Κεφάλαιο 14 Διασύνδεση με Εφαρμογές JAVA

Σύνοψη
Στο παρόν κεφάλαιο θα παρουσιασθούν βασικά στοιχεία όσον αφορά την σύνδεση ενός προγράμματος Java με την Mysql και την εκτέλεση εντολών σε αυτή.

Προαπαιτούμενη γνώση
Βασική γνώση συγγραφής προγραμμάτων σε Java καθώς και η ύλη των προηγούμενων κεφαλαίων του βιβλίου.

14.1 Εισαγωγικές Έννοιες

Το JAVA JDBC είναι ένα JAVA API με το οποίο μπορούμε να έχουμε πρόσβαση σε οποιαδήποτε μορφής δεδομένα πίνακα και ειδικότερα σε δεδομένα μιας σχεσιακής Βάσης Δεδομένων.

Με το JDBC μπορούμε να γράψουμε Java εφαρμογές, οι οποίες διαχειρίζονται τις παρακάτω λειτουργίες:

1. Σύνδεση στη Βάση Δεδομένων
2. Αποστολή ερωτημάτων και ενημέρωση δεδομένων στη Βάση
3. Ανάκτηση και επεξεργασία των αποτελεσμάτων που λαμβάνονται από τη βάση ως απάντηση στο ερώτημά μας.

Στο ακόλουθο παράδειγμα φαίνονται αυτές οι λειτουργίες:

```
public void connectToAndQueryDatabase(String username, String password) {
   Connection con = DriverManager.getConnection(
                    "jdbc:myDriver:myDatabase",
                    username,
                    password); // Εντολή σύνδεσης στο ΣΔΒΔ
   Statement stmt = con.createStatement();
   ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");
   while (rs.next()) { // Επανάληψη στις εγγραφές που έχουν επιστραφεί
      int x = rs.getInt("a");          // Επιλογή του χαρακτηριστικού a σε μία
                                       //μεταβλητή τύπου int.
      String s = rs.getString("b");
      float f = rs.getFloat("c");
   }
}
```

Στο παραπάνω παράδειγμα χρησιμοποιείται το αντικείμενο DriveManager το οποίο κάνει τη σύνδεση με τη Βάση, το αντικείμενο Statement το οποίο μεταφέρει το SQL ερώτημα στη Βάση και το ResultSet αντικείμενο το οποίο ανακτά τα αποτελέσματα από τα ερωτήματά μας και εκτελεί ένα απλό while loop για την ανάκτηση και εμφάνιση αυτών των αποτελεσμάτων.

Το JDBC API χρησιμοποιεί JDBC drivers για να συνδεθεί με τη βάση. Υπάρχουν αρκετές βιβλιοθήκες που υλοποιούν το API. Συνήθως κάθε σύστημα διαχείρισης βάσεων δεδομένων (MySQL, Oracle, SQL Server) δίνει και μία υλοποίηση αυτού. Συνεπώς με την εγκατάσταση του εξυπηρετητή πρέπει να επιλεγεί και κατάλληλα η JDBC βιβλιοθήκη.

Τι είναι το API (Application Programming Interface)

Είναι ένα έγγραφο τεκμηρίωσης που περιέχει την περιγραφή όλων των χαρακτηριστικών ενός προϊόντος ή λογισμικού ή βιβλιοθήκης. Αναπαριστά κλάσεις και διεπαφές που ακολουθούν τα προγράμματα λογισμικού για να επικοινωνούν μεταξύ τους. Μπορούμε να δημιουργήσουμε ένα API για εφαρμογές, βιβλιοθήκες, λειτουργικά συστήματα, κ.α.

JDBC Drivers

Ένας JDBC driver είναι ένα κομμάτι λογισμικού που επιτρέπει σε μια java εφαρμογή να αλληλοεπιδράσει με τη βάση δεδομένων. Υπάρχουν 4 τύποι JDBC drivers:

1. JDBC-ODBC bridge driver
2. Native-API driver
3. Network Protocol driver
4. Thin driver

Σύνδεση με τη Βάση Δεδομένων

Για να συνδεθεί μια java εφαρμογή με τη βάση δεδομένων πρέπει να γίνουν 5 βήματα:

1. Καταγραφή της driver class
2. Δημιουργία σύνδεσης
3. Δημιουργία statement
4. Εκτέλεση ερωτημάτων
5. Κλείσιμο σύνδεσης

1) Για να γίνει καταγραφή της κλάσης χρησιμοποιείται η μέθοδος forName(). Αυτή η μέθοδος φορτώνει τη driver class. Σύνταξη της forName() μεθόδου:

public static void forName(String className) **throws** ClassNotFoundException

2) Για να γίνει η σύνδεση με τη βάση δεδομένων χρησιμοποιείται η μέθοδος getConnection(). Σύνταξη της getConnection() μεθόδου:

public static Connection getConnection(String url) **throws** SQLException
public static Connection getConnection(String url, String name,
String password) **throws** SQLException

3) Για τη δημιουργία ενός statement χρησιμοποιούμε τη μέθοδο createStatement(). Σύνταξη της createStatement() μεθόδου:

public Statement createStatement() **throws** SQLException

4) Για να εκτελέσουμε ένα ερώτημα στη βάση δεδομένων χρησιμοποιούμε τη μέθοδο executeQuery(). Σύνταξη της executeQuery() μεθόδου:

public ResultSet executeQuery(String sql) **throws** SQLException

Παράδειγμα που εκτελεί ένα ερώτημα:

```
ResultSet rs=stmt.executeQuery("select * from emp");

while(rs.next()){
System.out.println(rs.getInt(1)+" "+rs.getString(2));
}
```

5) Για να κλείσουμε τη σύνδεση χρησιμοποιούμε τη μέθοδο close(). Σύνταξη της μεθόδου close():

```
public void close()throws SQLException
```

Παράδειγμα:

```
con.close();
```

Σύνδεση με MySQL χρησιμοποιώντας JDBC

Για να συνδεθεί μια εφαρμογή με τη mysql βάση δεδομένων πρέπει να γνωρίζουμε τα παρακάτω στοιχεία:

1) **Driver class:** Η driver class για τη mysql είναι **com.mysql.jdbc.Driver**.
2) **Connection URL:** Το connection url για τη mysql ΒΔ είναι **jdbc:mysql://localhost:3306/databasename**, όπου jdbc είναι το API, mysql είναι η βάση δεδομένων, localhost είναι ο server στον οποίο τρέχει η mysql(μπορούμε να βάλουμε και IP διεύθυνση), 3306 είναι η θύρα και databasename είναι το όνομα της βάσης δεδομένων.
3) **Username:** Το username που χρησιμοποιούμε για τη mysql ΒΔ (π.χ. root).
4) **Password:** Είναι ο κωδικός που έχουμε βάλει κατά την εγκατάσταση της mysql βάσης δεδομένων.

Παράδειγμα:

Δημιουργούμε μια ΒΔ company και τον πίνακα emp.

```
create database company;
use sonoo;
create table emp(id int(10),name varchar(40),age int(3));
```

Σύνδεση της java εφαρμογής με τη mysql ΒΔ:

```
import java.sql.*;
class MysqlCon{
public static void main(String args[]){
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/company","usename","password");

Statement stmt=con.createStatement();
ResultSet rs=stmt.executeQuery("select * from emp");
```

```
while(rs.next())  {
System.out.println(rs.getInt(1)+"  "+rs.getString(2)+"  "+rs.getString(3));
}
con.close();
}catch(Exception e){
System.out.println(e);}
}
}
```

Το παραπάνω παράδειγμα θα επιστρέψει και εκτυπώνει όλες τις εγγραφές του πίνακα emp.

Για να γίνει η σύνδεση της java εφαρμογής με τη mysql ΒΔ, πρέπει να φορτωθεί το αρχείο mysqlconnector.jar. Υπάρχουν δύο τρόποι για να φορτωθεί το αρχείο:

1. Επικόλληση στο φάκελο jre/lib/ext όπου βρίσκονται οι jar βιβλιοθήκες
2. Ορισμός της διαδρομής αρχείου του jar στο classpath

DriveManager class

Η κλάση DriverManager λειτουργεί ως διεπαφή μεταξύ χρηστών και drivers. Ελέγχει ποιοι drivers είναι διαθέσιμοι και διαχειρίζεται μια σύνδεση μεταξύ της βάσης και του κατάλληλου driver. Οι κυριότερες μέθοδοι που χρησιμοποιεί η κλάση είναι:

1. public static void registerDriver(Driver driver)
2. public static void deregisterDriver(Driver driver)
3. public static Connection getConnection(String url)
4. public static Connection getConnection(String url,String userName,String password)

Connection Interface

Η σύνδεση γίνεται μεταξύ της java εφαρμογής και της βάσης δεδομένων. Οι κυριότερες μέθοδοι είναι:

1. public Statement createStatement()
2. public Statement createStatement(int resultSetType,int resultSetConcurrency)
3. public void setAutoCommit(boolean status)
4. public void commit()
5. public void rollback()
6. public void close()

Statement Interface

Το Statement interface παρέχει μεθόδους οι οποίες εκτελούν ερωτήματα στη βάση δεδομένων. Οι κυριότερες μέθοδοι που χρησιμοποιούνται είναι:

1. public ResultSet executeQuery(String sql)
2. public int executeUpdate(String sql)
3. public boolean execute(String sql)
4. public int[] executeBatch()

Παράδειγμα (εισαγωγή, ενημέρωση και διαγραφή μιας εγγραφής)

```
import java.sql.*;
class FetchRecord{
public static void main(String args[])throws Exception{

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

```
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/company","usename","passwo
rd");
Statement stmt=con.createStatement();

//stmt.executeUpdate("insert into emp values(33,'Papadopoylos',15000)");
//int result=stmt.executeUpdate("update emp set name='Papadopoulos', salary=10000 where
id=33");
int result=stmt.executeUpdate("delete from emp where id=33");
 System.out.println(result+" records affected");
con.close();
}}
```

PreparedStatement Interface

Η συγκεκριμένη διεπαφή εκτελεί παραμετροποιημένα ερωτήματα.

Παράδειγμα

```
import java.sql.*;
class InsertPrepared{
public static void main(String args[]){
try{
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con= DriverManager.getConnection(
"jdbc:mysql://localhost:3306/company","usename","password");

PreparedStatement stmt=con.prepareStatement("insert into emp values(?,?)");
stmt.setInt(1,101);//1 specifies the first parameter in the query
stmt.setString(2,"Papadopoylos Kostas");
int i=stmt.executeUpdate();

System.out.println(i+" records inserted");
con.close();
}catch(Exception e){
System.out.println(e);
}

}
}
```

14.2 Παράδειγμα Εργαστηριακής Άσκησης

Εκφώνηση:	Στην Βάση Δεδομένων του ΑΕΙ της εργαστηριακής άσκησης 4 θα πρέπει να εκτελέσετε τις ακόλουθες ενέργειες: 3. Γράψτε μία κλάση που θα αναπαριστά τους φοιτητές. 4. Γράψτε μία κλάση που θα φορτώνει όλους τους φοιτητές από την ΒΔ στη κύρια μνήμη σε μία λίστα. 5. Γράψτε μέθοδο που θα αποθηκεύει ένα αντικείμενο χρήστη στη βάση. 6. Γράψτε μία κλάση mail που θα χρησιμοποιεί τις παραπάνω λειτουργίες.
Ζητούμενα:	Χρήση JDBC βιβλιοθήκης για εκτέλεση εντολών στην MySQL από προγράμματα java.

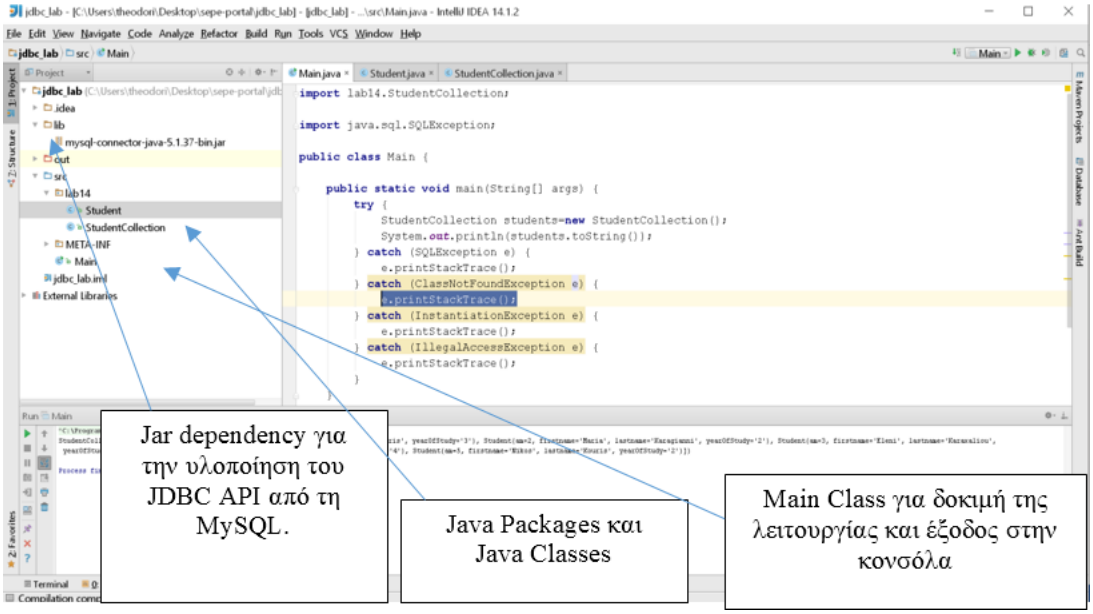
Κλάση αντικειμένων Φοιτητής (Student)

```
package lab14;


public class Student {
   private int am;
   private String firstname;
   private String lastname;
   private String yearOfStudy;

   public Student(int am, String firstname, String lastname, String yearOfStudy) {
      this.am = am;
      this.firstname = firstname;
      this.lastname = lastname;
      this.yearOfStudy = yearOfStudy;
   }

   public int getAm() {
      return am;
   }

   public void setAm(int am) {
      this.am = am;
   }

   public String getFirstname() {
      return firstname;
   }

   public void setFirstname(String firstname) {
      this.firstname = firstname;
   }

   public String getLastname() {
      return lastname;
```

```
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getYearOfStudy() {
        return yearOfStudy;
    }

    public void setYearOfStudy(String yearOfStudy) {
        this.yearOfStudy = yearOfStudy;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Student that = (Student) o;

        if (am != that.am) return false;
        if (firstname != null ? !firstname.equals(that.firstname) : that.firstname != null) return
false;
        if (lastname != null ? !lastname.equals(that.lastname) : that.lastname != null) return
false;
        if (yearOfStudy != null ? !yearOfStudy.equals(that.yearOfStudy) : that.yearOfStudy !=
null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = am;
        result = 31 * result + (firstname != null ? firstname.hashCode() : 0);
        result = 31 * result + (lastname != null ? lastname.hashCode() : 0);
        result = 31 * result + (yearOfStudy != null ? yearOfStudy.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "Student{" +
                "am=" + am +
                ", firstname='" + firstname + '\'' +
                ", lastname='" + lastname + '\'' +
                ", yearOfStudy='" + yearOfStudy + '\'' +
                '}';
    }
}
```

Κλάση συλλογής αντικειμένων Φοιτητής (StudentCollection)

```
package lab14;

import java.sql.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;


public class StudentCollection {
   List<Student> students;


   public StudentCollection() throws SQLException, ClassNotFoundException,
IllegalAccessException, InstantiationException {
       students=new ArrayList<Student>();
       Class.forName("com.mysql.jdbc.Driver").newInstance();
       Connection con = DriverManager.getConnection(
               "jdbc:mysql://localhost/university",
               "username",
               "password"); // Εντολή σύνδεσης στο ΣΔΒΔ
       Statement stmt = con.createStatement();
       ResultSet rs = stmt.executeQuery("SELECT * FROM student");

       while (rs.next()) { // Επανάληψη στις εγγραφές που έχουν επιστραφεί
          Student student=new
Student(rs.getInt("AM"),rs.getString("firstname"),rs.getString("lastname"),rs.getString("yea
rofstudy"));
          getStudents().add(student);
       }
con.close();


   }

   public List<Student> getStudents() {
      return students;
   }


   @Override
   public String toString() {
      return "StudentCollection{" +
              "students=" + Arrays.toString(students.toArray()) +
              '}';
   }
}
```



Εικόνα 14.1: *Στιγμιότυπο από το περιβάλλον ανάπτυξης.*

```
import lab14.StudentCollection;

import java.sql.SQLException;

public class Main {

   public static void main(String[] args) {
      try {
         StudentCollection students=new StudentCollection();
         System.out.println(students.toString());
      } catch (SQLException e) {
         e.printStackTrace();
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      } catch (InstantiationException e) {
         e.printStackTrace();
      } catch (IllegalAccessException e) {
         e.printStackTrace();
      }
   }
}
```

Console Output (βλέπε Εικόνα 14.1 και 14.2):

StudentCollection{students=[Student{am=1, firstname='Alexandros', lastname='Theocharis', yearOfStudy='3'}, Student{am=2, firstname='Maria', lastname='Karagianni', yearOfStudy='2'}, Student{am=3, firstname='Eleni', lastname='Karaxaliou', yearOfStudy='4'}, Student{am=4, firstname='Kosmas', lastname='Pitas', yearOfStudy='4'}, Student{am=5, firstname='Nikos', lastname='Kouris', yearOfStudy='2'}]}

Μεταγλώττιση Main κλάσης:
javac.exe Main.java

Εκτέλεση:
java -cp ..\lib\mysql-connector-java-5.1.37-bin.jar; Main

StudentCollection{students=[Student{am=1, firstname='Alexandros', lastname='Theocharis', yearOfStudy='3'}, Student{am=2, firstname='Maria', lastname='Karagianni', yearOfStudy='2'}, Student{am=3, firstname='Eleni', lastname='Karaxaliou', yearOfStudy='4'}, Student{am=4, firstname='Kosmas', lastname='Pitas', yearOfStudy='4'}, Student{am=5, firstname='Nikos', lastname='Kouris', yearOfStudy='2'}]}

Εικόνα 14.2: *Αποτέλεσμα στην κονσόλα των MS Windows.*

Κλάση Φοιτητή που έχει επεκταθεί με την μέθοδο `toUpdateSQLCommand` που επιστρέφει την κατάλληλη εντολή update που ενημερώνει τις τιμές του αντίστοιχου φοιτητή στη ΒΔ.

```
package lab14;


public class Student {
   private int am;
   private String firstname;
   private String lastname;
   private String yearOfStudy;

   public Student(int am, String firstname, String lastname, String yearOfStudy) {
      this.am = am;
      this.firstname = firstname;
      this.lastname = lastname;
      this.yearOfStudy = yearOfStudy;
   }

   public int getAm() {
      return am;
   }

   public void setAm(int am) {
      this.am = am;
   }

   public String getFirstname() {
```

```
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getYearOfStudy() {
        return yearOfStudy;
    }

    public void setYearOfStudy(String yearOfStudy) {
        this.yearOfStudy = yearOfStudy;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;

        Student that = (Student) o;

        if (am != that.am) return false;
        if (firstname != null ? !firstname.equals(that.firstname) : that.firstname != null) return
false;
        if (lastname != null ? !lastname.equals(that.lastname) : that.lastname != null) return
false;
        if (yearOfStudy != null ? !yearOfStudy.equals(that.yearOfStudy) : that.yearOfStudy !=
null) return false;

        return true;
    }

    @Override
    public int hashCode() {
        int result = am;
        result = 31 * result + (firstname != null ? firstname.hashCode() : 0);
        result = 31 * result + (lastname != null ? lastname.hashCode() : 0);
        result = 31 * result + (yearOfStudy != null ? yearOfStudy.hashCode() : 0);
        return result;
    }

    @Override
    public String toString() {
        return "Student{" +
                "am=" + am +
                ", firstname='" + firstname + '\'' +
```

```
            ", lastname='" + lastname + '\'' +
            ", yearOfStudy='" + yearOfStudy + '\'' +
            '}';
    }


    public String toUpdateSQLCommand() {
        return "update student set " +
                "am=" + am +
                " , firstname='" + firstname + '\'' +
                " , lastname='" + lastname + '\'' +
                " , yearOfStudy='" + yearOfStudy + '\'' +
                " where am=" +am;
    }

}
```

Κλάση StudentCollection που έχει επεκταθεί με την μέθοδο updateStudent που ενημερώνει ένα αντικείμενο Student στην ΒΔ.

```
package lab14;

import java.sql.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;


public class StudentCollection {
    List<Student> students;


    public StudentCollection() throws SQLException, ClassNotFoundException,
IllegalAccessException, InstantiationException {
        students=new ArrayList<Student>();
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost/university",
                "root",
                "5662420!");
        Statement stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM student");

        while (rs.next()) {
            Student student=new
Student(rs.getInt("AM"),rs.getString("firstname"),rs.getString("lastname"),rs.getString("yea
rofstudy"));
            getStudents().add(student);
        }
        con.close();

    }
```

```
    public boolean updateStudent(Student s) throws SQLException, ClassNotFoundException,
IllegalAccessException, InstantiationException {

        Class.forName("com.mysql.jdbc.Driver").newInstance();
        Connection con = DriverManager.getConnection(
                "jdbc:mysql://localhost/university",
                "username",
                "password");
        Statement stmt = con.createStatement();
        int result=stmt.executeUpdate(s.toUpdateSQLCommand());
        System.out.println(result+" records affected");
        con.close();
        if (result==1) return true;
        else return false;
    }

    public List<Student> getStudents() {
        return students;
    }


    @Override
    public String toString() {
        return "StudentCollection{" +
                "students=" + Arrays.toString(students.toArray()) +
                '}';
    }


}
```

Η κλάση Main που χρησιμοποιεί τις νέες λειτουργίες/μεθόδους.

```
import lab14.Student;
import lab14.StudentCollection;

import java.sql.SQLException;

public class Main {

    public static void main(String[] args) {
        try {
            StudentCollection students=new StudentCollection();
            System.out.println(students.toString());
            Student s=students.getStudents().get(0);
            System.out.println(s.toUpdateSQLCommand());
            s.setLastname("Papadopoulos");
            students.updateStudent(s);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
```

```
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
    }
}
```

14.3 Άλυτες Εργαστηριακές Ασκήσεις

Άσκηση 1

Δημιουργείστε και επεκτείνεται την παραπάνω εργαστηριακή άσκηση έτσι ώστε αυτή να εκτυπώνει στον χρήστη μία λίστα με τις ακόλουθες επιλογές:

1. Έξοδος
2. Αναζήτηση Φοιτητή: σε αυτή την επιλογή ο χρήστης θα πρέπει να δίνει είτε το ΑΜ είτε το Επώνυμο του φοιτητή και θα εκτυπώνονται στη κονσόλα τα στοιχεία του φοιτητή (ή φοιτητών) που βρέθηκαν
3. Επεξεργασία Φοιτητή: σε αυτή την επιλογή ο χρήστης θα μπορεί να επεξεργάζεται τα δεδομένα ενός φοιτητή
4. Διαγραφή Φοιτητή: δίνοντας το ΑΜ
5. Αποθήκευση στη ΒΔ: σε αυτή την επιλογή τα αντικείμενα-φοιτητές που έχουν επεξεργαστεί από τον χρήστη θα πρέπει να αποθηκεύονται στη ΒΔ.

Βιβλιογραφία/Αναφορές

R. Elmasri & S.B. Navathe "Θεμελιώδεις Αρχές Συστημάτων ΒΔ - 4η Έκδοση". Κεφάλαιο 9.
R. Ramakrishnan & J. Gehrke. 2002. Database Management Systems (3 ed.). McGraw-Hill, Inc., New York, NY, USA. Κεφάλαια 6, 7.
MySQL JDBC Connector. http://dev.mysql.com/doc/connector-j/en/index.html
JDBC Tutorial. http://www.tutorialspoint.com/jdbc/index.htm
MySQL – Java. http://www.vogella.com/tutorials/MySQLJava/article.html