جافا متقدمة

الدرس الخامس : جافا JDBC

أعداد: د. عبدالحميد الواعر

جامعة طرابلس
كلية تقنية المعلومات – قسم هندسة البرمجيات

الفهرس

2.جافا JDBC

1.1. مقدمة

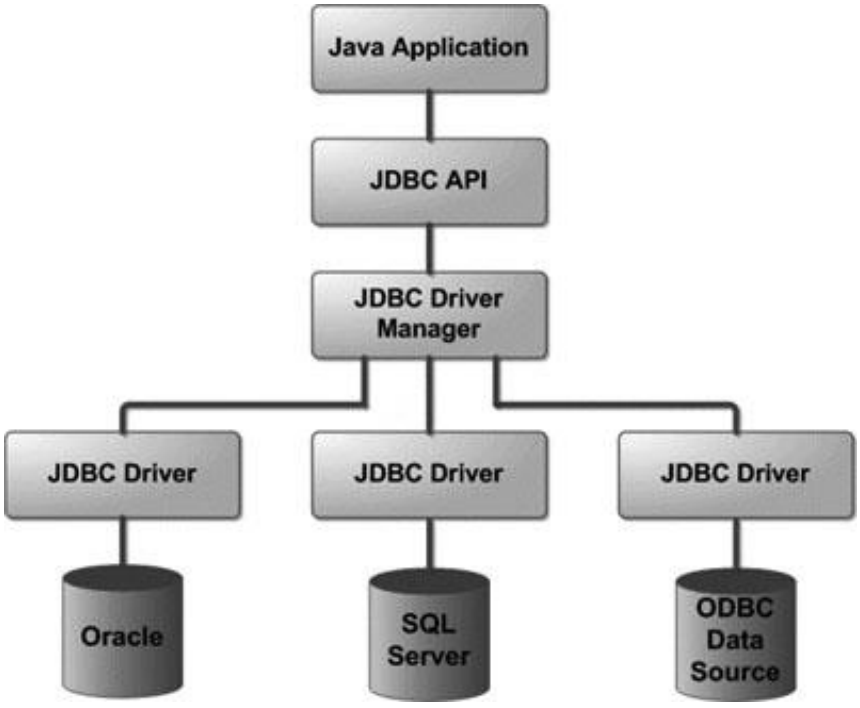
تستخدم قواعد البيانات العلائقية (Relational Database Systems) في تخزين البيانات والاستفادة منها عن طريق أجراء أستعلامات بأستخدام لغة الاستعلام الهيكلية (SQL) للحصول على المعلومات المطلوبة.

معظم لغات البرمجة توفر إمكانية عمل أتصال بقواعد البيانات المختلفة ,اجراء أستعلامات عليها بأستخدام (SQL) . لغة جافا تستخدم JDBC API لانشاء أتصال بقواعد البيانات والاستعلام منها.

JDBC هي أختصار ل **Java DataBase Connectivity** وهي المسئولة عن عملية الاتصال بقواعد البيانات المختلفة ولغة جافا.

مكتبة **JDBC** تحتوي على عددمن classes التي تستخدم في أجراء مختلف العمليات الخاصة بالتعامل مع قواعد البينات مثل:

- الاتصال بقاعدة البيانات
- أنشاء جمل SQL
- تنفيذ جمل SQL
- عرض وتعديل النتائج المتحصل عليها



1.2. المكونات الرئيسية ل *JDBC*

- **DriverManager** يستخدم مجموعة من database Drivers ويقوم بأستعمال المناسب منها في عملية الاتصال.
- **Drivers** تقوم بعملية الاتصال بخادم قاعدة البيانات.
- **Connection** يستخدم في أجراء عملية الاتصال بقاعدة البيانات.
- **Statement** تستخدم في أنشاء جمل SQL
- **ResultSet** يحتوي على البيانات المستخلصة من قاعدة البيانات.
- **SQLException** تقوم بالتعامل مع أي خطأ يظهر في التطبيق المستخدم لقاعدة البيانات.

1.3. أنشاء تطبيق يستخدم *JDBC*

- لانشاء تطبيق يستخدم JDBC نحتاج لتنفيذ الخطوات التالية:
- تضمين المكتبة الخاصة ب JDBC في التطبيق.
- تسجيل JDBC Driver .
- فتح أتصال مع قاعدة البيانات.
- تنفيذ جملة أستعلام.
- أستخلاص البيانات من نتيجة أستعلام قاعدة البيانات.
- أغلاق الاتصال مع قاعدة البيانات.

مثال:

البرنامج التالي يوضح عملية أنشاء تطبيق بأستخدام JDBC بأستخدام الخطوات السابقة

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

public class FirstExample {
    // 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 = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        try {
            Connection conn = null;
            Statement stmt = null;
            //STEP 2: Register JDBC driver
            Class.forName(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...");
            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();
            System.out.println("Goodbye!");
        } //end main
        catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }//end main
}//end FirstExample
```

1.4. الخطوات اللازمة للحصول على *JDBC Connection*

1. أستخدام المكتبة الخاص ب JDBC

```
import java.sql.*;
```

2. تسجيل JDBC Driver

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

3. تكوين Database URL

```
jdbc:mysql://hostname/databaseName
```

4. تكوين Connection Object

```
String URL = "jdbc:mysql://hostname/databaseName";
String USER = "username";
String PASS = "password"
Connection conn = DriverManager.getConnection(URL, USER,
PASS);
```

5. أغلاق الاتصال مع قاعدة البيانات

أخيراً بعد أستخدام قاعدة البيانات يتم أغلاق الاتصال وذلك عن طريق الامر التالي:

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

1.5. أنشاء *Statement Object*

بعد أنشاء الاتصال يمكن أستخدام قاعدة البيانات وذلك من خلال أستخدام JDBC Statement التي تستخدم في أنشاء جمل SQL للتعامل مع قاعدة البيانات.

مثال:

```
Connection conn = null;
Statement stmt = null;
String sql;
conn = DriverManager.getConnection(DB_URL,USER,PASS);
stmt = conn.createStatement();
sql = "SELECT id, first, last, age FROM Employees";
ResultSet rs = stmt.executeQuery(sql);
```

بعد أنشاء Statement يمكن أستخدامها مع أحد الدوال التالية:

```
boolean execute(String SQL) //Returns a boolean value of true
if a ResultSet object can be retrieved; otherwise, it returns
false.

int executeUpdate(String SQL) //Returns the numbers of rows
affected by the execution of the SQL statement.

ResultSet executeQuery(String SQL) //Returns a ResultSet
object. Use this method when you expect to get a result set.
```

بعد الانتهاء من أستخدام Statement يتم أغلاقها بأستخدام :

```
stmt.close();
```

1.6. أنشاء *PreparedStaement*

هذه الجملة تعطي بعض المرونة من ناحية أمكانية أستخدام المتغيرات خلال وقت التنفيذ في جملة Statement

```
PreparedStatement pstmt = null;
String SQL = "Update Employees SET age = ? WHERE id = ?";
pstmt = conn.prepareStatement(SQL);
pstmt.setInt(1,27);
pstmt.setInt(2,101);
int result = stmt.executeUpdate();
```

بعد الانتهاء من أستخدام PreparedStatement يتم أغلاقها بأستخدام :

```
pstmt.close();
```

1.7. التعامل مع *ResultSet Object*

بعد تنفيذ جملة SQL يتم الحصول على ResultSet object يحتوي على نتيجة الاستعلام يمكن من خلاله طباعة النتيجة أو أستخدام حسب مايتطلبه البرنامج.

مثال:

```
//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);
      }
      rs.close();
```

بعد الانتهاء من أستخدام ResultSet يتم أغلاقه عن طريق الامر close() .

1.8. معالجة الاستثناءات بأستخدام *SQLException*

عند أستخدام JDBC ممكن أن ينتج عنها بعض Exceptions فيتم التعامل معها من خلال تمرير Object من نوع SQLException إلى جملة catch ووضع فيها مجموعة من الاوامر للتعامل معه.

1.9. أمثلة عن أستخدام *JDBC*

1.3.1 أنشاء قاعدة بيانات جديدة

تستخدم SQL الامر التالي لانشاء قاعدة بيانات جديدة

```
CREATE DATABASE DATABASENAME
```

مثال:

```
CREATE DATABASE STUDENT
```

البرنامج التالي يقوم بأنشاء قاعدة بيانات جديدة تحت أسم STUDENTS .

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

public class CreateDatabaseExample {

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

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(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 database...");
            stmt = conn.createStatement();
            String sql = "CREATE DATABASE STUDENT";
            stmt.executeUpdate(sql);
            System.out.println("Database created successfully...");
            //STEP 6: Close resources
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.2 حذف قاعدة بيانات

تستخدم SQL الامر التالي لحذف قاعدة بيانات .

```
DROP DATABASE    DATABASENAME
```

مثال:

```
DROP  DATABASE  STUDENT
```

البرنامج التالي يقوم بحذف قاعدة بيانات STUDENT .

```java
//STEP 1. Import required packages
import java.sql.*;
public class DeleteDatabaseExample {
    // JDBC driver name and database URL

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/";
    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            System.out.println("Deleting database...");
            stmt = conn.createStatement();
            String sql = "DROP DATABASE STUDENT";
            stmt.executeUpdate(sql);
            System.out.println("Database deleted successfully...");

            //STEP 6: Close resources
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.3

1.3.4 أنشاء جدول في قاعدة البيانات

تستخدم SQL الامر التالي بأنشاء جدول في قاعدة البيانات.

```
CREATE TABLE table_name(
   column1 datatype,
   column2 datatype,
   column3 datatype,
   .....
   columnN datatype,
   PRIMARY KEY( one or more columns ));
```

مثال:

```
CREATE TABLE REGISTRATION(id INTEGER not NULL,
                          first VARCHAR(255),
                               last VARCHAR(255),
                             age INTEGER,
                               PRIMARY KEY ( id ))";
```

البرنامج التالي يقوم بأنشاء الجدول REGISTRATION في قاعدة البيانات STUDENT .

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

public class CreateTableExample {
    // JDBC driver name and database URL

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            System.out.println("Creating table in given database...");
            stmt = conn.createStatement();

            String sql = "CREATE TABLE REGISTRATION "
                    + "(id INTEGER not NULL, "
                    + " first VARCHAR(255), "
                    + " last VARCHAR(255), "
                    + " age INTEGER, "
                    + " PRIMARY KEY ( id ))";

            stmt.executeUpdate(sql);
            System.out.println("Created table in given database...");
             //STEP 6: Close resources
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.5 حذف جدول من قاعدة البيانات

تستخدم SQL الامر التالي لحذف جدول من قاعدة البيانات.

```
DROP TABLE    TABLENAME
```

مثال:

```
DROP  TABLE   REGISTRATION
```

البرنامج التالي يقوم بحذف الجدول REGISTRATION من قاعدة البيانات STUDENT .

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

public class DeleteTableExample {
    // JDBC driver name and database URL

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            System.out.println("Deleting Table...");
            stmt = conn.createStatement();
            String sql = "DROP TABLE REGISTRATION ";

            stmt.executeUpdate(sql);
            System.out.println("Table  deleted in given database...");

            //STEP 6: Close resources
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.6 أدخال حقل إلى الجدول

تستخدم SQL الامر التالي لادخال قيم الحقل إلى الجدول.

```
INSERT INTO TABLE_NAME  VALUES (value1, value2,
value3,...valueN);
```

مثال:

```
INSERT INTO Registration VALUES (100, 'Zara', 'Ali', 18);
```

البرنامج التالي يقوم بأدخال مجموعة من الحقول إلى الجدول REGISTRATION في قاعدة البيانات STUDENT.

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

public class InsertTableData {

    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            System.out.println("Inserting records into the table...");
            stmt = conn.createStatement();
            String sql = "INSERT INTO Registration VALUES (100, 'Zara', 'Ali', 18)";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO Registration VALUES (101, 'Fatma', 'Khalid', 25)";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO Registration VALUES (102, 'Zaid', 'Salem', 30)";
            stmt.executeUpdate(sql);
            sql = "INSERT INTO Registration VALUES(103, 'Salma', 'Ahmad', 28)";
            //STEP 6: Close resources
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
        System.out.println("Goodbye!");
    }
}
```

1.3.7 أختيار حقل من الجدول وعرض بياناته

تستخدم SQL الامر التالي لعرض محتويات الجدول .

```
SELECT column1, column2, columnN FROM table_name;
```

مثال:

```
SELECT id, first, last, age FROM Registration;
```

البرنامج التالي يقوم بأختيار جميع الحقول في الجدول REGISTRATION وعرض بياناتهم.

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

public class SelectExample {
    // JDBC driver name and database URL

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            System.out.println("Creating statement...");
            stmt = conn.createStatement();

            String sql = "SELECT id, first, last, age FROM Registration";
            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: Close resources
            rs.close();
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.8 تحديث بيانات حقل

تستخدم SQL الامر التالي لتحديث محتويات الجدول .

```
UPDATE table_name  SET column1 = value1, column2 =
value2...., columnN = valueN  WHERE [condition];
```

مثال:

```
UPDATE Registration SET age = 30 WHERE id in (100,101);
```

البرنامج التالي يقوم بتحديث بيانات age إلى 30 في كل من الحقل الذي id = 100 والحقل الذي id=101 في الجدول REGISTRATION .

```
//STEP 1. Import required packages
import java.sql.*;
public class UpdateExample {
    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";
    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";
    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);
            //STEP 3: Open a connection
            System.out.println("Connecting to a selected database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connected database successfully...");

            //STEP 4: Execute a query
            System.out.println("Creating statement...");
            stmt = conn.createStatement();
            String sql = "UPDATE Registration "+ "SET age = 30 WHERE id in (100, 101)";
            stmt.executeUpdate(sql);
            //STEP 6: Close resources
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.9 حذف حقل من جدول

تستخدم SQL الامر التالي لحذف صف من الجدول .

```
DELETE * FROM table_name   WHERE [condition];
```

مثال:

```
DELETE * FROM Registration WHERE id = 101;
```

البرنامج التالي يقوم بحذف الصف الذي id = 101 في الجدول REGISTRATION .

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

public class DeleteExample {
    // JDBC driver name and database URL

    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            System.out.println("Creating statement...");
            stmt = conn.createStatement();
            String sql = "DELETE * FROM Registration WHERE id = 101";

            stmt.executeUpdate(sql);

            //STEP 6: Close resources
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.10 أستخدام جملة WHERE

البرنامج التالي يوضح كيفية أستخدام جملة WHERE في أختيار الحقول التي id الخاص بها أكبر أو يساوي 101.

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

public class WhereExample {

    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";
    //  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(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            System.out.println("Creating statement...");
            stmt = conn.createStatement();
            // Select all records having ID equal or greater than 101
            String sql = "SELECT id, first, last, age FROM Registration WHERE id >= 101 ";
            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: Close resources
            rs.close();
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.11 أستخدام جملة LIKE

البرنامج التالي يوضح كيفية أستخدام جملة LIKE في أختيار الحقول التي id الخاص بها أكبر أو يساوي 101.

```
package Lecture5.JDBC;
//STEP 1. Import required packages
import java.sql.*;
public class LikeExample {

    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";
    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        Statement stmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);
            //STEP 3: Open a connection
            System.out.println("Connecting to a selected database...");
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            System.out.println("Connected database successfully...");
            //STEP 4: Execute a query
            System.out.println("Creating statement...");
            stmt = conn.createStatement();
            // Select all records having ID equal or greater than 101
            System.out.println("Fetching records with condition...");
            String sql = "SELECT id, first, last, age FROM Registration"
                    + " WHERE first LIKE '%za%' ";
            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: Close resources
            rs.close();
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```

1.3.12 ترتيب الحقول تصاعديا في الجدول

البرنامج التالي يوضح كيفية ترتيب الحقول تصاعديا أو تنازليا في الجدول Registration .

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

public class AscSortExample {

        // JDBC driver name and database URL
        static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
        static final String DB_URL = "jdbc:mysql://localhost/STUDENT";
        //  Database credentials
        static final String USER = "root";
        static final String PASS = "";
        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 a selected database...");
                conn = DriverManager.getConnection(DB_URL, USER, PASS);
                System.out.println("Connected database successfully...");
                //STEP 4: Execute a query
                System.out.println("Creating statement...");
                stmt = conn.createStatement();
                // Extract records in ascending order by first name.
                System.out.println("Fetching records in ascending order...");
                String sql = "SELECT id, first, last, age FROM Registration"
                        + " ORDER BY first ASC";
                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: Close resources
                rs.close();
                stmt.close();
                conn.close();
            } catch (ClassNotFoundException | SQLException ex) {
                //Handle errors for Class.forName and for JDBC
                System.out.println(ex);
            }
        }
    }
```

1.3.13 أستعمال جملة PreparedStatement

البرنامج التالي يوضح أستخدام PreparedStatement

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

public class PreparedStatementExample {

    // JDBC driver name and database URL
    static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    static final String DB_URL = "jdbc:mysql://localhost/STUDENT";

    //  Database credentials
    static final String USER = "root";
    static final String PASS = "";

    public static void main(String[] args) {
        Connection conn = null;
        PreparedStatement pstmt = null;
        try {
            //STEP 2: Register JDBC driver
            Class.forName(JDBC_DRIVER);

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

            //STEP 4: Execute a query
            String SQL = "Update Registration SET age = ? WHERE id = ?";
            pstmt = conn.prepareStatement(SQL);
            pstmt.setInt(1, 25);
            pstmt.setInt(2, 191);
            int rs = pstmt.executeUpdate();
            System.out.println("Number of rows effected is " + rs);

            //STEP 6: Close resources
            pstmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException ex) {
            //Handle errors for Class.forName and for JDBC
            System.out.println(ex);
        }
    }
}
```