ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εγχειρίδιο Εκμάθησης Java Database Connectivity (JDBC) και υλοποίηση ενός console application

Τσιαπάρας Ζαχαρίας

Εισηγητής: Αναστασία Βελώνη, Καθηγήτρια Εφαρμογών

ΑΘΗΝΑ
ΔΕΚΕΜΒΡΙΟΣ 2018

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εγχειρίδιο Εκμάθησης Java Database Connectivity (JDBC) και υλοποίηση ενός console application

Ζαχαρίας Τσιαπάρας
Α.Μ. ais0123

Εισηγητής:

Αναστασία Βελώνη, Καθηγήτρια Εφαρμογών

Εξεταστική Επιτροπή:

…….

…….

…….

Ημερομηνία εξέτασης 5/12/2018

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο/Η κάτωθι υπογεγραμμένος/η ……ΤΣΙΑΠΑΡΑΣ ΖΑΧΑΡΙΑΣ……, του ……ΔΗΜΗΤΡΙΟΥ……, με αριθμό μητρώου ……ais0123…….. φοιτητής/τρια του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα πτυχιακή εργασία ολοκληρώθηκε μετά από επίμονες προσπάθειες, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό της σύνδεσης ενός προγράμματος με μια βάση δεδομένων. Την προσπάθειά μου αυτή υποστήριξε η επιβλέπων καθηγήτρια μου, Κα. Αναστασία Βελώνη, την οποία θα ήθελα να ευχαριστήσω πάρα πολύ.

Ακόμα θα ήθελα να ευχαριστήσω την Κα. Χριστίνα Ισαράι για πολύτιμη υποστήριξη καθ' όλη την διάρκεια διεξαγωγής της πτυχιακής εργασίας και την οικογένειά μου για την ολόπλευρη ηθική και υλική υποστήριξη που παρείχαν κατά την διάρκεια των σπουδών μου.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία αφορά την μελέτη της τεχνολογίας JDBC (Java Data base Connectivity) και της εφαρμογής της στην δημιουργία μιας εφαρμογής ανταλλαγής μηνυμάτων. Η εφαρμογή περιλαμβάνει την ανάγνωση, διαγραφή και τροποποίηση των μηνυμάτων όπου αποθηκεύονται σε μια βάση δεδομένων MySQL.

Η κατασκευή της εφαρμογής επιτεύχθηκε με τη χρήση του ενοποιημένου περιβάλλοντος ανάπτυξης Eclipse IDE και η βάση δεδομένων όπου αποτελεί βασικό κομμάτι της εφαρμογής υλοποιήθηκε μέσω των ενεργειών που παρέχει η τεχνολογία JDBC.

ABSTRACT

The present thesis concerns the study of JDBC (Java Data Base Conectivity) technology and its use in the creation of a messaging application. That application includes reading, deleting and modifying the messages which are stored in a MySQL database.

The build of the application was achieved using the Eclipse IDE integrated development environment and the database which is the main part of the application. Furthermore it was implemented through the actions provided by the JDBC technology.

.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Περιγραφή του αντικειμένου της πτυχιακής εργασίας

Η παρούσα πτυχιακή εργασία αφορά την μελέτη της τεχνολογίας JDBC (Java Data base Conectivity) και της εφαρμογής της στην δημιουργία μιας εφαρμογής ανταλλαγής μηνυμάτων. Η εφαρμογή περιλαμβάνει την ανάγνωση, διαγραφή και τροποποίηση των μηνυμάτων όπου αποθηκεύονται σε μια βάση δεδομένων MySQL.

1.2 Ιστορική αναδρομή

Η Sun Microsystems κυκλοφόρησε την JDBC ως μέρος του JDK 1.1 στις 19 Φεβρουαρίου 1997. Από τότε είναι μέρος της Java Standard Edition. Οι JDBC κλάσεις περιέχονται στα πακέτα java.sql και javax.sql Java.

Ξεκινώντας με την έκδοση 3.1 η JDBC έχει αναπτυχθεί στο πλαίσιο της Java. Το JSR 114 καθορίζει τις προσθήκες των JDBC rowset και το JSR 221 αποτελεί τον προσδιορισμό του JDBC 4.0 (συμπεριλαμβανομένου της Java SE 6).

Τελευταία έκδοση, JDBC4.2, περιλαμβάνετε στην Java SE 8. [8]

1.3 Περιγραφή του JDBC

Το JDBC αντιπροσωπεύει το Java Data base Conectivity, το οποίο είναι ένα πρότυπο της Java API για συνδεσιμότητα ανεξάρτητη από τη βάση δεδομένων μεταξύ της γλώσσας προγραμματισμού Java και ενός ευρέος φάσματος βάσεων δεδομένων.[6]

Η βιβλιοθήκη JDBC περιλαμβάνει API για κάθε μια από τις παρακάτω εργασίες που σχετίζονται συνήθως με τη χρήση βάσεων δεδομένων.

- Δημιουργία σύνδεσης σε βάση δεδομένων.
- Δημιουργία δηλώσεων SQL ή MySQL.
- Εκτέλεση ερωτημάτων SQL ή MySQL στη βάση δεδομένων.

- Προβολή & Τροποποίηση των εγγραφών που προκύπτουν.

Βασικά, το JDBC είναι μια προδιαγραφή που παρέχει ένα πλήρες σύνολο διεπαφών που επιτρέπει τη φορητή πρόσβαση σε μια υποκείμενη βάση δεδομένων. Η Java μπορεί να χρησιμοποιηθεί για την εγγραφή διαφορετικών τύπων εκτελέσιμων αρχείων, όπως :

- Εφαρμογές Java
- Java Applets
- Java servlets
- Java ServerPages (JSPs)
- Enterprice JavaBeans (EJB).

Όλα αυτά τα διαφορετικά εκτελέσιμα μπορούν να χρησιμοποιήσουν ένα πρόγραμμα οδήγησης JDBC για να αποκτήσουν πρόσβαση σε μια βάση δεδομένων και να επωφεληθούν από τα αποθηκευμένα δεδομένα.

Το JDBC παρέχει τις ίδιες δυνατότητες με το ODBC, επιτρέποντας στα προγράμματα Java να περιέχουν κώδικα ανεξάρτητο από τη βάση δεδομένων.[6]

1.4 Αρχιτεκτονική JDBC

Το API JDBC υποστηρίζει μοντέλα επεξεργασίας δύο και τριών επιπέδων για πρόσβαση σε βάσεις δεδομένων, αλλά γενικά η JDBC Αρχιτεκτονική αποτελείται από δύο επίπεδα :

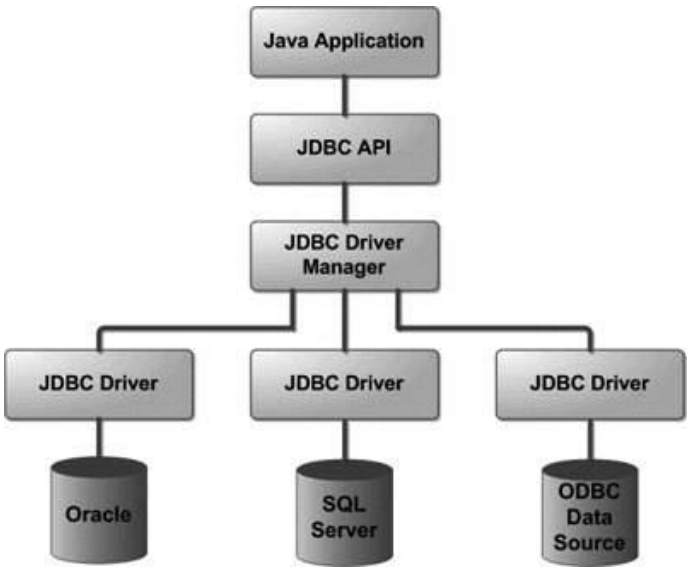
- JDBC API: Αυτό παρέχει τη σύνδεση εφαρμογής JDBC Manager.
- JDBC Driver API: Αυτό υποστηρίζει τη σύνδεση JDBC Manager-to-Driver.

Το JDBC API χρησιμοποιεί ένα πρόγραμμα οδήγησης και συγκεκριμένα προγράμματα οδήγησης για να παρέχει διαφανή συνδεσιμότητα σε ετερογενείς βάσεις δεδομένων.

Ο JDBC driver manager διασφαλίζει ότι χρησιμοποιείται το σωστό πρόγραμμα οδήγησης για την πρόσβαση σε κάθε πηγή δεδομένων. Ο διαχειριστής του

προγράμματος οδήγησης είναι ικανός να υποστηρίζει πολλαπλούς ταυτόχρονους οδηγούς συνδεδεμένους σε πολλές ετερογενείς βάσεις δεδομένων.[3]

Ακολουθεί το αρχιτεκτονικό διάγραμμα, το οποίο δείχνει τη θέση του διαχειριστή του προγράμματος οδήγησης σε σχέση με τα προγράμματα οδήγησης JDBC και την εφαρμογή Java :



Εικόνα 1.1: Αρχιτεκτονικό διάγραμμα JDBC [7]

1.5 Αρχιτεκτονική Κοινά στοιχεία JDBC

Το JDBC API παρέχει τις ακόλουθες διασυνδέσεις και κλάσεις :

- DriverManager: Αυτή η κλάση διαχειρίζεται μια λίστα με προγράμματα οδήγησης βάσεων δεδομένων. Αντιστοιχεί σε αιτήματα σύνδεσης από την εφαρμογή java με το κατάλληλο πρόγραμμα οδήγησης βάσης δεδομένων χρησιμοποιώντας υπο-πρωτόκολλο επικοινωνίας. Το πρώτο πρόγραμμα οδήγησης που αναγνωρίζει ένα συγκεκριμένο υποπρόγραμμα κάτω από το JDBC θα χρησιμοποιηθεί για τη δημιουργία σύνδεσης βάσης δεδομένων.

- Driver: Αυτή η διεπαφή χειρίζεται τις επικοινωνίες με το διακομιστή βάσης δεδομένων. Πολύ σπάνια θα επικοινωνείτε άμεσα με τα αντικείμενα του προγράμματος οδήγησης. Αντίθετα, χρησιμοποιείτε αντικείμενα τύπου DriverManager, τα οποία διαχειρίζονται αυτού του τύπου αντικείμενα. Απεικονίζει

επίσης τις λεπτομέρειες που σχετίζονται με την εργασία με τα αντικείμενα του προγράμματος οδήγησης.[1]

- Connection: Αυτή η διεπαφή περιέχει όλες τις μεθόδους επικοινωνίας με μια βάση δεδομένων. Το αντικείμενο σύνδεσης αντιπροσωπεύει το πλαίσιο επικοινωνίας, δηλαδή, όλη την επικοινωνία με τη βάση δεδομένων μέσω του αντικειμένου σύνδεσης.

- Statement: Χρησιμοποιείτε αντικείμενα που δημιουργούνται από αυτήν τη διεπαφή για να υποβάλλετε τις δηλώσεις SQL στη βάση δεδομένων. Ορισμένες διεπαφές δέχονται παραμέτρους εκτός από την εκτέλεση αποθηκευμένων διαδικασιών.

- ResultSet: Αυτά τα αντικείμενα κατέχουν τα δεδομένα που ανακτώνται από μια βάση δεδομένων μετά την εκτέλεση ενός ερωτήματος SQL. Λειτουργεί ως iterator για να σας επιτρέψει να μετακινήσετε στην ανάκτηση των δεδομένων του.

- SQLException: Αυτή η κλάση χειρίζεται τυχόν σφάλματα που συμβαίνουν σε μια εφαρμογή βάσης δεδομένων.

1.6 Τα πακέτα JDBC 4.0

Τα java.sql και javax.sql είναι τα πρωταρχικά πακέτα για το JDBC 4.0. Αυτή είναι η τελευταία έκδοση του JDBC τη στιγμή που γράψατε αυτή η πτυχιακή. Προσφέρει τις κύριες κλάσεις για αλληλεπίδραση με τις πηγές δεδομένων.Τα νέα χαρακτηριστικά σε αυτά τα πακέτα περιλαμβάνουν αλλαγές στις ακόλουθες περιοχές :

- Αυτόματη φόρτωση οδηγού βάσης δεδομένων.
- Εξαγωγή βελτιώσεων χειρισμού.
- Ενισχυμένη λειτουργικότητα BLOB / CLOB.
- Connection και διεπαφές statement.
- Υποστήριξη εθνικών χαρακτήρων.
- SQL ROWID πρόσβαση.
- Υποστήριξη τύπου δεδομένων SQL 2003 XML.
- Annotations. [4]

ΚΕΦΑΛΑΙΟ 2

Σύνταξη JDBC – SQL

2.1 Περιγραφή Structured Query Language (SQL)

Structured Query Language (SQL) είναι μια τυποποιημένη γλώσσα που σας επιτρέπει να εκτελέσετε λειτουργίες σε μια βάση δεδομένων, όπως η δημιουργία καταχωρήσεων, διαβάζοντας το περιεχόμενο, την ενημέρωση του περιεχομένου και διαγραφή εγγραφών.

Η SQL υποστηρίζεται από σχεδόν οποιαδήποτε βάση δεδομένων που πιθανόν να χρησιμοποιείτε και σας επιτρέπει να γράφετε κώδικα για τη βάση δεδομένων ανεξάρτητα από την υποκείμενη βάση δεδομένων.

Αυτό το κεφάλαιο παρέχει μια επισκόπηση της SQL, η οποία αποτελεί προϋπόθεση για την κατανόηση των εννοιών του JDBC. Μετά από αυτό το κεφάλαιο, θα είστε σε θέση να δημιουργήσετε, να διαβάσετε, να ενημερώσετε και να διαγράψετε (Create, Read, Update και Delete) ,που συχνά αναφέρεται ως CRUD δεδομένα, από μια βάση δεδομένων.[6]

2.2 Δημιουργία βάσης δεδομένων

Η εντολή CREATE DATABASE χρησιμοποιείται για τη δημιουργία μιας νέας βάσης δεδομένων. Η σύνταξη είναι : [2]

```
SQL> CREATE DATABASE DATABASE_NAME;
```

Παράδειγμα

Η ακόλουθη πρόταση SQL δημιουργεί μια βάση δεδομένων που ονομάζεται EMP:[7]

```
SQL> CREATE DATABASE EMP;
```

2.3 Drop Database

Η δήλωση DROP DATABASE χρησιμοποιείται για τη διαγραφή μιας υπάρχουσας βάσης δεδομένων. Η σύνταξη είναι : [2]

```
SQL> DROP DATABASE DATABASE_NAME;
```

Σημείωση: Για να δημιουργήσετε ή να αποθέσετε μια βάση δεδομένων, πρέπει να έχετε δικαιώματα διαχειριστή στον διακομιστή βάσης δεδομένων. Προσέξτε, η διαγραφή μιας βάσης δεδομένων θα έχει ως αποτέλεσμα την απώλεια όλων των δεδομένων που είναι αποθηκευμένα στη βάση δεδομένων.

2.4 Δημιουργία πίνακα

Η εντολή CREATE TABLE χρησιμοποιείται για τη δημιουργία ενός νέου πίνακα. Η σύνταξη της είναι : [2]

```
SQL> CREATE TABLE table_name
(
   column_name column_data_type,
   column_name column_data_type,
   column_name column_data_type
   ...
);
```

Παράδειγμα

Η ακόλουθη πρόταση SQL δημιουργεί έναν πίνακα που ονομάζεται Employees με τέσσερις στήλες :[7]

```
SQL> CREATE TABLE Employees
(
   id INT NOT NULL,
   age INT NOT NULL,
   first VARCHAR(255),
   last VARCHAR(255),
   PRIMARY KEY ( id )
);
```

2.5 Διαγραφή πίνακα

Η εντολή DROP TABLE χρησιμοποιείται για τη διαγραφή ενός υπάρχοντος πίνακα. Η σύνταξη της είναι : [2]

```
SQL> DROP TABLE table_name;
```

Παράδειγμα

Η ακόλουθη πρόταση SQL διαγράφει έναν πίνακα που ονομάζεται Employees: [7]

```
SQL> DROP TABLE Employees;
```

2.5 Εισαγωγή δεδομένων

Η σύνταξη για το INSERT, μοιάζει με την ακόλουθη, όπου η στήλη1, η στήλη2 κ.ο.κ. αναπαριστά τα νέα δεδομένα που εμφανίζονται στις αντίστοιχες στήλες : [2]

```
SQL> INSERT INTO table_name VALUES (column1, column2, ...);
```

Παράδειγμα

Η ακόλουθη πρόταση SQL INSERT εισάγει μια νέα σειρά στη βάση δεδομένων των Employees που δημιουργήθηκε νωρίτερα : [7]

```
SQL> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');
```

2.6 Ανάκτηση δεδομένων

Η εντολή SELECT χρησιμοποιείται για την ανάκτηση δεδομένων από μια βάση δεδομένων. Η σύνταξη για το SELECT είναι : [2]

```
SQL> SELECT column_name, column_name, ...
     FROM table_name
     WHERE conditions;
```

Η ρήτρα WHERE μπορεί να χρησιμοποιήσει τους χειριστές σύγκρισης όπως =,! =, <,>, <=, Και> =, καθώς και τους χειριστές BETWEEN και LIKE.

Παράδειγμα

Η ακόλουθη πρόταση SQL επιλέγει την ηλικία, την πρώτη και την τελευταία στήλη από τον πίνακα Employees, όπου η στήλη id είναι 100 : [7]

```
SQL> SELECT first, last, age
     FROM Employees
     WHERE id = 100;
```

Η ακόλουθη πρόταση SQL επιλέγει την ηλικία, την πρώτη και την τελευταία στήλη από τον πίνακα Employees όπου η πρώτη στήλη περιέχει το Zara : [7]

```
SQL> SELECT first, last, age
```

```
    FROM Employees

    WHERE first LIKE '%Zara%';
```

2.7 Ενημέρωση δεδομένων

Η εντολή UPDATE χρησιμοποιείται για την ενημέρωση των δεδομένων. Η σύνταξη για την UPDATE είναι : [2]

```
SQL> UPDATE table_name

    SET column_name = value, column_name = value, ...

    WHERE conditions;
```

Η ρήτρα WHERE μπορεί να χρησιμοποιήσει τους χειριστές σύγκρισης όπως =,! =, <,>, <=, Και> =, καθώς και τους χειριστές BETWEEN και LIKE.

Παράδειγμα

Η ακόλουθη πρόταση SQL UPDATE αλλάζει τη στήλη ηλικίας του υπαλλήλου του οποίου το id είναι 100 : [7]

```
SQL> UPDATE Employees SET age=20 WHERE id=100;
```

2.8 Διαγραφή δεδομένων

Η εντολή DELETE χρησιμοποιείται για τη διαγραφή δεδομένων από πίνακες. Η σύνταξη για το DELETE είναι : [2]

```
SQL> DELETE FROM table_name WHERE conditions;
```

Η ρήτρα WHERE μπορεί να χρησιμοποιήσει τους χειριστές σύγκρισης όπως =,! =, <,>, <=, Και> =, καθώς και τους χειριστές BETWEEN και LIKE.

Παράδειγμα

Η ακόλουθη παράθεση SQL DELETE διαγράφει την εγγραφή του υπαλλήλου του οποίου το id είναι 100 : [7]

```
SQL> DELETE FROM Employees WHERE id=100;
```

ΚΕΦΑΛΑΙΟ 3

JDBC - Ρύθμιση περιβάλλοντος

Για να ξεκινήσετε την ανάπτυξη με το JDBC, θα πρέπει να ρυθμίσετε το περιβάλλον του JDBC ακολουθώντας τα παρακάτω βήματα. Υποθέτουμε ότι εργάζεστε σε μια πλατφόρμα Windows.

3.1 Εγκαταστήστε τη Java

Εγκαταστήστε το J2SE Development Kit 5.0 (JDK 5.0) από την Επίσημη Ιστοσελίδα της Java (http://www.oracle.com).

Βεβαιωθείτε ότι οι ακόλουθες μεταβλητές περιβάλλοντος έχουν οριστεί όπως περιγράφεται παρακάτω: [5]

- **JAVA_HOME:** Αυτή η μεταβλητή περιβάλλοντος θα πρέπει να δείχνει στον κατάλογο όπου εγκαταστήσατε το JDK, π.χ. C: \ Program Files \ Java \ jdk1.5.0.
- **CLASSPATH:** Αυτή η μεταβλητή περιβάλλοντος θα πρέπει να έχει τις κατάλληλες διαδρομές, π.χ. C: \ Program Files \ Java \ jdk1.5.0_20 \ jre \ lib.
- **PATH:** Αυτή η μεταβλητή περιβάλλοντος θα πρέπει να δείχνει στον κατάλληλο JRE bin, π.χ. C: \ Program Files \ Java \ jre1.5.0_20 \ bin.

Είναι πιθανό να έχετε αυτά τα μεταβλητά σετ ήδη, αλλά απλώς για να βεβαιωθείτε πραγματοποιήστε τον παρακάτω έλεγχο.

- Μεταβείτε στον πίνακα ελέγχου και κάντε διπλό κλικ στο Σύστημα. Εάν είστε χρήστης των Windows XP, είναι πιθανό να πρέπει να ανοίξετε την απόδοση και τη συντήρηση, πριν να δείτε το εικονίδιο του συστήματος.
- Μεταβείτε στην καρτέλα Για προχωρημένους και κάντε κλικ στις μεταβλητές περιβάλλοντος.
- Τώρα ελέγξτε αν όλες οι παραπάνω μεταβλητές έχουν ρυθμιστεί σωστά.

Λαμβάνετε αυτόματα και τα δύο πακέτα JDBC java.sql και javax.sql, όταν εγκαθιστάτε το J2SE Development Kit 5.0 (JDK 5.0). [5]

3.2 Εγκατάσταση βάσης δεδομένων

Το πιο σημαντικό πράγμα που θα χρειαστείτε, φυσικά είναι μια πραγματική βάση δεδομένων που λειτουργεί με έναν πίνακα που μπορείτε να ζητήσετε και να τροποποιήσετε.

Εγκαταστήστε μια βάση δεδομένων που σας ταιριάζει περισσότερο. Μπορείτε να έχετε πολλές επιλογές και οι πιο συνηθισμένες είναι :

- **MySQL DB:** Η MySQL είναι μια βάση δεδομένων ανοιχτού κώδικα. Μπορείτε να το κατεβάσετε από την Επίσημη Ιστοσελίδα της MySQL (https://dev.mysql.com/downloads/mysql/). Σας συνιστούμε να κάνετε λήψη της πλήρους εγκατάστασης των Windows.

- Επιπλέον, κατεβάστε και εγκαταστήστε το MySQL Administrator καθώς και το MySQL Query Browser. Αυτά είναι εργαλεία βασισμένα σε GUI που θα κάνουν την ανάπτυξή σας πολύ πιο εύκολη.

- Τέλος, κατεβάστε και αποσυνδέστε το MySQL Connector / J (το πρόγραμμα οδήγησης JDBC της MySQL) σε έναν βολικό κατάλογο. Για τους σκοπούς αυτής της πτυχιακής θα υποθέσουμε ότι έχετε εγκαταστήσει το πρόγραμμα οδήγησης στο C: \ Program Files \ MySQL \ mysql-connector-java-5.1.8.

- Συνεπώς, ρυθμίστε τη μεταβλητή CLASSPATH σε C: \ Program Files \ MySQL \ mysql-connector-java-5.1.8 \ mysql-connector-java-5.1.8-bin.jar. Η έκδοση του προγράμματος οδήγησης μπορεί να διαφέρει ανάλογα με την εγκατάστασή σας.

- **PostgreSQL DB:** Η PostgreSQL είναι μια βάση δεδομένων ανοιχτού κώδικα. Μπορείτε να το κατεβάσετε από την Επίσημη Ιστοσελίδα PostgreSQL (https://www.postgresql.org/download).

- Η εγκατάσταση Postgres περιέχει ένα εργαλείο διαχείρισης με βάση το GUI που ονομάζεται pgAdmin III. Τα προγράμματα οδήγησης JDBC περιλαμβάνονται επίσης ως μέρος της εγκατάστασης.

- **Oracle DB:** Η Oracle DB είναι μια εμπορική βάση δεδομένων που πωλείται από την Oracle. Υποθέτουμε ότι έχετε τα απαραίτητα μέσα διανομής για να το εγκαταστήσετε.

- Η εγκατάσταση της Oracle περιλαμβάνει ένα εργαλείο διαχείρισης βασισμένο σε GUI που ονομάζεται Enterprise Manager. Τα προγράμματα οδήγησης JDBC περιλαμβάνονται επίσης ως μέρος της εγκατάστασης.

3.3 Εγκαταστήστε προγράμματα οδήγησης βάσεων δεδομένων

Το τελευταίο JDK περιλαμβάνει πρόγραμμα οδήγησης JDBC-ODBC Bridge, το οποίο κάνει τα περισσότερα προγράμματα οδήγησης Open Database Connectivity (ODBC) διαθέσιμα στους προγραμματιστές που χρησιμοποιούν το API JDBC.

Στις μέρες μας, οι περισσότεροι προμηθευτές βάσης δεδομένων παρέχουν τα κατάλληλα προγράμματα οδήγησης JDBC μαζί με την εγκατάσταση της βάσης δεδομένων. Έτσι, δεν πρέπει να ανησυχείτε για αυτό το κομμάτι.

3.4 Ορίστε την πιστοποίηση βάσης δεδομένων

Για αυτή τη πτυχιακή εργασία πρόκειται να χρησιμοποιήσουμε τη βάση δεδομένων MySQL. Όταν εγκαθιστάτε οποιαδήποτε από τις παραπάνω βάσεις δεδομένων, το αναγνωριστικό του διαχειριστή έχει οριστεί ως **root** και παρέχει τη δυνατότητα να ορίσετε έναν κωδικό πρόσβασης της επιλογής σας.

Χρησιμοποιώντας τον κωδικό αναγνώρισης και τον κωδικό πρόσβασης μπορείτε να δημιουργήσετε ένα άλλο αναγνωριστικό χρήστη και έναν κωδικό πρόσβασης ή μπορείτε να χρησιμοποιήσετε το αναγνωριστικό root και τον κωδικό πρόσβασης για την εφαρμογή JDBC.

Υπάρχουν διάφορες λειτουργίες βάσεων δεδομένων, όπως η δημιουργία και διαγραφή βάσεων δεδομένων, οι οποίες θα χρειάζονται ταυτότητα διαχειριστή και κωδικό πρόσβασης.

Για το υπόλοιπο του φροντιστηρίου JDBC, θα χρησιμοποιήσαμε τη βάση δεδομένων MySQL με **όνομα χρήστη : username** και **κωδικό πρόσβασης: password.**

3.5 Δημιουργία βάσης δεδομένων

Για να δημιουργήσετε τη βάση δεδομένων **EMP** , ακολουθήστε τα παρακάτω βήματα : [7]

Βήμα 1

Ανοίξτε μια γραμμή εντολών και αλλάξτε στον κατάλογο εγκατάστασης ως εξής :

```
C:\>
C:\>cd Program Files\MySQL\bin
C:\Program Files\MySQL\bin>
```

Σημείωση: Η διαδρομή προς το **mysqld.exe** ενδέχεται να διαφέρει ανάλογα με την τοποθεσία εγκατάστασης της MySQL στο σύστημά σας.

Βήμα 2

Ξεκινήστε τον διακομιστή βάσης δεδομένων εκτελώντας την ακόλουθη εντολή, αν δεν είναι ήδη ενεργοποιημένη.

```
C:\Program Files\MySQL\bin>mysqld
C:\Program Files\MySQL\bin>
```

Βήμα 3

Δημιουργήστε τη βάση δεδομένων EMP εκτελέστε την ακόλουθη εντολή :

```
C:\Program Files\MySQL\bin> mysqladmin create EMP -u root -p
Enter password: ********
C:\Program Files\MySQL\bin>
```

3.6 Δημιουργία πίνακα

Για να δημιουργήσετε τον πίνακα " Employee" στη βάση δεδομένων EMP, χρησιμοποιήστε τα παρακάτω βήματα : [7]

Βήμα 1

Ανοίξτε μια γραμμή εντολών και αλλάξτε στον κατάλογο εγκατάστασης ως εξής :

```
C:\>
C:\>cd Program Files\MySQL\bin
C:\Program Files\MySQL\bin>
```

Βήμα 2

Συνδεθείτε στη βάση δεδομένων ως εξής :

```
C:\Program Files\MySQL\bin>mysql -u root -p
Enter password: ********
mysql>
```

Βήμα 3

Δημιουργήστε τον πίνακα **Employee** ως εξής :

```
mysql> use EMP;
mysql> create table Employees
    -> (
    -> id int not null,
    -> age int not null,
    -> first varchar (255),
    -> last varchar (255)
    -> );
Query OK, 0 rows affected (0.08 sec)
mysql>
```

3.7 Δημιουργία αρχείων δεδομένων

Τέλος, δημιουργείτε λίγες εγγραφές στον πίνακα Employees ως εξής : [7]

```
mysql> INSERT INTO Employees VALUES (100, 18, 'Zara', 'Ali');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO Employees VALUES (101, 25, 'Mahnaz', 'Fatma');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (102, 30, 'Zaid', 'Khan');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO Employees VALUES (103, 28, 'Sumit', 'Mittal');
Query OK, 1 row affected (0.00 sec)
mysql>
```

ΚΕΦΑΛΑΙΟ 4

JDBC - Τύποι προγραμμάτων οδήγησης

4.1 Τι είναι το πρόγραμμα οδήγησης JDBC;

Τα προγράμματα οδήγησης JDBC υλοποιούν τις καθορισμένες διεπαφές στο API JDBC για αλληλεπίδραση με το διακομιστή της βάσης δεδομένων.

Για παράδειγμα, χρησιμοποιώντας τα προγράμματα οδήγησης JDBC, μπορείτε να ανοίξετε συνδέσεις με βάσεις δεδομένων και να αλληλεπιδράσετε με αυτές στέλνοντας εντολές SQL και έπειτα να λαμβάνετε αποτελέσματα με την Java.

Το πακέτο Java.sql που συνοδεύει το JDK, περιέχει διάφορες κατηγορίες με τις συμπεριφορές τους και οι πραγματικές υλοποιήσεις τους όπου γίνονται με προγράμματα οδήγησης άλλων κατασκευαστών. Οι προμηθευτές τρίτου μέρους εφαρμόζουν τη διασύνδεση java.sql.Driver στο πρόγραμμα οδήγησης της βάσης δεδομένων τους. [6]

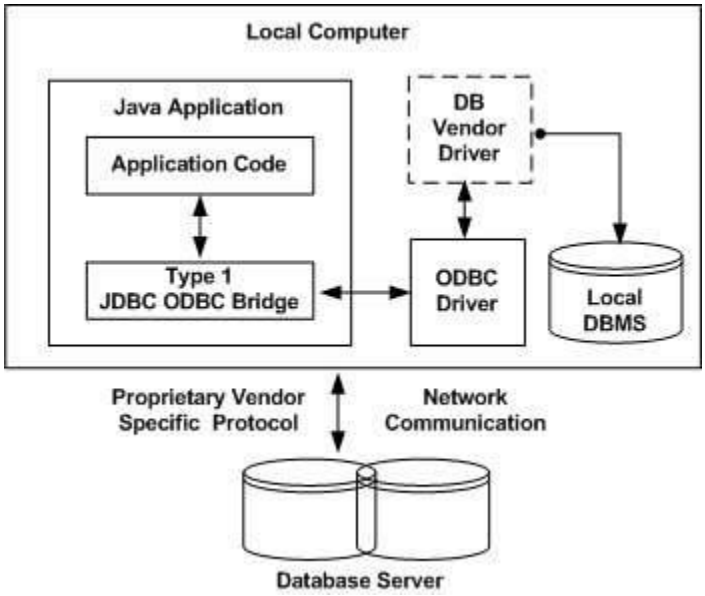
4.2 Τύποι προγραμμάτων οδήγησης JDBC

Οι υλοποιήσεις του προγράμματος οδήγησης JDBC ποικίλλουν λόγω της μεγάλης ποικιλίας των λειτουργικών συστημάτων και των πλατφορμών υλικού στα οποία λειτουργεί η Java. Η Sun έχει χωρίσει τους τύπους υλοποίησης σε τέσσερις κατηγορίες, τους τύπους 1, 2, 3 και 4, οι οποίοι εξηγούνται παρακάτω :

4.2.1 Τύπος 1: JDBC-ODBC Bridge Driver

Σε ένα πρόγραμμα οδήγησης τύπου 1, μια γέφυρα JDBC χρησιμοποιείται για την πρόσβαση σε προγράμματα οδήγησης ODBC εγκατεστημένα σε κάθε υπολογιστή-πελάτη. Χρησιμοποιώντας το ODBC, απαιτείται να διαμορφώσετε στο σύστημά σας ένα όνομα πηγής δεδομένων (DSN) που αντιπροσωπεύει τη βάση δεδομένων προορισμού.

Όταν ξεκίνησε η πρώτη έκδοση της Java, αυτό ήταν ένα χρήσιμο πρόγραμμα οδήγησης επειδή οι περισσότερες βάσεις δεδομένων υποστήριζαν μόνο την πρόσβαση ODBC, αλλά τώρα αυτός ο τύπος προγράμματος οδήγησης συνιστάται μόνο για πειραματική χρήση ή όταν δεν υπάρχει άλλη εναλλακτική λύση. [6]



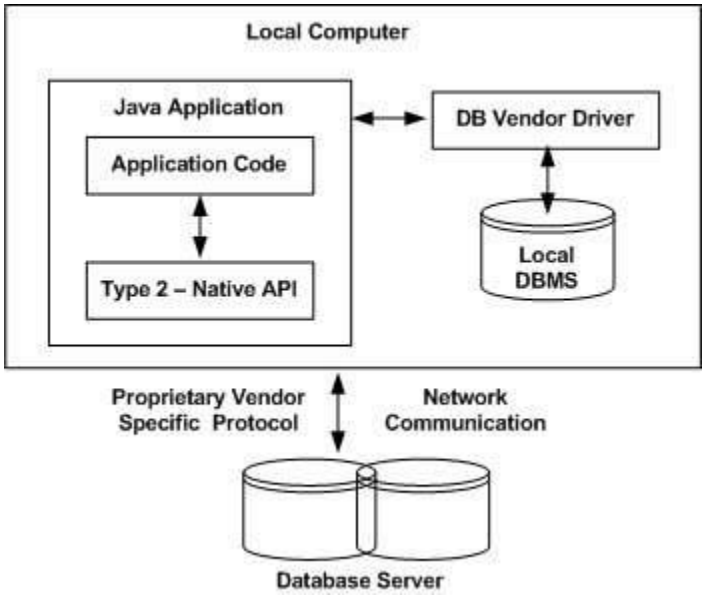
Εικόνα 4.1: JDBC-ODBC Bridge Driver [7]

Η Γέφυρα JDBC-ODBC που συνοδεύει το JDK 1.2 αποτελεί καλό παράδειγμα αυτού του τύπου οδηγού.

4.2.2 Τύπος 2: JDBC-Native API

Σε ένα πρόγραμμα οδήγησης τύπου 2, οι κλήσεις JDBC API μετατρέπονται σε native C / C++ API κλήσεις, οι οποίες είναι μοναδικές για τη βάση δεδομένων. Αυτά τα προγράμματα οδήγησης παρέχονται συνήθως από τους προμηθευτές της βάσης δεδομένων και χρησιμοποιούνται με τον ίδιο τρόπο όπως η γέφυρα JDBC-ODBC. Το συγκεκριμένο πρόγραμμα οδήγησης πρέπει να εγκατασταθεί σε κάθε υπολογιστή-πελάτη.

Αν αλλάξουμε τη Βάση Δεδομένων, πρέπει να αλλάξουμε το εγγενές API, δεδομένου ότι είναι συγκεκριμένο σε μια βάση δεδομένων, αλλά μπορεί να πραγματοποιήσετε κάποια αύξηση ταχύτητας με ένα πρόγραμμα οδήγησης τύπου 2, επειδή εξαλείφει τα γενικά έξοδα του ODBC. [6]



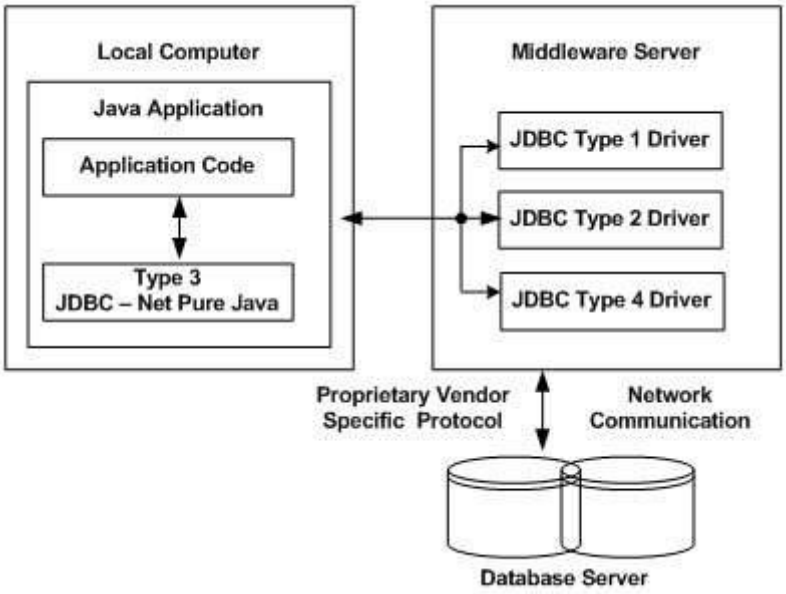
Εικόνα 4.2: JDBC-Native API [7]

Το πρόγραμμα οδήγησης Oracle Call Interface (OCI) είναι ένα παράδειγμα ενός προγράμματος οδήγησης τύπου 2.

4.2.3 Τύπος 3: JDBC-Net pure Java

Σε ένα πρόγραμμα οδήγησης τύπου 3, χρησιμοποιείται προσέγγιση τριών επιπέδων για την πρόσβαση σε βάσεις δεδομένων. Οι πελάτες JDBC χρησιμοποιούν πρότυπες υποδοχές δικτύου για να επικοινωνούν με ένα διακομιστή εφαρμογών middleware. Στη συνέχεια, οι πληροφορίες υποδοχής μεταφράζονται από τον διακομιστή εφαρμογών του μεσαίου λογισμικού στη μορφή κλήσης που απαιτείται από το DBMS και διαβιβάζονται στο διακομιστή βάσης δεδομένων.

Αυτό το είδος οδηγού είναι εξαιρετικά ευέλικτο, καθώς δεν απαιτεί κανένας κώδικας εγκατεστημένος στον πελάτη και ένας μόνο οδηγός μπορεί να παρέχει πρόσβαση σε πολλές βάσεις δεδομένων. [6]



Εικόνα 4.3: JDBC-Net pure Java [7]

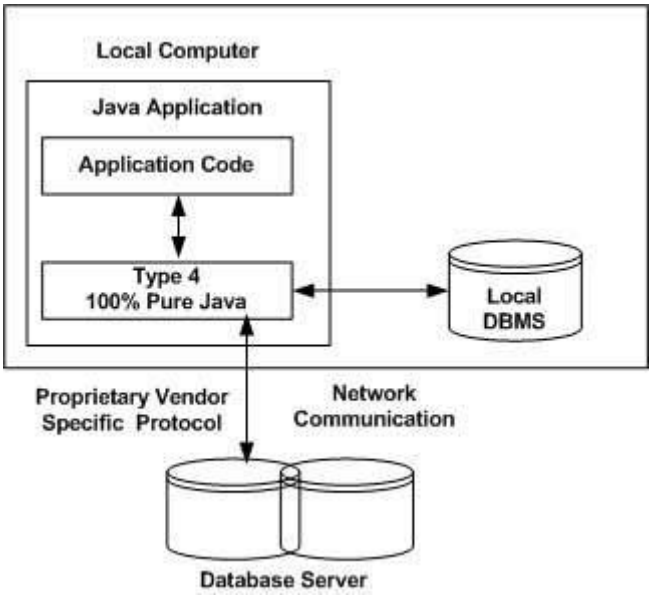
Μπορείτε να σκεφτείτε τον διακομιστή εφαρμογών ως JDBC "proxy", που σημαίνει ότι κάνει κλήσεις για την εφαρμογή του πελάτη. Ως αποτέλεσμα, χρειάζεστε κάποια γνώση σχετικά με τη διαμόρφωση του διακομιστή εφαρμογών, προκειμένου να χρησιμοποιήσετε αποτελεσματικά αυτόν τον τύπο προγράμματος οδήγησης.

Ο διακομιστής εφαρμογών σας ενδέχεται να χρησιμοποιήσει πρόγραμμα οδήγησης τύπου 1, 2 ή 4 για να επικοινωνήσει με τη βάση δεδομένων, καθώς η κατανόηση των αποχρώσεων θα αποδειχθεί χρήσιμη. [6]

4.2.4 Τύπος 4: 100% Pure Java

Σε ένα πρόγραμμα οδήγησης τύπου 4, ένα καθαρό πρόγραμμα οδήγησης βασισμένο σε Java επικοινωνεί απευθείας με τη βάση δεδομένων του προμηθευτή μέσω σύνδεσης υποδοχής. Αυτός είναι ο οδηγός υψηλότερης απόδοσης που είναι διαθέσιμος για τη βάση δεδομένων και συνήθως παρέχεται από τον ίδιο τον πωλητή.

Αυτό το είδος οδηγού είναι εξαιρετικά ευέλικτο, δεν χρειάζεται να εγκαταστήσετε ειδικό λογισμικό στον πελάτη ή στον διακομιστή. Επιπλέον, αυτοί οι οδηγοί μπορούν να μεταφορτωθούν δυναμικά. [6]



Εικόνα 4.4: 100% Pure Java [7]

Ο οδηγός Connector/J της MySQL είναι ένας οδηγός τύπου 4. Λόγω της ιδιόκτητης φύσης των πρωτοκόλλων δικτύου τους, οι προμηθευτές βάσεων δεδομένων παρέχουν συνήθως οδηγούς τύπου 4.

4.3 Ποιο πρόγραμμα οδήγησης πρέπει να χρησιμοποιηθεί;

Εάν έχετε πρόσβαση σε έναν τύπο βάσης δεδομένων, όπως Oracle, Sybase ή IBM, ο προτιμώμενος τύπος προγράμματος οδήγησης είναι ο τύπος 4.

Εάν η εφαρμογή Java σας έχει πρόσβαση σε πολλαπλούς τύπους βάσεων δεδομένων ταυτόχρονα, ο τύπος 3 είναι ο προτιμώμενος οδηγός.

Τα προγράμματα οδήγησης τύπου 2 είναι χρήσιμα σε καταστάσεις όπου ο οδηγός τύπου 3 ή τύπου 4 δεν είναι ακόμα διαθέσιμος για τη βάση δεδομένων σας.

Το πρόγραμμα οδήγησης τύπου 1 δεν θεωρείται πρόγραμμα οδήγησης σε επίπεδο ανάπτυξης και χρησιμοποιείται συνήθως μόνο για σκοπούς ανάπτυξης και δοκιμής. [6]

ΚΕΦΑΛΑΙΟ 5

JDBC - Συνδέσεις βάσης δεδομένων

5.1 Συνδέσεις βάσης δεδομένων

Αφού εγκαταστήσετε το κατάλληλο πρόγραμμα οδήγησης, είναι καιρός να δημιουργήσετε μια σύνδεση με τη βάση δεδομένων χρησιμοποιώντας το JDBC.

Ο προγραμματισμός που συνδέεται με τη δημιουργία μιας σύνδεσης JDBC είναι αρκετά απλός. Εδώ είναι αυτά τα απλά τέσσερα βήματα :

- **Εισαγωγή πακέτων JDBC:** Προσθέστε δηλώσεις εισαγωγής στο πρόγραμμα Java για να εισάγετε τις απαιτούμενες κλάσεις στον κώδικα Java.
- **Εγγραφή προγράμματος οδήγησης JDBC:** Αυτό το βήμα αναγκάζει το JVM να φορτώσει την επιθυμητή εφαρμογή του οδηγού στη μνήμη, ώστε να μπορεί να ικανοποιήσει τα αιτήματα JDBC.
- **Δήλωση διεύθυνσης URL βάσης δεδομένων:** Με αυτόν τον τρόπο δημιουργείται μια σωστά διαμορφωμένη διεύθυνση που δείχνει τη βάση δεδομένων με την οποία επιθυμείτε να συνδεθείτε.
- **Δημιουργία αντικειμένου σύνδεσης:** Τέλος, κωδικοποιήστε μια κλήση στη μέθοδο getConnection() του αντικειμένου DriverManager για να δημιουργήσετε μια πραγματική σύνδεση βάσης δεδομένων. [6]

5.2 Εισαγωγή πακέτων JDBC

Οι δηλώσεις εισαγωγής αναφέρουν στον μεταγλωττιστή της Java για τον τόπο εντοπισμού των κλάσεων που αναφέρατε στον κώδικα και τοποθετούνται στην αρχή του πηγαίου κώδικα.

Για να χρησιμοποιήσετε το τυπικό πακέτο JDBC, το οποίο σας επιτρέπει να επιλέξετε, να εισαγάγετε, να ενημερώσετε και να διαγράψετε δεδομένα σε πίνακες SQL, προσθέστε τις ακόλουθες εισαγωγές στον πηγαίο κώδικα : [7]

```
import java.sql.* ;  // for standard JDBC programs
import java.math.* ; // for BigDecimal and BigInteger support
```

5.3 Εγγραφή του προγράμματος οδήγησης JDBC

Πρέπει να καταχωρήσετε το πρόγραμμα οδήγησης στο πρόγραμμα σας προτού το χρησιμοποιήσετε. Η εγγραφή του προγράμματος οδήγησης είναι η διαδικασία με την οποία φορτώνεται το αρχείο κατηγορίας του οδηγού Oracle στη μνήμη, έτσι ώστε να μπορεί να χρησιμοποιηθεί ως εφαρμογή των διεπαφών JDBC.

Πρέπει να κάνετε αυτήν την εγγραφή μόνο μία φορά στο πρόγραμμά σας. Μπορείτε να καταχωρίσετε ένα πρόγραμμα οδήγησης με έναν από τους δύο τρόπους.

5.3.1 Προσέγγιση I - Class.forName()

Η πιο κοινή προσέγγιση για την καταχώρηση ενός προγράμματος οδήγησης είναι να χρησιμοποιήσετε τη μέθοδο Class.forName() της Java, για να φορτώσετε δυναμικά το αρχείο με την κλάση του οδηγού στη μνήμη, το οποίο αυτόματα καταχωρείτε. Αυτή η μέθοδος είναι προτιμότερη επειδή σας επιτρέπει να ρυθμίσετε την καταχώριση του προγράμματος οδήγησης και να είναι φορητή.

Το ακόλουθο παράδειγμα χρησιμοποιεί το Class.forName() για την καταχώρηση του προγράμματος οδήγησης Oracle : [7]

```
try {
   Class.forName("oracle.jdbc.driver.OracleDriver");
}
catch(ClassNotFoundException ex) {
   System.out.println("Error: unable to load driver class!");
   System.exit(1);
}
```

Μπορείτε να χρησιμοποιήσετε τη μέθοδο getInstance() για να επεξεργαστείτε μη συμβατές JVM, αλλά στη συνέχεια θα πρέπει να κωδικοποιήσετε για δύο επιπλέον Εξαιρέσεις ως εξής : [7]

```
try {
   Class.forName("oracle.jdbc.driver.OracleDriver").newInstance();
}
catch(ClassNotFoundException ex) {
   System.out.println("Error: unable to load driver class!");
   System.exit(1);
catch(IllegalAccessException ex) {
   System.out.println("Error: access problem while loading!");
   System.exit(2);
catch(InstantiationException ex) {
   System.out.println("Error: unable to instantiate driver!");
   System.exit(3);
}
```

5.3.2 Προσέγγιση II - DriverManager.registerDriver()

Η δεύτερη προσέγγιση που μπορείτε να χρησιμοποιήσετε για να καταχωρίσετε ένα πρόγραμμα οδήγησης είναι να χρησιμοποιήσετε τη στατική μέθοδο DriverManager.registerDriver().

Πρέπει να χρησιμοποιήσετε τη μέθοδο registerDriver() εάν χρησιμοποιείτε JVM μη συμβατό με JDK, όπως αυτή που παρέχεται από τη Microsoft.

Το ακόλουθο παράδειγμα χρησιμοποιεί το registerDriver() για την καταχώρηση του προγράμματος οδήγησης Oracle : [7]

```
try {
   Driver myDriver = new oracle.jdbc.driver.OracleDriver();
   DriverManager.registerDriver( myDriver );
}
catch(ClassNotFoundException ex) {
   System.out.println("Error: unable to load driver class!");
   System.exit(1);
}
```

5.4 Σύνταξη διεύθυνσης URL βάσης δεδομένων

Αφού τοποθετήσετε το πρόγραμμα οδήγησης, μπορείτε να δημιουργήσετε μια σύνδεση χρησιμοποιώντας τη μέθοδο DriverManager.getConnection() . Για εύκολη αναφορά, επιτρέψτε μου να απαριθμήσω τις τρεις υπερφορτωμένες μεθόδους DriverManager.getConnection()

- **getConnection (String url)**
- **getConnection (String url, Properties prop)**
- **getConnection (String url, String user, String password)**

Κάθε φόρμα απαιτεί μια διεύθυνση URL της βάσης δεδομένων . Μια διεύθυνση URL της βάσης δεδομένων είναι μια διεύθυνση που δείχνει στη βάση δεδομένων σας.

Η διατύπωση μιας διεύθυνσης URL της βάσης δεδομένων είναι εκεί όπου συμβαίνουν τα περισσότερα από τα προβλήματα που σχετίζονται με τη δημιουργία μιας σύνδεσης.

Ο παρακάτω πίνακας αναγράφει τα δημοφιλή ονόματα προγραμμάτων οδήγησης JDBC και τη διεύθυνση URL της βάσης δεδομένων. [6]

Πίνακας 5.1: Όνομα προγράμματος οδήγησης JDBC και διεύθυνση URL [7]

RDBMS	Όνομα προγράμματος οδήγησης JDBC	Μορφή URL
MySQL	com.mysql.jdbc.Driver	**jdbc: mysql: //** hostname / databaseName
ORACLE	oracle.jdbc.driver.OracleDriver	**jdbc:oracle:thin:@**hostname:port Number:databaseName
DB2	COM.ibm.db2.jdbc.net.DB2Driver	**jdbc:db2:**hostname:port Number/databaseName
Sybase	com.sybase.jdbc.SybDriver	**jdbc:sybase:Tds:**hostname: port Number/databaseName

Όλο το επισημασμένο τμήμα στη μορφή URL είναι στατικό και θα πρέπει να αλλάξετε μόνο το υπόλοιπο κομμάτι σύμφωνα με τη ρύθμιση της βάσης δεδομένων.

5.5 Δημιουργία αντικειμένου σύνδεσης

Έχουμε καταγράψει τρεις μορφές της μεθόδου DriverManager.getConnection() για να δημιουργήσετε ένα αντικείμενο σύνδεσης.

5.5.1 Χρησιμοποιώντας μια διεύθυνση URL βάσης δεδομένων με όνομα χρήστη και κωδικό πρόσβασης

Η πιο συχνά χρησιμοποιούμενη μορφή του getConnection() απαιτεί να περάσετε μια διεύθυνση URL της βάσης δεδομένων, ένα όνομα χρήστη και έναν κωδικό πρόσβασης :

Υποθέτοντας ότι χρησιμοποιείτε το thin πρόγραμμα οδήγησης της Oracle , θα καθορίσετε μια τιμή host: port: databaseName για το τμήμα βάσης δεδομένων της διεύθυνσης URL.

Εάν έχετε έναν κεντρικό υπολογιστή στη διεύθυνση TCP / IP 192.0.0.1 με όνομα host του amrood και ο ακροατής Oracle έχει ρυθμιστεί να ακούει στη θύρα 1521 και το όνομα της βάσης δεδομένων σας είναι EMP, τότε η πλήρης διεύθυνση URL της βάσης δεδομένων θα είναι : [7]

```
jdbc:oracle:thin:@amrood:1521:EMP
```

Τώρα πρέπει να καλέσετε τη μέθοδο getConnection() με το κατάλληλο όνομα χρήστη και κωδικό πρόσβασης για να αποκτήσετε ένα αντικείμενο Connection ως εξής : [7]

```
String URL = "jdbc:oracle:thin:@amrood:1521:EMP";

String USER = "username";

String PASS = "password";

Connection conn = DriverManager.getConnection(URL, USER, PASS);
```

5.5.2 Χρήση μόνο μιας διεύθυνσης URL βάσης δεδομένων

Μια δεύτερη μορφή της μεθόδου DriverManager.getConnection() απαιτεί μόνο μια διεύθυνση βάσης δεδομένων :

```
DriverManager.getConnection(String url);
```

Ωστόσο, στην περίπτωση αυτή, η διεύθυνση URL της βάσης δεδομένων περιλαμβάνει το όνομα χρήστη και τον κωδικό πρόσβασης και έχει την ακόλουθη γενική μορφή :

```
jdbc:oracle:driver:username/password@database
```

Έτσι, η παραπάνω σύνδεση μπορεί να δημιουργηθεί ως εξής : [7]

```
String URL = "jdbc:oracle:thin:username/password@amrood:1521:EMP";

Connection conn = DriverManager.getConnection(URL);
```

5.5.3 Χρησιμοποιώντας μια διεύθυνση URL βάσης δεδομένων και ένα αντικείμενο ιδιοτήτων

Μια τρίτη μορφή της μεθόδου DriverManager.getConnection() απαιτεί μια διεύθυνση URL βάσης δεδομένων και ένα αντικείμενο Properties:

```
DriverManager.getConnection(String url, Properties info);
```

Ένα αντικείμενο Properties διατηρεί ένα σύνολο ζευγών λέξης-τιμής. Χρησιμοποιείται για να μεταβιβάσει τις ιδιότητες του προγράμματος οδήγησης στο ίδιο το πρόγραμμα οδήγησης κατά τη διάρκεια μιας κλήσης της μεθόδου getConnection().[7]

```
import java.util.*;


String URL = "jdbc:oracle:thin:@amrood:1521:EMP";

Properties info = new Properties();

info.put( "user", "username" );

info.put( "password", "password" );


Connection conn = DriverManager.getConnection(URL, info);
```

5.6 Κλείσιμο συνδέσεων JDBC

Στο τέλος του προγράμματος JDBC, απαιτείται ρητά να κλείσετε όλες τις συνδέσεις στη βάση δεδομένων για να τερματίσετε κάθε σύνοδο βάσης δεδομένων. Ωστόσο, εάν ξεχάσετε, ο συλλέκτης απορριμμάτων της Java θα κλείσει τη σύνδεση όταν καθαρίζει τα αντικείμενα που έχουν παραμείνει.

Στηριζόμενη στη συλλογή απορριμμάτων, ειδικά στον προγραμματισμό της βάσης δεδομένων, είναι μια πολύ κακή πρακτική προγραμματισμού. Θα πρέπει να κάνετε μια συνήθεια να κλείνετε πάντα τη σύνδεση με τη μέθοδο close() που σχετίζεται με το αντικείμενο σύνδεσης.

Για να διασφαλίσετε ότι μια σύνδεση είναι κλειστή, θα μπορούσατε να δώσετε ένα μπλοκ " finally " στον κωδικό σας. Ένα " finally " μπλοκ εκτελείτε πάντα, ανεξάρτητα από την περίπτωση μιας εξαίρεσης ή όχι.

Για να κλείσετε την παραπάνω ανοιχτή σύνδεση, θα πρέπει να καλέσετε τη μέθοδο close() ως εξής : [7]

```
conn.close();
```

Το ρητό κλείσιμο μιας σύνδεσης διατηρεί τους πόρους του DBMS, γεγονός που θα κάνει τον διαχειριστή της βάσης δεδομένων σας ευτυχισμένο.

ΚΕΦΑΛΑΙΟ 6

JDBC - Statements, PreparedStatement και CallableStatement

6.1 JDBC - Statements, PreparedStatement και CallableStatement

Μόλις ληφθεί μια σύνδεση, μπορούμε να αλληλεπιδράσουμε με τη βάση δεδομένων. Οι διεπαφές της JDBC, Statement, CallableStatement και PreparedStatement καθορίζουν τις μεθόδους και τις ιδιότητες που σας επιτρέπουν να στέλνετε εντολές SQL ή PL/SQL και να λαμβάνετε δεδομένα από τη βάση δεδομένων σας.

Επίσης, καθορίζουν μεθόδους που συμβάλλουν στη γεφύρωση των διαφορών των δεδομένων μεταξύ των τύπων δεδομένων Java και SQL που χρησιμοποιούνται σε μια βάση δεδομένων.

Ο παρακάτω πίνακας παρέχει μια περίληψη του σκοπού κάθε διεπαφής.

Πίνακας 6.1: Περίληψη του σκοπού κάθε διεπαφής [7]

Διεπαφές	**Συνιστώμενη χρήση**
Statement	Χρησιμοποιήστε τη γενική πρόσβαση στη βάση δεδομένων σας. Χρήσιμο όταν χρησιμοποιείτε στατικές δηλώσεις SQL κατά το χρόνο εκτέλεσης. Η διασύνδεση αυτή δεν μπορεί να δεχθεί παραμέτρους.
PreparedStatement	Χρησιμοποιήστε το όποτε σκοπεύετε να χρησιμοποιήσετε τις δηλώσεις SQL πολλές φορές. Η διεπαφή PreparedStatement δέχεται παραμέτρους εισόδου κατά το χρόνο εκτέλεσης.
CallableStatement	Χρησιμοποιήστε το όποτε θέλετε να έχετε πρόσβαση στις αποθηκευμένες διαδικασίες της βάσης δεδομένων. Η διεπαφή CallableStatement μπορεί επίσης να δεχθεί παραμέτρους εισόδου χρόνου εκτέλεσης.

6.2 Δημιουργία αντικειμένου Statement

Για να μπορέσετε να χρησιμοποιήσετε ένα αντικείμενο τύπου Statement και για να εκτελέσετε μια εντολή SQL, πρέπει να το δημιουργήσετε χρησιμοποιώντας τη μέθοδο creationStatemen() του αντικειμένου σύνδεσης, όπως στο παρακάτω παράδειγμα : [7]

```
Statement stmt = null;
try {
   stmt = conn.createStatement();
   . . .
}
catch (SQLException e) {
   . . .
}
finally {
   . . .
}
```

Μόλις δημιουργήσετε ένα αντικείμενο Statement, μπορείτε στη συνέχεια να το χρησιμοποιήσετε για να εκτελέσετε μια εντολή SQL με μία από τις τρεις μεθόδους εκτέλεσης.

- **boolean execute(String SQL):** Επιστρέφει μια τιμή boolean όπου είναι true εάν μπορεί να ανακτηθεί ένα αντικείμενο ResultSet. Αλλιώς επιστρέφει false. Χρησιμοποιήστε αυτήν τη μέθοδο για την εκτέλεση εντολών SQL DDL ή όταν πρέπει να χρησιμοποιήσετε πραγματικά δυναμική SQL.

- **int executeUpdate(String SQL):** Επιστρέφει τον αριθμό των γραμμών που επηρεάζονται από την εκτέλεση της εντολής SQL. Χρησιμοποιήστε αυτήν τη μέθοδο για να εκτελέσει εντολές SQL για την οποία περιμένετε να πάρετε έναν αριθμό γραμμών που επηρεάζονται. Παράδειγμα δηλώσεων, INSERT, UPDATE ή DELETE.

- **ResultSet executeQuery(String SQL):** Εμφανίζει ένα αντικείμενο ResultSet. Χρησιμοποιήστε αυτήν τη μέθοδο όταν αναμένετε να πάρετε ένα σύνολο αποτελεσμάτων, όπως θα κάνατε με μια εντολή SELECT.

6.3 Τερματισμός αντικειμένου Statment

Ακριβώς όπως κλείνετε ένα αντικείμενο Connection για να αποθηκεύσετε πόρους της βάσης δεδομένων, για τον ίδιο λόγο θα πρέπει επίσης να κλείσετε το αντικείμενο Statement.

Μια απλή κλήση στη μέθοδο close() θα κάνει τη δουλειά. Αν κλείσετε πρώτα το αντικείμενο Connection, θα κλείσει και το αντικείμενο Statement. Ωστόσο, θα πρέπει πάντα να κλείσετε ρητά το αντικείμενο της Statement για να διασφαλίσετε τον σωστό καθαρισμό. [7]

```
Statement stmt = null;
try {
   stmt = conn.createStatement( );
   . . .
}
catch (SQLException e) {
   . . .
}
finally {
   stmt.close();
}
```

6.4 Τα αντικείμενα PreparedStatement

Η διεπαφή PreparedStatement επεκτείνει τη διασύνδεση Statement , η οποία σας δίνει πρόσθετη λειτουργικότητα με μερικά πλεονεκτήματα σε σχέση με ένα αντικείμενο Statement.

Αυτή η δήλωση σας δίνει την ευελιξία να παρέχετε δυναμικά επιχειρήματα.

6.4.1 Δημιουργία αντικειμένου PreparedStatement

```
PreparedStatement pstmt = null;

try {

   String SQL = "Update Employees SET age = ? WHERE id = ?";

   pstmt = conn.prepareStatement(SQL);

   . . .

}

catch (SQLException e) {

   . . .

}

finally {

   . . .

}
```

Όλες οι παράμετροι στο JDBC αντιπροσωπεύονται από το σύμβολο **?** , το οποίο είναι γνωστό ως δείκτης παραμέτρων. Πρέπει να δώσετε τιμές για κάθε παράμετρο πριν εκτελέσετε τη δήλωση SQL.

Οι μέθοδοι setXXX() δεσμεύουν τις τιμές στις παραμέτρους, όπου το XXX αντιπροσωπεύει τον τύπο δεδομένων Java της τιμής που θέλετε να δεσμεύσετε στην παράμετρο εισαγωγής. Εάν ξεχάσετε να δώσετε τις τιμές, θα λάβετε ένα SQLException.

Κάθε δείκτης παραμέτρων αναφέρεται από την κανονική του θέση. Ο πρώτος δείκτης αντιπροσωπεύει τη θέση 1, την επόμενη θέση 2 και ούτω καθεξής. Αυτή η μέθοδος διαφέρει από εκείνη των δεικτών ενός πίνακα Java, η οποία ξεκινά από το 0.

Όλες οι μέθοδοι του αντικειμένου Statement για αλληλεπίδραση με τη βάση δεδομένων (a) execute(), (b) executeQuery(), και (c) executeUpdate() λειτουργούν επίσης και με το αντικείμενο PreparedStatement. Ωστόσο, οι μέθοδοι τροποποιούνται για να χρησιμοποιούν δηλώσεις SQL που μπορούν να εισαγάγουν τις παραμέτρους. [7]

6.5 Κλείνοντας το αντικείμενο PreparedStatement

Ακριβώς όπως κλείνετε ένα αντικείμενο Statment, για τον ίδιο λόγο θα πρέπει επίσης να κλείσετε το αντικείμενο PreparedStatement.

Μια απλή κλήση στη μέθοδο close() θα κάνει τη δουλειά. Αν κλείσετε πρώτα το αντικείμενο Connection, θα κλείσει επίσης το αντικείμενο PreparedStatement. Ωστόσο, θα πρέπει πάντα να κλείσετε ρητά το αντικείμενο PreparedStatement για να διασφαλίσετε την σωστή εκκαθάριση. [7]

```
PreparedStatement pstmt = null;
try {
   String SQL = "Update Employees SET age = ? WHERE id = ?";
   pstmt = conn.prepareStatement(SQL);
   . . .
}
catch (SQLException e) {
   . . .
}
finally {
   pstmt.close();
}
```

6.6 Τα αντικείμενα CallableStatement

Ακριβώς όπως ένα αντικείμενο Connection δημιουργεί τα αντικείμενα Statement και PreparedStatement, δημιουργεί επίσης το αντικείμενο CallableStatement, το οποίο θα χρησιμοποιηθεί για την εκτέλεση μιας κλήσης σε μια αποθηκευμένη διαδικασία βάσης δεδομένων.

6.6.1 Δημιουργία αντικειμένου CallableStatement

Ας υποθέσουμε ότι πρέπει να εκτελέσετε την ακόλουθη αποθηκευμένη διαδικασία Oracle :

```
CREATE OR REPLACE PROCEDURE getEmpName
   (EMP_ID IN NUMBER, EMP_FIRST OUT VARCHAR) AS
```

```
BEGIN
   SELECT first INTO EMP_FIRST
   FROM Employees
   WHERE ID = EMP_ID;
END;
```

ΣΗΜΕΙΩΣΗ: Η παραπάνω αποθηκευμένη διαδικασία γράφτηκε για την Oracle, αλλά δουλεύουμε με τη βάση δεδομένων MySQL έτσι ας γράψουμε την ίδια αποθηκευμένη διαδικασία για την MySQL ως εξής για να την δημιουργήσετε στη βάση δεδομένων EMP : [7]

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `EMP`.`getEmpName` $$
CREATE PROCEDURE `EMP`.`getEmpName`
   (IN EMP_ID INT, OUT EMP_FIRST VARCHAR(255))
BEGIN
   SELECT first INTO EMP_FIRST
   FROM Employees
   WHERE ID = EMP_ID;
END $$

DELIMITER ;
```

Υπάρχουν τρεις τύποι παραμέτρων: IN, OUT και INOUT. Το αντικείμενο PreparedStatement χρησιμοποιεί μόνο την παράμετρο IN. Το αντικείμενο CallableStatement μπορεί να χρησιμοποιήσει και τα τρία.

Πίνακας 6.2: Περιγραφή παραμέτρων αντικειμένου PreparedStatement [7]

Παράμετρος	Περιγραφή
IN	Μια παράμετρος της οποίας η τιμή είναι άγνωστη όταν δημιουργείται η εντολή SQL. Μπορείτε να δεσμεύσετε τις τιμές στις παραμέτρους IN με τις μεθόδους setXXX ().
OUT	Μια παράμετρος της οποίας η τιμή παρέχεται από τη δήλωση SQL που επιστρέφει. Μπορείτε να ανακτήσετε τιμές από τις παραμέτρους OUT με τις μεθόδους getXXX ().
INOUT	Μια παράμετρος που παρέχει τιμές εισόδου και εξόδου. Μπορείτε να συνδέσετε μεταβλητές με τις μεθόδους setXXX () και να ανακτήσετε τιμές με τις μεθόδους getXXX ().

Το ακόλουθο απόσπασμα κώδικα παρουσιάζει τον τρόπο χρήσης της μεθόδου **Connection.prepareCall()** για την εμφάνιση ενός αντικειμένου **CallableStatement** με βάση την προηγούμενη αποθηκευμένη διαδικασία : [7]

```
CallableStatement cstmt = null;
try {
   String SQL = "{call getEmpName (?, ?)}";
   cstmt = conn.prepareCall (SQL);
   . . .
}
catch (SQLException e) {
   . . .
}
finally {
   . . .
}
```

Η μεταβλητή String SQL, αντιπροσωπεύει την αποθηκευμένη διαδικασία, με εντολές κράτησης παραμέτρων.

Η χρήση των αντικειμένων CallableStatement μοιάζει πολύ με τη χρήση των αντικειμένων PreparedStatement. Πρέπει να συνδέσετε τιμές σε όλες τις παραμέτρους πριν εκτελέσετε τη δήλωση ή αλλιώς θα λάβετε μια SQLException.

Αν έχετε παραμέτρους IN, ακολουθήστε τους ίδιους κανόνες και τεχνικές που ισχύουν για ένα αντικείμενο PreparedStatement. χρησιμοποιήστε τη μέθοδο setXXX() που αντιστοιχεί στον τύπο δεδομένων Java που δεσμεύετε.

Όταν χρησιμοποιείτε παραμέτρους OUT και INOUT, πρέπει να χρησιμοποιήσετε μια επιπλέον μέθοδο CallableStatement, registerOutParameter(). Η μέθοδος registerOutParameter() συνδέει τον τύπο δεδομένων JDBC με τον τύπο δεδομένων που αναμένεται να επιστρέψει η αποθηκευμένη διαδικασία.

Αφού καλέσετε την αποθηκευμένη σας διαδικασία, ανακτάτε την τιμή από την παράμετρο OUT με την κατάλληλη μέθοδο getXXX(). Αυτή η μέθοδος μεταφέρει την ανακτούμενη τιμή του τύπου SQL σε έναν τύπο δεδομένων Java.

6.7 Κλείσιμο αντικειμένου CallableStatement

Ακριβώς όπως κλείνετε άλλο αντικείμενο Statement, για τον ίδιο λόγο θα πρέπει επίσης να κλείσετε το αντικείμενο CallableStatement.

Μια απλή κλήση στη μέθοδο close() θα κάνει τη δουλειά. Εάν κλείσετε πρώτα το αντικείμενο Connection, θα κλείσει επίσης το αντικείμενο CallableStatement. Ωστόσο, θα πρέπει πάντα να κλείσετε ρητά το αντικείμενο CallableStatement για να διασφαλίσετε την σωστή εκκαθάριση. [7]

```
CallableStatement cstmt = null;
try {
   String SQL = "{call getEmpName (?, ?)}";
   cstmt = conn.prepareCall (SQL);
   . . .
}
catch (SQLException e) {
   . . .
}
finally {
   cstmt.close();}
```

6.8 Παράδειγμα αντικειμένου JDBC - CallableStatement Object

Ακολουθεί το παράδειγμα, το οποίο χρησιμοποιεί το CallableStatement μαζί με την ακόλουθη **getEmpName()** MySQL αποθηκευμένη διαδικασία.

Βεβαιωθείτε ότι έχετε δημιουργήσει αυτήν την αποθηκευμένη διαδικασία στη βάση δεδομένων EMP. Μπορείτε να χρησιμοποιήσετε το MySQL Query Browser για να το ολοκληρώσετε. [7]

```
DELIMITER $$


DROP PROCEDURE IF EXISTS `EMP`.`getEmpName` $$
CREATE PROCEDURE `EMP`.`getEmpName`
   (IN EMP_ID INT, OUT EMP_FIRST VARCHAR(255))
BEGIN
   SELECT first INTO EMP_FIRST
   FROM Employees
   WHERE ID = EMP_ID;
END $$


DELIMITER ;
```

Αυτός ο κώδικας δείγματος έχει γραφτεί με βάση το περιβάλλον και τη βάση δεδομένων που έγινε στα προηγούμενα κεφάλαια.

Αντιγράψτε και περάστε από το ακόλουθο παράδειγμα στο JDBCExample.java, μεταγλωττίστε και εκτελέστε ως εξής : [7]

```
//STEP 1. Import required packages
import java.sql.*;


public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";


   //  Database credentials
   static final String USER = "username";
```

```
   static final String PASS = "password";


   public static void main(String[] args) {
   Connection conn = null;
   CallableStatement stmt = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");


      //STEP 3: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);


      //STEP 4: Execute a query
      System.out.println("Creating statement...");
      String sql = "{call getEmpName (?, ?)}";
      stmt = conn.prepareCall(sql);


      //Bind IN parameter first, then bind OUT parameter
      int empID = 102;
      stmt.setInt(1, empID); // This would set ID as 102
      // Because second parameter is OUT so register it
      stmt.registerOutParameter(2, java.sql.Types.VARCHAR);


      //Use execute method to run stored procedure.
      System.out.println("Executing stored procedure..." );
      stmt.execute();


      //Retrieve employee name with getXXX method
      String empName = stmt.getString(2);
      System.out.println("Emp Name with ID:" +
               empID + " is " + empName);
      stmt.close();
```

```
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main
}//end JDBCExample
```

Τώρα ας καταρτίσουμε το παραπάνω παράδειγμα ως εξής :

```
C:\>javac JDBCExample.java
C:\>
```

Όταν εκτελείτε το **JDBCExample** , παράγει το ακόλουθο αποτέλεσμα :

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
Executing stored procedure...
```

```
Emp Name with ID:102 is Zaid

Goodbye!

C:\>
```

ΚΕΦΑΛΑΙΟ 7

JDBC - Σύνολα Αποτελεσμάτων

7.1 JDBC - Σύνολα Αποτελεσμάτων

Οι δηλώσεις SQL που διαβάζουν δεδομένα από ένα ερώτημα βάσης δεδομένων, επιστρέφουν τα δεδομένα σε ένα σύνολο αποτελεσμάτων. Η εντολή SELECT είναι ο τυπικός τρόπος για να επιλέξετε σειρές από μια βάση δεδομένων και να τις προβάλετε σε ένα σύνολο αποτελεσμάτων. Η διεπαφή **java.sql.ResultSet** αντιπροσωπεύει το σύνολο αποτελεσμάτων ενός ερωτήματος βάσης δεδομένων.

Ένα αντικείμενο ResultSet διατηρεί έναν δρομέα που δείχνει την τρέχουσα σειρά στο σύνολο αποτελεσμάτων. Ο όρος "σύνολο αποτελεσμάτων" αναφέρεται στα δεδομένα γραμμών και στηλών που περιέχονται σε ένα αντικείμενο ResultSet.

Οι μέθοδοι της διεπαφής ResultSet μπορούν να ταξινομηθούν σε τρεις κατηγορίες :

- **Navigational methods:** Χρησιμοποιείται για τη μετακίνηση του δρομέα.
- **Get methods:** Χρησιμοποιείται για την προβολή των δεδομένων στις στήλες της τρέχουσας σειράς που υποδεικνύονται από το δρομέα.
- **Update methods:** Χρησιμοποιείται για την ενημέρωση των δεδομένων στις στήλες της τρέχουσας σειράς. Οι ενημερώσεις μπορούν στη συνέχεια να ενημερωθούν και στην υποκείμενη βάση δεδομένων.

Ο δρομέας είναι κινητός με βάση τις ιδιότητες του ResultSet. Αυτές οι ιδιότητες ορίζονται όταν δημιουργείται το αντίστοιχο Statment που δημιουργεί το ResultSet.

Το JDBC παρέχει τις ακόλουθες μεθόδους σύνδεσης για τη δημιουργία δηλώσεων με το επιθυμητό αποτέλεσμα ResultSet :

- **createStatement(int RSType, int RSConcurrency);**
- **prepareStatement(String SQL, int RSType, int RSConcurrency);**
- **prepareCall(String sql, int RSType, int RSConcurrency);**

Το πρώτο όρισμα υποδεικνύει τον τύπο ενός αντικειμένου ResultSet και το δεύτερο όρισμα είναι μία από τις δύο σταθερές ResultSet για να καθορίσετε εάν ένα σετ αποτελεσμάτων είναι μόνο για ανάγνωση ή μπορεί να ενημερωθεί. [7]

7.2 Τύπος αποτελέσματος

Οι πιθανοί τύποι RSType δίνονται παρακάτω. Εάν δεν καθορίσετε κάποιο τύπο ResultSet, θα λάβετε αυτόματα έναν τύπο που είναι TYPE_FORWARD_ONLY.

Πίνακας 7.1: Τύποι RSType [6]

Τύπος	**Περιγραφή**
ResultSet.TYPE_FORWARD_ONLY	Ο κέρσορας μπορεί να προχωρήσει μόνο στο σύνολο αποτελεσμάτων.
ResultSet.TYPE_SCROLL_INSENSITIVE	Ο δρομέας μπορεί να μετακινηθεί προς τα εμπρός και προς τα πίσω και το σύνολο αποτελεσμάτων δεν είναι ευαίσθητο στις αλλαγές που πραγματοποίησαν άλλοι στη βάση δεδομένων που συμβαίνουν μετά τη δημιουργία του συνόλου αποτελεσμάτων.
ResultSet.TYPE_SCROLL_SENSITIVE.	Ο δρομέας μπορεί να μετακινηθεί προς τα εμπρός και προς τα πίσω και το σύνολο αποτελεσμάτων είναι ευαίσθητο στις αλλαγές που πραγματοποίησαν άλλοι στη βάση δεδομένων που συνέβησαν μετά τη δημιουργία του συνόλου αποτελεσμάτων.

7.3 Συναλλαγή του ResultSet

Τα πιθανά RSConcurrency δίνονται παρακάτω. Εάν δεν καθορίσετε κανένα τύπο Concurrency, θα λάβετε αυτόματα ένα που είναι CONCUR_READ_ONLY.

Πίνακας 7.2: Τύποι RSConcurrency [6]

Concurrency	Περιγραφή
ResultSet.CONCUR_READ_ONLY	Δημιουργεί ένα σύνολο αποτελεσμάτων μόνο για ανάγνωση. Αυτή είναι η προεπιλογή.
ResultSet.CONCUR_UPDATABLE	Δημιουργεί ένα σύνολο αποτελεσμάτων με δυνατότητα ενημέρωσης.

Όλα τα παραδείγματα που έχουν υποθεί μέχρι τώρα μπορούν να γραφτούν ως εξής, τα οποία προετοιμάζουν ένα αντικείμενο της Statment για να δημιουργήσουν ένα αντικείμενό ResultSet μόνο προς τα εμπρός : [7]

```
try {
   Statement stmt = conn.createStatement(
                            ResultSet.TYPE_FORWARD_ONLY,
                            ResultSet.CONCUR_READ_ONLY);
}
catch(Exception ex) {
   ....
}
finally {
   ....
}
```

7.3 Πλοήγηση σε ένα Result Set

Υπάρχουν πολλές μέθοδοι στη διεπαφή ResultSet που περιλαμβάνουν τη μετακίνηση του δρομέα, συμπεριλαμβανομένων :

Πίνακας 7.3: Μέθοδοι διεπαφής ResultSet για τη μετακίνηση του δρομέα [6]

α/α	**Μέθοδοι & περιγραφή**
1	public void beforeFirst() throws SQLException Μετακινεί το δρομέα λίγο πριν την πρώτη σειρά.
2	public void afterLast() throws SQLException Μετακινεί το δρομέα αμέσως μετά την τελευταία σειρά.
3	public boolean first() throws SQLException Μετακινεί το δρομέα στην πρώτη σειρά.
4	public void last() throws SQLException Μετακινεί το δρομέα στην τελευταία σειρά.
5	public boolean absolute(int row) throws SQLException Μετακινεί το δρομέα σε συγκεκριμένη σειρά.
6	public boolean relative(int row) throws SQLException Μετακινεί το δρομέα στον δεδομένο αριθμό γραμμών προς τα εμπρός ή προς τα πίσω, από εκεί όπου δείχνει προς το παρόν.
7	public boolean previous() throws SQLException Μετακινεί το δρομέα στην προηγούμενη σειρά. Αυτή η μέθοδος επιστρέφει το ψευδές αν η προηγούμενη σειρά είναι εκτός σειράς αποτελεσμάτων.
8	public boolean next() throws SQLException Μετακινεί το δρομέα στην επόμενη σειρά. Αυτή η μέθοδος επιστρέφει ψευδής αν δεν υπάρχουν περισσότερες σειρές στο σύνολο αποτελεσμάτων.
9	public int getRow() throws SQLException Επιστρέφει τον αριθμό σειράς στον οποίο δείχνει ο δρομέας.
10	public void moveToInsertRow() throws SQLException

	Μετακινεί το δρομέα σε μια ειδική σειρά στο σύνολο αποτελεσμάτων που μπορεί να χρησιμοποιηθεί για την εισαγωγή μιας νέας γραμμής στη βάση δεδομένων.
11	public void moveToCurrentRow() throws SQLException Μετακινεί το δρομέα στην τρέχουσα σειρά αν ο δρομέας βρίσκεται στη γραμμή εισαγωγής. διαφορετικά, αυτή η μέθοδος δεν κάνει τίποτα

7.4 Προβολή ενός συνόλου αποτελεσμάτων

Η διεπαφή ResultSet περιέχει δεκάδες μεθόδους για τη λήψη των δεδομένων της τρέχουσας σειράς.

Υπάρχει μια μέθοδος get για κάθε έναν από τους πιθανούς τύπους δεδομένων και δύο εκδόσεις :

- Μια που έχει όνομα στήλης.
- Μια που παίρνει έναν δείκτη στήλης.

Για παράδειγμα, εάν η στήλη που σας ενδιαφέρει για προβολή περιέχει ένα int, πρέπει να χρησιμοποιήσετε μία από τις μεθόδους getInt() του ResultSet :

Πίνακας 7.4: Μέθοδοι διεπαφής ResultSet για τη λήψη των δεδομένων [7]

α/α	**Μέθοδοι & περιγραφή**
1	public int getInt(String columnName) throws SQLException Επιστρέφει το int στην τρέχουσα σειρά στη στήλη columnName.
2	public int getInt(int columnIndex) throws SQLException Επιστρέφει το int στην τρέχουσα σειρά στο καθορισμένο δείκτη στήλης. Ο δείκτης στήλης ξεκινά από το 1, δηλαδή η πρώτη στήλη μιας σειράς είναι 1, η δεύτερη στήλη μιας σειράς είναι 2 και ούτω καθεξής.

Παρομοίως, υπάρχουν μέθοδοι get στη διεπαφή ResultSet για καθέναν από τους οκτώ πρωτόγονες τύπους Java καθώς και συνηθισμένους τύπους όπως java.lang.String, java.lang.Object και java.net.URL.

Υπάρχουν επίσης μέθοδοι για τη λήψη SQL τύπων δεδομένων java.sql.Date, java.sql.Time, java.sql.TimeStamp, java.sql.Clob και java.sql.Blob. Ελέγξτε την τεκμηρίωση για περισσότερες πληροφορίες σχετικά με τη χρήση αυτών των τύπων δεδομένων SQL.

7.5 Ενημέρωση ενός συνόλου αποτελεσμάτων

Η διεπαφή ResultSet περιέχει μια συλλογή μεθόδων ενημέρωσης για την ενημέρωση των δεδομένων ενός συνόλου αποτελεσμάτων.

Όπως και με τις μεθόδους get, υπάρχουν δύο μέθοδοι ενημέρωσης για κάθε τύπο δεδομένων :

- Μια που έχει όνομα στήλης.
- Μια που παίρνει έναν δείκτη στήλης.

Για παράδειγμα, για να ενημερώσετε μια στήλη String της τρέχουσας σειράς ενός συνόλου αποτελεσμάτων, θα χρησιμοποιήσετε μία από τις ακόλουθες μεθόδους updateString().

Πίνακας 7.5: Μέθοδοι διεπαφής ResultSet για τη ενημέρωση των δεδομένων [7]

α/α	**Μέθοδοι & περιγραφή**
1	public void updateString(int columnIndex, String s) throws SQLException Αλλάζει τη συμβολοσειρά στη συγκεκριμένη στήλη στην τιμή s.
2	public void updateString(String columnName, String s) throws SQLException Παρόμοια με την προηγούμενη μέθοδο, εκτός από το ότι η στήλη καθορίζεται από το όνομά της αντί του δείκτη.

Υπάρχουν μέθοδοι ενημέρωσης για τους οκτώ τύπους πρωτόγονων δεδομένων, καθώς και για το στοιχείο String, Object, URL και τους τύπους δεδομένων SQL στο πακέτο java.sql.

Η ενημέρωση μιας σειράς στο σύνολο αποτελεσμάτων αλλάζει τις στήλες της τρέχουσας σειράς στο αντικείμενο ResultSet, αλλά όχι στην υποκείμενη βάση δεδομένων. Για να ενημερώσετε τις αλλαγές σας στη σειρά στη βάση δεδομένων, πρέπει να χρησιμοποιήσετε μία από τις ακόλουθες μεθόδους.

Πίνακας 7.6: Μέθοδοι διεπαφής ResultSet για τη ενημέρωση των δεδομένων στη εκάστοτε γραμμή της βάσης δεδομένων [7]

α/α	**Μέθοδοι & περιγραφή**
1	public void updateRow() Ενημερώνει την τρέχουσα σειρά ενημερώνοντας την αντίστοιχη γραμμή στη βάση δεδομένων.
2	public void deleteRow() Διαγράφει την τρέχουσα σειρά από τη βάση δεδομένων
3	public void refreshRow() Ανανεώνει τα δεδομένα στο σύνολο αποτελεσμάτων ώστε να αντανακλούν τυχόν πρόσφατες αλλαγές στη βάση δεδομένων.
4	public void cancelRowUpdates() Ακυρώνει τυχόν ενημερώσεις στην τρέχουσα σειρά.
5	public void insertRow() Εισάγει μια σειρά στη βάση δεδομένων. Αυτή η μέθοδος μπορεί να χρησιμοποιηθεί μόνο όταν ο δρομέας δείχνει στη γραμμή εισαγωγής.

7.6 JDBC - Ενημέρωση ενός συνόλου αποτελεσμάτων - Παράδειγμα

Ακολουθεί το παράδειγμα, το οποίο κάνει χρήση των αποτελεσμάτων **ResultSet.CONCUR_UPDATABLE** και **ResultSet.TYPE_SCROLL_INSENSITIVE**. Αυτό το παράδειγμα θα εξηγούσε τη λειτουργία **INSERT**, **UPDATE** και **DELETE** σε ένα table της SQL.

Πρέπει να σημειωθεί ότι οι πίνακες στους οποίους εργάζεστε πρέπει να έχουν ρυθμιστεί σωστά το πρωτεύον κλειδί.

Αυτός ο κώδικας δείγματος έχει γραφτεί με βάση το περιβάλλον και τη βάση δεδομένων που έγινε στα προηγούμενα κεφάλαια.

Αντιγράψτε και περάστε από το ακόλουθο παράδειγμα στο **JDBCExample.java,** μεταγλωττίστε και εκτελέστε ως εξής : [7]

```
//STEP 1. Import required packages
import java.sql.*;


public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";


   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";


 public static void main(String[] args) {
   Connection conn = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");


      //STEP 3: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);


      //STEP 4: Execute a query to create statment with
      // required arguments for RS example.
      System.out.println("Creating statement...");
      Statement stmt = conn.createStatement(
                           ResultSet.TYPE_SCROLL_INSENSITIVE,
                           ResultSet.CONCUR_UPDATABLE);
     //STEP 5: Execute a query
```

```
      String sql = "SELECT id, first, last, age FROM Employees";
      ResultSet rs = stmt.executeQuery(sql);


      System.out.println("List result set for reference....");
      printRs(rs);


      //STEP 6: Loop through result set and add 5 in age
      //Move to BFR postion so while-loop works properly
      rs.beforeFirst();
      //STEP 7: Extract data from result set
      while(rs.next()){
         //Retrieve by column name
         int newAge = rs.getInt("age") + 5;
         rs.updateDouble( "age", newAge );
         rs.updateRow();
      }
      System.out.println("List result set showing new ages...");
      printRs(rs);
      // Insert a record into the table.
      //Move to insert row and add column data with updateXXX()
      System.out.println("Inserting a new record...");
      rs.moveToInsertRow();
      rs.updateInt("id",104);
      rs.updateString("first","John");
      rs.updateString("last","Paul");
      rs.updateInt("age",40);
      //Commit row
      rs.insertRow();


      System.out.println("List result set showing new set...");
      printRs(rs);


      // Delete second record from the table.
```

```
      // Set position to second record first
      rs.absolute( 2 );
      System.out.println("List the record before deleting...");
      //Retrieve by column name
      int id  = rs.getInt("id");
      int age = rs.getInt("age");
      String first = rs.getString("first");
      String last = rs.getString("last");

      //Display values
      System.out.print("ID: " + id);
      System.out.print(", Age: " + age);
      System.out.print(", First: " + first);
      System.out.println(", Last: " + last);

     //Delete row
      rs.deleteRow();
      System.out.println("List result set after \
                                deleting one records...");
      printRs(rs);

      //STEP 8: Clean-up environment
      rs.close();
      stmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
```

```
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main


   public static void printRs(ResultSet rs) throws SQLException{
      //Ensure we start with first row
      rs.beforeFirst();
      while(rs.next()){
         //Retrieve by column name
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String first = rs.getString("first");
         String last = rs.getString("last");


         //Display values
         System.out.print("ID: " + id);
         System.out.print(", Age: " + age);
         System.out.print(", First: " + first);
         System.out.println(", Last: " + last);
      }
      System.out.println();
   }//end printRs()
}//end JDBCExample
```

Τώρα ας καταρτίσουμε το παραπάνω παράδειγμα ως εξής :

```
C:\>javac JDBCExample.java

C:\>
```

Όταν εκτελείτε το JDBCExample , παράγει το ακόλουθο αποτέλεσμα :

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
List result set for reference....
ID: 100, Age: 33, First: Zara, Last: Ali
ID: 101, Age: 40, First: Mahnaz, Last: Fatma
ID: 102, Age: 50, First: Zaid, Last: Khan
ID: 103, Age: 45, First: Sumit, Last: Mittal

List result set showing new ages...
ID: 100, Age: 38, First: Zara, Last: Ali
ID: 101, Age: 45, First: Mahnaz, Last: Fatma
ID: 102, Age: 55, First: Zaid, Last: Khan
ID: 103, Age: 50, First: Sumit, Last: Mittal

Inserting a new record...
List result set showing new set...
ID: 100, Age: 38, First: Zara, Last: Ali
ID: 101, Age: 45, First: Mahnaz, Last: Fatma
ID: 102, Age: 55, First: Zaid, Last: Khan
ID: 103, Age: 50, First: Sumit, Last: Mittal
ID: 104, Age: 40, First: John, Last: Paul

List the record before deleting...
ID: 101, Age: 45, First: Mahnaz, Last: Fatma
List result set after deleting one records...
ID: 100, Age: 38, First: Zara, Last: Ali
ID: 102, Age: 55, First: Zaid, Last: Khan
ID: 103, Age: 50, First: Sumit, Last: Mittal
ID: 104, Age: 40, First: John, Last: Paul

Goodbye!
C:\>
```

ΚΕΦΑΛΑΙΟ 8

JDBC - Τύποι Δεδομένων

8.1 JDBC -Τύποι Δεδομένων

Το πρόγραμμα οδήγησης JDBC μετατρέπει τον τύπο δεδομένων της Java στον κατάλληλο τύπο για τη JDBC, πριν τα αποστείλει στη βάση δεδομένων. Χρησιμοποιεί μια προεπιλεγμένη αντιστοίχιση για τους περισσότερους τύπους δεδομένων. Για παράδειγμα, ένα Java int μετατρέπεται σε SQL INTEGER. Οι προεπιλεγμένες αντιστοιχίσεις δημιουργήθηκαν για να παρέχουν συνοχή μεταξύ των οδηγών.

Ο παρακάτω πίνακας συνοψίζει τον προεπιλεγμένο τύπο δεδομένων JDBC στον οποίο μετατρέπεται ο τύπος δεδομένων Java, όταν καλείτε τη μέθοδο setXXX() του αντικειμένου PreparedStatement ή CallableStatement ή της μεθόδου ResultSet.updateXXX().

Πίνακας8.1: Προεπιλεγμένος τύπος δεδομένων JDBC όπου μετατρέπετε σε τύπο δεδομένων Java [7]

SQL	JDBC / Java	setXXX	updateXXX
VARCHAR	java.lang.String	setString	updateString
CHAR	java.lang.String	setString	updateString
LONGVARCHAR	java.lang.String	setString	updateString
BIT	boolean	setBoolean	updateBoolean
NUMERIC	java.math.BigDecimal	setBigDecimal	updateBigDecimal
TINYINT	byte	setByte	updateByte

SMALLINT	short	setShort	updateShort
INTEGER	int	setInt	updateInt
BIGINT	long	setLong	updateLong
REAL	float	setFloat	updateFloat
FLOAT	float	setFloat	updateFloat
DOUBLE	double	setDouble	updateDouble
VARBINARY	byte []	setBytes	updateBytes
BINARY	byte []	setBytes	updateBytes
DATE	java.sql.Date	setDate	updateDate
TIME	java.sql.Time	setTime	updateTime
TIMESTAMP	java.sql.Timestamp	setTimestamp	updateTimestamp
CLOB	java.sql.Clob	setClob	updateClob
BLOB	java.sql.Blob	setBlob	updateBlob
ARRAY	java.sql.Array	setARRAY	updateRARAY
REF	java.sql.Ref	SetRef	updateRef
STRUCT	java.sql.Struct	SetStruct	updateStruct

Το JDBC 3.0 έχει ενισχυμένη υποστήριξη για τους τύπους δεδομένων BLOB, CLOB, ARRAY και REF. Το αντικείμενο ResultSet έχει πλέον τις μεθόδους

updateBLOB(), updateCLOB(), updateArray() και updateRef() που σας δίνουν τη δυνατότητα να χειρίζεστε άμεσα τα αντίστοιχα δεδομένα στο διακομιστή.

Οι μέθοδοι setXXX() και updateXXX() σάς επιτρέπουν να μετατρέπετε συγκεκριμένους τύπους Java σε συγκεκριμένους τύπους δεδομένων JDBC. Οι μέθοδοι, setObject() και updateObject(), σας επιτρέπουν να χαρτογραφείτε σχεδόν οποιοδήποτε τύπο Java σε έναν τύπο δεδομένων JDBC.

Το αντικείμενο ResultSet παρέχει την αντίστοιχη μέθοδο getXXX() για κάθε τύπο δεδομένων για να ανακτήσει την τιμή της στήλης. Κάθε μέθοδος μπορεί να χρησιμοποιηθεί με το όνομα της στήλης ή με την κανονική της θέση.

8.2 Τύποι δεδομένων ημερομηνίας και ώρας

Η κλάση java.sql.Date αντιστοιχεί στον τύπο SQL DATE και οι κλάσεις java.sql.Time και java.sql.Timestamp αντιστοιχούν στους τύπους δεδομένων SQL TIME και SQL TIMESTAMP αντίστοιχα.

Το παρακάτω παράδειγμα δείχνει με ποιον τρόπο οι μορφές ημερομηνίας και ώρας μορφοποιούν τις τυπικές τιμές ημερομηνίας και ώρας της Java για να ταιριάζουν με τις απαιτήσεις τύπου δεδομένων SQL. [7]

```
import java.sql.Date;
import java.sql.Time;
import java.sql.Timestamp;
import java.util.*;

public class SqlDateTime {
   public static void main(String[] args) {
      //Get standard date and time
      java.util.Date javaDate = new java.util.Date();
      long javaTime = javaDate.getTime();
      System.out.println("The Java Date is:" +
            javaDate.toString());

      //Get and display SQL DATE
      java.sql.Date sqlDate = new java.sql.Date(javaTime);
      System.out.println("The SQL DATE is: " +
```

```
            sqlDate.toString());


      //Get and display SQL TIME
      java.sql.Time sqlTime = new java.sql.Time(javaTime);
      System.out.println("The SQL TIME is: " +
            sqlTime.toString());
      //Get and display SQL TIMESTAMP
      java.sql.Timestamp sqlTimestamp =
      new java.sql.Timestamp(javaTime);
      System.out.println("The SQL TIMESTAMP is: " +
            sqlTimestamp.toString());
     }//end main
}//end SqlDateTime
```

Τώρα ας καταρτίσουμε το παραπάνω παράδειγμα ως εξής :

```
C:\>javac SqlDateTime.java
C:\>
```

Όταν εκτελείτε το **JDBCExample** , παράγει το ακόλουθο αποτέλεσμα :

```
C:\>java SqlDateTime
The Java Date is:Tue Aug 18 13:46:02 GMT+04:00 2009
The SQL DATE is: 2009-08-18
The SQL TIME is: 13:46:02
The SQL TIMESTAMP is: 2009-08-18 13:46:02.828
C:\>
```

8.3 Διαχείριση τιμών NULL

Η χρήση από τη SQL των τιμών NULL και η χρήση της null από τη Java είναι διαφορετικές έννοιες. Έτσι, για να χειριστείτε τις τιμές SQL NULL στη Java, υπάρχουν τρεις τακτικές που μπορείτε να χρησιμοποιήσετε :

- Αποφύγετε τη χρήση μεθόδων getXXX() που επιστρέφουν primitive τύπους δεδομένων.
- Χρησιμοποιήστε τις κατηγορίες περιτύλιξης για primitive τύπους δεδομένων και χρησιμοποιήστε τη μέθοδο wasNull() του αντικειμένου ResultSet για να ελέγξετε αν η μεταβλητή κλάσης περιτύλιξης που έλαβε την τιμή που επέστρεψε με τη μέθοδο getXXX() θα πρέπει να οριστεί σε null.
- Χρησιμοποιήστε πρωτότυπους τύπους δεδομένων και τη μέθοδο wasNull() του αντικειμένου ResultSet για να ελέγξετε αν η primitive μεταβλητή που έλαβε την τιμή που επέστρεψε με τη μέθοδο getXXX() θα πρέπει να οριστεί σε μια αποδεκτή τιμή που έχετε επιλέξει να αντιπροσωπεύει μια τιμή NULL.

Εδώ είναι ένα παράδειγμα για να χειριστεί μια τιμή NULL : [7]

```
Statement stmt = conn.createStatement( );

String sql = "SELECT id, first, last, age FROM Employees";

ResultSet rs = stmt.executeQuery(sql);


int id = rs.getInt(1);

if( rs.wasNull( ) ) {

   id = 0;

}
```

ΚΕΦΑΛΑΙΟ 9

JDBC – Συναλλαγές

9.1 JDBC - Συναλλαγές

Εάν η σύνδεσή σας JDBC είναι σε λειτουργία αυτόματης δέσμευσης, η οποία είναι από προεπιλογή, τότε κάθε δήλωση SQL δεσμεύεται στη βάση δεδομένων κατά την ολοκλήρωσή της.

Αυτό μπορεί να είναι καλό για απλές εφαρμογές, αλλά υπάρχουν τρεις λόγοι για τους οποίους ίσως θελήσετε να απενεργοποιήσετε την αυτόματη διεκπεραίωση και να διαχειριστείτε τις δικές σας συναλλαγές :

- Για να αυξήσετε την απόδοση.
- Για να διατηρηθεί η ακεραιότητα των επιχειρηματικών διαδικασιών.
- Χρήση κατανεμημένων συναλλαγών.

Οι συναλλαγές σας επιτρέπουν να ελέγχετε αν και πότε εφαρμόζονται οι αλλαγές στη βάση δεδομένων. Αντιμετωπίζει μια μεμονωμένη δήλωση SQL ή μια ομάδα δηλώσεων SQL ως μία λογική μονάδα και εάν αποτύχει η οποιαδήποτε δήλωση, αποτυγχάνει ολόκληρη η συναλλαγή.

Για να ενεργοποιήσετε την υποστήριξη χειρωνακτικών συναλλαγών αντί για τη λειτουργία αυτόματης δέσμευσης που χρησιμοποιεί το πρόγραμμα οδήγησης JDBC από προεπιλογή, χρησιμοποιήστε τη μέθοδο setAutoCommit() του αντικειμένου σύνδεσης . Εάν περάσετε ένα boolean false στο **setAutoCommit(),** απενεργοποιείτε την αυτόματη διεκπεραίωση. Μπορείτε να περάσετε ένα boolean true για να το ενεργοποιήσετε ξανά.

Για παράδειγμα, εάν έχετε ένα αντικείμενο Connection με όνομα conn, κωδικοποιήστε τα παρακάτω για να απενεργοποιήσετε την αυτόματη διεκπεραίωση : [6]

```
conn.setAutoCommit(false);
```

9.2 Επαναφορά και επαναφορά

Μόλις τελειώσετε με τις αλλαγές σας και θέλετε να δεσμευτείτε τις αλλαγές, στη συνέχεια, καλέστε τη μέθοδο **commit()** στο αντικείμενο σύνδεσης ως εξής:

```
conn.commit();
```

Διαφορετικά, για να επαναφέρετε τις ενημερώσεις στη βάση δεδομένων που έγιναν χρησιμοποιώντας τη σύνδεση με όνομα conn, χρησιμοποιήστε τον ακόλουθο κώδικα :

```
conn.rollback();
```

Το παρακάτω παράδειγμα δείχνει τη χρήση ενός αντικειμένου δέσμευσης και επαναφοράς : [7]

```
try{
   //Assume a valid connection object conn
   conn.setAutoCommit(false);
   Statement stmt = conn.createStatement();

   String SQL = "INSERT INTO Employees  " +
                "VALUES (106, 20, 'Rita', 'Tez')";
   stmt.executeUpdate(SQL);
   //Submit a malformed SQL statement that breaks
   String SQL = "INSERTED IN Employees  " +
                "VALUES (107, 22, 'Sita', 'Singh')";
   stmt.executeUpdate(SQL);
   // If there is no error.
   conn.commit();
}catch(SQLException se){
   // If there is any error.
   conn.rollback();
}
```

Σε αυτήν την περίπτωση, καμία από τις παραπάνω δηλώσεις INSERT δεν θα επέτρεπε την επιτυχία και όλα θα επανέλθουν στο προηγούμενο στάδιο.

9.3 Χρήση σημείων αποθήκευσης

Η νέα διασύνδεση JDBC 3.0 Savepoint σας δίνει τον πρόσθετο έλεγχο συναλλαγών. Τα περισσότερα σύγχρονα DBMS, υποστηρίζουν σημεία αποθήκευσης σε περιβάλλοντα όπως το PL/SQL της Oracle.

Όταν ορίσετε ένα σημείο αποθήκευσης, ορίζετε ένα λογικό σημείο επαναφοράς σε μια συναλλαγή. Εάν εμφανιστεί ένα σφάλμα μετά από ένα σημείο αποθήκευσης, μπορείτε να χρησιμοποιήσετε τη μέθοδο επαναφοράς για να αναιρέσετε όλες τις αλλαγές ή μόνο τις αλλαγές που έγιναν μετά το σημείο αποθήκευσης.

Το αντικείμενο Connection έχει δύο νέες μεθόδους που σας βοηθούν να διαχειριστείτε σημεία αποθήκευσης :

- **setSavepoint(String savepointName):** Ορίζει ένα νέο σημείο αποθήκευσης. Επιστρέφει επίσης ένα αντικείμενο Savepoint.
- **releaseSavepoint(Savepoint savepointName):** Διαγράφει ένα σημείο αποθήκευσης. Παρατηρήστε ότι απαιτείται ένα αντικείμενο Savepoint ως παράμετρος. Αυτό το αντικείμενο είναι συνήθως ένα σημείο αποθήκευσης που παράγεται από τη μέθοδο setSavepoint().[6]

Υπάρχει μια μέθοδος rollback(String savepointName) , η οποία επαναφέρει την εργασία στο καθορισμένο σημείο αποθήκευσης.

Το ακόλουθο παράδειγμα απεικονίζει τη χρήση ενός αντικειμένου Savepoint: [7]

```
try{
   //Assume a valid connection object conn
   conn.setAutoCommit(false);
   Statement stmt = conn.createStatement();


   //set a Savepoint
   Savepoint savepoint1 = conn.setSavepoint("Savepoint1");
   String SQL = "INSERT INTO Employees " +
                "VALUES (106, 20, 'Rita', 'Tez')";
   stmt.executeUpdate(SQL);
   //Submit a malformed SQL statement that breaks
```

```
   String SQL = "INSERTED IN Employees " +
                "VALUES (107, 22, 'Sita', 'Tez')";
   stmt.executeUpdate(SQL);
   // If there is no error, commit the changes.
   conn.commit();

}catch(SQLException se){
   // If there is any error.
   conn.rollback(savepoint1);
}
```

Σε αυτήν την περίπτωση, καμία από τις παραπάνω δηλώσεις INSERT δεν θα επέτρεπε την επιτυχία και όλα θα επανέλθουν.

9.3 JDBC - setSavepoint, releaseSavepoint - Παράδειγμα

Ακολουθεί το παράδειγμα, το οποίο χρησιμοποιεί το **setSavepoint** και την **rollback** .

Αντιγράψτε και περάστε από το ακόλουθο παράδειγμα στο JDBCExample.java, μεταγλωττίστε και εκτελέστε ως εξής : [7]

```
//STEP 1. Import required packages
import java.sql.*;

public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";

public static void main(String[] args) {
```

```
Connection conn = null;
Statement stmt = null;
try{
   //STEP 2: Register JDBC driver
   Class.forName("com.mysql.jdbc.Driver");


   //STEP 3: Open a connection
   System.out.println("Connecting to database...");
   conn = DriverManager.getConnection(DB_URL,USER,PASS);


   //STEP 4: Set auto commit as false.
   conn.setAutoCommit(false);


   //STEP 5: Execute a query to delete statment with
   // required arguments for RS example.
   System.out.println("Creating statement...");
   stmt = conn.createStatement();


        //STEP 6: Now list all the available records.
   String sql = "SELECT id, first, last, age FROM Employees";
   ResultSet rs = stmt.executeQuery(sql);
   System.out.println("List result set for reference....");
   printRs(rs);


   // STEP 7: delete rows having ID grater than 104
   // But save point before doing so.
   Savepoint savepoint1 = conn.setSavepoint("ROWS_DELETED_1");
   System.out.println("Deleting row....");
   String SQL = "DELETE FROM Employees " +
                "WHERE ID = 110";
   stmt.executeUpdate(SQL);
   // oops... we deleted too wrong employees!
   //STEP 8: Rollback the changes afetr save point 2.
```

```
      conn.rollback(savepoint1);


    // STEP 9: delete rows having ID grater than 104
      // But save point before doing so.
      Savepoint savepoint2 = conn.setSavepoint("ROWS_DELETED_2");
      System.out.println("Deleting row....");
      SQL = "DELETE FROM Employees " +
                   "WHERE ID = 95";
      stmt.executeUpdate(SQL);


           //STEP 10: Now list all the available records.
      sql = "SELECT id, first, last, age FROM Employees";
      rs = stmt.executeQuery(sql);
      System.out.println("List result set for reference....");
      printRs(rs);


      //STEP 10: Clean-up environment
      rs.close();
      stmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
      // If there is an error then rollback the changes.
      System.out.println("Rolling back data here....");
           try{
                   if(conn!=null)
            conn.rollback();
      }catch(SQLException se2){
         se2.printStackTrace();
      }//end try


   }catch(Exception e){
```

```
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main


   public static void printRs(ResultSet rs) throws SQLException{
      //Ensure we start with first row
      rs.beforeFirst();
      while(rs.next()){
         //Retrieve by column name
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String first = rs.getString("first");
         String last = rs.getString("last");


         //Display values
         System.out.print("ID: " + id);
         System.out.print(", Age: " + age);
         System.out.print(", First: " + first);
```

```
         System.out.println(", Last: " + last);

      }

      System.out.println();

   }//end printRs()

}//end JDBCExample
```

Τώρα, ας καταρτίσουμε το παραπάνω παράδειγμα ως εξής :

```
C:\>javac JDBCExample.java

C:\>
```

Όταν εκτελείτε το **JDBCExample** , παράγει το ακόλουθο αποτέλεσμα :

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
List result set for reference....
ID: 95, Age: 20, First: Sima, Last: Chug
ID: 100, Age: 18, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 30, First: Sumit, Last: Mittal
ID: 110, Age: 20, First: Sima, Last: Chug

Deleting row....
Deleting row....
List result set for reference....
ID: 100, Age: 18, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 30, First: Sumit, Last: Mittal
ID: 110, Age: 20, First: Sima, Last: Chug

Goodbye!
C:\>
```

ΚΕΦΑΛΑΙΟ 10

JDBC – Διαχείριση Εξαιρέσεων

10.1 JDBC - Διαχείριση Εξαιρέσεων

Ο χειρισμός εξαιρέσεων σας επιτρέπει να χειρίζεστε εξαιρετικές συνθήκες, όπως τα προγραμματισμένα σφάλματα με ελεγχόμενο τρόπο.

Ο όρος που της εξαίρεσης όπου πραγματοποιείτε σημαίνει ότι η τρέχουσα εκτέλεση του προγράμματος σταματά και ο έλεγχος ανακατευθύνεται στην πλησιέστερη ισχύουσα ρήτρα σύλληψης. Εάν δεν υπάρχει ρήτρα εφαρμογής, η εκτέλεση του προγράμματος τελειώνει.

Το χειρισμό εξαιρέσεων στη JDBC είναι πολύ παρόμοιο με το χειρισμό εξαιρέσεων στη Java, αλλά για το JDBC, η πιο συνηθισμένη εξαίρεση που θα αντιμετωπίσετε είναι η **java.sql.SQLException**. [3]

10.2 Μέθοδοι SQLException

Μια SQLException μπορεί να συμβεί τόσο στο πρόγραμμα οδήγησης όσο και στη βάση δεδομένων. Όταν συμβαίνει μια τέτοια εξαίρεση, ένα αντικείμενο τύπου SQLException θα περάσει στη ρήτρα σύλληψης.

Το μεταβιβαζόμενο αντικείμενο SQLException έχει τις ακόλουθες διαθέσιμες μεθόδους για την ανάκτηση πρόσθετων πληροφοριών σχετικά με την εξαίρεση :

Πίνακας 10.1: Περιγραφή Μεθόδων SQLException [7]

Μέθοδος	Περιγραφή
getErrorCode()	Λαμβάνει τον αριθμό σφάλματος που σχετίζεται με την εξαίρεση.
getMessage()	Παίρνει το μήνυμα σφάλματος του προγράμματος οδήγησης JDBC για ένα σφάλμα που χειρίζεται το πρόγραμμα οδήγησης ή παίρνει τον αριθμό του σφάλματος της Oracle και μήνυμα για σφάλμα βάσης δεδομένων.

getSQLState()	Λαμβάνει τη συμβολοσειρά XOPEN SQLstate. Για σφάλμα του προγράμματος οδήγησης JDBC, δεν επιστρέφονται χρήσιμες πληροφορίες από αυτήν τη μέθοδο. Για σφάλμα της βάσης δεδομένων, επιστρέφεται ο πενταψήφιος κωδικός XOPEN SQLstate. Αυτή η μέθοδος μπορεί να επιστρέψει null.
getNextException()	Παρέχει το επόμενο αντικείμενο εξαίρεσης στην αλυσίδα εξαίρεσης.
printStackTrace()	Εκτυπώνει την τρέχουσα εξαίρεση, ή μπορεί να εκτοξευθεί, και είναι backtrace σε μια τυπική ροή λάθους.
printStackTrace (PrintStream s)	Εκτυπώνει την επιστροφή του στη ροή εκτύπωσης που καθορίζετε.
printStackTrace (PrintWriter w)	Εκτυπώνει την επιστροφή στον εκτυπωτή γραφής που καθορίζετε.

Χρησιμοποιώντας τις διαθέσιμες πληροφορίες από το αντικείμενο Εξαίρεση, μπορείτε να πάρετε μια εξαίρεση και να συνεχίσετε το πρόγραμμα κατάλληλα. Εδώ είναι η γενική μορφή ενός μπλοκ δοκιμής : [7]

```
try {
   // Your risky code goes between these curly braces!!!
}
catch(Exception ex) {
   // Your exception handling code goes between these
   // curly braces, similar to the exception clause
   // in a PL/SQL block.
}
finally {
   // Your must-always-be-executed code goes between these curly braces. Like closing
database connection.
}
```

10.2.1 Παράδειγμα

Μελετήστε το ακόλουθο παράδειγμα κώδικα για να κατανοήσετε τη χρήση της **try.... catch ... finally**. [7]

```
//STEP 1. Import required packages
import java.sql.*;


public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";


   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";


   public static void main(String[] args) {
   Connection conn = null;
   try{
      //STEP 2: Register JDBC driver
      Class.forName("com.mysql.jdbc.Driver");


      //STEP 3: Open a connection
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);


      //STEP 4: Execute a query
      System.out.println("Creating statement...");
      Statement stmt = conn.createStatement();
      String sql;
      sql = "SELECT id, first, last, age FROM Employees";
      ResultSet rs = stmt.executeQuery(sql);
```

```
      //STEP 5: Extract data from result set
      while(rs.next()){
         //Retrieve by column name
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String first = rs.getString("first");
         String last = rs.getString("last");

         //Display values
         System.out.print("ID: " + id);
         System.out.print(", Age: " + age);
         System.out.print(", First: " + first);
         System.out.println(", Last: " + last);
      }
      //STEP 6: Clean-up environment
      rs.close();
      stmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
```

```
   System.out.println("Goodbye!");
}//end main
}//end JDBCExample
```

Τώρα, ας καταρτίσουμε το παραπάνω παράδειγμα ως εξής :

```
C:\>javac JDBCExample.java
C:\>
```

Όταν εκτελείτε το **JDBCExample**, παράγει το ακόλουθο αποτέλεσμα αν δεν υπάρχει πρόβλημα, διαφορετικά το αντίστοιχο σφάλμα θα πιαστεί και θα εμφανιστεί μήνυμα σφάλματος :

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
ID: 100, Age: 18, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 28, First: Sumit, Last: Mittal
C:\>
```

Δοκιμάστε το παραπάνω παράδειγμα περνώντας λάθος όνομα βάσης δεδομένων ή λάθος όνομα χρήστη ή κωδικό πρόσβασης και ελέγξτε το αποτέλεσμα.

ΚΕΦΑΛΑΙΟ 11

JDBC – Επεξεργασία παρτίδας (Batch Processing)

11.1 JDBC - Επεξεργασία παρτίδας (Batch Processing)

Η Παρτίδα Επεξεργασίας (Batch Processing) σάς επιτρέπει να ομαδοποιείτε σχετικές δηλώσεις SQL σε μια παρτίδα και να τις υποβάλλετε με μία κλήση στη βάση δεδομένων.

Όταν στέλνετε πολλές δηλώσεις SQL στη βάση δεδομένων ταυτόχρονα, μειώνετε το ποσό των γενικών εξόδων επικοινωνίας, βελτιώνοντας έτσι την απόδοση.

- Τα προγράμματα οδήγησης JDBC δεν απαιτείτε για την υποστήριξη αυτής της δυνατότητας. Πρέπει να χρησιμοποιήσετε τη μέθοδο **DatabaseMetaData.supportsBatchUpdates()** για να προσδιορίσετε εάν η βάση δεδομένων προορισμού υποστηρίζει την επεξεργασία ενημέρωσης παρτίδας. Η μέθοδος επιστρέφει true εάν το πρόγραμμα οδήγησης JDBC υποστηρίζει αυτή τη λειτουργία.

- Η μέθοδος **addBatch()** του Statement, PreparedStatement και CallableStatement χρησιμοποιείται για την προσθήκη μεμονωμένων δηλώσεων στην παρτίδα. Το **executeBatch()** χρησιμοποιείται για να ξεκινήσει την εκτέλεση όλων των καταστάσεων που ομαδοποιούνται.

- Το **executeBatch()** επιστρέφει μια σειρά από ακέραιους αριθμούς και κάθε στοιχείο του πίνακα αντιπροσωπεύει τον αριθμό ενημερώσεων για την αντίστοιχη δήλωση ενημέρωσης.

- Όπως μπορείτε να προσθέσετε δηλώσεις σε μια παρτίδα για επεξεργασία, μπορείτε να τις καταργήσετε με τη μέθοδο **clearBatch()**. Αυτή η μέθοδος καταργεί όλες τις δηλώσεις που προσθέσατε με τη μέθοδο addBatch(). Ωστόσο, δεν μπορείτε επιλεκτικά να επιλέξετε ποια δήλωση να καταργήσετε. [6]

11.2 Δοσοληψία με Αντικείμενο Δήλωσης

Ακολουθεί μια τυπική ακολουθία βημάτων για τη χρήση της επεξεργασίας παρτίδας με το αντικείμενο δήλωσης :

1. Δημιουργήστε ένα αντικείμενο Δήλωσης χρησιμοποιώντας είτε τις μεθόδους createStatement() .
2. Ορίστε την αυτόματη δέσμευση σε ψευδή χρησιμοποιώντας το setAutoCommit() .
3. Προσθέστε όσες δηλώσεις SQL σας αρέσει σε παρτίδα χρησιμοποιώντας τη μέθοδο addBatch() στο αντικείμενο δημιούργημα.
4. Εκτελέστε όλες τις εντολές SQL χρησιμοποιώντας τη μέθοδο executeBatch() στο αντικείμενο της δηλωμένης εντολής.
5. Τέλος, δεσμεύστε όλες τις αλλαγές χρησιμοποιώντας τη μέθοδο commit() .

Το ακόλουθο απόσπασμα κώδικα παρέχει ένα παράδειγμα μιας ενημέρωσης παρτίδας χρησιμοποιώντας το αντικείμενο της δήλωσης : [7]

```
// Create statement object
Statement stmt = conn.createStatement();

// Set auto-commit to false
conn.setAutoCommit(false);

// Create SQL statement
String SQL = "INSERT INTO Employees (id, first, last, age) " +
             "VALUES(200,'Zia', 'Ali', 30)";
// Add above SQL statement in the batch.
stmt.addBatch(SQL);

// Create one more SQL statement
String SQL = "INSERT INTO Employees (id, first, last, age) " +
             "VALUES(201,'Raj', 'Kumar', 35)";
// Add above SQL statement in the batch.
stmt.addBatch(SQL);

// Create one more SQL statement
```

```
String SQL = "UPDATE Employees SET age = 35 " +
             "WHERE id = 100";
// Add above SQL statement in the batch.
stmt.addBatch(SQL);

// Create an int[] to hold returned values
int[] count = stmt.executeBatch();

//Explicitly commit statements to apply changes
conn.commit();
```

11.3 Παρτίδα με αντικείμενο PrepareStatement

Ακολουθεί μια τυπική ακολουθία βημάτων για τη χρήση της επεξεργασίας παρτίδας με το αντικείμενο PrepareStatement :

1. Δημιουργήστε εντολές SQL με placeholders.
2. Δημιουργήστε το αντικείμενο PrepareStatement χρησιμοποιώντας είτε τιςμεθόδους prepareStatement() .
3. Ορίστε την αυτόματη δέσμευση σε ψευδή χρησιμοποιώντας το setAutoCommit() .
4. Προσθέστε όσες δηλώσεις SQL σας αρέσει σε παρτίδα χρησιμοποιώντας τη μέθοδο addBatch() στο αντικείμενο δημιούργημα.
5. Εκτελέστε όλες τις εντολές SQL χρησιμοποιώντας τη μέθοδο executeBatch() στο αντικείμενο της δηλωμένης εντολής.
6. Τέλος, δεσμεύστε όλες τις αλλαγές χρησιμοποιώντας τη μέθοδο commit().

Το ακόλουθο απόσπασμα κώδικα παρέχει ένα παράδειγμα μιας ενημέρωσης παρτίδας χρησιμοποιώντας το αντικείμενο PrepareStatement : [7]

```
// Create SQL statement
String SQL = "INSERT INTO Employees (id, first, last, age) " +
             "VALUES(?, ?, ?, ?)";
```

```
// Create PrepareStatement object
PreparedStatemen pstmt = conn.prepareStatement(SQL);

//Set auto-commit to false
conn.setAutoCommit(false);

// Set the variables
pstmt.setInt( 1, 400 );
pstmt.setString( 2, "Pappu" );
pstmt.setString( 3, "Singh" );
pstmt.setInt( 4, 33 );
// Add it to the batch
pstmt.addBatch();

// Set the variables
pstmt.setInt( 1, 401 );
pstmt.setString( 2, "Pawan" );
pstmt.setString( 3, "Singh" );
pstmt.setInt( 4, 31 );
// Add it to the batch
pstmt.addBatch();

//add more batches
.
.
.
.
//Create an int[] to hold returned values
int[] count = stmt.executeBatch();

//Explicitly commit statements to apply changes
conn.commit();
```

11.3 Δοσολογία JDBC με αντικείμενο PrepareStatement

Ακολουθεί μια τυπική ακολουθία βημάτων για τη χρήση της επεξεργασίας παρτίδας με το αντικείμενο PrepareStatement :

- Δημιουργήστε εντολές SQL με placeholders.
- Δημιουργήστε το αντικείμενο PrepareStatement χρησιμοποιώντας είτε τις μεθόδους prepareStatement () .
- Ορίστε την αυτόματη δέσμευση σε ψευδή χρησιμοποιώντας το setAutoCommit () .
- Προσθέστε όσες δηλώσεις SQL σας αρέσει σε παρτίδα χρησιμοποιώντας τη μέθοδο addBatch () στο αντικείμενο δημιούργημα.
- Εκτελέστε όλες τις εντολές SQL χρησιμοποιώντας τη μέθοδο executeBatch () στο αντικείμενο της δηλωμένης εντολής.
- Τέλος, δεσμεύστε όλες τις αλλαγές χρησιμοποιώντας τη μέθοδο commit () .

Αυτός ο κώδικας δείγματος έχει γραφτεί με βάση το περιβάλλον και τη βάση δεδομένων που έγινε στα προηγούμενα κεφάλαια.

Αντιγράψτε και περάστε από το ακόλουθο παράδειγμα στο **JDBCExample.java,** μεταγλωττίστε και εκτελέστε ως εξής : [7]

```
// Import required packages
import java.sql.*;

public class JDBCExample {
   // JDBC driver name and database URL
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/EMP";

   //  Database credentials
   static final String USER = "username";
   static final String PASS = "password";

   public static void main(String[] args) {
   Connection conn = null;
```

```
PreparedStatement stmt = null;
try{
   // Register JDBC driver
   Class.forName("com.mysql.jdbc.Driver");

   // Open a connection
   System.out.println("Connecting to database...");
   conn = DriverManager.getConnection(DB_URL,USER,PASS);

   // Create SQL statement
   String SQL = "INSERT INTO Employees(id,first,last,age) " +
                "VALUES(?, ?, ?, ?)";

   // Create preparedStatemen
   System.out.println("Creating statement...");
   stmt = conn.prepareStatement(SQL);

   // Set auto-commit to false
   conn.setAutoCommit(false);

   // First, let us select all the records and display them.
   printRows( stmt );

   // Set the variables
   stmt.setInt( 1, 400 );
   stmt.setString( 2, "Pappu" );
   stmt.setString( 3, "Singh" );
   stmt.setInt( 4, 33 );
   // Add it to the batch
   stmt.addBatch();

   // Set the variables
   stmt.setInt( 1, 401 );
```

```
      stmt.setString( 2, "Pawan" );
      stmt.setString( 3, "Singh" );
      stmt.setInt( 4, 31 );
      // Add it to the batch
      stmt.addBatch();

      // Create an int[] to hold returned values
      int[] count = stmt.executeBatch();

      //Explicitly commit statements to apply changes
      conn.commit();

      // Again, let us select all the records and display them.
      printRows( stmt );

      // Clean-up environment
      stmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
```

```
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main

public static void printRows(Statement stmt) throws SQLException{
   System.out.println("Displaying available rows...");
   // Let us select all the records and display them.
   String sql = "SELECT id, first, last, age FROM Employees";
   ResultSet rs = stmt.executeQuery(sql);

   while(rs.next()){
      //Retrieve by column name
      int id  = rs.getInt("id");
      int age = rs.getInt("age");
      String first = rs.getString("first");
      String last = rs.getString("last");

      //Display values
      System.out.print("ID: " + id);
      System.out.print(", Age: " + age);
      System.out.print(", First: " + first);
      System.out.println(", Last: " + last);
   }
   System.out.println();
   rs.close();
}//end printRows()
}//end JDBCExample
```

Τώρα ας καταρτίσουμε το παραπάνω παράδειγμα ως εξής -

```
C:\>javac JDBCExample.java

C:\>
```

Όταν εκτελείτε το **JDBCExample**, παράγει το ακόλουθο αποτέλεσμα :

```
C:\>java JDBCExample
Connecting to database...
Creating statement...
Displaying available rows...
ID: 95, Age: 20, First: Sima, Last: Chug
ID: 100, Age: 35, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 30, First: Sumit, Last: Mittal
ID: 110, Age: 20, First: Sima, Last: Chug
ID: 200, Age: 30, First: Zia, Last: Ali
ID: 201, Age: 35, First: Raj, Last: Kumar

Displaying available rows...
ID: 95, Age: 20, First: Sima, Last: Chug
ID: 100, Age: 35, First: Zara, Last: Ali
ID: 101, Age: 25, First: Mahnaz, Last: Fatma
ID: 102, Age: 30, First: Zaid, Last: Khan
ID: 103, Age: 30, First: Sumit, Last: Mittal
ID: 110, Age: 20, First: Sima, Last: Chug
ID: 200, Age: 30, First: Zia, Last: Ali
ID: 201, Age: 35, First: Raj, Last: Kumar
ID: 400, Age: 33, First: Pappu, Last: Singh
ID: 401, Age: 31, First: Pawan, Last: Singh
Goodbye!
C:\>
```

ΚΕΦΑΛΑΙΟ 12

JDBC – Αποθηκευμένη διαδικασία

12.1 JDBC - Αποθηκευμένη διαδικασία

Ακριβώς όπως ένα αντικείμενο Connection δημιουργεί τα αντικείμενα Statement και PreparedStatement, δημιουργεί επίσης το αντικείμενο CallableStatement, το οποίο θα χρησιμοποιηθεί για την εκτέλεση μιας κλήσης σε μια αποθηκευμένη διαδικασία βάσης δεδομένων.

12.2 Δημιουργία αντικειμένου CallableStatement

Ας υποθέσουμε ότι πρέπει να εκτελέσετε την ακόλουθη αποθηκευμένη διαδικασία Oracle : [7]

```
CREATE OR REPLACE PROCEDURE getEmpName
   (EMP_ID IN NUMBER, EMP_FIRST OUT VARCHAR) AS
BEGIN
   SELECT first INTO EMP_FIRST
   FROM Employees
   WHERE ID = EMP_ID;
END;
```

ΣΗΜΕΙΩΣΗ: Η παραπάνω αποθηκευμένη διαδικασία γράφτηκε για την Oracle, αλλά δουλεύουμε με τη βάση δεδομένων MySQL έτσι ας γράψουμε την ίδια αποθηκευμένη διαδικασία για την MySQL ως εξής για να την δημιουργήσετε στη βάση δεδομένων EMP : [7]

```
DELIMITER $$

DROP PROCEDURE IF EXISTS `EMP`.`getEmpName` $$
CREATE PROCEDURE `EMP`.`getEmpName`
   (IN EMP_ID INT, OUT EMP_FIRST VARCHAR(255))
BEGIN
   SELECT first INTO EMP_FIRST
   FROM Employees
```

```
   WHERE ID = EMP_ID;
END $$


DELIMITER ;
```

Υπάρχουν τρεις τύποι παραμέτρων: IN, OUT και INOUT. Το αντικείμενο PreparedStatement χρησιμοποιεί μόνο την παράμετρο IN. Το αντικείμενο CallableStatement μπορεί να χρησιμοποιήσει και τα τρία.

Πίνακας 12.1: Ορισμοί παραμέτρων IN, OUT και INOUT [7]

Παράμετρος	**Περιγραφή**
IN	Μια παράμετρος της οποίας η τιμή είναι άγνωστη όταν δημιουργείται η εντολή SQL. Μπορείτε να δεσμεύσετε τις τιμές στις παραμέτρους IN με τις μεθόδους setXXX().
OUT	Μια παράμετρος της οποίας η τιμή παρέχεται από τη δήλωση SQL που επιστρέφει. Μπορείτε να ανακτήσετε τιμές από τις παραμέτρους OUT με τις μεθόδους getXXX().
INOUT	Μια παράμετρος που παρέχει τιμές εισόδου και εξόδου.Μπορείτε να συνδέσετε μεταβλητές με τις μεθόδους setXXX() και να ανακτήσετε τιμές με τις μεθόδους getXXX().

Το ακόλουθο απόσπασμα κώδικα παρουσιάζει τον τρόπο χρήσης της μεθόδου **Connection.prepareCall()** για την εμφάνιση ενός αντικειμένου **CallableStatement** με βάση την προηγούμενη αποθηκευμένη διαδικασία : [7]

```
CallableStatement cstmt = null;
try {
   String SQL = "{call getEmpName (?, ?)}";
   cstmt = conn.prepareCall (SQL);
   . . .
```

```
}
catch (SQLException e) {
   . . .
}
finally {
   . . .
}
```

Η μεταβλητή String SQL αντιπροσωπεύει την αποθηκευμένη διαδικασία, με εντολές κράτησης των παραμέτρων.

Η χρήση αντικειμένων CallableStatement μοιάζει πολύ με τη χρήση των αντικειμένων PreparedStatement. Πρέπει να συνδέσετε τιμές σε όλες τις παραμέτρους πριν εκτελέσετε τη δήλωση ή θα λάβετε μια SQLException.

Αν έχετε παραμέτρους IN, ακολουθήστε τους ίδιους κανόνες και τεχνικές που ισχύουν για ένα αντικείμενο PreparedStatement. Χρησιμοποιήστε τη μέθοδο setXXX() που αντιστοιχεί στον τύπο δεδομένων Java που δεσμεύετε.

Όταν χρησιμοποιείτε τις παραμέτρους OUT και INOUT, πρέπει να χρησιμοποιήσετε μια επιπλέον μέθοδο CallableStatement, την registerOutParameter(). Η μέθοδος registerOutParameter() δεσμεύει τον τύπο δεδομένων JDBC στον τύπο δεδομένων που αναμένεται να επιστρέψει η αποθηκευμένη διαδικασία.

Αφού καλέσετε την αποθηκευμένη σας διαδικασία, ανακτάτε την τιμή από την παράμετρο OUT με την κατάλληλη μέθοδο getXXX(). Αυτή η μέθοδος μεταφέρει την ανακτώμενη τιμή του τύπου SQL σε έναν τύπο δεδομένων Java. [6]

12.3 Κλείσιμο αντικειμένου CallableStatement

Ακριβώς όπως κλείνετε άλλο αντικείμενο Statement, για τον ίδιο λόγο θα πρέπει επίσης να κλείσετε το αντικείμενο CallableStatement.

Μια απλή κλήση στη μέθοδο close() θα κάνει τη δουλειά. Εάν κλείσετε πρώτα το αντικείμενο σύνδεσης, θα κλείσει επίσης το αντικείμενο CallableStatement. Ωστόσο, θα πρέπει πάντα να κλείσετε ρητά το αντικείμενο CallableStatement για να διασφαλίσετε την σωστή εκκαθάριση. [7]

```
CallableStatement cstmt = null;
try {
```

```
   String SQL = "{call getEmpName (?, ?)}";
   cstmt = conn.prepareCall (SQL);
   . . .
}
catch (SQLException e) {
   . . .
}
finally {
   cstmt.close();
}
```

12.4 JDBC SQL Escape Syntax

Η σύνταξη διαφυγής σάς δίνει την ευελιξία να χρησιμοποιήσετε συγκεκριμένες λειτουργίες της βάσης δεδομένων που δεν είναι διαθέσιμες για εσάς χρησιμοποιώντας τις τυπικές μεθόδους και ιδιότητες του JDBC.

Η γενική μορφή σύνταξης διαφυγής SQL έχει ως εξής:

```
{keyword 'parameters'}
```

Ακολουθούν οι ακολουθίες διαφυγής, τις οποίες θα βρείτε πολύ χρήσιμες κατά την εκτέλεση του προγραμματισμού JDBC :

12.5 d, t, ts Keywords

Βοηθούν στον προσδιορισμό των ημερομηνιών, του χρόνου και των κυριοτέρων σημάτων. Όπως γνωρίζετε, κανένα από τα δύο DBMS δεν αντιπροσωπεύει το χρόνο και την ημερομηνία με τον ίδιο τρόπο. Αυτή η σύνταξη διαφυγής αποκαλύπτει στον οδηγό την εμφάνιση της ημερομηνίας ή της ώρας στη μορφή της βάσης δεδομένων προορισμού. Για παράδειγμα : [7]

```
{d 'yyyy-mm-dd'}
```

Όπου yyyy = έτος, mm = μήνα, dd = ημερομηνία. Χρησιμοποιώντας αυτή τη σύνταξη {d '2009-09-03'} είναι η 9η Μαρτίου 2009.

Εδώ είναι ένα απλό παράδειγμα που δείχνει πώς να εισάγετε την ημερομηνία σε έναν πίνακα :

```
//Create a Statement object
stmt = conn.createStatement();
//Insert data ==> ID, First Name, Last Name, DOB
String sql="INSERT INTO STUDENTS VALUES" +
             "(100,'Zara','Ali', {d '2001-12-16'})";


stmt.executeUpdate(sql);
```

Ομοίως, μπορείτε να χρησιμοποιήσετε μία από τις ακόλουθες δύο συνταγές, είτε **t** είτε **ts** :

```
{t 'hh:mm:ss'}
```

Όπου hh = ώρα. mm = λεπτό. ss = δευτερόλεπτο. Η χρήση αυτής της σύνταξης {t '13: 30: 29 '} είναι 1:30:29 μ.μ.

```
{ts 'yyyy-mm-dd hh:mm:ss'}
```

Αυτή είναι η συνδυασμένη σύνταξη των δύο παραπάνω συντακτικών για το 'd' και το 't' για την απεικόνιση της χρονικής σήμανσης. [7]

12.6 escape Keyword

Αυτή η λέξη-κλειδί αναγνωρίζει το χαρακτήρα διαφυγής που χρησιμοποιείται στις ρήτρες LIKE. Χρήσιμο όταν χρησιμοποιείτε το wildcard SQL%, το οποίο αντιστοιχεί σε μηδέν ή περισσότερους χαρακτήρες. Για παράδειγμα : [7]

```
String sql = "SELECT symbol FROM MathSymbols
              WHERE symbol LIKE '\%' {escape '\'}";
stmt.execute(sql);
```

Εάν χρησιμοποιείτε τον χαρακτήρα αντίστροφης κάθετος (\) ως χαρακτήρα διαφυγής, πρέπει επίσης να χρησιμοποιήσετε δύο χαρακτήρες αντίστροφης κάθετος στη λέξη Java String σας, επειδή η αντίστροφη κάθετος είναι επίσης ένας χαρακτήρας διαφυγής Java.

12.7 fn Keyword

Αυτή η λέξη-κλειδί αντιπροσωπεύει τις κλιμακωτές συναρτήσεις που χρησιμοποιούνται σε ένα DBMS. Για παράδειγμα, μπορείτε να χρησιμοποιήσετε το SQL function length για να πάρετε το μήκος μιας συμβολοσειράς : [7]

```
{fn length('Hello World')}
```

Αυτό επιστρέφει 11, το μήκος της συμβολοσειράς χαρακτήρων 'Hello World'.

12.8 call Keyword

Αυτή η λέξη-κλειδί χρησιμοποιείται για την κλήση των αποθηκευμένων διαδικασιών. Για παράδειγμα, για μια αποθηκευμένη διαδικασία που απαιτεί μια παράμετρο IN, χρησιμοποιήστε την ακόλουθη σύνταξη : [7]

```
{call my_procedure(?)};
```

Για μια αποθηκευμένη διαδικασία που απαιτεί μια παράμετρο IN και επιστρέφει μια παράμετρο OUT, χρησιμοποιήστε την ακόλουθη σύνταξη :

```
{? = call my_procedure(?)};
```

12.9 oj Keyword

Αυτή η λέξη-κλειδί χρησιμοποιείται για την ένδειξη outer joins. Η σύνταξη είναι η εξής : [7]

```
{oj outer-join}
```

Όπου outer-join = table {LEFT|RIGHT|FULL} OUTERJOIN {table | outer-join} στην κατάσταση αναζήτησης. Για παράδειγμα : [7]

```
String sql = "SELECT Employees
              FROM {oj ThisTable RIGHT
              OUTER JOIN ThatTable on id = '100'}";
stmt.execute(sql);
```

ΚΕΦΑΛΑΙΟ 13

JDBC - Streaming ASCII και δυαδικά δεδομένα

13.1 JDBC - Streaming ASCII και δυαδικά δεδομένα

Ένα αντικείμενο PreparedStatement έχει τη δυνατότητα να χρησιμοποιεί ροές εισόδου και εξόδου για την παροχή δεδομένων από τις παραμέτρους. Αυτό σας επιτρέπει να τοποθετήσετε ολόκληρα αρχεία σε στήλες της βάσης δεδομένων με αποτέλεσμα να διατηρούν έτσι μεγάλες τιμές, όπως τύποι δεδομένων CLOB και BLOB.

Υπάρχουν οι ακόλουθες μέθοδοι, οι οποίες μπορούν να χρησιμοποιηθούν για τη ροή δεδομένων :

- **setAsciiStream():** Αυτή η μέθοδος χρησιμοποιείται για την παροχή μεγάλων τιμών ASCII.
- **setCharacterStream():** Αυτή η μέθοδος χρησιμοποιείται για την παροχή μεγάλων τιμών UNICODE.
- **setBinaryStream():** Αυτή η μέθοδος χρησιμοποιείται για την παροχή μεγάλων δυαδικών τιμών.

Η μέθοδος setXXXStream() απαιτεί μια επιπλέον παράμετρο, το μέγεθος του αρχείου, εκτός από το σύμβολο κράτησης παραμέτρων. Αυτή η παράμετρος ενημερώνει τον οδηγό σχετικά με το πόσα δεδομένα πρέπει να αποστέλλονται στη βάση δεδομένων. [6]

13.2 Παράδειγμα

Σκεφτείτε ότι θέλουμε να φορτώσετε ένα XML αρχείο XML_Data.xml σε έναν πίνακα βάσης δεδομένων. Εδώ είναι το περιεχόμενο αυτού του αρχείου XML : [7]

```
<?xml version="1.0"?>
<Employee>
<id>100</id>
<first>Zara</first>
```

```
<last>Ali</last>

<Salary>10000</Salary>

<Dob>18-08-1978</Dob>

<Employee>
```

Κρατήστε αυτό το αρχείο XML στον ίδιο κατάλογο όπου θα εκτελέσετε αυτό το παράδειγμα.

Αυτό το παράδειγμα θα δημιουργούσε έναν πίνακα βάσης δεδομένων XML_Data και στη συνέχεια το αρχείο XML_Data.xml θα αποθηκευτεί σε αυτόν τον πίνακα.

Αντιγράψτε και περάστε από το ακόλουθο παράδειγμα στο **JDBCExample.java,** μεταγλωττίστε και εκτελέστε ως εξής : [7]

```
// Import required packages

import java.sql.*;

import java.io.*;

import java.util.*;


public class JDBCExample {

   // JDBC driver name and database URL

   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";

   static final String DB_URL = "jdbc:mysql://localhost/EMP";


   //  Database credentials

   static final String USER = "username";

   static final String PASS = "password";


   public static void main(String[] args) {

   Connection conn = null;

   PreparedStatement pstmt = null;

   Statement stmt = null;

   ResultSet rs = null;

   try{

      // Register JDBC driver
```

```
Class.forName("com.mysql.jdbc.Driver");


// Open a connection
System.out.println("Connecting to database...");
conn = DriverManager.getConnection(DB_URL,USER,PASS);


//Create a Statement object and build table
stmt = conn.createStatement();
createXMLTable(stmt);


//Open a FileInputStream
File f = new File("XML_Data.xml");
long fileLength = f.length();
FileInputStream fis = new FileInputStream(f);


//Create PreparedStatement and stream data
String SQL = "INSERT INTO XML_Data VALUES (?,?)";
pstmt = conn.prepareStatement(SQL);
pstmt.setInt(1,100);
pstmt.setAsciiStream(2,fis,(int)fileLength);
pstmt.execute();


//Close input stream
fis.close();


// Do a query to get the row
SQL = "SELECT Data FROM XML_Data WHERE id=100";
rs = stmt.executeQuery (SQL);
// Get the first row
if (rs.next ()){
   //Retrieve data from input stream
   InputStream xmlInputStream = rs.getAsciiStream (1);
   int c;
```

```
         ByteArrayOutputStream bos = new ByteArrayOutputStream();
         while (( c = xmlInputStream.read ()) != -1)
            bos.write(c);
         //Print results
         System.out.println(bos.toString());
      }
      // Clean-up environment
      rs.close();
      stmt.close();
      pstmt.close();
      conn.close();
   }catch(SQLException se){
      //Handle errors for JDBC
      se.printStackTrace();
   }catch(Exception e){
      //Handle errors for Class.forName
      e.printStackTrace();
   }finally{
      //finally block used to close resources
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(pstmt!=null)
            pstmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
```

```
         se.printStackTrace();
      }//end finally try
   }//end try
   System.out.println("Goodbye!");
}//end main


public static void createXMLTable(Statement stmt)
   throws SQLException{
   System.out.println("Creating XML_Data table..." );
   //Create SQL Statement
   String streamingDataSql = "CREATE TABLE XML_Data " +
                             "(id INTEGER, Data LONG)";
   //Drop table first if it exists.
   try{
      stmt.executeUpdate("DROP TABLE XML_Data");
   }catch(SQLException se){
   }// do nothing
   //Build table.
   stmt.executeUpdate(streamingDataSql);
}//end createXMLTable
}//end JDBCExample
```

Τώρα ας καταρτίσουμε το παραπάνω παράδειγμα ως εξής :

```
C:\>javac JDBCExample.java
C:\>
```

Όταν εκτελείτε το **JDBCExample**, παράγει το ακόλουθο αποτέλεσμα :

```
C:\>java JDBCExample
Connecting to database...
Creating XML_Data table...
<?xml version="1.0"?>
<Employee>
<id>100</id>
<first>Zara</first>
```

```
<last>Ali</last>

<Salary>10000</Salary>

<Dob>18-08-1978</Dob>

<Employee>

Goodbye!

C:\>
```

ΚΕΦΑΛΑΙΟ 14

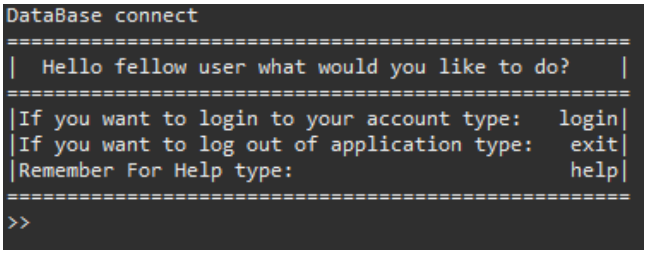
JDBC – Πρόγραμμα επίδειξης Πτυχιακής Εργασίας

14.1 Περιγραφή του Προγράμματος της πτυχιακής εργασίας

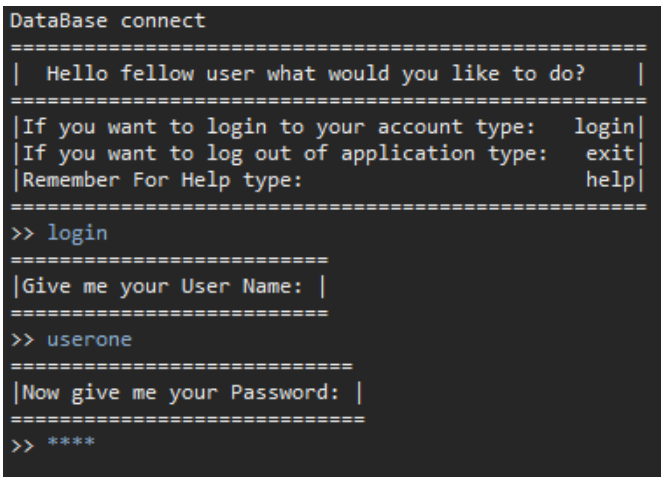
Αντικείμενο του προγράμματος της παρούσας πτυχιακής εργασίας είναι η ανταλλαγή γραπτών μηνυμάτων μέσω καταχώρισης τους σε μια βάση δεδομένων ανάμεσα στους χρήστες.

Κατά την εκκίνηση του προγράμματος παρουσιάζετε το κεντρικό μενού για τις βασικές επιλογές που μπορεί να πραγματοποιήσει ο χρήστης.

- Login : Εκκινεί το κυρίως πρόγραμμα όπου ο χρήστης θα εισάγει το όνομα χρήστη (username) και τον κωδικό πρόσβασης (password) .
- Exit : Τερματισμός του προγράμματος.
- Help : Εμφανίζει το μενού της Εικόνας 14.1 με τις εντολές που μπορεί να εισάγει ο χρήστης στο επίπεδο που βρίσκετε.



Εικόνα 14.1: Αρχικό Μενού Προγράμματος



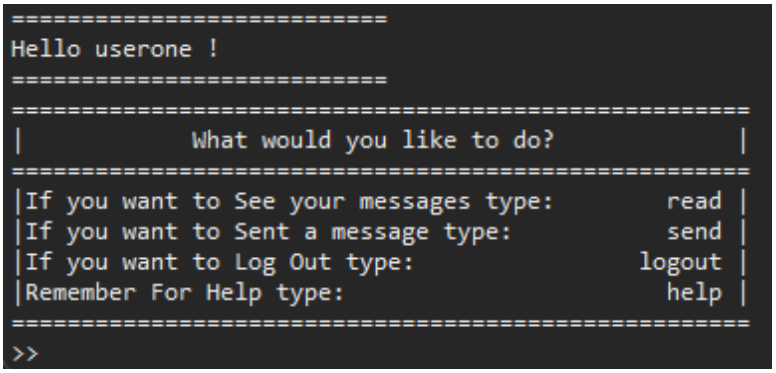
Εικόνα 14.2: Μενού για τον έλεγχο του Ονόματος Χρήστη και του κωδικού πρόσβασης

14.2 Κατηγορία χρήστη επιπέδου «Α»

Η Κατηγορία χρηστών επιπέδου «Α» περιλαμβάνει τις λειτουργίες της αποστολής και ανάγνωσης μηνυμάτων.

Στο σημείο αυτό αφού έχει καταχωρηθεί σωστά το όνομα χρήστη και ο κωδικός πρόσβασης του χρήστη και αφού αναγνωριστεί ότι τα δικαιώματα του ταυτίζονται με τον Κατηγορία επιπέδου «Α» τότε εμφανίζετε το όνομα του χρήστη και το παρακάτω μενού.

- Send : εκκίνηση διαδικασίας για αποστολή μηνύματος σε άλλον χρήστη
- Read : εκκίνηση διαδικασίας για την ανάγνωση όλων των μηνυμάτων του χρήστη.
- Logout : αποσύνδεση και επιστροφή στο αρχικό μενού
- Help : εμφάνιση μενού της Εικόνας 14.3 με τις εντολές που μπορεί να εισάγει ο χρήστης στο επίπεδο που βρίσκετε.



Εικόνα 14.3: Κύριο Μενού Προγράμματος για την Κατηγορία χρηστών «Α»

14.2.1 Διαδικασία αποστολής μηνύματος

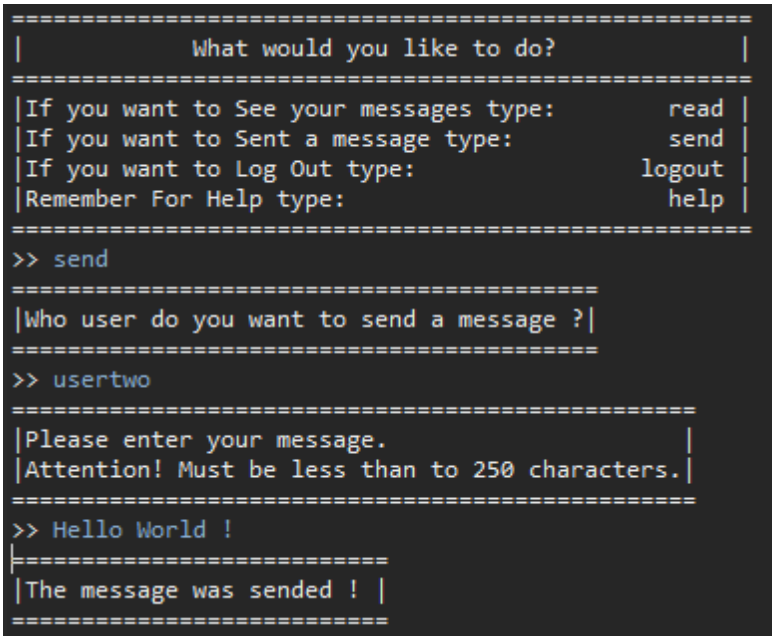
Για την εκκίνηση της διαδικασίας αποστολής μηνύματος ο χρήστης πρέπει να εισάγει την εντολή “send” στο κεντρικό μενού.

Με την εισαγωγή της εντολής το πρόγραμμα ζητά να εισαχθεί σε ποιόν άλλον χρήστη επιθυμούμε να στείλουμε μήνυμα. Στο σημείο αυτό πραγματοποιείτε έλεγχος ώστε να επιβεβαιωθεί αν το όνομα του παραλήπτη του μηνύματος υφίσταται στη βάση δεδομένων. Αν δεν υπάρχει ο αποστολέας τότε το

πρόγραμμα επαναλαμβάνει την ερώτηση μέχρι να δοθεί σωστά το όνομα χρήστη του αποστολέα.

Έπειτα από την επιτυχημένη εισαγωγή του ονόματος του αποστολέα ζητείτε από τον χρήστη να πληκτρολογήσει το μήνυμα που επιθυμεί με την προϋπόθεση ότι δεν ξεπερνά τους 250 χαρακτήρες.

Τέλος αφού πληκτρολογηθεί το μήνυμα τότε το πρόγραμμα το αποθηκεύει στη βάση δεδομένων και το παρουσιάζει στον αποστολέα σαν εισερχόμενο μήνυμα.

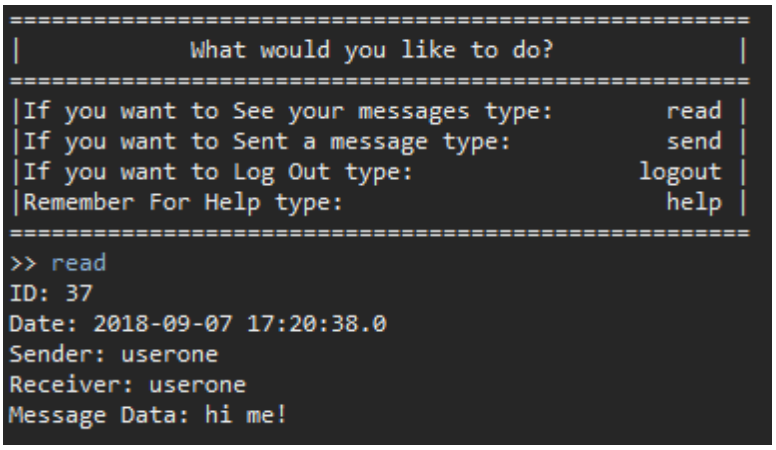


Εικόνα 14.4: Επιτυχή αποστολή Μηνύματος.

14.2.2 Διαδικασία ανάγνωσης μηνυμάτων

Για την εκκίνηση της διαδικασίας αποστολής μηνύματος ο χρήστης πρέπει να εισάγει την εντολή “read” στο κεντρικό μενού.

Με την εισαγωγή της εντολής το πρόγραμμα εμφανίζει όλα τα εισερχόμενα μηνύματα του χρήστη. Συγκεκριμένα εμφανίζει τον μοναδικό αριθμό του μηνύματος “ID”, την ημερομηνία και ώρα αποστολής “Date”, τον αποστολέα “Sender”, τον παραλήπτη “Receiver” και τέλος το μήνυμα “Message Data”.

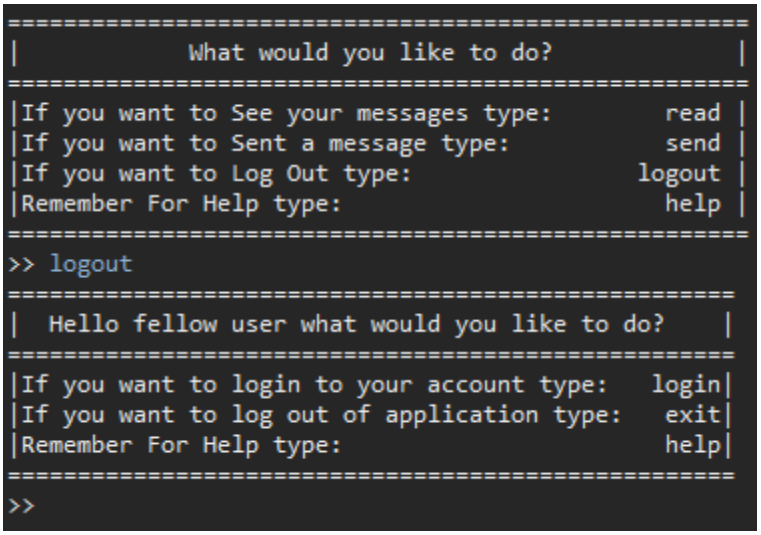


Εικόνα 14.5: Ανάγνωση όλων των Μηνυμάτων του χρήστη.

14.2.3 Διαδικασία αποσύνδεσης από τον λογαριασμό

Για την εκκίνηση της διαδικασίας αποστολής μηνύματος ο χρήστης πρέπει να εισάγει την εντολή “logout” στο κεντρικό μενού.

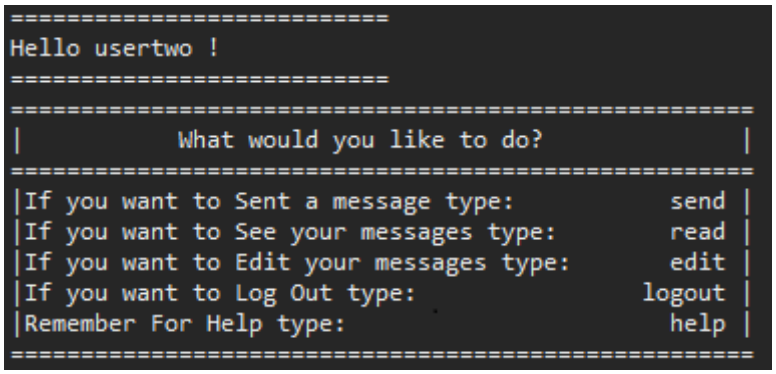
Με την εισαγωγή της εντολής το πρόγραμμα επιστρέφει στο αρχικό μενού όπου εμφανίζετε κατά την εκκίνηση του προγράμματος.



Εικόνα 14.6: Έξοδος από το Πρόγραμμα

14.3 Κατηγορία χρήστη επιπέδου «Β»

Η Κατηγορία χρηστών επιπέδου «Β» περιλαμβάνει όλες τις λειτουργίες της Κατηγορίας «Α» συν μια επιπρόσθετη, την δυνατότητα τροποποίησης των μηνυμάτων από τον χρήστη.



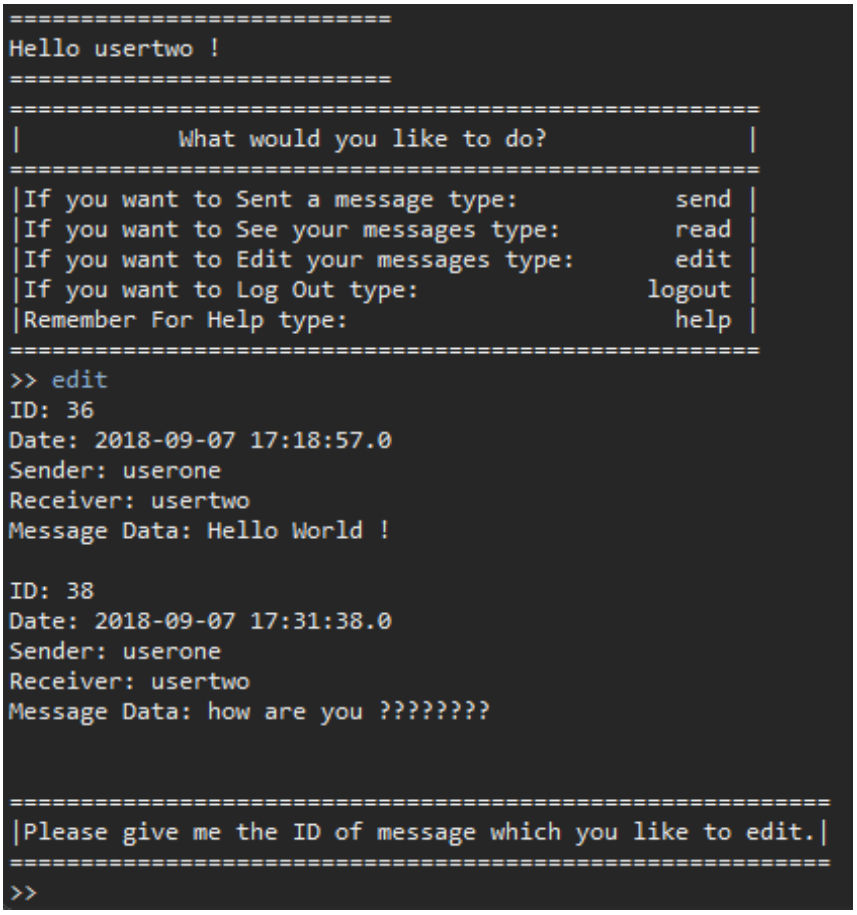
Εικόνα 14.7: Κύριο Μενού Προγράμματος για την Κατηγορία χρηστών «Β»

Για την εκκίνηση της διαδικασίας τροποποίησης ενός μηνύματος ο χρήστης πρέπει να εισάγει την εντολή “edit” στο κεντρικό μενού.

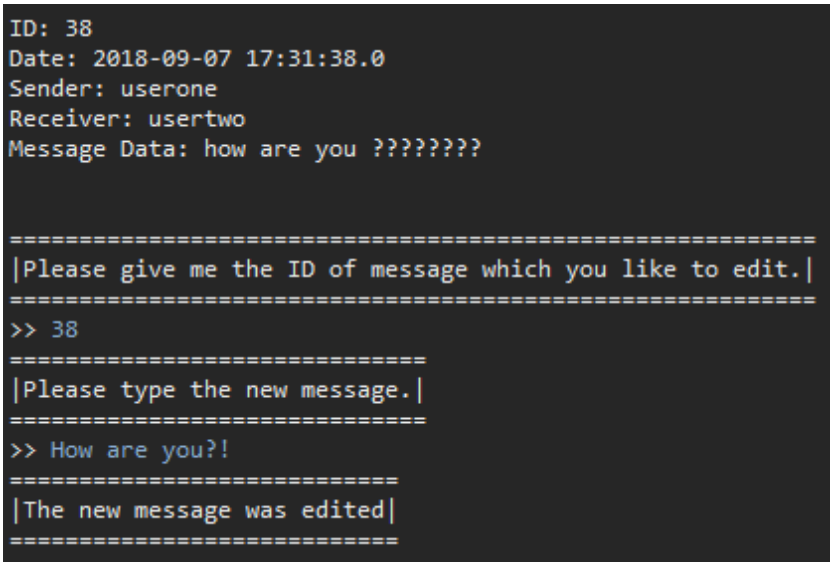
Με την εισαγωγή της εντολής το πρόγραμμα πρώτα εμφανίζει όλα τα μηνύματα όπου έχει λάβει ο χρήστης. Έπειτα ζητά να δοθεί ο μοναδικός αριθμός “ID” του μηνύματος όπου ο χρήστης επιθυμεί να τροποποιήσει.

Αφού πραγματοποιηθεί έλεγχος για την εγκυρότητα του μοναδικού αριθμού το πρόγραμμα ζητά από τον χρήστη να εισάγει το νέο και τροποποιημένο μήνυμα. Φυσικά ισχύει η προϋπόθεση να μην ξεπερνάει τους 250 χαρακτήρες.

Τέλος αφού πληκτρολογηθεί το μήνυμα τότε το πρόγραμμα το αποθηκεύει στη βάση δεδομένων.



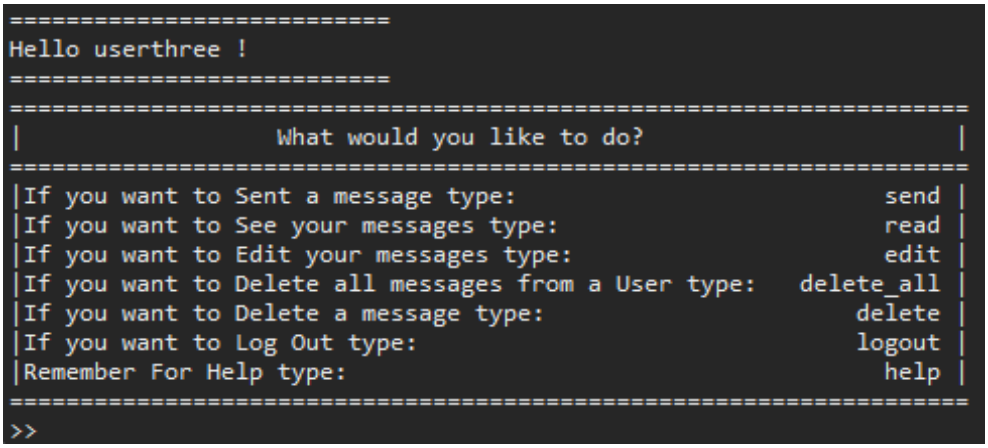
Εικόνα 14.8: Εμφάνιση όλων των μηνυμάτων για να επιλέξει ο χρήστης ποιο μήνυμα θα τροποποιήσει.



Εικόνα 14.9: Συγγραφή και αποθήκευση τροποποιημένου μηνύματος.

14.4 Κατηγορία χρήστη επιπέδου «C»

Η Κατηγορία χρηστών επιπέδου «C» περιλαμβάνει όλες τις λειτουργίες της Κατηγορίας «A» και «B» συν την δυνατότητα διαγραφής ενός μηνύματος ή όλων των μηνυμάτων από έναν επιλεγμένο αποστολέα.



Εικόνα 14.10: Κύριο Μενού Προγράμματος για την Κατηγορία χρηστών «C»

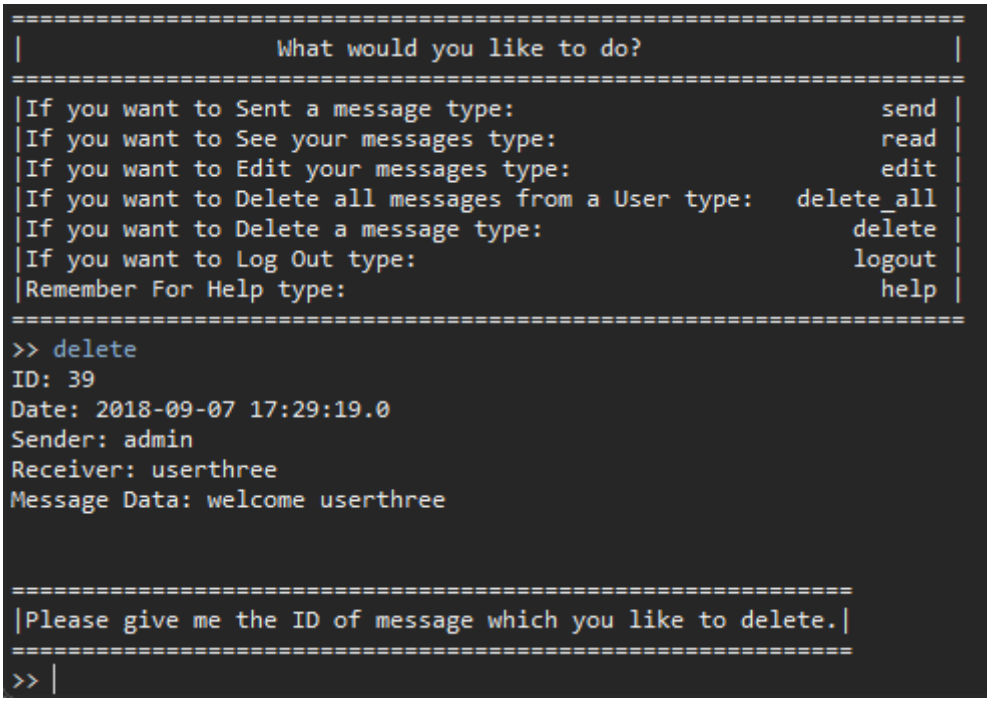
14.4.1 Διαδικασία διαγραφής ενός επιλεγμένου μηνύματος.

Για την εκκίνηση της διαδικασίας διαγραφής ενός μηνύματος ο χρήστης πρέπει να εισάγει την εντολή “delete” στο κεντρικό μενού.

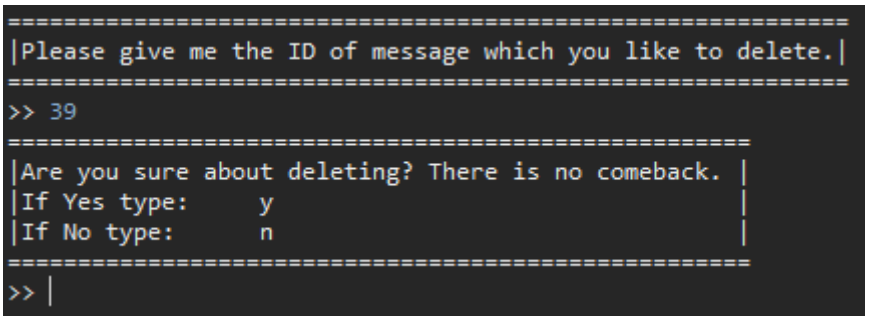
Με την εισαγωγή της εντολής το πρόγραμμα πρώτα εμφανίζει όλα τα μηνύματα όπου έχει λάβει ο χρήστης. Έπειτα ζητά να δοθεί ο μοναδικός αριθμός “ID” του μηνύματος όπου ο χρήστης επιθυμεί να διαγραφεί.

Αφού πραγματοποιηθεί έλεγχος για την εγκυρότητα του μοναδικού αριθμού το πρόγραμμα ζητά από τον χρήστη να επιβεβαιώσει ότι σίγουρα επιθυμεί την μόνιμη διαγραφή του μηνύματος.

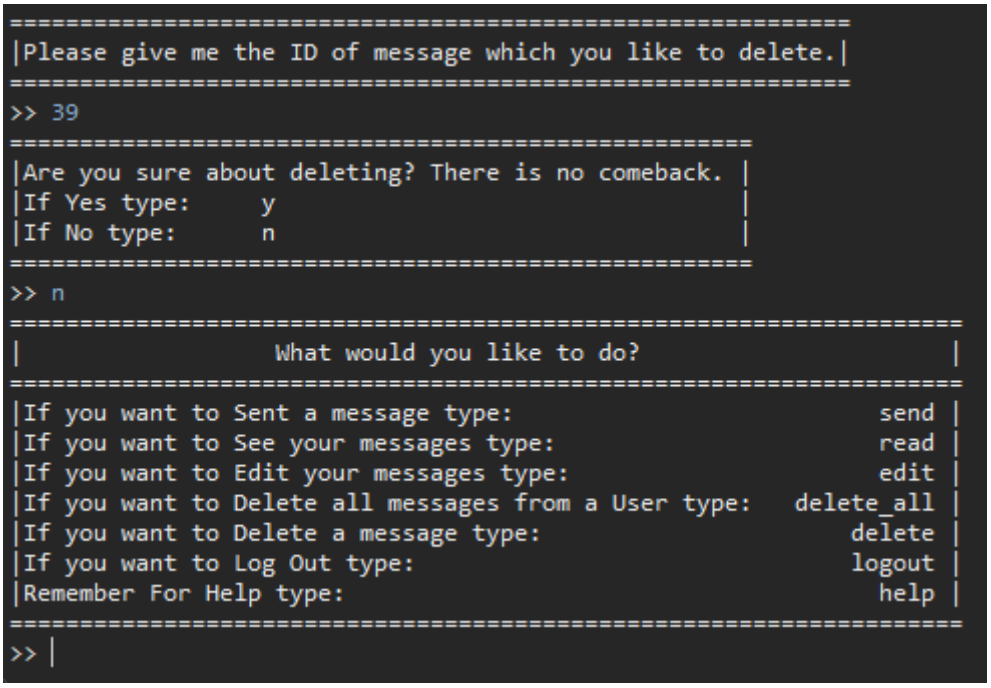
Για την επιβεβαίωση αυτή το πρόγραμμα εμφανίζει το μενού της Εικόνας 14.12 όπου το πρόγραμμα ζητά επιβεβαίωση για το αν ο χρήστης άλλαξε γνώμη και δεν επιθυμεί την μόνιμη διαγραφή του μηνύματος. Στην περίπτωση όπου ο χρήστης άλλαξε γνώμη και δεν επιθυμεί να διαγράψει του μήνυμα τότε πληκτρολογεί τον χαρακτήρα “n” και το πρόγραμμα επιστρέφει στο αρχικό μενού του χρήστη της κατηγορίας αυτής. Αντίθετα αν ο χρήστης είναι απόλυτα σίγουρος ότι επιθυμεί να προβεί σε μόνιμή διαγραφή τότε πληκτρολογεί τον χαρακτήρα “y”, με αποτέλεσμα η βάση δεδομένων να διαγράφει ολόκληρη την εγγραφή στον πίνακα όπου αφορά το συγκεκριμένο μήνυμα.



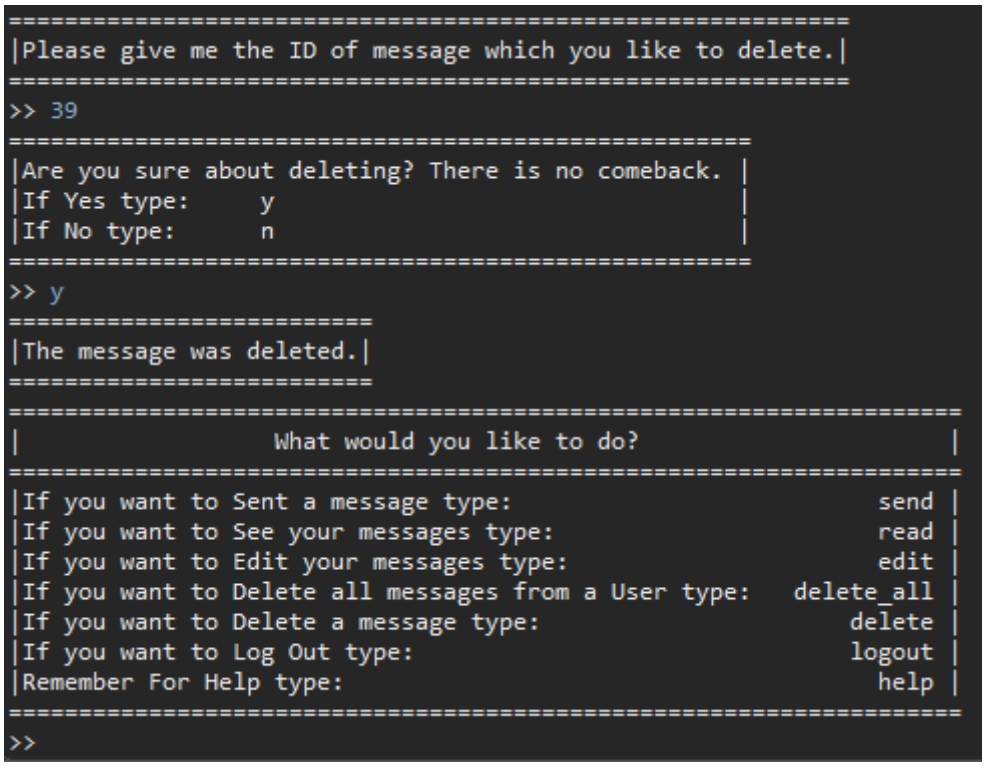
Εικόνα 14.11: Μενού για την διαγραφή ενός επιλεγμένου μηνύματος.



Εικόνα 14.12: Επιβεβαίωση για την μόνιμη διαγραφή του μηνύματος.



Εικόνα 14.13: Επιβεβαίωση για την μόνιμη διαγραφή του μηνύματος με αρνητική απάντηση από τον χρήστη.



Εικόνα 14.14: Επιβεβαίωση για την μόνιμη διαγραφή του μηνύματος με θετική απάντηση από τον χρήστη.

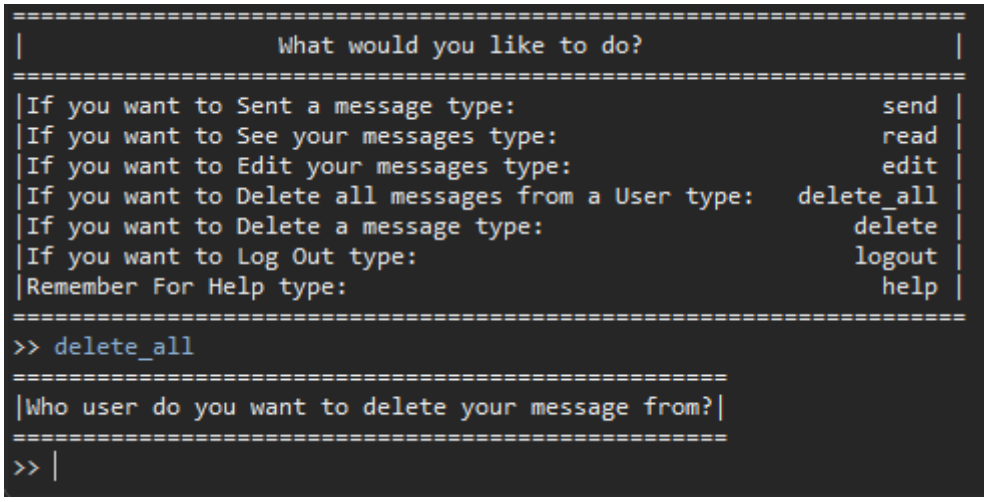
14.4.2 Διαδικασία διαγραφής όλων των μηνυμάτων από έναν επιλεγμένο αποστολέα.

Για την εκκίνηση της διαδικασίας διαγραφής όλων των μηνυμάτων από έναν επιλεγμένο αποστολέα, ο χρήστης πρέπει να εισάγει την εντολή “delete_all” στο κεντρικό μενού.

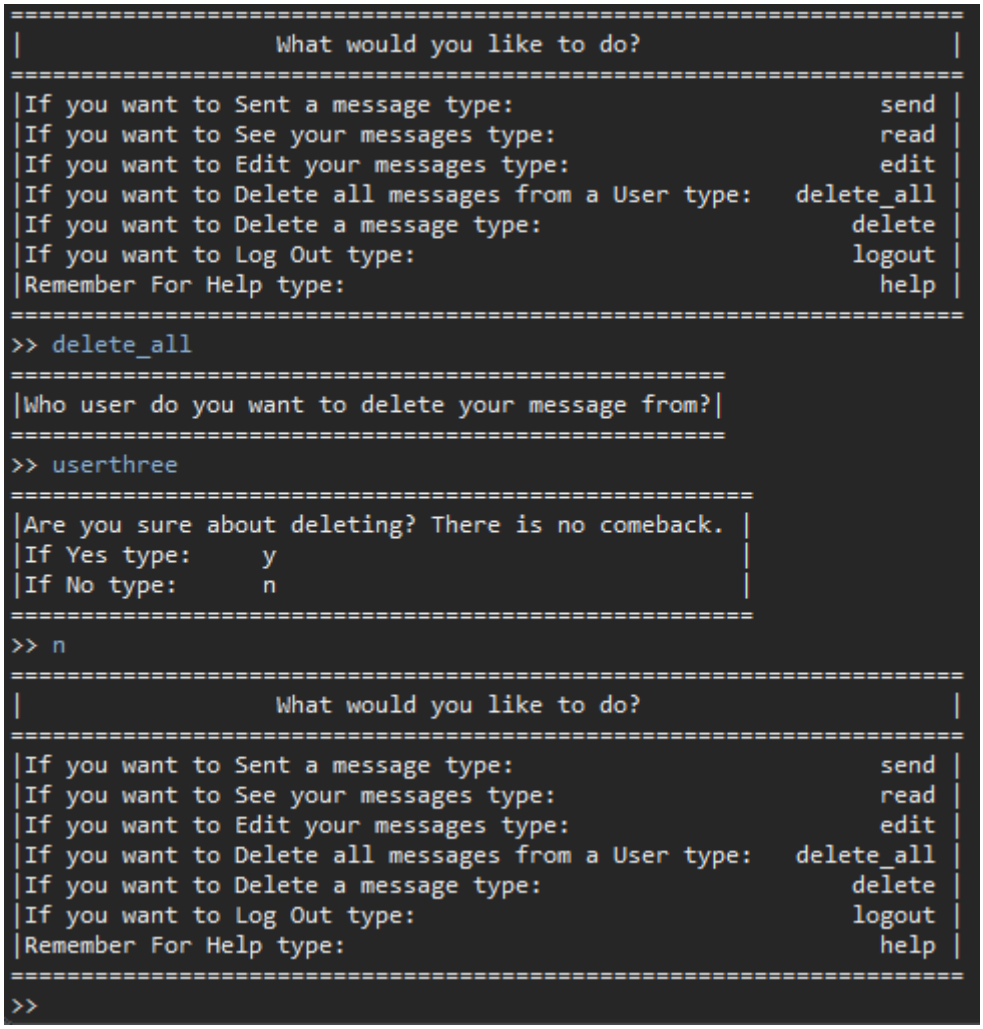
Με την εισαγωγή της εντολής το πρόγραμμα πρώτα ζητά να δοθεί το όνομα χρήστη του αποστολέα των μηνυμάτων “user name” όπου ο χρήστης επιθυμεί να διαγραφούν.

Αφού πραγματοποιηθεί έλεγχος για την εγκυρότητα του ονόματος χρήστη του αποστολέα, το πρόγραμμα ζητά από τον χρήστη να επιβεβαιώσει ότι σίγουρα επιθυμεί την μόνιμη διαγραφή του μηνύματος.

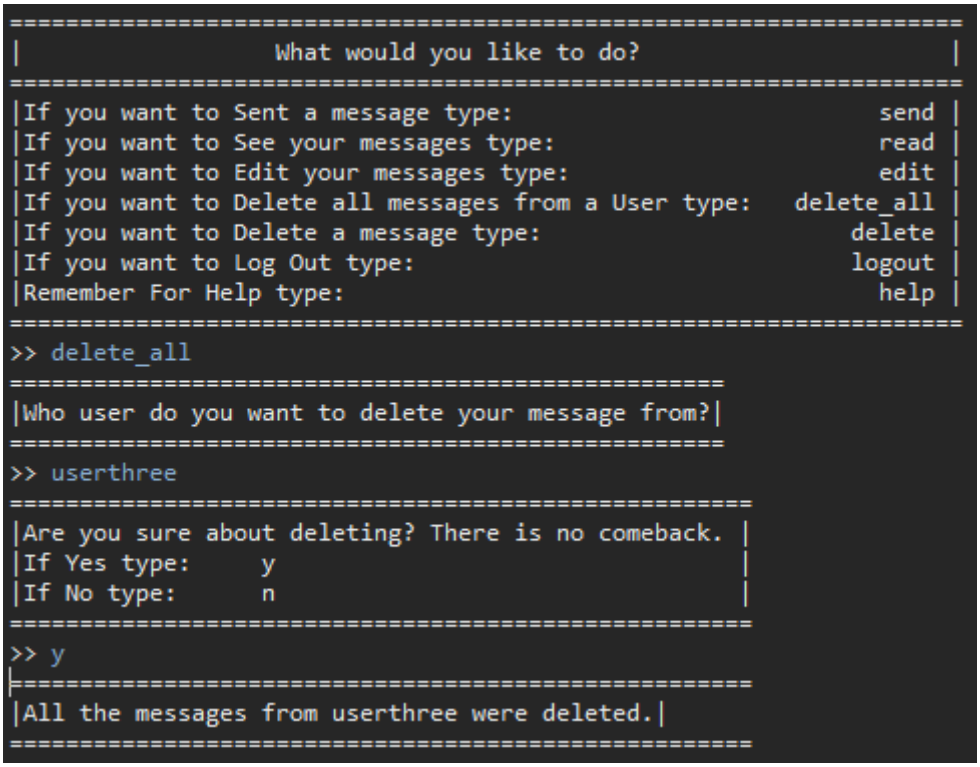
Για την επιβεβαίωση αυτή το πρόγραμμα ζητά από τον χρήστη αν άλλαξε γνώμη και δεν επιθυμεί την μόνιμη διαγραφή του μηνύματος, πληκτρολογώντας τον χαρακτήρα “n” και το πρόγραμμα επιστρέφει στο αρχικό μενού του χρήστη της κατηγορίας αυτής. Αντίθετα αν ο χρήστης είναι απόλυτα σίγουρος ότι επιθυμεί να προβεί σε μόνιμή διαγραφή τότε πληκτρολογεί τον χαρακτήρα “y”, με αποτέλεσμα η βάση δεδομένων να διαγράφει όλες τις εγγραφές στον πίνακα όπου αφορά το συγκεκριμένο μήνυμα με αποστολέα το επιλεγμένο όνομα χρήστη.



Εικόνα 14.15: Μενού για την διαγραφή όλων των μηνυμάτων από έναν επιλεγμένο αποστολέα.



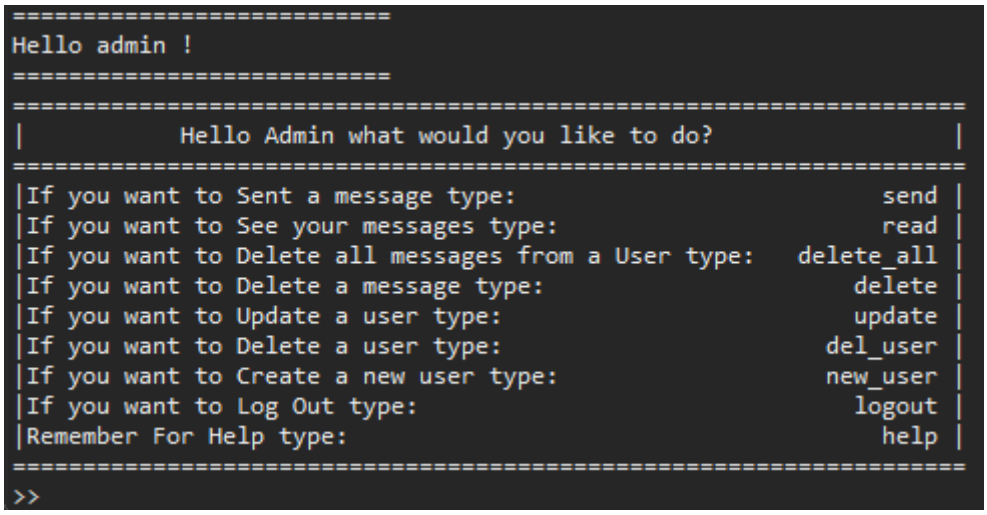
Εικόνα 14.16: Επιβεβαίωση για την μόνιμη διαγραφή των μηνυμάτων με αρνητική απάντηση από τον χρήστη.



Εικόνα 14.17: Επιβεβαίωση για την μόνιμη διαγραφή των μηνυμάτων με θετική απάντηση από τον χρήστη.

14.5 Κατηγορία χρήστη επιπέδου «Admin»

Η Κατηγορία χρηστών επιπέδου «Admin» περιλαμβάνει όλες τις λειτουργίες των Κατηγοριών «A», «B» και «C» συν την δυνατότητα δημιουργίας χρήστη, διαγραφή ενός χρήστη και την αλλαγή της κατηγορίας ενός υφιστάμενου χρήστη.



Εικόνα 14.18: Κύριο Μενού Προγράμματος για την Κατηγορία «Admin»

14.5.1 Διαδικασία δημιουργίας ενός νέου χρήστη.

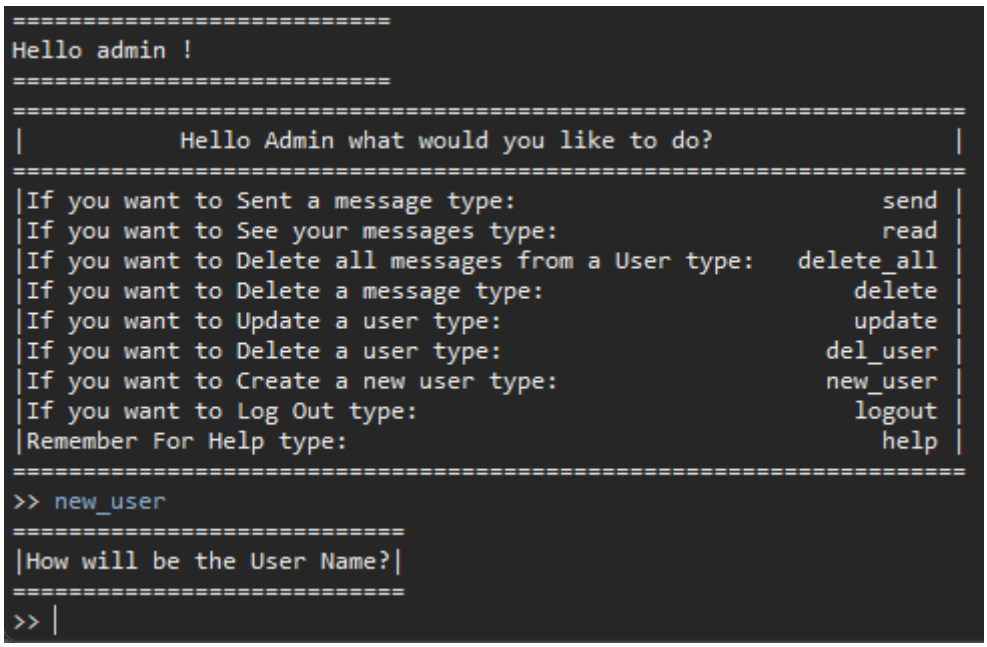
Για την εκκίνηση της διαδικασίας δημιουργίας ενός νέου χρήστη, ο χρήστης με δικαιώματα “Admin” πρέπει να εισάγει την εντολή “new_user” στο κεντρικό μενού.

Με την εισαγωγή της εντολής το πρόγραμμα πρώτα ζητά να δοθεί το όνομα χρήστη του νέου χρήστη “user name” όπου έπειτα από τον έλεγχο ότι δεν ανήκει σε άλλον χρήστη αποθηκεύετε προσωρινά μέχρι να συμπληρωθούν τα υπόλοιπα στοιχεία.

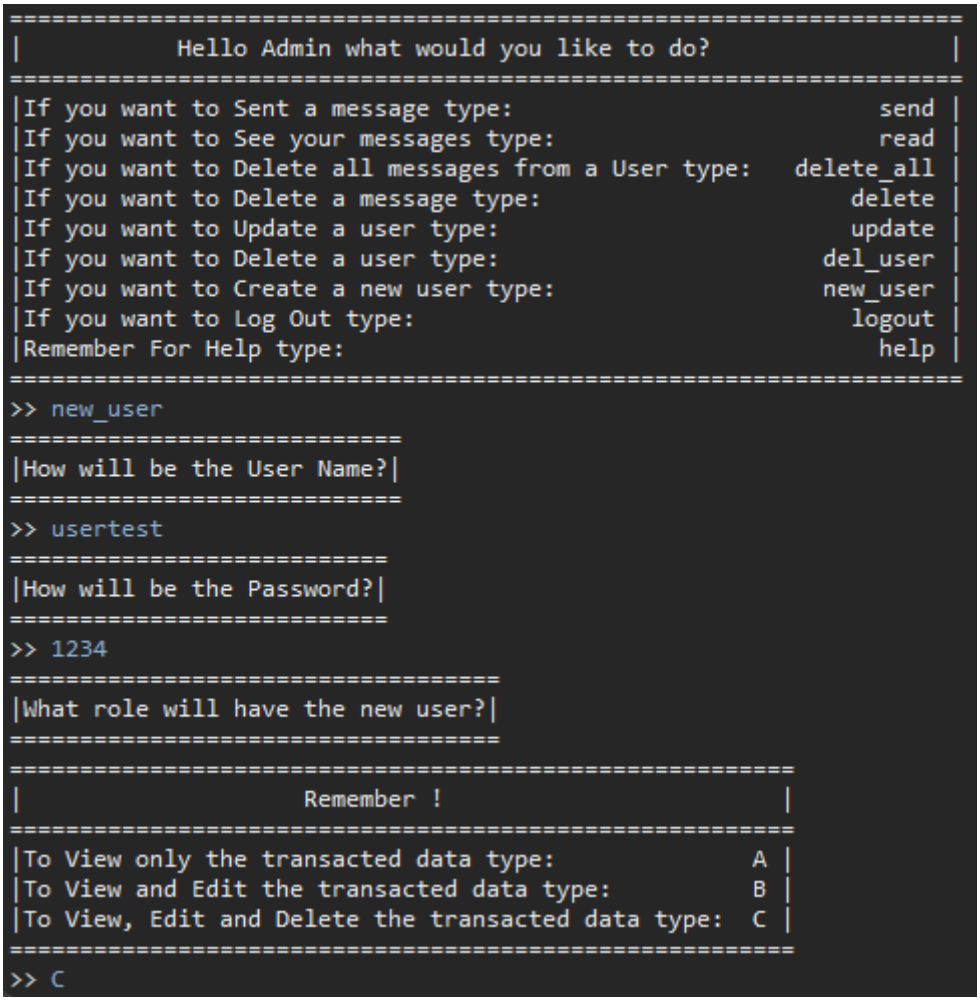
Έπειτα το πρόγραμμα ζητά να δοθεί ένας κωδικός πρόσβασης “password” όπου αφορά τον νέο χρήστη. Πρέπει να σημειωθεί ότι δεν υπάρχει κάποιος περιορισμός για τον κωδικό του νέου χρήστη με σκοπό να επιτευχθεί απλότητα στο πρόγραμμα.

Τέλος παρουσιάζετε πίνακας με τις συντομεύσεις και επεξηγήσεις των Κατηγοριών με τα δικαιώματα όπου μπορεί να δώσει ο “Admin” για τον νέο χρήστη.

Όταν συμπληρωθούν όλα τα αναγκαία πεδία για την δημιουργία του νέου χρήστη τότε το πρόγραμμα τα καταχωρεί σαν μια νέα εγγραφή στη βάση δεδομένων.



Εικόνα 14.19: Μενού για την δημιουργία νέου χρήστη. Εισαγωγή ονόματος χρήστη ”user name”.



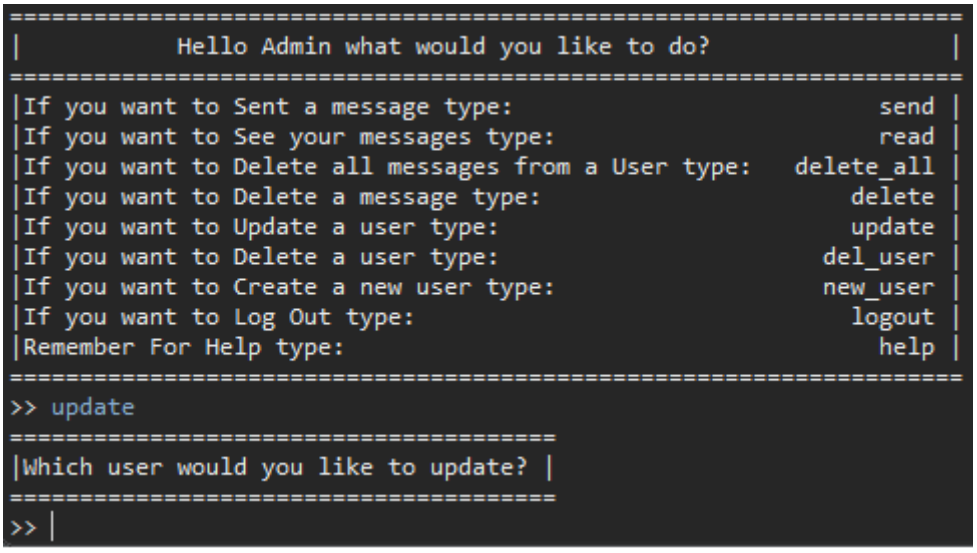
Εικόνα 14.20: Μενού για την δημιουργία νέου χρήστη. Εισαγωγή κωδικού και Κατηγορίας δικαιωμάτων του νέου χρήστη.

14.5.2 Διαδικασία αλλαγής ονόματος χρήστη ή κωδικό ή κατηγορία δικαιωμάτων ενός επιλεγμένου χρήστη.

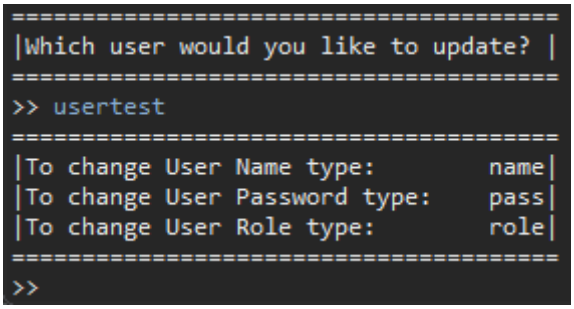
Για την εκκίνηση της διαδικασίας αλλαγής ονόματος χρήστη ή κωδικό ή κατηγορία δικαιωμάτων ενός επιλεγμένου χρήστη, ο χρήστης με δικαιώματα “Admin” πρέπει να εισάγει την εντολή “update” στο κεντρικό μενού.

Με την εισαγωγή της εντολής το πρόγραμμα πρώτα ζητά από τον χρήστη να επιλέξει ποια από τις τρεις μεταβλητές επιθυμεί να αλλάξει, το όνομα χρήστη (username), τον κωδικό πρόσβασης του (password) ή την κατηγορία δικαιωμάτων του (role).

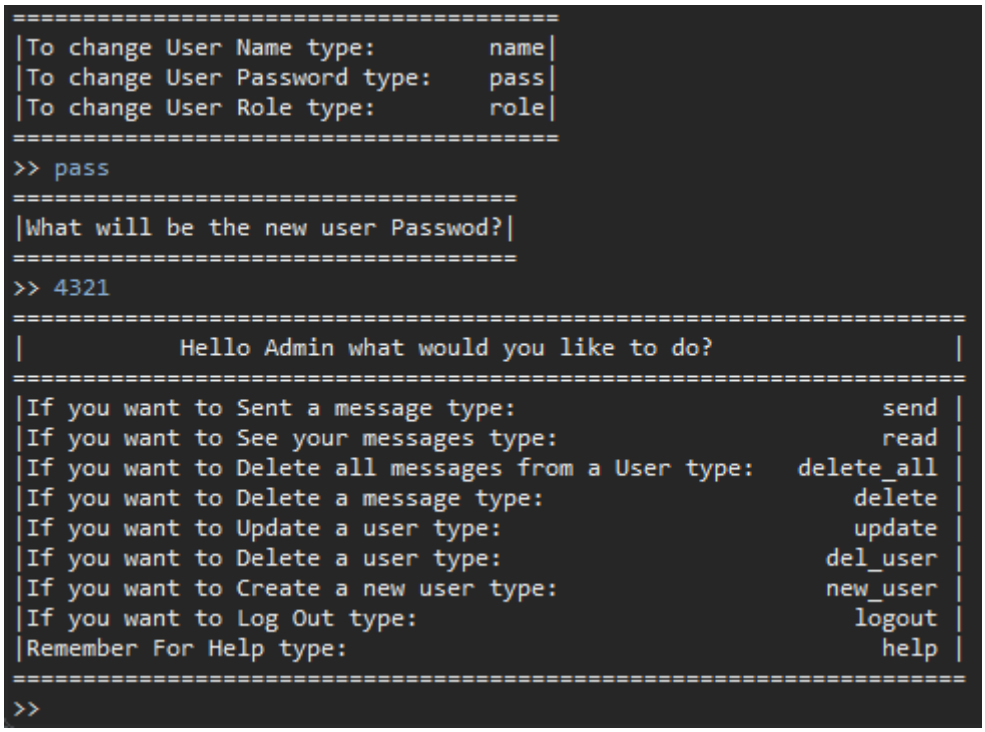
Η διαδικασία είναι κοινή για οποιαδήποτε από τις τρεις προαναφερθέντες μεταβλητές επιθυμεί να τροποποιήσει ο χρήστης. Το πρώτο βήμα όπου ζητά το πρόγραμμα είναι να δοθεί το όνομα του χρήστη όπου επιθυμείτε η οποιαδήποτε αλλαγή. Έπειτα εμφανίζετε ένας πίνακας με τις εντολές για τον καθορισμό ποιάς μεταβλητής θα τροποποιηθεί. Τέλος το πρόγραμμα ζητά την νέα τιμή της μεταβλητής όπου επιλέχτηκε από το προηγούμενο βήμα και έπειτα την αποθηκεύει στη βάση δεδομένων.



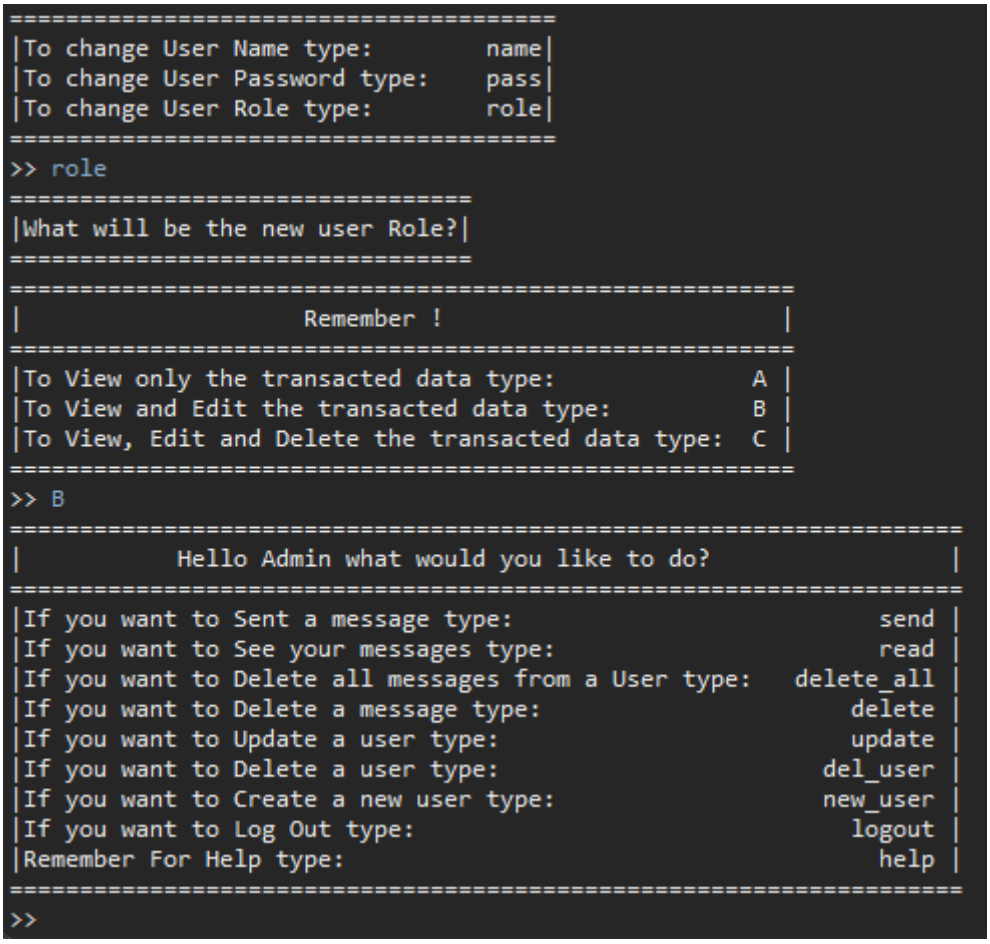
Εικόνα 14.21: Μενού για την τροποποίηση ενός χρήστη. Εισαγωγή εντολής ”update”.



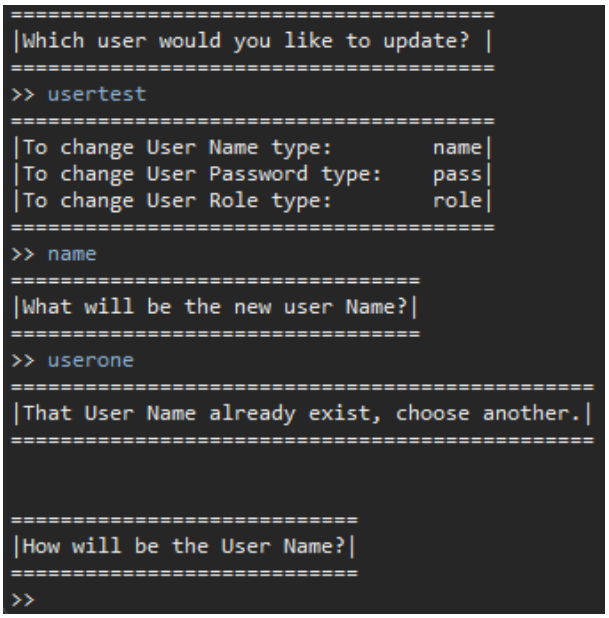
Εικόνα 14.22: Μενού για την επιλογή μεταβλητής τροποποίησης του επιλεγμένου χρήστη.



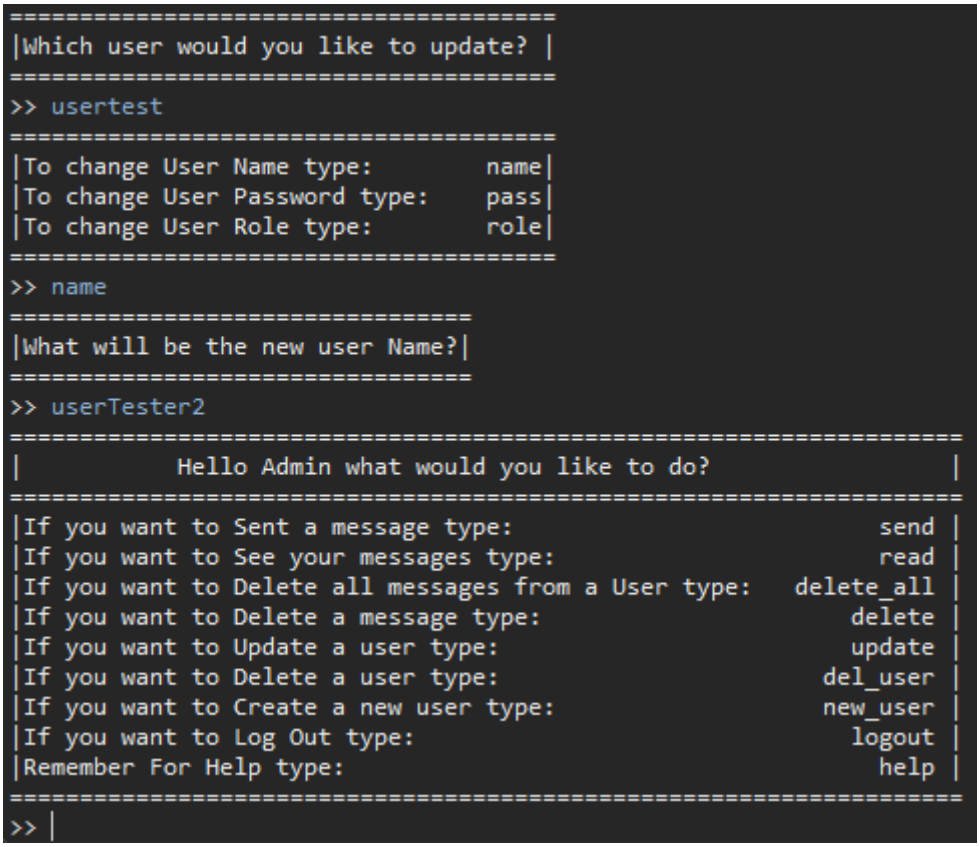
Εικόνα 14.23: Τροποποίηση κωδικού πρόσβασης (password)



Εικόνα 14.24: Τροποποίηση κατηγορίας δικαιωμάτων του χρήστη (role)



Εικόνα 14.25: Μη επιτρεπτή τροποποίηση ονόματος χρήστη (username) για την περίπτωση όπου το νέο όνομα χρήστη χρησιμοποιείτε από άλλον χρήστη.



Εικόνα 14.26: Επιτρεπτή τροποποίηση ονόματος χρήστη (username) για την περίπτωση όπου το νέο όνομα χρήστη δεν χρησιμοποιείτε από άλλον χρήστη.

14.5.3 Διαδικασία διαγραφής ενός χρήστη.

Για την εκκίνηση της διαδικασίας διαγραφής ενός επιλεγμένου χρήστη, ο χρήστης με δικαιώματα “Admin” πρέπει να εισάγει την εντολή “del_user” στο κεντρικό μενού.

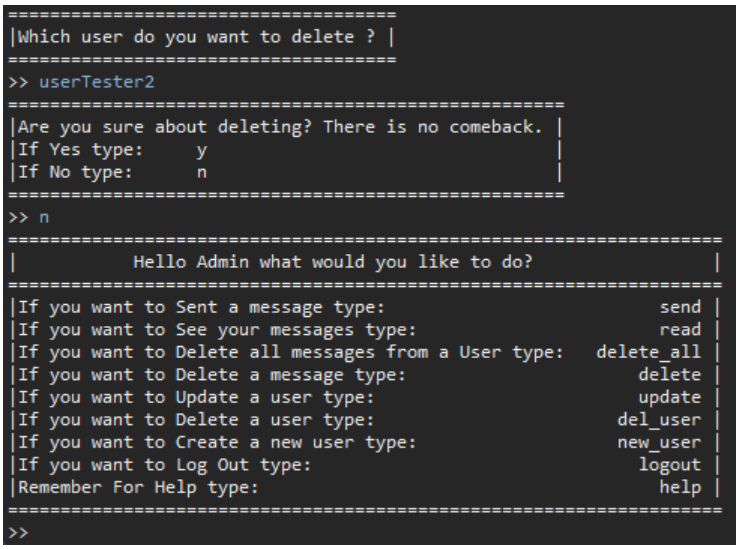
Με την εισαγωγή της εντολής το πρόγραμμα πρώτα ζητά από τον χρήστη να επιλέξει το όνομα χρήστη (username) όπου επιθυμεί να διαγράψει.

Αφού πραγματοποιηθεί έλεγχος για την εγκυρότητα του ονόματος χρήστη, το πρόγραμμα ζητά επιβεβαίωση ότι σίγουρα επιθυμείτε η μόνιμη διαγραφή του επιλεγμένου χρήστη.

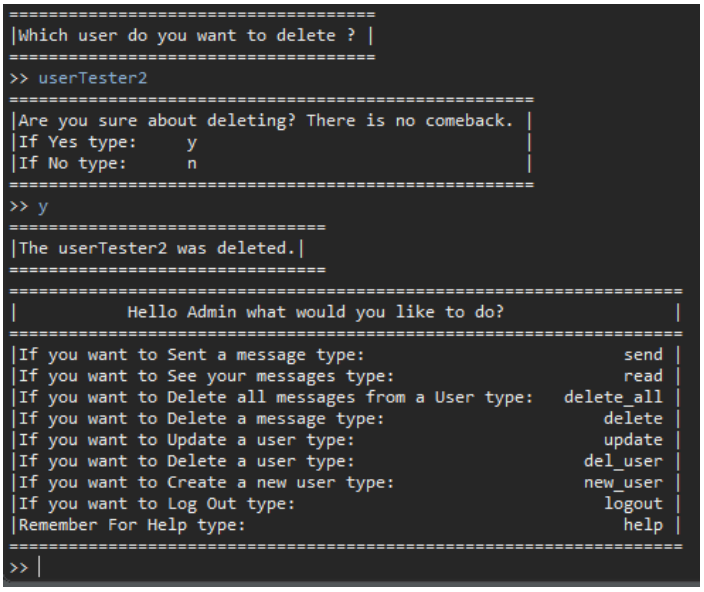
Για την επιβεβαίωση αυτή το πρόγραμμα ζητά από τον χρήστη αν άλλαξε γνώμη και δεν επιθυμεί την μόνιμη διαγραφή του μηνύματος, πληκτρολογώντας τον χαρακτήρα “n” και το πρόγραμμα επιστρέφει στο αρχικό μενού του χρήστη της κατηγορίας αυτής. Αντίθετα αν ο χρήστης είναι απόλυτα σίγουρος ότι επιθυμεί να

προβεί σε μόνιμή διαγραφή τότε πληκτρολογεί τον χαρακτήρα “y”, με αποτέλεσμα η βάση δεδομένων να αλλάζει τη τιμή της στήλης “EXIST” από “YES” σε “NO” στον πίνακα όπου αφορά το επιλεγμένο όνομα χρήστη.

Αυτή η ενέργεια πραγματοποιείτε διότι αν διαγραφόταν η εγγραφή του χρήστη θα έπρεπε εν συνεχεία να διαγραφούν και όλα τα μηνύματα όπου έχει αποστείλει ή λάβει.



Εικόνα 14.27: Μενού για την διαγραφή ενός χρήστη. Αλλαγή απόφασης και διατήρησης του χρήστη ενεργού.



Εικόνα 14.28: Μενού για την διαγραφή ενός χρήστη. Επιβεβαίωση απόφασης και μη διατήρησης του χρήστη ενεργού.

ΠΑΡΑΡΤΗΜΑ Α’

Στο παράρτημα αυτό παρατίθεται ο κώδικας ανάπτυξης της παρούσας εφαρμογής.

Το πρόγραμμα επίδειξης της πτυχιακής εργασίας χωρίζετε σε τέσσερα πακέτα και σε επτά κλάσεις.

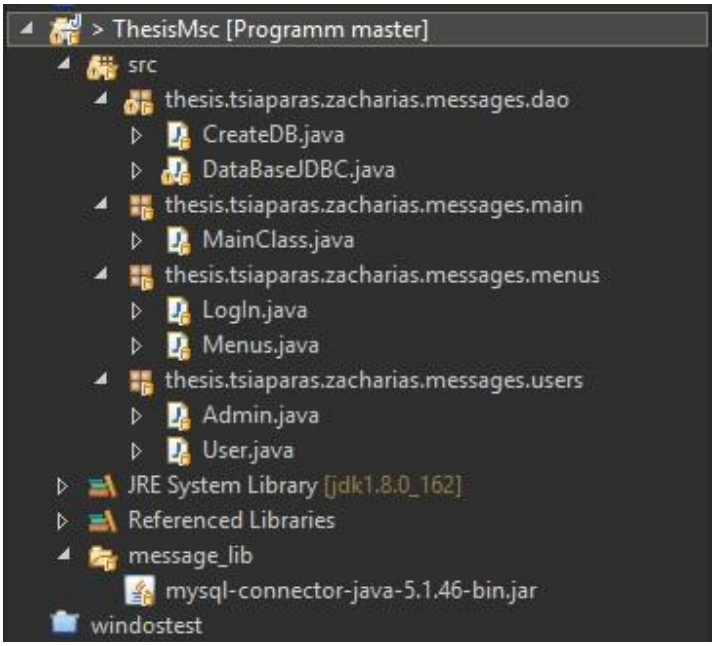
Το πρώτο πακέτο “thesis.tsiaparas.zacharias.messages.dao;” περιλαμβάνει τις κλάσεις όπου αφορούν τη σύνδεση και αλληλεπίδραση με τη βάση δεδομένων καθώς και τη δημιουργία και αρχικοποίηση των πινάκων για την περίπτωση όπου δεν υφίσταντο την πρώτη φορά όπου θα τρέξει το πρόγραμμα.

Το δεύτερο πακέτο “thesis.tsiaparas.zacharias.messages.main;” περιλαμβάνει την κύρια κλάση όπου θα ξεκινήσει όλο το πρόγραμμα.

Το τρίτο πακέτο “thesis.tsiaparas.zacharias.messages.menus;” περιλαμβάνει τις κλάσεις για την παρουσίαση όλων των μενού του προγράμματος.

Το τέταρτο πακέτο “thesis.tsiaparas.zacharias.messages.users;” περιλαμβάνει τις κλάσεις όπου αφορούν τις ξεχωριστές κατηγορίες δικαιωμάτων των χρηστών.

Τέλος πρέπει να αναφερθεί ότι πρέπει να γίνει εισαγωγή της βιβλιοθήκης “mysql-connector-java-5.1.46-bin.jar” για την πραγματοποίηση της σύνδεσης με την βάση δεδομένων.



Εικόνα 15.1: Απεικόνιση πακέτων και κλάσεων του προγράμματος επίδειξης.

Πακέτο “thesis.tsiaparas.zacharias.messages.dao;” Κλάση CreateDB.

Στην συγκεκριμένη κλάση παρουσιάζετε ο τρόπος σύνδεσης με τη βάση δεδομένων καθώς και η δημιουργία και αρχικοποίηση των πινάκων όπου θα χρειαστούν από το πρόγραμμα στην περίπτωση όπου δεν υπάρχουν.

```
package thesis.tsiaparas.zacharias.messages.dao;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;


/**
 * @author Zacharias Tsiaparas
 */
public class CreateDB {
    private static final String MYSQLURL =
"jdbc:mysql://localhost/bootcampdb?autoReconnect=true&useSSL=false";
    private static final String USERNAME = "root";
    private static final String PASS = "root";
    private Connection con = null;
    private String sqlselect;
    private Statement statement = null;
    private String sqlselect1 = "USE Project1";


    /*
     * ================================================
     * =============CREATE A DATABASE=================
     * ================================================
     */
    public CreateDB() throws ClassNotFoundException {
       connectDB();
       createDB();
       createTableRegistration();
       createTableMessages();
```

```
    }// End of constructor

    // ======================================================

    private void connectDB() throws ClassNotFoundException {
       try {
          Class.forName("com.mysql.jdbc.Driver");
          con = DriverManager.getConnection(MYSQLURL, USERNAME, PASS);
          // System.out.println("DataBase connect");
       } catch (SQLException ex) {
          Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
       }
    }// End of connectDB()

    // ======================================================

    private void createDB() {
       try {
          sqlselect = "CREATE DATABASE IF NOT EXISTS Project1 DEFAULT CHARACTER SET utf8";
          statement = con.createStatement();
          statement.executeUpdate(sqlselect);
          System.out.println("DataBase create");
       } catch (SQLException ex) {
          Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
       }
    }// End of createDB()

    // ======================================================

    private void createTableRegistration() {
       try {
          String sqlselect2 = "DROP TABLE IF EXISTS REGISTRATION";
          System.out.println("Creating table REGISTRATION in given database...");
          statement = con.createStatement();
          String sql = "CREATE TABLE REGISTRATION "
          + "(ID_REGIST INTEGER not NULL AUTO_INCREMENT, "
          + " USERNAME VARCHAR(25), "
```

```
          + " PASSWORD VARCHAR(25), " + " ROLE ENUM('A','B','C'), "
          + " EXIST ENUM('YES','NO')," + " PRIMARY KEY ( ID_REGIST ))";
          statement.executeUpdate(sqlselect1);
          statement.executeUpdate(sqlselect2);
          statement.executeUpdate(sql);
          System.out.println("Created table in given database...");
          insertTableRegistration();
       } catch (SQLException ex) {
          Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
       }
    }// End of createTableRegistration()

    // ======================================================

    private void insertTableRegistration() {
       String sql = "INSERT INTO REGISTRATION VALUES (1, 'admin', 'aDmI3$', 'C','YES')";
       try {
          statement.executeUpdate(sqlselect1);
          statement.executeUpdate(sql);
          System.out.println("The admin were created");
       } catch (SQLException e) {
          e.printStackTrace();
       }
    }// End of insertTableRegistration()

    // ======================================================

    private void createTableMessages() {
       try {
          String sqlselect2 = "DROP TABLE IF EXISTS MESSAGES";
          System.out.println("Creating table MESSAGES in given database...");
          statement = con.createStatement();
          String sql = "CREATE TABLE MESSAGES " + "(ID_MESSAGE int not NULL
AUTO_INCREMENT, " + " DATE DATETIME, "
          + " SENDER VARCHAR(20), " + " RECEIVER VARCHAR(20), "
          + " MESSAGE_DATA VARCHAR(250), "
          + " USER_ID INT, " + " PRIMARY KEY (ID_MESSAGE)," + "CONSTRAINT FK_USER_ID "
          + " FOREIGN KEY (USER_ID) REFERENCES REGISTRATION(ID_REGIST)"
```

```
          + "ON DELETE CASCADE)";
          statement.executeUpdate(sqlselect1);
          statement.executeUpdate(sqlselect2);
          statement.executeUpdate(sql);
          System.out.println("Created table in given database...");
          insertTableMessages();
       } catch (SQLException ex) {
          Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
       }
    }// End of createTableMessages()

    // ======================================================

    private void insertTableMessages() {
       String sql = "INSERT INTO MESSAGES VALUES (1, '2018-04-15 12:00:00', 'admin',
'admin','hello ',1)";
       try {
          statement.executeUpdate(sqlselect1);
          statement.executeUpdate(sql);
          System.out.println("The dummy message of user were created");
       } catch (SQLException e) {
          e.printStackTrace();
       }
    }// End of insertTableMessages()

}// End of Class CreateDB
```

Πακέτο “thesis.tsiaparas.zacharias.messages.dao;” Κλάση DataBaseJDBC.

Στην συγκεκριμένη κλάση παρουσιάζονται οι μέθοδοι για την επικοινωνία με την βάση δεδομένων καθώς και η προετοιμασία και έλεγχος των συναλλαγών που θα πραγματοποιήσει το πρόγραμμα με την βάση δεδομένων.

```
package thesis.tsiaparas.zacharias.messages.dao;


/**
 * @author Zacharias Tsiaparas
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;


import thesis.tsiaparas.zacharias.messages.menus.Menus;


public class DataBaseJDBC {


    private static final String MYSQLURL =
"jdbc:mysql://localhost/bootcampdb?autoReconnect=true&useSSL=false";
    private static final String USERNAME = "root";
    private static final String PASS = "agent900";


    private static Connection con = null;
    private static ResultSet rs = null;
    private static PreparedStatement stm = null;
    private static Statement statement = null;
    private static String userName;
    private static String password;
    private static String userId;
    private static String sqlselect1 = "USE Project1";
```

```
public DataBaseJDBC() {
   try {
      connectDB();
   } catch (ClassNotFoundException e) {
      e.printStackTrace();
   }
}// End of constructor

// ================================================================================

public DataBaseJDBC(String userName, String password) {
   DataBaseJDBC.userName = userName;
   DataBaseJDBC.password = password;
   try {
      connectDB();
   } catch (ClassNotFoundException e) {
      e.printStackTrace();
   }
}// End of constructor

// ================================================================================

public void connectDB() throws ClassNotFoundException {
   try {
      Class.forName("com.mysql.jdbc.Driver");
      con = DriverManager.getConnection(MYSQLURL, USERNAME, PASS);
      System.out.println("==================");
      System.out.println("|DataBase connect|");
      System.out.println("==================");
   } catch (SQLException ex) {
      Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
   }
}// End of connectDB()

// ================================================================================
```

```
    public static String checkPassword(String userName) {
       try {
          statement = con.createStatement();
          statement.executeUpdate(sqlselect1);
          String sql = "SELECT PASSWORD FROM REGISTRATION WHERE USERNAME='" + userName +
"';";
          rs = statement.executeQuery(sql);
          rs.next();
          return rs.getString(1);
       } catch (SQLException e) {
          e.printStackTrace();
          return "|  The user does not exist.  |";
       }
    }// End of checkPassword()

    // ================================================================================

    public static boolean checkUserName(String userName) {
       ArrayList<String> userNames = new ArrayList<>();
       try {
          statement = con.createStatement();
          statement.executeUpdate(sqlselect1);
          String sql = "SELECT USERNAME FROM REGISTRATION ";
          rs = statement.executeQuery(sql);
          while (rs.next()) {
             userNames.add(rs.getString(1));
          }
       } catch (SQLException e) {
          e.printStackTrace();
       }
       for (String e : userNames) {
          if (userNames.contains(userName)) {
             return true;
          }
       }
       return false;
    }// End of checkUserName()
```

```
    // ================================================================================


    public static void insertTableMessages(String date, String sender, String receiver,
String messageData) {
       userId = getUserId(receiver);
       if (userId != "null") {
          String sql = "INSERT INTO
    MESSAGES(DATE,SENDER,RECEIVER,MESSAGE_DATA,USER_ID)VALUES ('"
          + date + "', '"
          + sender + "', '" + receiver + "','" + messageData + "'," + userId + ")";
          try {
             statement = con.createStatement();
             statement.executeUpdate(sqlselect1);
             statement.executeUpdate(sql);
             System.out.println("============================");
             System.out.println("|The message was sended ! |");
             System.out.println("============================");
             new Menus(DataBaseJDBC.getUserName(),
              DataBaseJDBC.getPassword(),
DataBaseJDBC.getUserRole(DataBaseJDBC.getUserName()));
          } catch (SQLException e) {
             e.printStackTrace();
          }
       }
    }// End of insertTableMessages()


    // ================================================================================


    public String getMessageId() {
       try {
          statement = con.createStatement();
          statement.executeUpdate(sqlselect1);
          String sqlselect = "SELECT COUNT(ID_MESSAGE) FROM MESSAGES";
          rs = statement.executeQuery(sqlselect);
          rs.next();
          String idMessage = rs.getString(1);
          int newId = Integer.parseInt(idMessage);
          newId += 1;
          String id_Message = Integer.toString(newId);
```

```
          return id_Message;
       } catch (SQLException e) {
          e.printStackTrace();
          return "1";
       }
    }// End of getMessageId()


    // ================================================================================


    public static String getUserId(String receiver) {
       String sqlselect = "SELECT ID_REGIST FROM REGISTRATION WHERE USERNAME='" +
receiver + "'";
       try {
          statement = con.createStatement();
          statement.executeUpdate(sqlselect1);
          rs = statement.executeQuery(sqlselect);
          rs.next();
          String id_receiver = rs.getString(1);
          if (id_receiver != null) {
             return id_receiver;
          } else {
             return "null";
          }
       } catch (SQLException e) {
          e.printStackTrace();
          return "null";
       }
    }// End of getUserId()


    // ================================================================================


    public static void deleteTableMessages(String sender, String receiver) {
       try {
          String sqlselect2 = "DELETE FROM MESSAGES WHERE SENDER='" + sender + "' AND
RECEIVER='" + receiver + "';";
          statement = con.createStatement();
          statement.executeUpdate(sqlselect1);
          statement.executeUpdate(sqlselect2);
```

```
            System.out.println("=====================================================");
            System.out.println("|All the messages from " + receiver + " were deleted.|");
            System.out.println("=====================================================");
        } catch (SQLException ex) {
            Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
        }
    }// End of deleteTableMessages()

    // ============================================================================

    public static void deleteMessage(String id) {
        try {
            String sqlselect2 = "DELETE FROM MESSAGES WHERE ID_MESSAGE='" + id + "';";
            statement = con.createStatement();
            statement.executeUpdate(sqlselect1);
            statement.executeUpdate(sqlselect2);
            System.out.println("=========================");
            System.out.println("|The message was deleted.|");
            System.out.println("=========================");
        } catch (SQLException ex) {
            Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
        }
    }// End of deleteMessage()

    // ============================================================================

    public static String readTableMessages(String userName) {
        String sqlselect = "SELECT * FROM MESSAGES WHERE RECEIVER='" + userName + "'";
        try {
            statement = con.createStatement();
            statement.executeUpdate(sqlselect1);
            rs = statement.executeQuery(sqlselect);
            StringBuilder message = new StringBuilder();
            while (rs.next()) {
                String id = rs.getString(1);
                String date = rs.getString(2);
                String sender = rs.getString(3);
```

```
            String receiver = rs.getString(4);
            String messageDb = rs.getString(5);
            message.append("ID: ");
            message.append(id + "\n");
            message.append("Date: ");
            message.append(date + "\n");
            message.append("Sender: ");
            message.append(sender + "\n");
            message.append("Receiver: ");
            message.append(receiver + "\n");
            message.append("Message Data: ");
            message.append(messageDb + "\n\n");
         }
         return message.toString();
      } catch (SQLException e) {
         e.printStackTrace();
         return "null";
      }
   }// End of readTableMessages()

   // ========================================================================

   public static void editDataBase(String id, String date, String newMessage) {
      String sqlselect = "UPDATE MESSAGES SET DATE ='" + date + "', MESSAGE_DATA = '" +
newMessage
            + "' WHERE ID_MESSAGE='" + id + "'";
      try {
         statement = con.createStatement();
         statement.executeUpdate(sqlselect1);
         statement.executeUpdate(sqlselect);
         System.out.println("============================");
         System.out.println("|The new message was edited|");
         System.out.println("============================");
      } catch (SQLException e) {
         e.printStackTrace();
      }
   }// End of editDataBase()
   // ========================================================================
```

```
    public static String getUserRole(String user) {
       String sqlselect = "SELECT ROLE FROM REGISTRATION WHERE USERNAME='" + user + "'";
       try {
          statement = con.createStatement();
          statement.executeUpdate(sqlselect1);
          rs = statement.executeQuery(sqlselect);
          String receiverRole = null;

          rs.next();
          receiverRole = rs.getString(1);

          if (receiverRole != null) {
             return receiverRole;
          } else {
             return "null";
          }

       } catch (SQLException e) {
          e.printStackTrace();
          return "null";
       }
    }// End of getUserRole()

    // ================================================================================

    public static void createUserIntoDB(String newUserName, String newPassword, String
role) {
       String sql = "INSERT INTO REGISTRATION(USERNAME,PASSWORD,ROLE,EXIST) VALUES ('" +
newUserName + "', '"
             + newPassword + "', '" + role + "','YES')";
       try {
          statement.executeUpdate(sqlselect1);
          statement.executeUpdate(sql);
          System.out.println("The " + newUserName + " were created");
       } catch (SQLException e) {
          e.printStackTrace();
       }
    }// End of createUserIntoDB()
```

```
    // ================================================================================

    public static void deleteUserFromDB(String user) {
       try {
          String sqlselect2 = "UPDATE REGISTRATION SET EXIST = 'NO' WHERE USERNAME='" +
user + "';";
          statement = con.createStatement();
          statement.executeUpdate(sqlselect1);
          statement.executeUpdate(sqlselect2);
          System.out.println("================================");
          System.out.println("|The " + user + " was deleted.|");
          System.out.println("================================");
       } catch (SQLException ex) {
          Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
       }
    }// End of deleteUserFromDB()

    // ================================================================================

    public static void updateUserFromDB(int i, String user, String change) {
       String sql = null;
       try {
          switch (i) {
          case 1:
             sql = "UPDATE REGISTRATION set USERNAME='" + change + "' WHERE USERNAME='"
             + user + "'";
             break;
          case 2:
             sql = "UPDATE REGISTRATION set PASSWORD='" + change + "' WHERE USERNAME='"
             + user + "'";
             break;
          case 3:
             sql = "UPDATE REGISTRATION set ROLE='" + change + "' WHERE USERNAME='"
             + user + "'";
             break;
          }
          statement.executeUpdate(sqlselect1);
```

```
            stm = con.prepareStatement(sql);
            stm.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }// End of updateUserFromDB()

    // ============================================================================

    public static Boolean getExist(String user) {
        String result = null;
        String sql = null;
        try {
            sql = "SELECT EXIST FROM REGISTRATION WHERE USERNAME='" + user + "';";
            statement = con.createStatement();
            statement.executeUpdate(sqlselect1);
            rs = statement.executeQuery(sql);
            rs.next();
            result = rs.getString(1);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        if (result.equals("YES")) {
            return true;
        } else {
            return false;
        }
    }// End of getExist()

    // ============================================================================

    public static void closeDB() {
        try {
            if (rs != null) {
                rs.close();
                //System.out.println("ResultSet closed!");
            }
```

```
          if (statement != null) {
             statement.close();
             //System.out.println("Statement closed!");
          }
          if (con != null) {
             con.close();
             //System.out.println("Connection closed!");
          }
          //System.out.println("All closed!");
       } catch (Exception e) {
       }
    }// End of closeDB()
    // ================================================================================
    public static String getUserName() {
       return userName;
    }

    public static String getPassword() {
       return password;
    }

    public static void setUserName(String userName) {
       DataBaseJDBC.userName = userName;
    }

    public static void setPassword(String password) {
       DataBaseJDBC.password = password;
    }

}// End of Class DataBaseJDBC
```

Πακέτο “thesis.tsiaparas.zacharias.messages.menus;” Κλάση Login.

Στην συγκεκριμένη κλάση παρουσιάζονται οι μέθοδοι για την έλεγχο της εγκυρότητας του χρήστη. Δηλαδή αν το όνομα χρήστη και ο κωδικός όπου δίνετε είναι ίδιος με την εγγραφή στο πίνακα της βάσης δεδομένων.

```
package thesis.tsiaparas.zacharias.messages.menus;

import java.util.Scanner;
import thesis.tsiaparas.zacharias.messages.dao.DataBaseJDBC;

/**
 * @author Zacharias Tsiaparas
 */
public class LogIn {

	private String userName;
	private String password, passwordCheck, role;
	private Boolean userNameCheck;
	private Scanner scanner;

	public LogIn(String userName, String password) {
	    this.userName = userName;
	    this.password = password;
	    new DataBaseJDBC();
	    checkAutheticationUserName();
	}// End of constructor

	// ======================================================

	private void checkAutheticationUserName() {
	    userNameCheck = DataBaseJDBC.checkUserName(getUserName());
	    if (userNameCheck && (DataBaseJDBC.getExist(getUserName()) == true)) {
	        DataBaseJDBC.setUserName(getUserName());
	        this.passwordCheck = DataBaseJDBC.checkPassword(getUserName());
	        checkAutheticationPassword();
```

```
        } else {
            System.out.println("================================");
            System.out.println("|The User Name is not correct. |");
            System.out.println("================================");
            userNameRepeat();
        }
    }// End of checkAutheticationUserName()

    // ========================================================


    private void userNameRepeat() {
        scanner = new Scanner(System.in);
        System.out.println("=========================");
        System.out.println("|Give me your User Name: |");
        System.out.println("=========================");
        System.out.print(">> ");
        userName = scanner.nextLine();
        setUserName(userName);
        checkAutheticationUserName();
    }// End of userNameRepeat()

    // ========================================================

    private void checkAuthetication Password() {
        if (this.passwordCheck.equals(this.getPassword())) {
            System.out.println("=========================");
            System.out.println("Hello " + getUserName().toString() + " !");
            System.out.println("=========================");
            DataBaseJDBC.setPassword(getPassword());
            goToMenu();
        } else {
            System.out.println("============================");
            System.out.println("|The password is not correct.|");
            System.out.println("============================");
```

```
            passwordRepeat();
        }
    }// End of checkAutheticationPassword()

    // ========================================================

    private void passwordRepeat() {
        scanner = new Scanner(System.in);
        System.out.println("========================");
        System.out.println("|Give me your Password: |");
        System.out.println("========================");
        System.out.print(">> ");
        password = scanner.nextLine();
        setPassword(password);
        checkAutheticationPassword();
    }// End of passwordRepeat()

    // ========================================================

    private void goToMenu() {
        this.role = DataBaseJDBC.getUserRole(getUserName());
        if (getUserName().equals("admin")) {
            new Menus(getUserName(), getPassword(), getRole());
        } else {
            new Menus(getUserName(), getPassword(), getRole());
        }
    }// End of goToMenu()

    // ========================================================

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
```

```
        this.userName = userName;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getPassword() {
        return password;
    }

    public String getRole() {
        return role;
    }

}// End of Class LogIn
```

Πακέτο “thesis.tsiaparas.zacharias.messages.menus;” Κλάση Menus.

Στην συγκεκριμένη κλάση παρουσιάζονται οι μέθοδοι για την παρουσίαση όλων των μηνυμάτων που απεικονίζονται στα εκάστοτε μενού στο πρόγραμμα.

```
package thesis.tsiaparas.zacharias.messages.menus;

import java.util.Scanner;
import thesis.tsiaparas.zacharias.messages.dao.DataBaseJDBC;
import thesis.tsiaparas.zacharias.messages.users.Admin;
import thesis.tsiaparas.zacharias.messages.users.User;

/**
 * @author Zacharias Tsiaparas
 */
public class Menus {

    private String userName = null;
    private String password = null;
    private String command = null;
    private User user;
    private String role;
    Scanner scanner = new Scanner(System.in);

    public Menus() {
        starterMenu();
    }// End of constructor

    // ============================================================

    public Menus(String userName, String password, String role) {
        this.userName = userName;
        this.password = password;
        this.role = role;
        if (getUserName().equals("admin")) {
            adminMenu(getUserName(), getPassword());
```

```
        } else {
            user = new User(getUserName(), getPassword());
            switch (getRole()) {
            case "A":
                userMenuA(getUserName(), getPassword());
                break;
            case "B":
                userMenuB(getUserName(), getPassword());
                break;
            case "C":
                userMenuC(getUserName(), getPassword());
                break;
            }
        }
    }// End of constructor

    // ============================================================

    private void starterMenu() {
        helpMenuStart();
        do {
            System.out.print(">> ");
            command = scanner.nextLine();
            switch (command) {
            case "login":
                logInMenu();
                break;
            case "exit":
                System.exit(0);
                break;
            case "help":
                helpMenuStart();
                break;
            default:
                System.out.println("================================");
```

```
                System.out.println("|Remember For Help type:  help |");
                System.out.println("================================");
                break;
            }
        } while (true);
    }// End of starterMenu()

    // ============================================================

    private void helpMenuStart() {
        System.out.println("================================================");
        System.out.println("|  Hello fellow user what would you like to do?    |");
        System.out.println("================================================");
        System.out.println("|If you want to login to your account type:   login|");
        System.out.println("|If you want to log out of application type:   exit|");
        System.out.println("|Remember For Help type:                      help|");
        System.out.println("================================================");
    }// End of helpMenuStart()

    // ============================================================

    private void logInMenu() {
        System.out.println("=========================");
        System.out.println("|Give me your User Name: |");
        System.out.println("=========================");
        System.out.print(">> ");
        this.userName = scanner.nextLine();
        System.out.println("============================");
        System.out.println("|Now give me your Password: |");
        System.out.println("============================");
        System.out.print(">> ");
        this.password = scanner.nextLine();
        new LogIn(userName, password);
    }// End of logInMenu()
```

```
// ============================================================


private void userMenuA(String userName, String password) {
    do {
        helpUserMenuA();
        System.out.print(">> ");
        command = scanner.nextLine();
        switch (command) {
        case "read":
            user.readFromDataBase();
            break;
        case "send":
            user.typeMessage();
            break;
        case "logout":
            DataBaseJDBC.closeDB();
            starterMenu();
            break;
        case "help":
            helpUserMenuA();
            break;
        default:
            System.out.println("================================");
            System.out.println("|Remember For Help type:  help |");
            System.out.println("================================");
            break;
        }
    } while (true);
}// End of userMenuA()

// ============================================================

private void helpUserMenuA() {

System.out.println("=================================================");
```

```
        System.out.println("|           What would you like to do?            |");

    System.out.println("=====================================================");
        System.out.println("|If you want to See your messages type:        read |");
        System.out.println("|If you want to Sent a message type:           send |");
        System.out.println("|If you want to Log Out type:                logout |");
        System.out.println("|Remember For Help type:                        help |");

    System.out.println("=====================================================");
    }// End of helpUserMenuA()

    // ============================================================

    private void userMenuB(String userName, String password) {
        do {
            helpUserMenuB();
            System.out.print(">> ");
            command = scanner.nextLine();
            switch (command) {
            case "send":
                user.typeMessage();
                break;
            case "read":
                user.readFromDataBase();
                break;
            case "edit":
                user.editMessage();
                break;
            case "logout":
                DataBaseJDBC.closeDB();
                starterMenu();
                break;
            case "help":
                helpUserMenuB();
                break;
            default:
```

```
                System.out.println("================================");
                System.out.println("|Remember For Help type:  help |");
                System.out.println("================================");
                break;
            }
        } while (true);
    }// End of userMenuB()

    // ============================================================

    private void helpUserMenuB() {

    System.out.println("=====================================================");
        System.out.println("|          What would you like to do?             |");

    System.out.println("=====================================================");
        System.out.println("|If you want to Sent a message type:          send |");
        System.out.println("|If you want to See your messages type:        read |");
        System.out.println("|If you want to Edit your messages type:       edit |");
        System.out.println("|If you want to Log Out type:                logout |");
        System.out.println("|Remember For Help type:                        help |");

    System.out.println("=====================================================");
    }// End of helpUserMenuB()

    // ============================================================

    private void userMenuC(String userName, String password) {
        do {
            helpUserMenuC();
            System.out.print(">> ");
            command = scanner.nextLine();
            switch (command) {
            case "send":
                user.typeMessage();
                break;
            case "read":
```

```
                user.readFromDataBase();
                break;
            case "edit":
                user.editMessage();
                break;
            case "logout":
                DataBaseJDBC.closeDB();
                starterMenu();
                break;
            case "delete_all":
                user.deleteAllMessagesFromDataBase();
                break;
            case "delete":
                user.deleteMessage();
                break;
            case "help":
                helpUserMenuC();
                break;
            default:
                System.out.println("Remember For Help type: help");
                break;
            }
        } while (true);
    }// End of userMenuC()

    // ============================================================

    private void helpUserMenuC() {
        System.out.println("=================================================");
        System.out.println("|                What would you like to do?                |");
        System.out.println("=================================================");
        System.out.println("|If you want to Sent a message type:                send |");
        System.out.println("|If you want to See your messages type:               read |");
        System.out.println("|If you want to Edit your messages type:              edit |");
        System.out.println("|If you want to Delete all messages from a User type:   delete_all |");
```

```
        System.out.println("|If you want to Delete a message type:                delete |");
        System.out.println("|If you want to Log Out type:                       logout |");
        System.out.println("|Remember For Help type:                              help |");
        System.out.println("================================================");
    }// End of helpUserMenuC()

    // ============================================================

    private void adminMenu(String userName, String password) {
        Admin admin = new Admin(userName, password);
        do {
            helpAdminMenu();
            System.out.print(">> ");
            command = scanner.nextLine();
            switch (command) {
            case "send":
                admin.typeMessage();
                break;
            case "read":
                admin.readFromDataBase();
                break;
            case "logout":
                DataBaseJDBC.closeDB();
                starterMenu();
                break;
            case "delete_all":
                admin.deleteAllMessagesFromDataBase();
                break;
            case "delete":
                admin.deleteMessage();
                break;
            case "update":
                admin.updateUser();
                break;
            case "del_user":
```

```
                admin.deleteUser();
                break;
            case "new_user":
                admin.createUser();
                break;
            case "help":
                helpAdminMenu();
                break;
            default:
                System.out.println("Remember For Help type:  help");
                break;
            }
        } while (true);
    }// End of adminMenu()

    // ============================================================

    private void helpAdminMenu() {
        System.out.println("=====================================================");
        System.out.println("|          Hello Admin what would you like to do?               |");
        System.out.println("=====================================================");
        System.out.println("|If you want to Sent a message type:                    send |");
        System.out.println("|If you want to See your messages type:                  read |");
        System.out.println("|If you want to Delete all messages from a User type:   delete_all |");
        System.out.println("|If you want to Delete a message type:                  delete |");
        System.out.println("|If you want to Update a user type:                     update |");
        System.out.println("|If you want to Delete a user type:                    del_user |");
        System.out.println("|If you want to Create a new user type:                 new_user |");
        System.out.println("|If you want to Log Out type:                           logout |");
        System.out.println("|Remember For Help type:                                  help |");
        System.out.println("=====================================================");
    }// End of helpAdminMenu()

    // ============================================================
```

```
    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }

    public String getRole() {
        return role;
    }

}// End of Class Menus
```

Πακέτο “thesis.tsiaparas.zacharias.messages.users;” Κλάση User.

Στην συγκεκριμένη κλάση παρουσιάζονται οι μέθοδοι όπου εκτελούνται από τους χρήστες ανεξαρτήτως κατηγορίας δικαιωμάτων. Ο καθορισμός ποιων μεθόδων μπορεί να χρησιμοποιήσει η εκάστοτε κατηγορία πραγματοποιείται στη κλάση Menus.

```
package thesis.tsiaparas.zacharias.messages.users;

import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;
import thesis.tsiaparas.zacharias.messages.dao.DataBaseJDBC;

/**
 * @author Zacharias Tsiaparas
 */
public class User {

	private String userName;
	private String password;
	private String date, sender, receiver, messageData;
	private LocalDate now = LocalDate.now();
	private LocalTime time = LocalTime.now();
	private final int messageLength = 250;
	private String newMessage;
	Scanner scanner;

	public User(String userName, String password) {
	    this.userName = userName;
	    this.password = password;
	    scanner = new Scanner(System.in);
	}// End of constructor

	// ============================================================
```

```
public void typeMessage() {
    String input;
    System.out.println("========================================");
    System.out.println("|Who user do you want to send a message ?|");
    System.out.println("========================================");
    System.out.print(">> ");
    input = scanner.nextLine();
    this.receiver = input;
    if (DataBaseJDBC.checkUserName(receiver)) {
        writeMessage();
    } else {

System.out.println("======================================================");
        System.out.println("|The user who you want to send the message does not exist! |");

System.out.println("======================================================");
    }
}// End of typeMessage()

// ============================================================

protected void writeMessage() {
    String input;
    System.out.println("================================================");
    System.out.println("|Please enter your message.                    |");
    System.out.println("|Attention! Must be less than to 250 characters.|");
    System.out.println("================================================");
    System.out.print(">> ");
    do {
        input = scanner.nextLine();
        messageData = input;
        if (messageData.length() <= messageLength) {
            break;
        }
    } while (true);
    date = now.toString() + " " + time.toString();
```

```
            sender = getUserName();
            writeToDatabase();
        }// End of writeMessage()

        // ============================================================


        protected void writeToDatabase() {
            DataBaseJDBC.insertTableMessages(this.date, this.sender, this.receiver,
this.messageData);
        }// End of writeToDatabase()

        // ============================================================


        public void readFromDataBase() {
            date = now.toString() + " " + time.toString();
            String message = DataBaseJDBC.readTableMessages(getUserName());
            System.out.println(message);
        }// End of readFromDataBase()

        // ============================================================


        public void editMessage() {
            readFromDataBase();

        System.out.println("=================================================");
            System.out.println("|Please give me the ID of message which you like to edit.|");

        System.out.println("=================================================");
            System.out.print(">> ");
            String id = scanner.nextLine();
            System.out.println("==============================");
            System.out.println("|Please type the new message.|");
            System.out.println("==============================");
            System.out.print(">> ");
            newMessage = scanner.nextLine();
            date = now.toString() + " " + time.toString();
            DataBaseJDBC.editDataBase(id, date, newMessage);
```

```
    }// End of editMessage()

    // ============================================================

    public void deleteMessage() {
        String input;
        readFromDataBase();

    System.out.println("=================================================");
        System.out.println("|Please give me the ID of message which you like to delete.|");

    System.out.println("=================================================");
        System.out.print(">> ");
        String id = scanner.nextLine();

    System.out.println("=================================================");
        System.out.println("|Are you sure about deleting? There is no comeback. |");
        System.out.println("|If Yes type:     y                        |");
        System.out.println("|If No type:      n                        |");

    System.out.println("=================================================");
        System.out.print(">> ");
        input = scanner.nextLine();
        if (input.equals("y") || input.equals("Y")) {
            DataBaseJDBC.deleteMessage(id);
        }
    }// End of deleteMessage()

    // ============================================================

    public void deleteAllMessagesFromDataBase() {
        String input;
        scanner = new Scanner(System.in);
        System.out.println("================================================");
        System.out.println("|Who user do you want to delete your message from?|");
        System.out.println("================================================");
        System.out.print(">> ");
```

```
            input = scanner.nextLine();
            receiver = input;

        System.out.println("=====================================================");
            System.out.println("|Are you sure about deleting? There is no comeback. |");
            System.out.println("|If Yes type:     y                               |");
            System.out.println("|If No type:      n                               |");

        System.out.println("=====================================================");
            System.out.print(">> ");
            input = scanner.nextLine();
            if (input.equals("y") || input.equals("Y")) {
                DataBaseJDBC.deleteTableMessages(getUserName(), receiver);
            }
        }// End of deleteAllMessagesFromDataBase()

        // ============================================================

        public String getUserName() {
            return this.userName;
        }

        public String getPassword() {
            return this.password;
        }

}// End of Class User
```

Πακέτο “thesis.tsiaparas.zacharias.messages.users;” Κλάση Admin.

Στην συγκεκριμένη κλάση παρουσιάζονται οι μέθοδοι όπου εκτελούνται από τον χρήστη με δικαιώματα Admin.

```
package thesis.tsiaparas.zacharias.messages.users;

import java.util.Scanner;
import thesis.tsiaparas.zacharias.messages.dao.DataBaseJDBC;

/**
 * @author Zacharias Tsiaparas
 */
public class Admin extends User {

    Scanner scanner = new Scanner(System.in);
    private String newUserName;
    private String newPassword;
    private String role;

    public Admin(String userName, String password) {
        super(userName, password);
    }// End of constructor

    // ================================================================

    public void createUser() {
        System.out.println("============================");
        System.out.println("|How will be the User Name?|");
        System.out.println("============================");
        System.out.print(">> ");
        newUserName = scanner.next();
        if (DataBaseJDBC.checkUserName(newUserName)) {
            System.out.println("==============================================");
            System.out.println("|That User Name already exist, choose another.|");
            System.out.println("==============================================");
            System.out.print("\n\n");
            createUser();
```

```
        }
        System.out.println("============================");
        System.out.println("|How will be the Password?|");
        System.out.println("============================");
        System.out.print(">> ");
        newPassword = scanner.next();
        System.out.println("==================================");
        System.out.println("|What role will have the new user?|");
        System.out.println("==================================");
        printRole();
        role = scanner.next();
        DataBaseJDBC.createUserIntoDB(newUserName, newPassword, role);
    }// End of createUser()

    // ================================================================

    public void deleteUser() {
        String input;
        scanner = new Scanner(System.in);
        System.out.println("====================================");
        System.out.println("|Which user do you want to delete ? |");
        System.out.println("====================================");
        System.out.print(">> ");
        input = scanner.nextLine();
        String user = input;

    System.out.println("=================================================");
        System.out.println("|Are you sure about deleting? There is no comeback. |");
        System.out.println("|If Yes type:     y                         |");
        System.out.println("|If No type:      n                         |");

    System.out.println("=================================================");
        System.out.print(">> ");
        input = scanner.nextLine();
        if (input.equals("y") || input.equals("Y")) {
            DataBaseJDBC.deleteUserFromDB(user);
```

```
        }
    }// End of deleteUser()

    // ================================================================

    public void updateUser() {
        System.out.println("====================================");
        System.out.println("|Which user would you like to update? |");
        System.out.println("====================================");
        System.out.print(">> ");
        String user = scanner.next();
        UpdateUserMenu();
        String choise = scanner.next();
        switch (choise) {
        case "name":
            System.out.println("================================");
            System.out.println("|What will be the new user Name?|");
            System.out.println("================================");
            System.out.print(">> ");
            String newName = scanner.next();
            if (DataBaseJDBC.checkUserName(newName)) {
                System.out.println("============================================");
                System.out.println("|That User Name already exist, choose another.|");
                System.out.println("============================================");
                System.out.print("\n\n");
                createUser();
            } else {
                DataBaseJDBC.updateUserFromDB(1, user, newName);
            }
            break;
        case "pass":
            System.out.println("================================");
            System.out.println("|What will be the new user Passwod?|");
            System.out.println("================================");
            System.out.print(">> ");
```

```
            String newPass = scanner.next();
            DataBaseJDBC.updateUserFromDB(2, user, newPass);
            break;
        case "role":
            System.out.println("================================");
            System.out.println("|What will be the new user Role?|");
            System.out.println("================================");
            printRole();
            String newRoll = scanner.next();
            DataBaseJDBC.updateUserFromDB(3, user, newRoll);
            break;
        case "help":
            UpdateUserMenu();
            break;
        default:
            System.out.println("==============================");
            System.out.println("|Remember For Help type:  help|");
            System.out.println("==============================");
            break;
        }
    }// End of updateUser()

    // ================================================================

    protected void UpdateUserMenu() {
        System.out.println("======================================");
        System.out.println("|To change User Name type:        name|");
        System.out.println("|To change User Password type:    pass|");
        System.out.println("|To change User Role type:        role|");
        System.out.println("======================================");
        System.out.print(">> ");
    }// End of UpdateUserMenu()

    // ================================================================
```

```
    protected void printRole() {
        System.out.println("=================================================");
        System.out.println("|                Remember !                     |");
        System.out.println("=================================================");
        System.out.println("|To View only the transacted data type:          A |");
        System.out.println("|To View and Edit the transacted data type:      B |");
        System.out.println("|To View, Edit and Delete the transacted data type:  C |");
        System.out.println("=================================================");
        System.out.print(">> ");
    }// End of printRole()


}// End of Class Admin
```

Πακέτο “thesis.tsiaparas.zacharias.messages.users;” Κλάση Main.

Στην συγκεκριμένη κλάση παρουσιάζονται οι μέθοδοι όπου εκτελούνται κατά την εκκίνηση του προγράμματος. Πρώτα ελέγχετε αν υπάρχει η βάση δεδομένων (αν όχι τότε την δημιουργεί) και έπειτα εκτελεί την κλάση Menus.

```
package thesis.tsiaparas.zacharias.messages.main;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

import thesis.tsiaparas.zacharias.messages.dao.CreateDB;
import thesis.tsiaparas.zacharias.messages.dao.DataBaseJDBC;
import thesis.tsiaparas.zacharias.messages.menus.Menus;

/**
 * @author Zacharias Tsiaparas
 */
public class MainClass {

    private static final String MYSQLURL =
"jdbc:mysql://localhost/bootcampdb?autoReconnect=true&useSSL=false";
    private static final String USERNAME = "root";
    private static final String PASS = "root";
    private static Connection con = null;
    private static Statement statement = null;
    private static ResultSet rs = null;

    private static boolean dbexist() {
        try {
            try {
                Class.forName("com.mysql.jdbc.Driver");
```

```
            } catch (ClassNotFoundException e) {
                e.printStackTrace();
            }
            con = DriverManager.getConnection(MYSQLURL, USERNAME, PASS);
            System.out.println("DataBase connect");
            // existsDB();
        } catch (SQLException ex) {
            Logger.getLogger(DataBaseJDBC.class.getName()).log(Level.SEVERE, null, ex);
        }
        try {
            statement = con.createStatement();
            String sqlselect1 = "USE Project1";
            statement.executeUpdate(sqlselect1);
            return true;
        } catch (SQLException e) {
            return false;
        }
    }// End of dbexist()

    // ============================================================

    private static void closedb() {
        try {
            if (rs != null) {
                rs.close();
                //System.out.println("ResultSet closed!");
            }
            if (statement != null) {
                statement.close();
                //System.out.println("Statement closed!");
            }
            if (con != null) {
                con.close();
                //System.out.println("Connection closed!");
            }
```

```
            //System.out.println("All close!");
        } catch (Exception e) {
        }
    }// End of closedb()

    // ============================================================

    public static void main(String[] args) {

        while (true) {
            if (dbexist()) {
                closedb();
                new Menus();
            } else {
                try {
                    new CreateDB();
                    closedb();
                } catch (ClassNotFoundException e) {
                    e.printStackTrace();
                }
            }
        }
    }// End of main

}// End of Class MainClass
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Sun's Site on JDBC − *Sun Developer Network giving link on JDBC material.*
[2] MySQL Connector/J − *MySQL Connector/J is the official JDBC driver for MySQL.*
[3] The JavaTM Tutorials − *The Java Tutorials are practical guides for programmers who want to use the Java programming language to create applications.*
[4] JavaTM 2 SDK, Standard Edition − *Official site for JavaTM 2 SDK, Standard Edition*
[5] Free Java Download − *Download Java for your desktop computer now!*
[6] Sun Developer Network − *Sun Microsystem's official website listing down all the API documentation, latest Java Technologies, Books and other resource.*
[7] https://www.tutorialspoint.com/jdbc/
[8] https://en.wikipedia.org/wiki/Java_Database_Connectivity