
     n.getFather() != n Λ
     n.getMother() != n Λ
     tree->include(n.getFather()) Λ
     tree->include(n.getMother())

context UtenteBase::addNode(n) post:
     tree->include(n) Λ
     tree->n.getFather().getDescendants()->asSet()->include(n) Λ
     tree->n.getMother().getDescendants()->asSet()->include(n)

context UtenteBase::modifyNode(m,n) pre:
     n != null Λ
     m != null Λ
     tree->include(m) Λ
     m.getFather() = n.getFather() Λ
     m.getMother() = n.getMother()

context UtenteBase::modifyNode(m,n) post:
     !tree->include(m) Λ
     tree->include(n)

context UtenteBase::deleteNode(n) pre:
     n != null Λ
```

```
     tree->include(n) Λ
     n != getUserNode()


context UtenteBase::deleteNode(n) post:
     !tree->include(n) Λ
     forAll(s|n.getDescendants()) !tree->include(s)



context Node inv:
     (getBirthDate() != null Λ getDeathDate() != null) ⇒
          getBirthDate() < getDeathDate()

context Node::bindDocument(d) pre:
     d != null Λ
     !self.getBoundDocuments()->asSet()->include(d)

context Node::bindDocument(d) post:
     self.getBoundDocuments()->asSet()->include(d)

context Node::unbindDocument(d) pre:
     d != null Λ
     self.getBoundDocuments()->asSet()->include(d)

context Node::unbindDocument(d) post:
     !self.getBoundDocuments()->asSet()->include(d)
```