**Università degli Studi di Salerno**

**Corso di Ingegneria del Software**

**BookPoint**

**ODD**

**Versione 1.0**

Partecipanti:

|  |  |
| --- | --- |
| **Nome Cognome** | **Matricola** |
| **Massimo Giordano** | **0512104480** |
| **Giovanni Buonocore** | **0512104612** |
| **Gennaro Teodoro** | **0512104876** |

Sommario

[1 Introduzione 3](#_Toc563263)

[1.1 Object Design Trade-offs 3](#_Toc563264)

[1.2 Linee Guida per la documentazione delle interfacce 3](#_Toc563265)

[1.3 Definizioni, acronimi e abbreviazioni 4](#_Toc563266)

[1.4 Riferimenti 5](#_Toc563267)

[2 Packages 5](#_Toc563268)

[2.1 control 5](#_Toc563269)

[2.1.1 gestioneAccount 5](#_Toc563270)

[2.1.2 gestioneAmministratore 6](#_Toc563271)

[2.1.3 gestioneAmministratoreOrdini 6](#_Toc563272)

[2.1.4 gestioneRicerca 7](#_Toc563273)

[2.1.5 gestioneOrdine 7](#_Toc563274)

[2.1.6 gestioneInterazioneLibro 7](#_Toc563275)

[2.1.7 gestioneAcquisto 7](#_Toc563276)

[2.2 model 8](#_Toc563277)

[2.3 bean 8](#_Toc563278)

[2.4 Carrello 9](#_Toc563279)

[2.5 connectionPool 9](#_Toc563280)

[2.6 presentation 9](#_Toc563281)

[2.6.1 presentation amministratore 9](#_Toc563282)

[2.6.2 presentation amministratore ordini 10](#_Toc563283)

[2.6.3 presentation cliente 10](#_Toc563284)

[3 Class interfaces 11](#_Toc563285)

[3.1 Interfacce Bean 11](#_Toc563286)

[3.2 AccountManager 12](#_Toc563287)

[3.3 AmministratoreManager 13](#_Toc563288)

[3.4 AmministratoreOrdiniManager 15](#_Toc563289)

[3.5 LibroManager 16](#_Toc563290)

[3.6 InterazioneLibroManager 17](#_Toc563291)

[3.7 GestioneOrdineManager 18](#_Toc563292)

[3.8 OrdineManager 19](#_Toc563293)

[4 Design Pattern 20](#_Toc563294)

[4.1 Object Pool Pattern 20](#_Toc563295)

# Introduzione

## Object Design Trade-offs

**Comprensibilità vs Tempo:**Il codice del sistema deve essere comprensibile, in modo da facilitare la fase di testing ed eventuali future modifiche da apportare. Al fine di rispettare queste linee guida il codice sarà integrato da commenti volti a migliorarne la leggibilità; tuttavia questo richiederà una maggiore quantità di tempo necessario per lo sviluppo del nostro progetto.

**Prestazioni vs Costi:**Dal momento che il budget allocato è spendibile principalmente in risorse umane e non consente l’acquisto di tecnologie proprietarie specifiche verranno utilizzati template open source e componenti hardware di nostra proprietà.

**Interfaccia vs Usabilità:**Verrà realizzata un’interfaccia grafica chiara e concisa, usando form e pulsanti predefiniti che hanno lo scopo di rendere semplice l’utilizzo del sistema da parte dell’utente finale.

**Sicurezza vs Efficienza:**La sicurezza rappresenta uno degli aspetti principali del sistema. Tuttavia, a causa di tempi di sviluppo molto limitati, ci limiteremo ad implementare sistemi di sicurezza basati su email e password.

## Linee Guida per la documentazione delle interfacce

Gli sviluppatori dovranno seguire precise linee guida per la stesura del codice:

**Naming Convention:**

Per la documentazione delle interfacce bisognerà utilizzare nomi:

* Descrittivi
* Pronunciabili;
* Di lunghezza medio-corta;
* Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

**Variabili:**

* I nomi delle variabili dovranno iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola.
* In ogni riga dovrà esserci un’unica variabile dichiarata, eventualmente allineata con quelle del blocco dichiarativo.
* In determinati casi, è possibile utilizzare il carattere underscore “\_”: il caso principale previsto è quello relativo alla dichiarazione di costanti oppure di proprietà statiche.

**Metodi:**

* I nomi dei metodi dovranno iniziare con la lettera minuscola, e le parole successive con la lettera maiuscola, secondo la “Camel Notation”.
* Il nome del metodo sarà costituito da un verbo che ne identifica l’azione seguito da un sostantivo, eventualmente aggettivato.
* Il nome dei metodi accessori e modificatori seguirà, rispettivamente, i pattern getNomeVariabile e setNomeVariabile.
* Ai metodi verrà aggiunto un commento JavaDoc, il quale deve essere posizionata prima della dichiarazione del metodo, e deve descriverne lo scopo.

**Classi Java e pagine JSP:**

* I nomi delle classi e delle pagine dovrà iniziare con la lettera maiuscola, così come le parole successive all’interno del nome.
* I nomi delle classi e delle pagine dovrà corrispondere alle informazioni e le funzioni fornite da quest’ultime.
* Le classi saranno strutturate prevedendo rispettivamente:

1. Dichiarazione della classe pubblica;
2. Dichiarazione di costanti;
3. Dichiarazioni di variabili di classe;
4. Dichiarazioni di variabili di istanza;
5. Costruttore;
6. Metodi;

**Packages:**

* I nomi dei packages dovranno essere scritti in minuscolo concatenando insieme diversi sostantivi o sigle, separate dal carattere “.” .
* Non saranno ammessi caratteri speciali.

## Definizioni, acronimi e abbreviazioni

**Acronimi:**

* RAD: Requirements Analysis Document
* SDD: System Design Document
* ODD: Object Design Document

## Riferimenti

Documento RAD\_BookPoint.pdf

Documento SDD\_BookPoint.pdf

# Packages

Il sistema è in diviso orizzontalmente in package come segue:

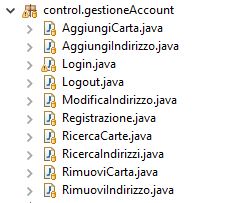
* control
* model
* bean
* carrello
* connectionPool
* presentation
* test

## **control**

Il package control è suddiviso verticalmente in package come segue:

* gestioneAccount
* gestioneAmministratore
* gestioneAmministratoreOrdini
* gestioneRicerca
* gestioneOrdine
* gestioneInterazioneLibro
* gestioneAcquisto

### **gestioneAccount**



|  |  |
| --- | --- |
| Classe | Descrizione |
| AggiungiCarta | Servlet che gestisce l’aggiunta di una carta di credito |
| AggiungiIndirizzo | Servlet che gestisce l’aggiunta di un nuovo indirizzo |
| Login | Servlet che permette di effettuare il login |
| Logout | Servlet che permette di effettuare il logout |
| ModificaIndirizzo | Servlet che gestisce la modifica di un indirizzo |
| Registrazione | Servlet che gestisce la registrazione di una nuovo account |
| RicercaCarte | Servlet che gestisce la ricerca di carte |
| RicercaIndirizzi | Servlet che gestisce la ricerca di indirizzi |
| RimuoviCarta | Servlet che gestisce la rimozione di una carta di credito |
| RimuoviIndirizzo | Servlet che gestisce la rimozione di un indirizzo |

### gestioneAmministratore

****

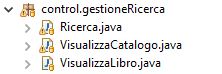
|  |  |
| --- | --- |
| Classe | Descrizione |
| AggiungiLibro | Servlet che gestisce l’aggiunta di un libro |
| CambiaTipo | Servlet che gestisce il cambiamento del tipo di un libro |
| EliminaLibro | Servlet che gestisce l’eliminazione di un libro |
| EliminaRecensione | Servlet che gestisce l’eliminazione di una recensione |
| EliminaUtente | Servlet che gestisce l’eliminazione di un utente |
| ModificaAttributo | Servlet che gestisce la modifica di un attributo |
| RicercaAccount | Servlet che gestisce la ricerca di un account |

### gestioneAmministratoreOrdini

****

|  |  |
| --- | --- |
| Classe | Descrizione |
| CambiaDataEOra | Servlet che gestisce il cambiamento della data e l’ora |
| CambiaStato | Servlet che gestisce il cambiamento dello stato di un ordine |
| RicercaOrdine | Servlet che gestisce la ricerca di un ordine |

### gestioneRicerca

****

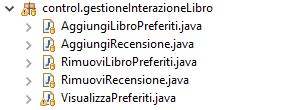
|  |  |
| --- | --- |
| Classe | Descrizione |
| Ricerca | Servlet che gestisce la ricerca di un libro |
| VisualizzaCatalogo | Servlet che gestisce la visualizzazione del catalogo |
| VisualizzaLibro | Servlet che gestisce la visualizzazione di un libro |

### gestioneOrdine

****

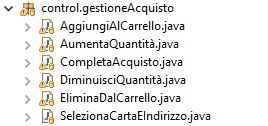
|  |  |
| --- | --- |
| Classe | Descrizione |
| VisualizzaFattura | Servlet che gestisce la visualizzazione della fattura |
| VisualizzaStorico | Servlet che gestisce la visualizzazione dello storico |

### gestioneInterazioneLibro

****

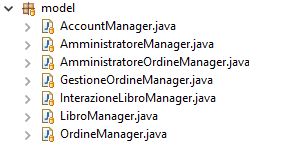
|  |  |
| --- | --- |
| Classe | Descrizione |
| AggiungiLibroPreferiti | Servlet che gestisce l’aggiunta dei libri ai preferiti |
| AggiungiRecensione | Servlet che gestisce l’aggiunta di una recensione ad un libro |
| RimuoviLibroPreferiti | Servlet che gestisce l’eliminazione dei libri dai preferiti |
| RimuoviRecensione | Servlet che gestisce l’eliminazione di una recensione di un libro |
| VisualizzaPreferiti | Servlet che gestice la visualizzazione dei libri preferiti |

### gestioneAcquisto

****

|  |  |
| --- | --- |
| Classe | Descrizione |
| AggiungiAlCarrello | Servlet che gestisce l’aggiunta al carrello di un libro |
| AumentaQuantità | Servlet che gestisce l’aumento della quantità di un libro |
| CompletaAcquisto | Servlet che gestisce il completamento dell’acquisto di un libro |
| DiminuisciQuantità | Servlet che gestisce la diminuzione della quantità di un libro |
| EliminaDalCarrello | Servlet che gestisce l’eliminazione di un libro dal carrello |
| SelezionaCartaEIndirizzo | Servlet che gestisce la gestione di una carta e di un indirizzo |

## model

****

|  |  |
| --- | --- |
| Classe | Descrizione |
| AccountManager | Classe che implementa le operazioni riguardanti l’account |
| AmministratoreManager | Classe che implementa le operazioni effettuabili dall’amministratore |
| AmmministratoreOrdineManager | Classe che implementa le operazioni effettuabili dall’amministratore degli ordini |
| GestioneOrdineManager | Classe che implementa le operazioni di gestione degli ordini e visualizzazione della fattura |
| InterazioneLibroManager | Classe che implementa le operazioni di gestione delle recensioni e gestione dei preferiti |
| LibroManager | Classe che implementa la ricerca e visualizzazione dei libri |
| OrdineManager | Classe che implementa la memorizzazione di un ordine |

## bean

****

|  |  |
| --- | --- |
| Classe | Descrizione |
| Autore | Classe che rappresenta le informazioni relative ad un autore |
| CartaDiCredito | Classe che rappresenta le informazioni relative ad una carta di credito |
| Indirizzo | Classe che rappresenta le informazioni relative ad un indirizzo |
| Libro | Classe che rappresenta le informazioni relative ad un libro |
| Ordine | Classe che rappresenta le informazioni relative ad un ordine |
| Recensione | Classe che rappresenta le informazioni relative ad una recensione |
| Utente | Classe che rappresenta le informazioni relative ad un utente |

## Carrello

****

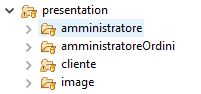
|  |  |
| --- | --- |
| Classe | Descrizione |
| Carrello | Classe che rappresenta un carrello, con le relative operazioni |

## connectionPool

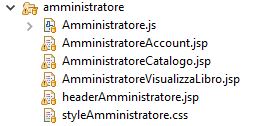
****

|  |  |
| --- | --- |
| Classe | Descrizione |
| DriverMaagerConnectionPool | Classe che implementa l’Object pool pattern, responsabile di fornire le connessioni al database. |

## presentation

****

### presentation amministratore

****

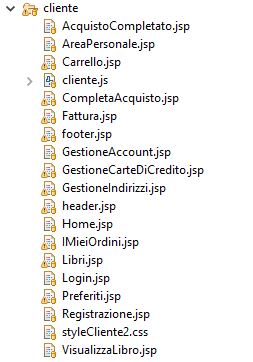
|  |  |
| --- | --- |
| Classe | Descrizione |
| AmministratoreAccount | Sezione in cui l’amministratore può gestire gli account, cambiando il loro tipo o eliminandoli |
| AmministratoreCatalogo | Sezione in cui l’amministratore può visualizzare e aggiungere i libri |
| AmministratoreVisualizzaLibro | Sezione in cui l’amministratore può visualizzare e modificare le informazioni di un libro |

### presentation amministratore ordini

****

|  |  |
| --- | --- |
| Classe | Descrizione |
| AmministratoreOrdiniOrdine | Sezione in cui l’amministratore può visualizzare un ordine e modificarne lo stato, la data e l’ora di consegna |

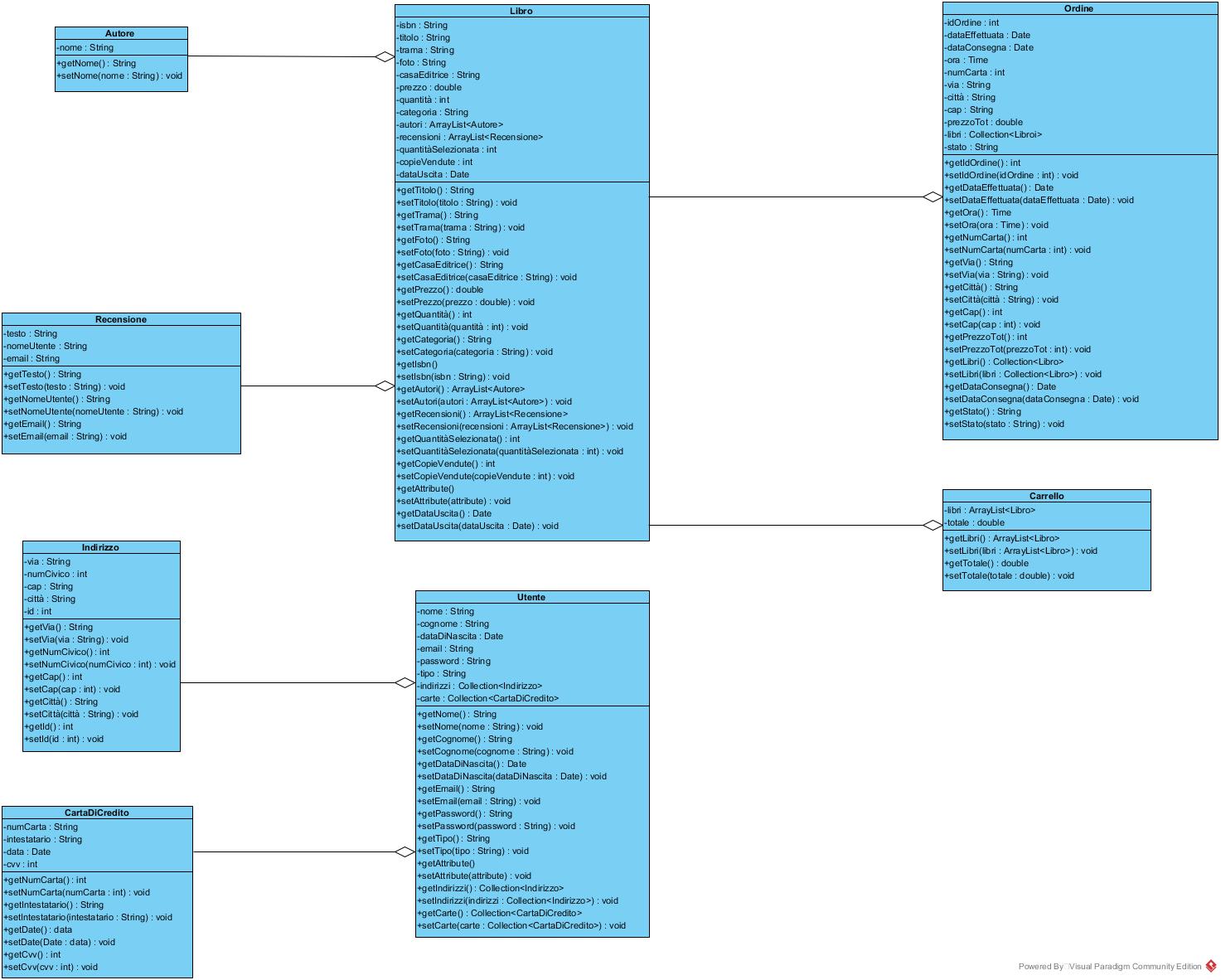
### presentation cliente

****

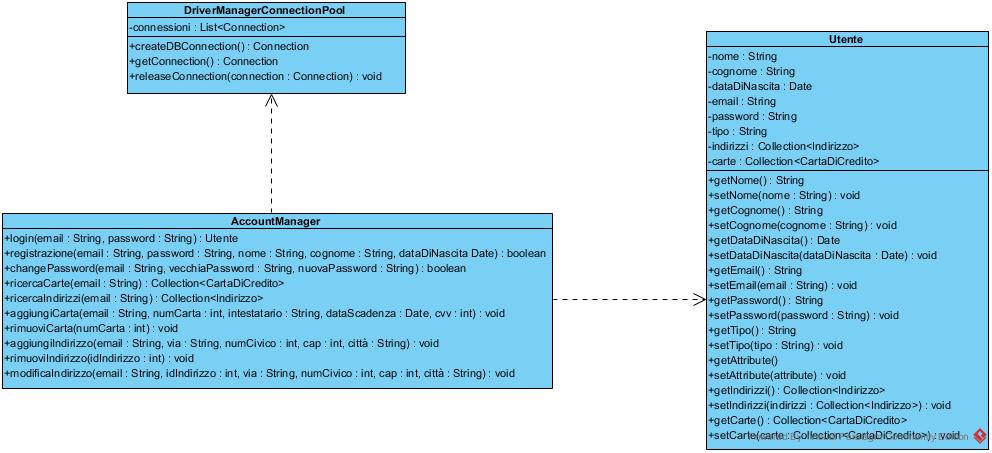
|  |  |
| --- | --- |
| Classe | Descrizione |
| AcquistoCompletato | Pagina che mostra all’utente l’avvenuto completamento dell’acquisto |
| AreaPersonale | Sezione che mostra l’area personale di un utente registrato |
| Carrello | Pagina che mostra il carrello |
| CompletaAcquisto | Pagina che permette di completare un acquisto |
| Fattura | Pagina che mostra la fattura |
| GestioneAccount | Sezione dove l’utente può cambiare la password |
| GestioneCarteDiCredito | Sezione dove l’utente può visualizzare le proprie carte registrate |
| GestioneIndirizzi | Sezione dove l’utente può visualizzare gli indirizzi registrati |
| Home | Pagina iniziale del sistema |
| IMieiOrdini | Sezione dove l’utente può visualizzare lo storico degli ordini |
| Libri | Pagina che mostra i libri |
| Login | Pagina che permette di effettuare il login |
| Preferiti | Pagina che mostra i libri aggiunti ai preferiti |
| Registrazione | Pagina che permette di registrare un nuovo account |
| VisualizzaLibro | Pagina che mostra le informazioni riguardanti un libro con le relative recensioni |

# Class interfaces

## Interfacce Bean

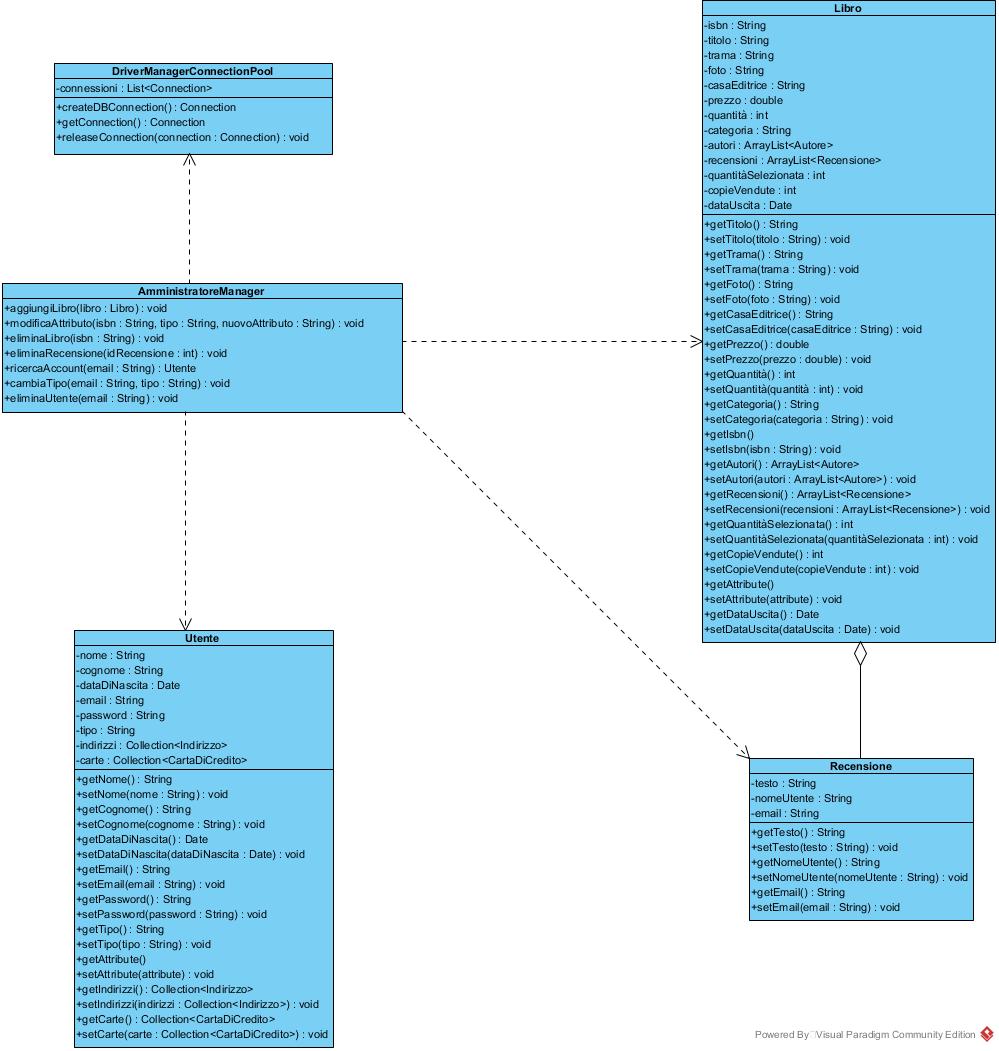


## AccountManager



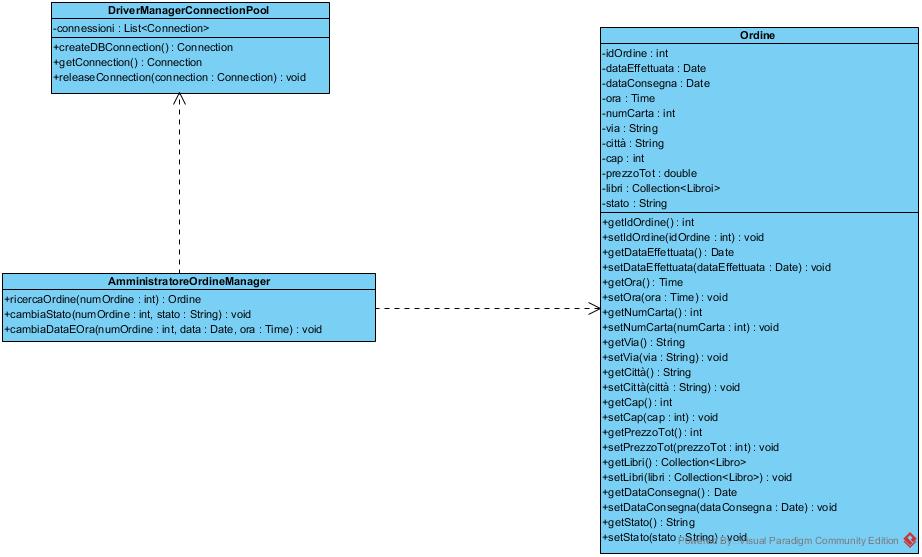
|  |  |
| --- | --- |
| Nome Classe | AccountManager |
| Pre-condizione | **context** AccountManager::login(email: String, password: String) : Utente **pre** email!=null && password != null;  **context** AccountManager::registrazione(Utente utente) : boolean **pre** utente!=null;  **context** AccountManager:: changePassword(email: String, vecchiaPassword: String, nuovaPassword: String) : boolean **pre** email!= null && vecchiaPassword!= null && nuovaPassword!=null;  **context** AccountManager:: ricercaCarte(email : String) : Collection<CarteDiCredito> **pre** email!=null ;  **context** AccountManager:: ricercaIndirizzi(email : String) : Collection<Indirizzo> **pre** email!=null ;  **context** AccountManager:: aggiungiCarta(email : String, carta : CartaDiCredito) : void **pre** email!=null && utente!= null ;  **context** AccountManager:: aggiungiIndirizzo(email : String, indirizzo : Indirizzo) : void **pre** email!=null && indirizzo != null; |
| Post-condizione | **context** AccountManager:: changePassword(email: String, vecchiaPassword: String, nuovaPassword: String) : boolean **post** utente.password=nuovaPassword  **context** AccountManager:: aggiungiCarta(email : String, carta : CartaDiCredito) : void **post** getCarte.size()= @pre.getCarte.size()+1  **context** AccountManager:: aggiungiIndirizzo(email : String, carta : CartaDiCredito) : void **post** getIndirizzi.size()= @pre.getIndirizzi.size()+1 |
| Invarianti |  |

## AmministratoreManager



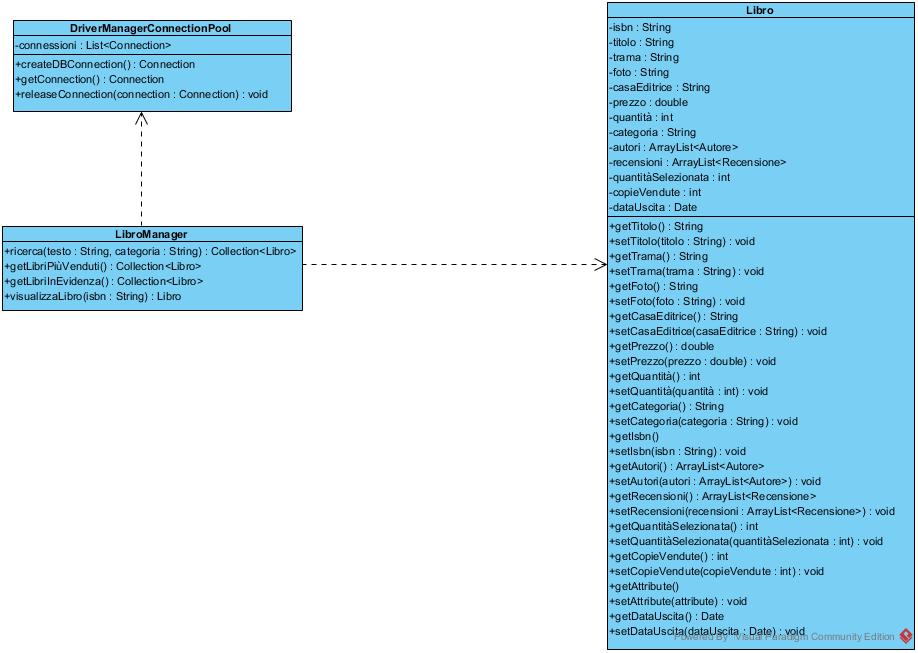
|  |  |
| --- | --- |
| Nome Classe | AmministratoreManager |
| Pre-condizione | **context** AmministratoreManager::aggiungiLibro(libro : Libro) : void **pre**: libro!=null;  **context** AmministratoreManager::modificaTitolo(isbn : String, nuovoAttributo : String) : void **pre:** isbn!=null && nuovoAttributo!= null;  **context** AmministratoreManager::modificaTrama(isbn : String, nuovoAttributo : String) : void **pre:** isbn!=null && nuovoAttributo!= null;  **context** AmministratoreManager::modificaFoto(isbn : String, nuovoAttributo : String) : void **pre:** isbn!=null && nuovoAttributo!= null;  **context** AmministratoreManager::modificaCasaEditrice(isbn : String, nuovoAttributo : String) : void **pre:** isbn!=null && nuovoAttributo!= null;  **context** AmministratoreManager::modificaPrezzo(isbn : String, nuovoAttributo : Double) : void **pre:** isbn!=null && nuovoAttributo!= null && nuovoAttributo>0;  **context** AmministratoreManager::modificaQuatità(isbn : String, nuovoAttributo : Int) : void **pre:** isbn!=null && nuovoAttributo!= null && nuovoAttributo>0;  **context** AmministratoreManager::modificaCategoria(isbn : String, nuovoAttributo : String) : void **pre:** isbn!=null && nuovoAttributo!= null;  **context** AmministratoreManager::modificaQuantitàSelezionata(isbn : String, nuovoAttributo : Int) : void **pre:** isbn!=null && nuovoAttributo!= null && nuovoAttributo>0;  **context** AmministratoreManager::modificaCopieVendute(isbn : String, nuovoAttributo : Int) : void **pre:** isbn!=null && nuovoAttributo!= null && nuovoAttributo>0;  **context** AmministratoreManager::modificaDataUscita(isbn : String, nuovoAttributo : Date) : void **pre:** isbn!=null && nuovoAttributo!= null ;  **context** AmministratoreManager::eliminaLibro(isbn : String) : void **pre** isbn!=null;  **context** AmministratoreManager::eliminaRecensione(idRecensione : int) : void **pre** idRecensione!=null;  **context** AmministratoreManager::ricercaAccount(email : String) : Utente **pre** email!=null;  **context** AmministratoreManager::cambiaTipo(email : String, tipo : String) : void **pre** isbn!=null && tipo!=null;  **context** AmministratoreManager::eliminaUtente(email : String) : void **pre** email!=null; |
| Post-condizione | **context** AmministratoreManager::aggiungiLibro(libro : Libro) : void **post**: libri.size()[=@pre.libri.size+1](mailto:=@pre.libri.size+1)  **context** AmministratoreManager::modificaTitolo(isbn : String, nuovoAttributo : String) : void **post:** libro.titolo=nuovoAttributo;  **context** AmministratoreManager::modificaTrama(isbn : String, nuovoAttributo : String) : void **post:** libro.trama=nuovoAttributo;  **context** AmministratoreManager::modificaFoto(isbn : String, nuovoAttributo : String) : void **post:** libro.foto=nuovoAttributo;  **context** AmministratoreManager::modificaCasaEditrice(isbn : String, nuovoAttributo : String) : void **post:** libro.casaEditrice=nuovoAttributo;  **context** AmministratoreManager::modificaPrezzo(isbn : String, nuovoAttributo : Double) : void **post:** libro.prezzo=nuovoAttributo;  **context** AmministratoreManager::modificaQuatità(isbn : String, nuovoAttributo : Int) : void **post:** libro.quantità=nuovoAttributo;  **context** AmministratoreManager::modificaCategoria(isbn : String, nuovoAttributo : String) : void **post:** libro.categoria=nuovoAttributo;  **context** AmministratoreManager::modificaQuantitàSelezionata(isbn : String, nuovoAttributo : Int) : void **post:** libro.quantitàSelezionata=nuovoAttributo;  **context** AmministratoreManager::modificaCopieVendute(isbn : String, nuovoAttributo : Int) : void **post:** libro.copieVendute=nuovoAttributo;  **context** AmministratoreManager::modificaDataUscita(isbn : String, nuovoAttributo : Date) : void **post:** libro.dataUscita=nuovoAttributo;  **context** AmministratoreManager::eliminaLibro(isbn : String) : void **pre** isbn!=null;  **post:** libri.size()[=@pre.libri.size+1](mailto:=@pre.libri.size+1);  **context** AmministratoreManager::eliminaRecensione(idRecensione : int) : void **post:** libro.getRecensioni().size=@pre.libro.getRecensioni().size()-1;  **context** AmministratoreManager::cambiaTipo(email : String, tipo : String) : void **post:** utente.getTipo=tipo; |
| Invarianti |  |

## AmministratoreOrdiniManager



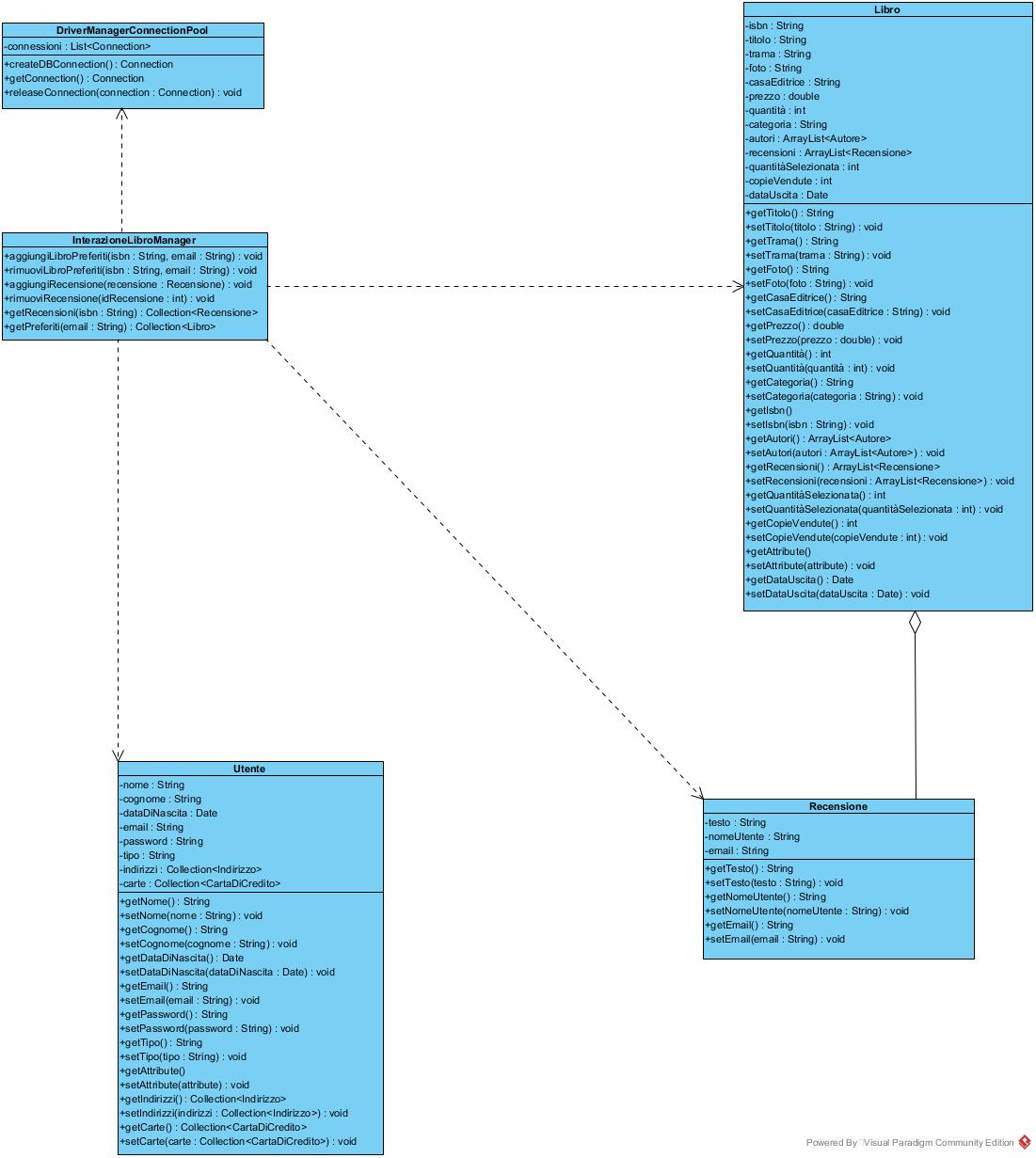
|  |  |
| --- | --- |
| Nome Classe | AmministratoreOrdineManager |
| Pre-condizione | **context** AmministratoreOrdineManager::ricercaOrdine( numOrdine : int) : Ordine **pre** numOrdine!=null;  **context** AmministratoreOrdineManager::cambiaStato(numOrdine : int, stato : String) : void **pre** numOrdine!=null && stato!=null;  **contex** AmministratoreOrdineManager::cambiaDataEOra(numOrdine : int, data : Date, ora : Time) : void **pre** numOrdine!=null && data!=null && ora!=null |
| Post-condizione |  |
| Invarianti |  |

## LibroManager



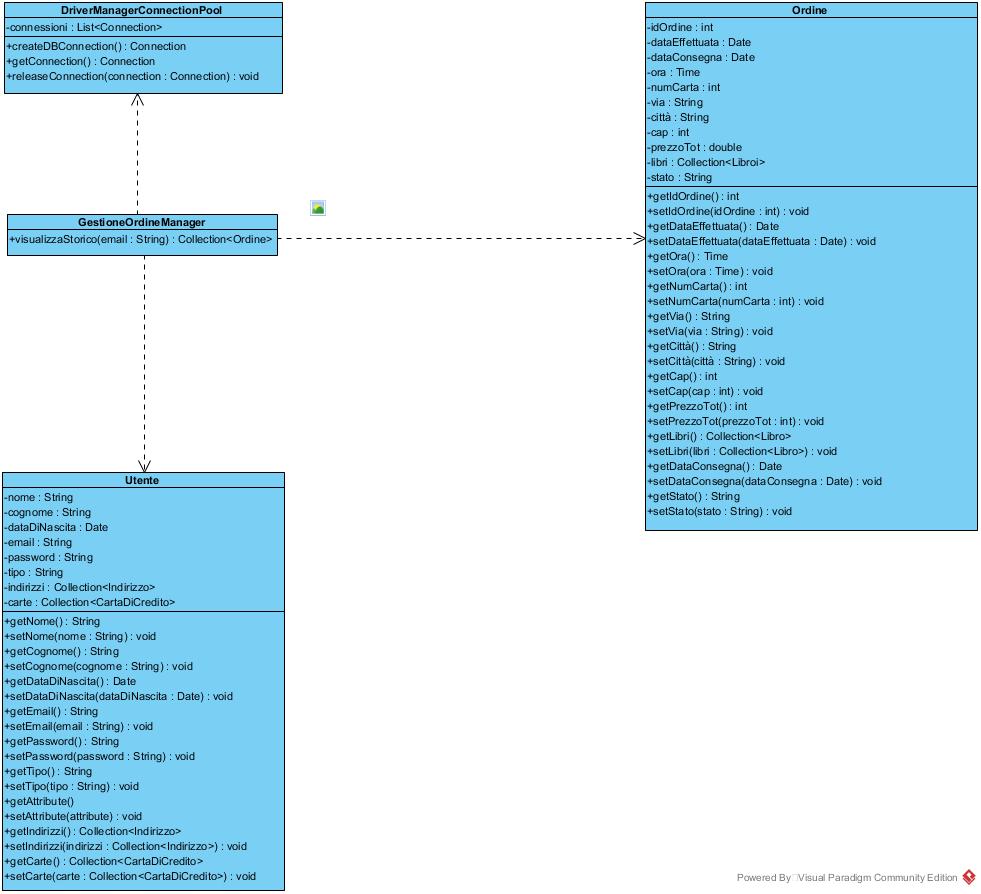
|  |  |
| --- | --- |
| Nome Classe | LibroManager |
| Pre-condizione | **context** LibroManager::ricerca(testo : String, categoria : String) : Collection<Libro> **pre**: testo!=null && categoria!= null;  **context** LibroManager::visualizzaLibro(isbn : String): Libro **pre** : isbn!=null |
| Post-condizione |  |
| Invarianti |  |

## InterazioneLibroManager



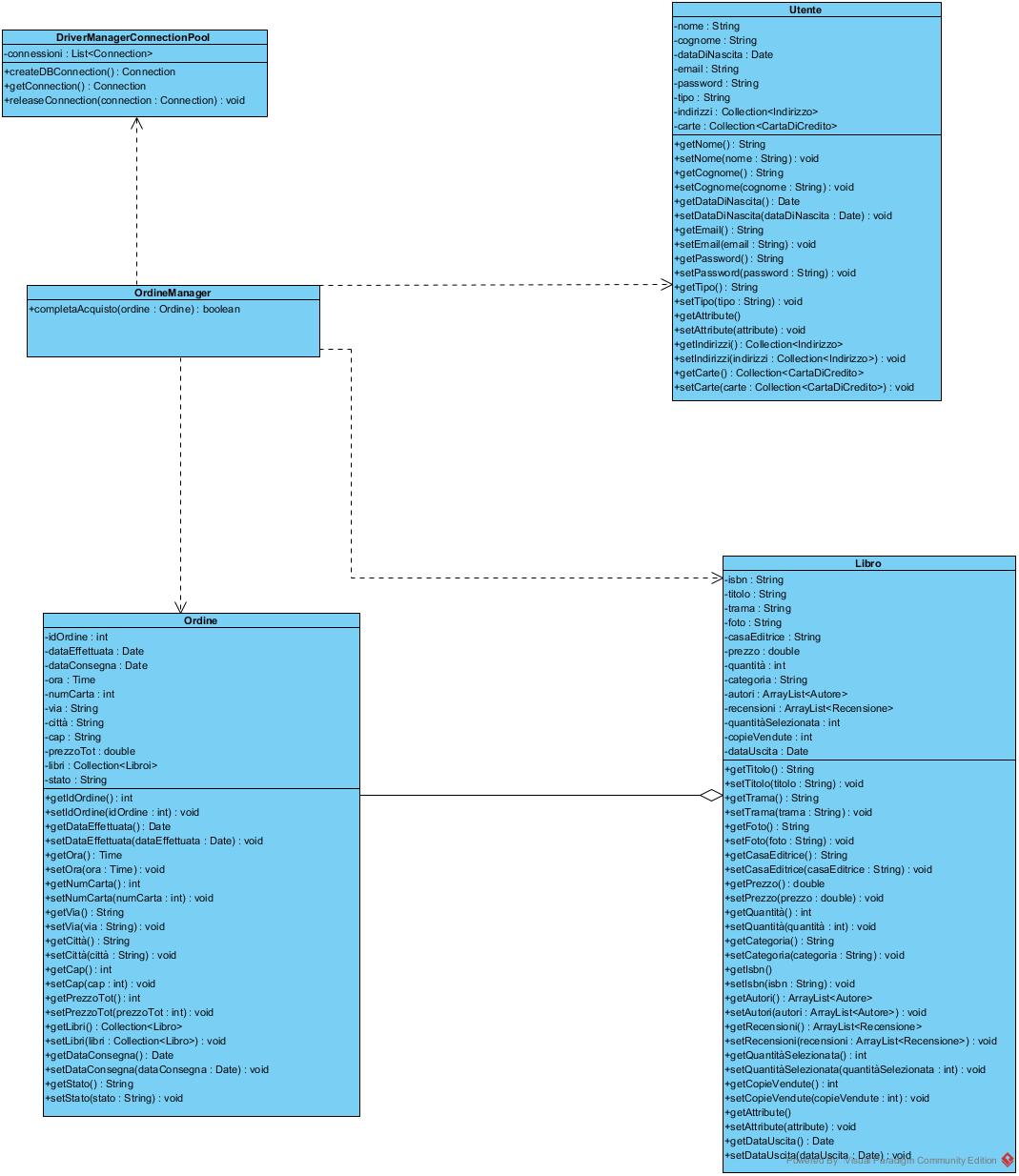
|  |  |
| --- | --- |
| Nome Classe | InterazioneLibroManager |
| Pre-condizione | **context** InterazioneLibroManager::aggiungiLibroPreferiti(isbn : String, email: String): void **pre**: isbn!=null && email!=null;  **context** InterazioneLibroManager::rimuoviLibroPreferiti(isbn : String, email: String) : void **pre**: isbn!=null && email!=null;  **context** InterazioneLibroManager::aggiungiRecensione(recensione : Recensione): void **pre**: recensione!=null;  **context** InterazioneLibroManager::rimuoviRecensione(idRecensione : int) : void **pre**: idRecensione!=null;  **context** InterazioneLibroManager::getRecensioni(isbn : String) Collection<Libro> **pre**: isbn!=null;  **context** InterazioneLibroManager::getPreferiti( email: String) Collection<Libro> **pre**: email!=null; |
| Post-condizione |  |
| Invarianti |  |

## GestioneOrdineManager



|  |  |
| --- | --- |
| Nome Classe | GestioneOrdineManager |
| Pre-condizione | **context** GestioneOrdineManager::visualizzaStorico(email : String): Collection<Libro> **pre**: email!=null |
| Post-condizione |  |
| Invarianti |  |

## OrdineManager



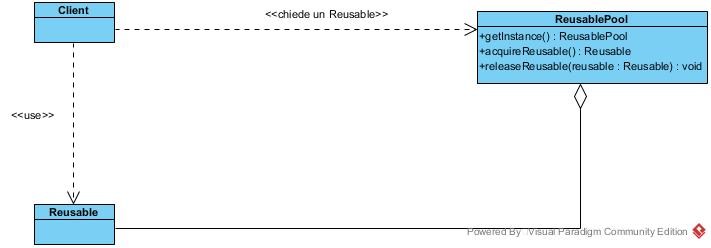
|  |  |
| --- | --- |
| Nome Classe | OrdineManager |
| Pre-condizione | **context** OrdineManager::completaAcquisto(ordine : Ordine): boolean **pre**: ordine!=null;  **context** OrdineManager::aggiungiAlCarrello(carrello : Carrello, isbn : String): Carrello **pre**: carrello!=null && isbn!=null;  **context** OrdineManager::aumentaQuantità(carrello : Carrello, isbn : String): Carrello **pre**: carrello!=null && isbn!=null;  **context** OrdineManager::diminuisciQuantità(carrello : Carrello, isbn : String): Carrello **pre**: carrello!=null && isbn!=null;  **context** OrdineManager::rimuoviDalCarrello(carrello : Carrello, isbn : String): Carrello **pre**: carrello!=null && isbn!=null; |
| Post-condizione | **context** OrdineManager::rimuoviDalCarrello(carrello : Carrello, isbn : String): Carrello **post**: carrello.size()=@pre.carrello.size()-1; |
| Invarianti |  |

# Design Pattern

## Object Pool Pattern

L’Object pool pattern è un design pattern creazionale che usa un insieme di oggetti inizializzati pronti per l’uso mantenuti in una “pool”, che si occupa di allocarli e de-allocherà su richiesta. Il client della pool invierà richiesta a un oggetto nella pool ed eseguirà operazioni sull’oggetto ritornato. Quando il client ha finito, ritorna l’oggetto alla pool che lo de-allocherà.

*Object pool pattern*

****

Utilizzo: L’Object Pool Pattern sarà utilizzato per gestire le connessioni con il database. Più precisamente, un oggetto Model richiederà connessioni al DriverManagerPool che ritornerà un oggetto Connection, l’oggetto Model effettuerà operazioni con l’oggetto connection e successivamente richiederà al DriverManagerPool la de-allocazione.

