Università Politecnica delle Marche – Facoltà di Ingegneria

**INGEGNERIA INFORMATICA E DELL’AUTOMAZIONE**

******

**APPLICAZIONE PER LA GESTIONE DI UN AUTONOLEGGIO**

**RELAZIONE**

ABBADINI LORENZO, COMPAGNONI PAOLO, SCISCI VALERIO

**OOP1617Gruppo08**

Sommario

[**Introduzione** 2](#_Toc491102455)

[Note 2](#_Toc491102456)

[**Analisi del dominio di interesse** 3](#_Toc491102457)

[Tipologie di clienti 3](#_Toc491102458)

[Tipologie di noleggio 3](#_Toc491102459)

[Tipologie di utenti dell’applicazione 4](#_Toc491102460)

[Note 4](#_Toc491102461)

[**Strutturazione dei requisiti** 5](#_Toc491102462)

[Diagramma dei casi d’uso 5](#_Toc491102463)

[Gestione clienti 6](#_Toc491102464)

[Gestione contratti 6](#_Toc491102465)

[Gestione flotta 7](#_Toc491102466)

[Funzionalità extra 7](#_Toc491102467)

[Progettazione del database 8](#_Toc491102468)

[Note 9](#_Toc491102469)

[**Struttura del progetto** 10](#_Toc491102470)

[Struttura della cartella “src” 10](#_Toc491102471)

[Diagramma delle classi 11](#_Toc491102472)

[**Implementazione** 12](#_Toc491102473)

[Autonoleggio (package) 12](#_Toc491102474)

[Autonoleggio (Classe) 12](#_Toc491102475)

[Login 13](#_Toc491102476)

[Db 14](#_Toc491102477)

[Bibliografia 15](#_Toc491102478)

# **Introduzione**

Il progetto è stato sviluppato per informatizzare la gestione dell’autonoleggio RentForYou situato a Giulianova (TE).

L’applicazione offre all’utente le seguenti funzionalità:

* calcolo del profitto mensile/annuale
* creazione di statistiche
* gestione dei clienti
* gestione dei contratti di noleggio
* gestione del parco veicoli
* monitoraggio delle scadenze

Per un corretto utilizzo dell’applicazione si raccomanda di consultare il *Manuale di Prima Esecuzione* ed il *Manuale di Utilizzo*.

## Note

Le figure contenute nella presente relazione sono riportate nella cartella **indicare path cartella** per una migliore leggibilità.

# **Analisi del dominio di interesse**

## Tipologie di clienti

* Associazioni
* Aziende
* Privati

Le **associazioni** di qualsiasi genere (sportive, culturali, ecc.) hanno diritto ad uno sconto del 20% sul costo totale del noleggio.

Le **aziende** hanno diritto ad uno sconto del 10% sul costo totale, nel caso di noleggio a lungo termine.

## Tipologie di noleggio

* Noleggio a breve termine
* Noleggio a lungo termine

Il **noleggio a breve termine**, riguarda tutti quei casi in cui l’auto viene noleggiata temporaneamente, da un giorno fino anche a sei mesi. Il noleggio a breve termine prevede tariffe che variano in base al periodo, e sono più basse quanto più si allunga il periodo di noleggio. Questa formula è una soluzione molto utilizzata da privati ed aziende che necessitano di un’auto per spostamenti brevi (da uno a più giorni), per sostituire veicoli in assistenza, o per specifiche esigenze aziendali (dotare di auto un collaboratore temporaneo, ad esempio).

Il **noleggio a lungo termine** è una formula di abbonamento mensile che permette di utilizzare un’auto senza acquistarla e senza spendere soldi per la sua manutenzione. Il noleggio a lungo termine prevede un contratto che va dai 12 ai 36 mesi, con rate annuali o mensili fisse che variano in funzione del veicolo e dei chilometri che si prevedono di effettuare. Le rate mensili vanno da un minimo di 200€ ad un massimo di 500€. Per il lungo termine le tariffe vanno calcolate in funzione della quota mensile.

## Tipologie di utenti dell’applicazione

* Amministratore
* Utente standard

L’**utente standard** ha accesso solo alla gestione dei clienti e dei contratti di noleggio.

L'**amministratore** ha accesso completo a tutte le funzionalità del programma.

## Note

Per informazioni più dettagliate circa le tipologie di veicoli, le tariffe e le dinamiche interne dell’attività si consiglia di consultare le specifiche di progetto fornite con l’applicazione.

# **Strutturazione dei requisiti**

In seguito allo studio del dominio di interesse, abbiamo strutturato i requisiti dell’applicazione utilizzando il linguaggio UML.

## Diagramma dei casi d’uso

Seguendo le indicazioni del testo di riferimento sul linguaggio UML (Fowler, 2010), abbiamo cercato di rendere il tutto più chiaro trattando separatamente i diversi casi d’uso.

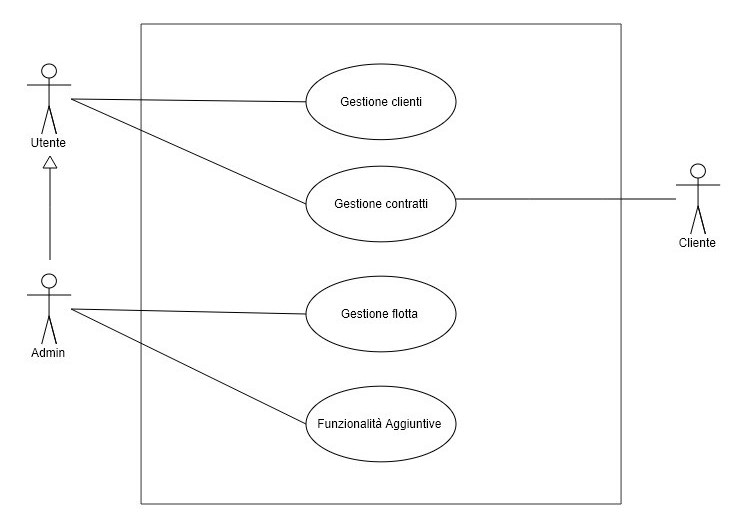
Gli attori del diagramma dei casi d’uso sono:

* Amministratore
* Cliente
* Utente standard

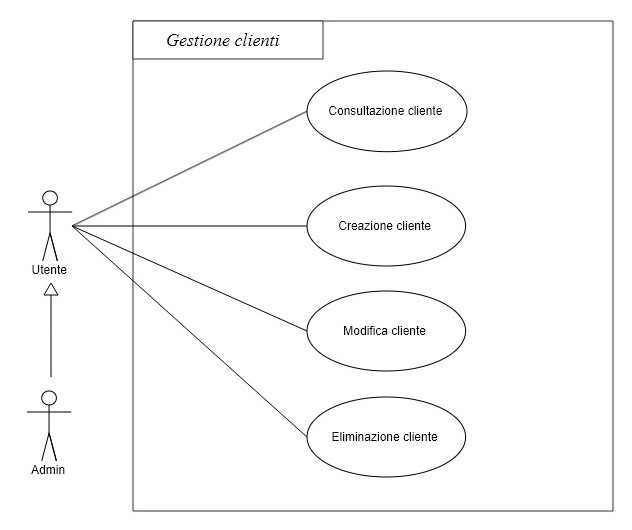
L’amministratore è il titolare dell’attività. I casi d’uso ad esso associati rappresentano le principali funzionalità fornite dall’applicazione.

L’utente standard è l’impiegato dell’attività. I casi d’uso ad esso associati sono un sottoinsieme di quelli associati all’amministratore.

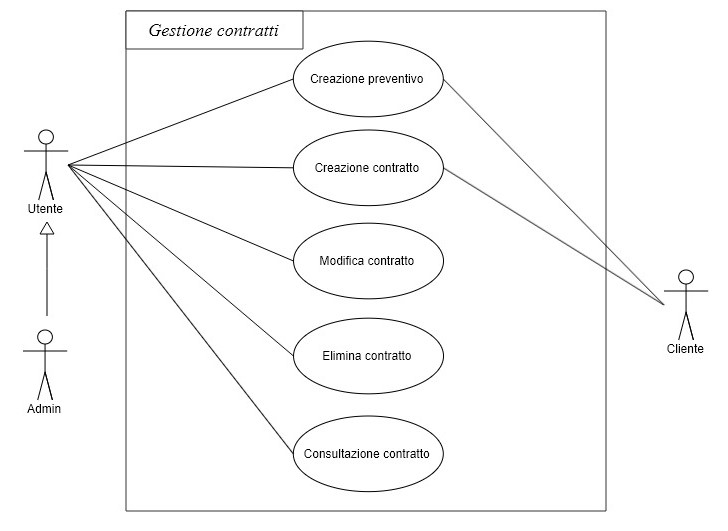
Al cliente è associato un solo caso d’uso. Come evidenziato nella sezione Gestione contratti, esso può soltanto richiedere un preventivo o stipulare un contratto di noleggio.



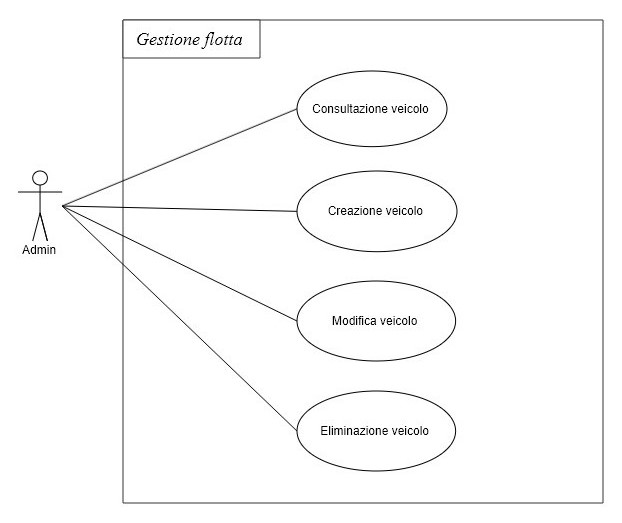
### Gestione clienti



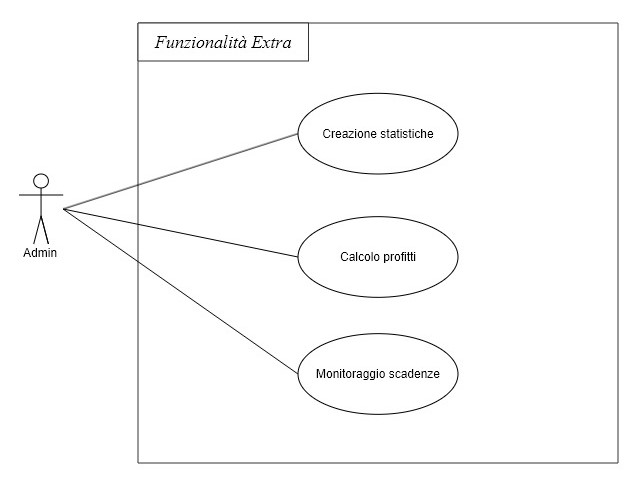
### Gestione contratti



### Gestione flotta



### Funzionalità extra



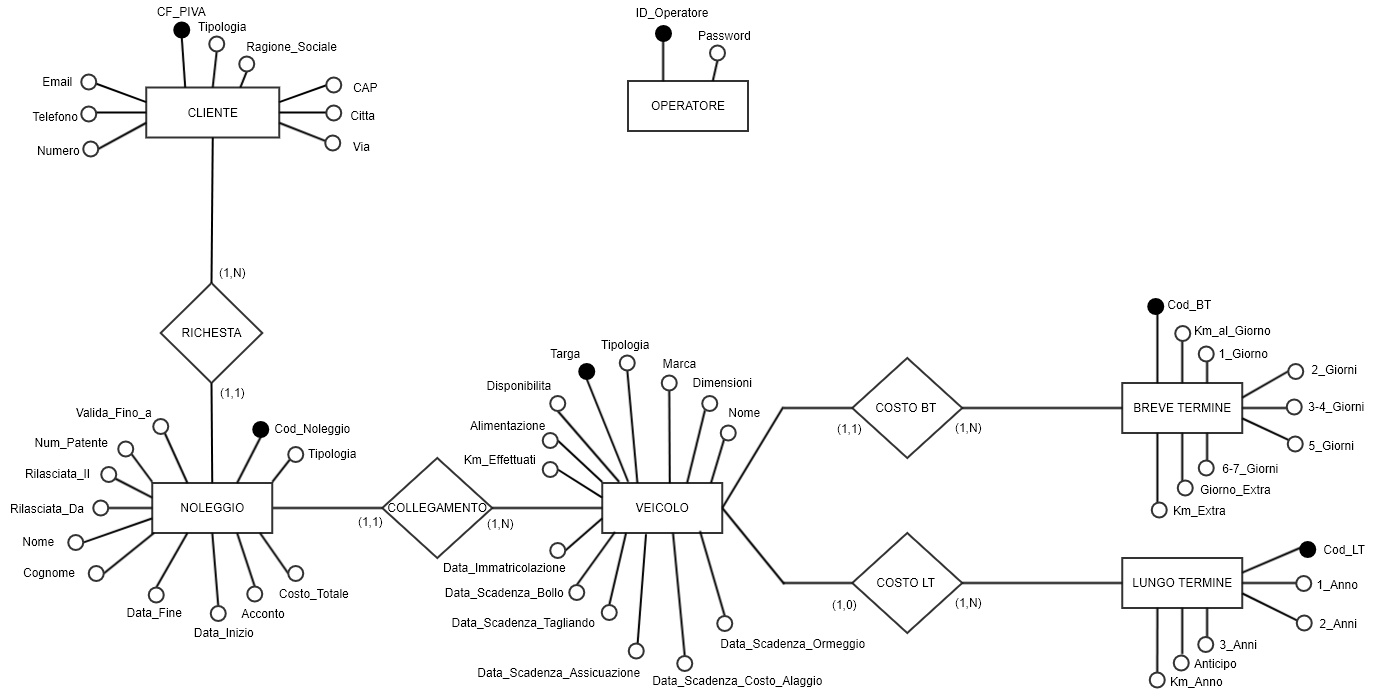
## Progettazione del database

Come già visto nel Manuale di Prima Esecuzione e nel Manuale di Utilizzo, per il corretto funzionamento dell’applicazione è necessaria la presenza di un database. In particolare si fa uso dell’applicazione multipiattaforma XAMPP contenente il DBMS MySQL.

In fase di progettazione è stata adottata la strategia top-down. Il lavoro ha seguito le seguenti fasi:

* analisi del dominio di interesse ed individuazione delle entità principali
* costruzione dello scheletro dello schema concettuale
* specializzazione dello schema concettuale
* ristrutturazione dello schema concettuale

Lo schema E-R ottenuto al termine della progettazione è il seguente.



L’entità Operatore rappresenta gli utenti che utilizzeranno il software. Ciascun utente è caratterizzato da un username (ID\_Operatore) e da una password.

L’entità Cliente rappresenta i clienti dell’autonoleggio; per ciascuno di essi vengono memorizzati tutti i dati anagrafici.

L’entità Noleggio rappresenta i contratti di noleggio; per ciascuno di essi vengono memorizzati anche i dati anagrafici del conducente, che in generale può essere diverso dal cliente che ha stipulato il contratto.

L’entità Veicolo rappresenta i mezzi posseduti dall’autonoleggio; per ciascuno di essi vengono memorizzati tutti i dati di interesse. Il campo Disponibilità indica se un veicolo è impegnato in operazioni di manutenzione.

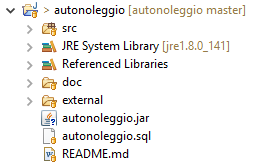
L’entità Breve Termine rappresenta il tariffario per i noleggi a breve termine.

L’entità Lungo Termine rappresenta il tariffario per i noleggi a lungo termine.

### Note

Un preventivo può essere scartato oppure trasformato in un contratto di noleggio a tutti gli effetti. Questo non rende necessario tenere uno storico dei preventivi calcolati; per questo motivo, nello schema E-R non è presente una entità Preventivo.

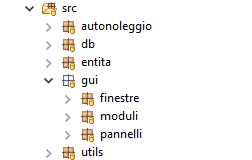
# **Struttura del progetto**



La cartella principale del progetto contiene tre sottocartelle:

* doc contiene il javaDoc
* external contiene le librerie importate all’interno del programma in fase di configurazione
  + car per l’icona dell’applicazione
  + jcalendar per la gestione delle date all’interno dell’applicazione
  + mysqlconnector per la connettività al database
  + seaglasslookandfeel per il look and feel dell’applicazione
* source contiene il codice sorgente dell’applicazione

## Struttura della cartella “src”



La cartella src contiene tutto il codice sorgente organizzato in package:

* main contiene la classe necessaria all’avvio dell’applicazione e la classe per il login
* db contiene la classe necessaria per effettuare la connessione al database
* entita contiene le classi che mappano le entità del database
* gui contiene le classi che implementano l’interfaccia grafica dell’applicazione ed è organizzato nei seguenti sottopackage:
  + finestre
  + moduli
  + pannelli
* utils contiene classi che implementano funzionalità necessarie per l’applicazione

## Diagramma delle classi

Il diagramma delle classi mette in relazione le classi dell’applicazione. Per la sua creazione sono stati adottati due criteri:

* sono state riportate tutte le classi dell’applicazione per rendere la comprensione della parte implementativa più semplice.
* si è scelto di suddividere il diagramma delle classi in più parti in modo tale da facilitarne la lettura.

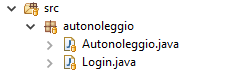
**Mettere immagini class diagram!!**

# **Implementazione**

Si è accennato nel capitolo relativo alla struttura del progetto, all’organizzazione dei package. In questo capitolo si andrà più a fondo nell’analisi, specificando le classi contenute in ogni package.

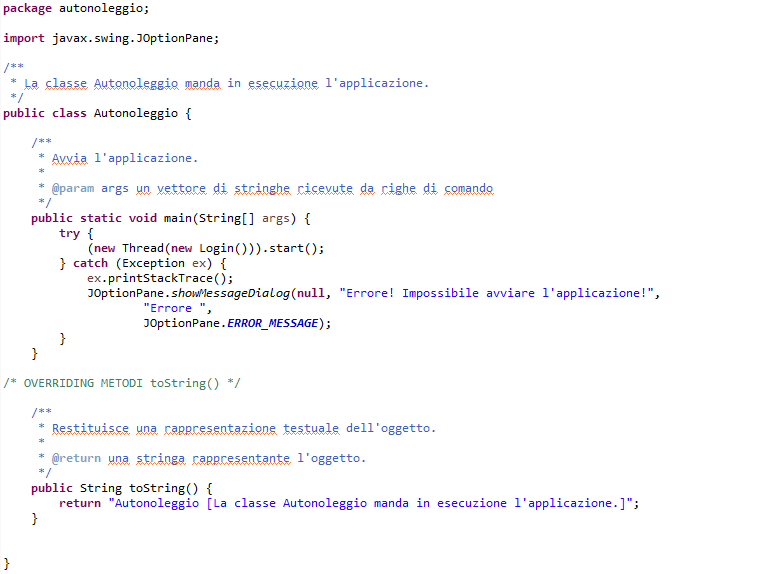
## Autonoleggio (package)

Il package autonoleggio si occupa di far partire l’applicazione e di verificare le credenziali di accesso



### Autonoleggio (Classe)

La classe Autonoleggio si occupa di far partire l’applicazione in quanto contiene l’implementazione del metodo main.



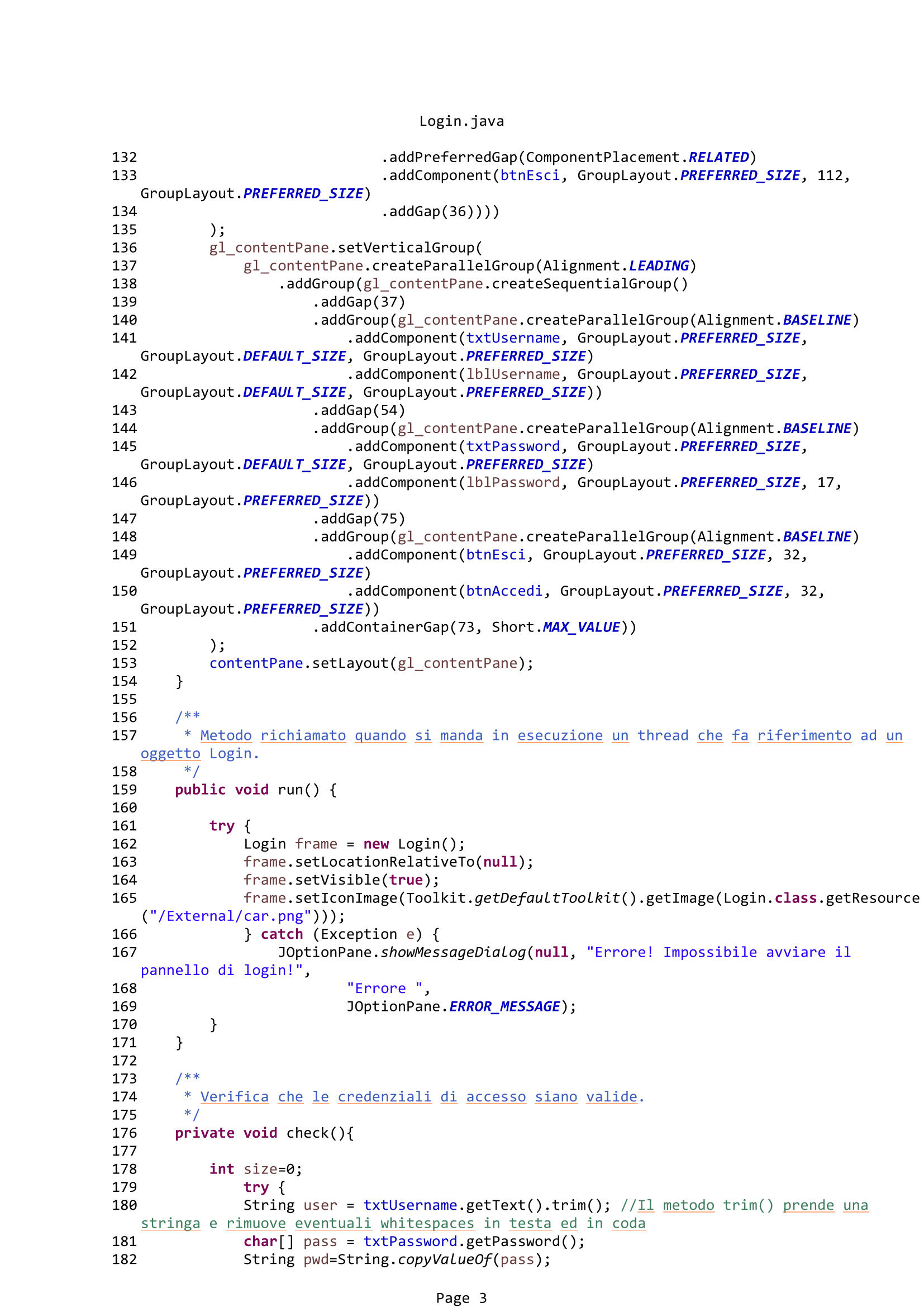
### Login

La classe Login si occupa di controllare le credenziali di accesso.

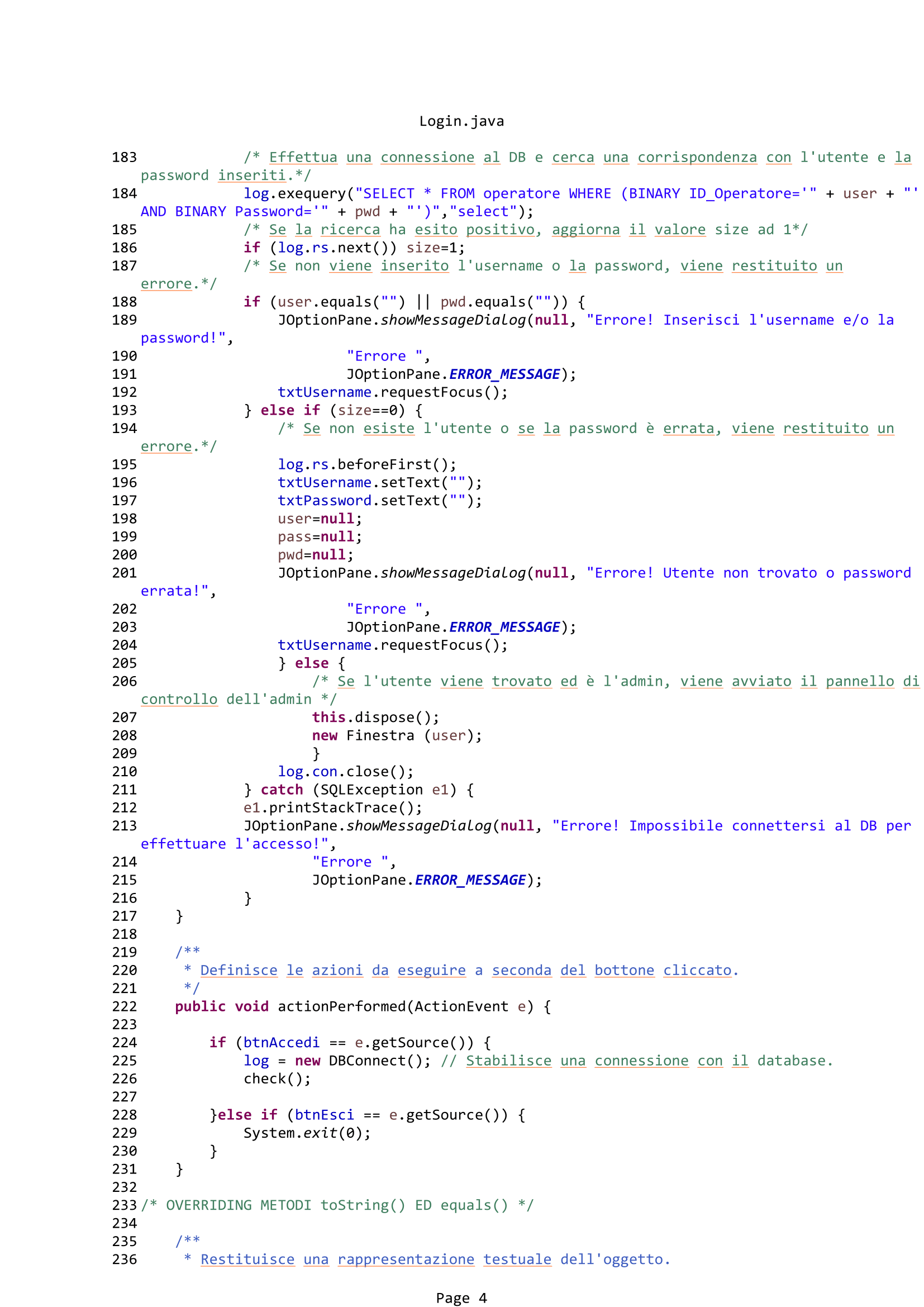
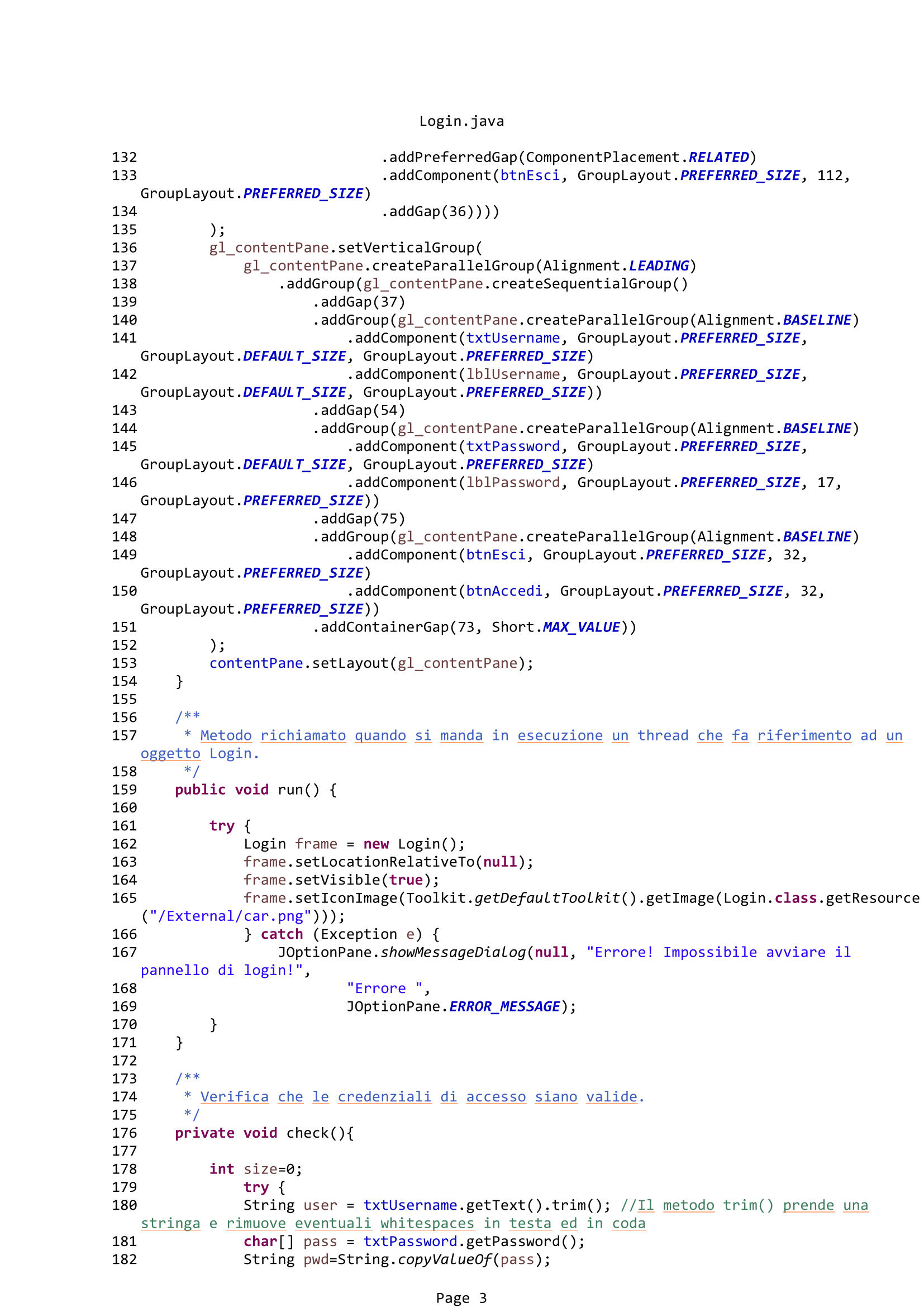
Nel metodo main va analizzata la seguente riga di codice :

(**new** Thread (**new** Login())).start();

Il primo new da sinistra crea un oggetto a partire dall’ interfaccia Thread. Un’interfaccia è simile ad una classe astratta che dichiara solo metodi astratti senza fornirne un’implementazione: è compito di un’altra classe implementare tutti i metodi astratti. L’interfaccia Thread viene implementata richiamando il metodo start (si occupa di richiamare il metodo run) dell’oggetto Login passato come parametro. Infatti la classe Login contiene l’implementazione del metodo run come riportato di seguito.

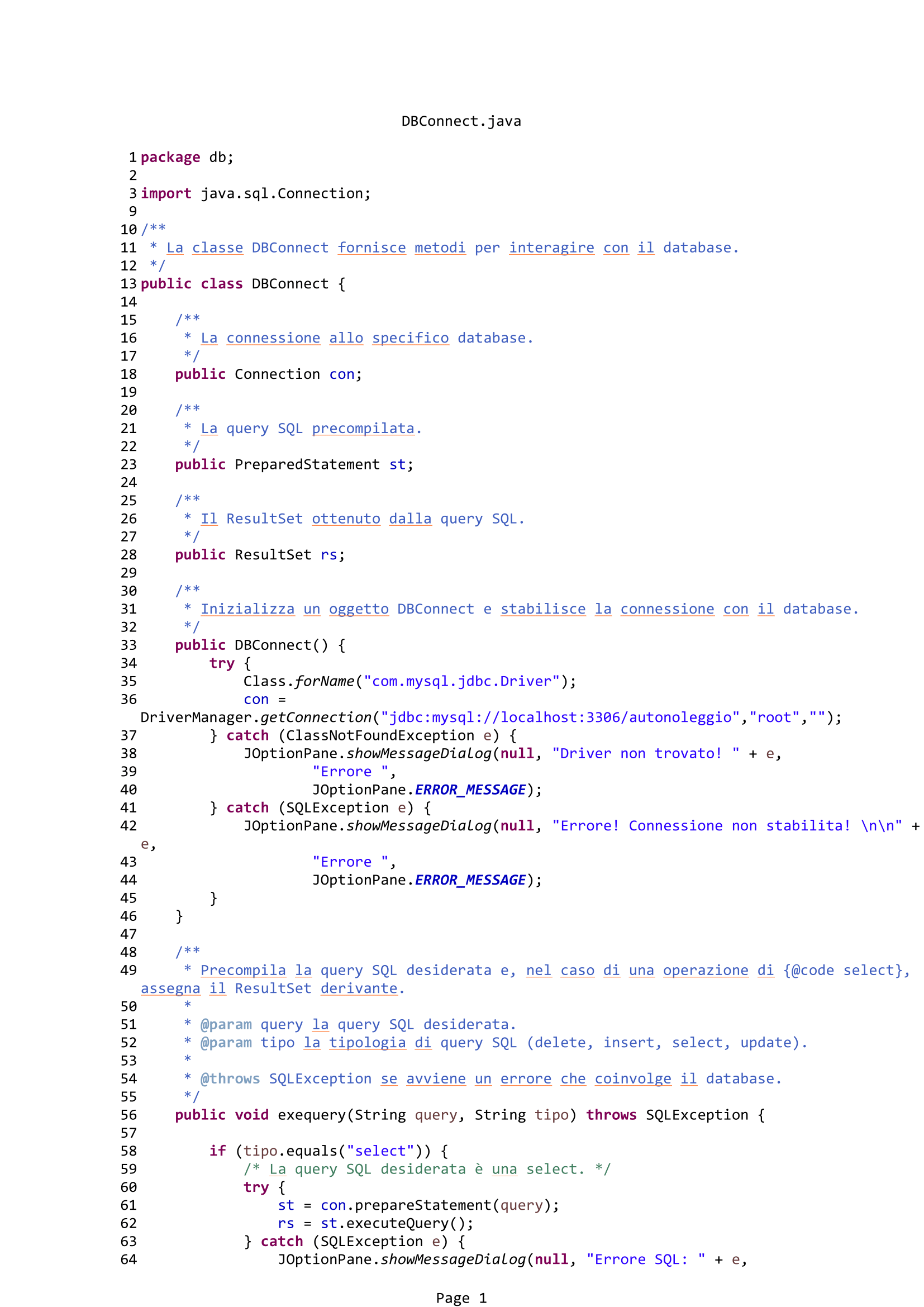


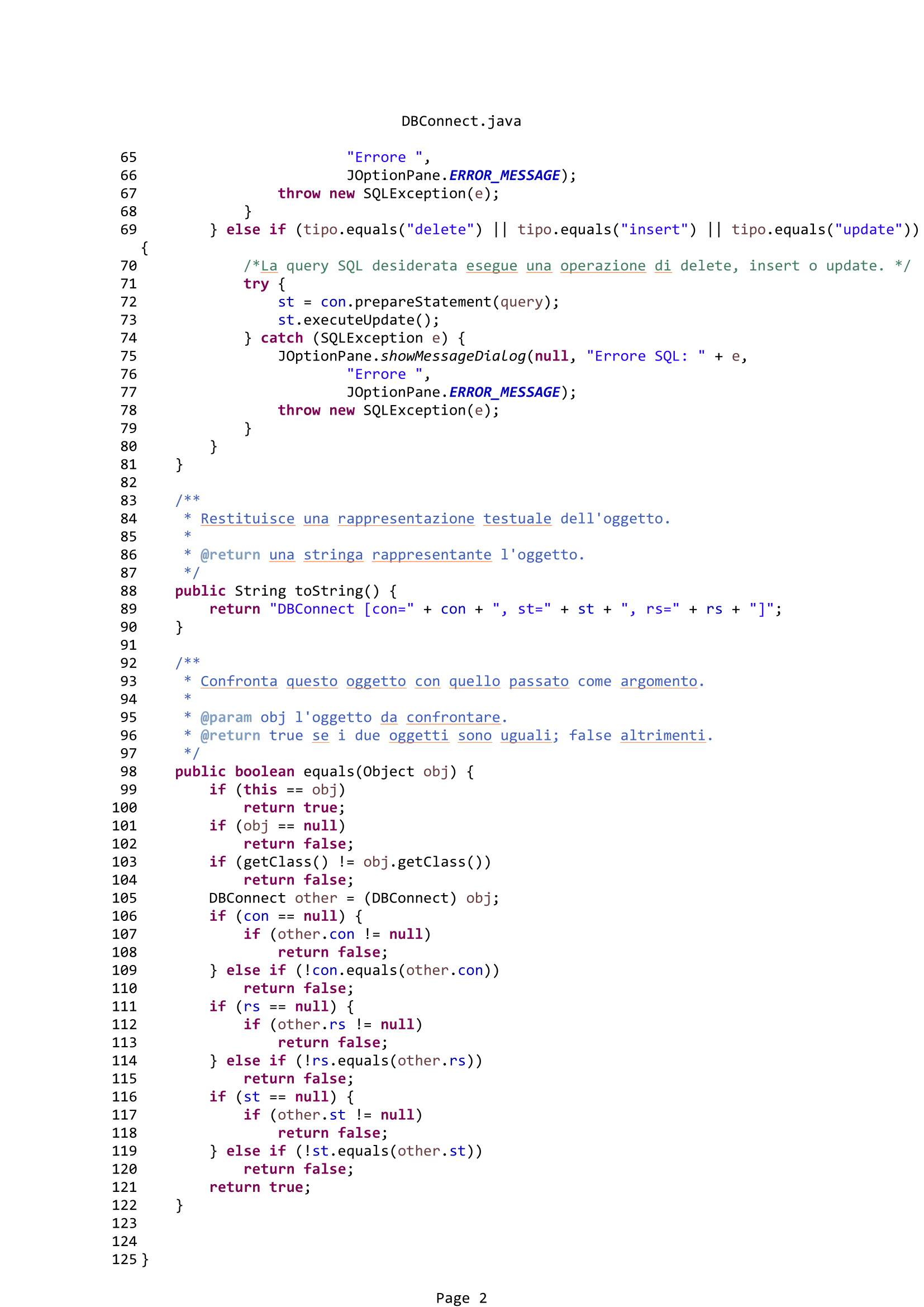
Il metodo check è il metodo che verifica le credenziali di accesso.



## Db

Il package db contiene la classe DBConnect che permette la connessione al database. Al suo interno esiste un metodo chiamato exequery che prende come parametri una stringa che rappresenta la query da analizzare e un’altra stringa che rappresenta il tipo di query. Se l’operazione richiesta è una select, viene creata la connessione compilando la query passata come argomento. Successivamente viene creato il resultSet. Altrimenti, per tutte le altre operazioni viene sempre creata la connessione allo stesso modo di prima ma successivamente viene eseguita la query senza creare un resultSet.





# Bibliografia

Fowler, M. (2010). *UML Distilled. Guida rapida al linguaggio di modellazione standard.* Pearson.