Hyperiondev

TASK

Java Database Programming: The JDBC

Visit our website

Introduction

Welcome to the Introduction to the JDBC Task!

In this task, we learn how to connect to a MySQL database with Java JDBC.

Get in touch

Connect for support

Remember that with our courses, you're not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

Introduction to JDBC

JDBC is the Java API for developing Java database applications. It is a set of Java interfaces and classes used to write Java programs for accessing and manipulating relational databases. JDBC is not an acronym, it is however thought to stand for Java Database Connectivity. JDBC provides a uniform interface to Java programmers for accessing and manipulating a wide range of relational databases. Applications written in Java can execute SQL statements, retrieve results, present data in a user-friendly interface and make changes to the database, using the JDBC API.

A JDBC driver serves as the interface to facilitate communications between JDBC and different databases. They are database specific and are normally provided by the vendor of a specific database. For example, to access the MySQL database you need the MySQL JDBC drivers.

In summary with the JDBC API, you are able to:

1. establish a connection with a database or access any tabular data source
2. send SQL statements
3. process the results

Setting Up Your Environment

Before you start developing with JDBC you need to setup your environment. This includes downloading and installing MySQL, installing the JDK and a text editor. At this point, you should have already installed the JDK and a text editor such as TextPad, NotePad++ or Sublime, however if not, you can download the JDK **here** and the text editor Sublime **here**.

We will now explain the steps to download and install MySQL.

Download and Install MySQL

For Windows:

- Download the MySQL ZIP ARCHIVE from the MySQL website. Your operating system should be automatically detected, however, if it is not:
    - Select the "General Available (GA) Releases" tab.
    - In “Select Operating System:”, choose “Microsoft Windows”

    - Under “Other downloads”, download “Windows (x86, 32-bit), ZIP Archive” or “Windows (x86, 64-bit), ZIP Archive” depending on your version of Windows.
        - You can check whether your Windows is 32-bit or 64-bit by going to “Control Panel” -> System and Security (optional) -> System -> Under "System Type".
    - You don’t need to Login or Sign Up, simply select “No thanks, just start my download.” at the bottom of the page.

- Create a project directory, for example: "d:\myProject" (or "c:\myProject"), and UNZIP the downloaded file into your newly created project directory. MySQL will be unzipped as "d:\myProject\mysql-5.7.{xx}-winx64".

- You now need to initialise the database. Start a Command Prompt (as administrator) and enter the following commands:

```
// Change your current directory to the MySQL installed directory
// For this example we suppose that your MySQL installed directory is
//d:\myProject\mysql-5.7.21-winx64
d:
cd \myProject\mysql-5.7.21-winx64\bin

// Initialize the database. Create a root user without a password.
//Show the message on console
mysqld --initialize --console
......
...... [Note] A temporary password is generated for root@localhost:
xxxxxxxx
```

  A superuser called root is created with a temporary password during the installation. **It is extremely important that you take note of the password.** You can copy and save it somewhere or take a picture of the screen. Just make sure you don’t lose it!

If you make a mistake you will need to DELETE the entire MySQL directory and repeat the above two steps.

For Mac:

- Download the MySQL DMG Archive **MySQL website**. Your operating system should be automatically detected, however, if it is not:
    - Select the "General Available (GA) Releases" tab.
    - In “Select Operating System:”, choose “macOS”

    - You don't need to Login or Sign Up, simply select "No thanks, just start my download." at the bottom of the page.

- Install MySQL:
    - Go to "Downloads" and Double-click ".dmg" file.
    - Double-click the "mysql-5.7.{xx}-osx10.x-xxx.pkg.
    - Follow the on-screen instructions to install MySQL.
    - A superuser called root is created with a temporary password during the installation. **It is extremely important that you take note of the password.** You can copy and save it somewhere or take a screenshot. Just make sure you don't lose it!
    - MySQL will be installed in "/usr/local/mysql". Take note of this installed directory!
    - Eject the ".dmg" file

- If you make a mistake you need to:
    - stop the server by clicking on the 'Apple' Icon -> System Preferences -> MySQL -> Stop
    - remove the directories "/usr/local/mysql-5.7.{xx}..."
    - re-run the installation
    - restart the server by clicking on the 'Apple' Icon -> System Preferences -> MySQL -> Start
    - You may also need to reboot your machine.

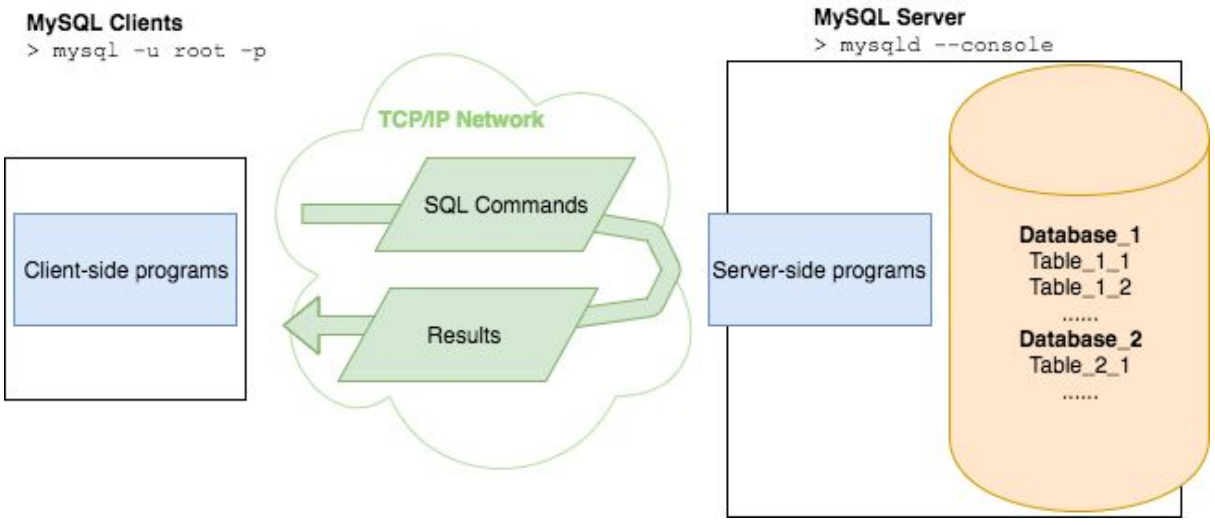
Remember to take note of your MySQL installed directory.

Using MySQL

We will now briefly discuss the basics of using MySQL.

Starting the Server

MySQL is a client-server system. This means that the database is run as a server application and users can access the database server locally or remotely by using a client program. This is demonstrated in the diagram below:



As you can see from the diagram, the server program is called "mysqld" and the client program is called "mysql". These programs are kept in the "bin" subdirectory, located in the MySQL installed directory. To start up the server you need to do the following:

For Windows:

- Start a Command Prompt and enter the following:

```
// Assume that the MySQL installed directory is
// "d:\myProject\mysql-5.7.21-winx64"
// Change Directory to the MySQL's bin directory
d:
cd \myProject\mysqlmysql-5.7.21-winx64\bin

//Start the MySQL Database Server
mysqld --console
......
......
XXXXXX XX:XX:XX [Note] mysqld: ready for connections.
Version: '5.7.xx'  socket: ''  port: 3306  MySQL Community Server (GPL)
```

The --console option directs output messages from the server to the console. You would end up seeing a blank screen without this option.

For Mac:

- Simply click on the 'Apple' Icon -> System Preferences -> MySQL -> Start MySQL Server

The MySQL server should now be started and able to handle requests from the client.

Shutting down the Server

For Windows:

- Press Ctrl+C. This initiates a normal shut down. Do not use the windows close button to close the server.

  You should get the following message from the MySQL server console:

```
XXXXXX XX:XX:XX [Note] mysqld: Normal shutdown
......
XXXXXX XX:XX:XX  InnoDB: Starting shutdown...
XXXXXX XX:XX:XX  InnoDB: Shutdown completed; log sequence number 0 44233
......
XXXXXX XX:XX:XX [Note] mysqld: Shutdown complete
```

For Mac:

- Simply click on the 'Apple' Icon -> System Preferences -> MySQL -> Stop MySQL Server

Please ensure you shutdown the MySQL server correctly, otherwise, you might corrupt the database and have problems restarting it.

Starting a Client

As previously stated, MySQL is a client-server system. Once you start the server one or more clients can be connected to it. A client can be local, meaning that it is run on the same machine, or remote, meaning it is from another machine on the same network.

In order to login to the MySQL server, you need to provide a username and password. If you can remember MySQL created a superuser called "root" with a temporary password during installation. Hopefully you took note of the password, otherwise, you will have to re-install MySQL all over again.

MySQL provides a command-line client called "mysql". We will now start a command-line client with the superuser "root", however, firstly, make sure that the

server is running. If it is not, restart it.

For Windows:

- Start a Command Prompt to run the client and enter the following:

```
// Assume that the MySQL installed directory is
// "d:\myProject\mysql-5.7.21-winx64"
// Change Directory to the MySQL's bin directory
d:
cd \myProject\mysql\bin

// Start a client as superuser "root"
mysql -u root -p
Enter password:
   // Enter the root's password which was set during installation.
   // Note that nothing will be shown for security reasons
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.39-community MySQL Community Server (GPL)
Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
// When the Client has started, the prompt changes to "mysql>".
// You can now enter SQL commands.
```

For Mac:

- Open a new terminal run the client and enter the following:

```
// Change the current directory to /usr/local/mysql/bin.
cd /usr/local/mysql/bin

// Start a client with superuser "root"
./mysql -u root -p
Enter password:
   // Enter the root's password which was set during installation.
   // Note that nothing will be shown for security reasons

Welcome to the MySQL monitor.  Commands end with ; or \g.
......
mysql>
// When the Client has started, the prompt changes to "mysql>".
// You can now enter SQL commands.
```

Changing the Password for "root"

As previously mentioned, the MySQL installation created a superuser named “root” with a temporary password. The superuser "root" can do anything to the databases. This includes deleting all of them. Security is therefore imperative when it comes to the “root” superuser and you are required to change the root's temporary password immediately after logging in.

Your client should now be started. We will therefore continue with the current client session. Simply enter the following:

```
// Change password for 'root'@'localhost'.
// You need to replace XXXXX with your chosen password
// Please note that strings need to be enclosed by a pair of
// single-quotes.

mysql> alter user 'root'@'localhost' identified by 'XXXXX';
Query OK, 0 rows affected (0.00 sec)

// logout and terminate the client program
mysql> quit
Bye
```

By entering the statements above, you have changed the password for root and quit the client. You can now restart the client “root” with the new password. To do so enter the following:

For Windows:

```
// Change Directory to the MySQL's bin directory

mysql -u root -p
Enter password:    // Type your new password and hit enter.
Welcome to the MySQL monitor.
......
mysql>
// client is started again and ready to issue SQL commands
```

For Mac:

```
// Change the current working directory to /usr/local/mysql/bin
cd /usr/local/mysql/bin
./mysql -u root -p
Enter password:    // Type your new password and hit enter.
Welcome to the MySQL monitor.
......
mysql>
// client is started again and ready to issue SQL commands
```

Create a New User

The superuser “root” is a privileged user and is meant to be used for database administration. It is not meant to be used for everyday operational purposes. We therefore should create another user with fewer privileges. We will call this user “myuser”. To create a new user you need to start a client with the “root” superuser.

```
// If it is not already started, start a client.
mysql -u root -p     // Windows
./mysql -u root -p   // Mac OS

// Create a new user called "myuser" using the command “create user”,
// which can login locally from the same machine with password "xxxx"
mysql> create user 'myuser'@'localhost' identified by 'xxxx';
Query OK (0.01 sec)

// The newly created user has no privileges. This includes the select.
// We grant all privileges to “myuser” using the grant command.
// This includes select, insert, delete, etc.
// “on *.*” means that we grant all privileges to all the databases and
// all the tables.
// This means that the new user has all the privileges as the root user,
// except the grant command.
mysql> grant all on *.* to 'myuser'@'localhost';
Query OK (0.01 sec)

mysql> quit
```

Client Session Tips

Before we go any further, here are some tips on using the client:

- Terminate your command with a semicolon (;), this sends the command to the server for processing.

```
mysql> select * from java_programming ;
```

- A single command can span many lines. To denote continuation, the new line prompt changes to ->. Remember to terminate the command with a semicolon. If you forgot to type “;” before hitting enter, you can type it on the next line.

```
mysql> select *
    -> from java_programming
```

```
    ->
    -> ;
```

- Use \c to cancel (abort) the current command

```
mysql> select * from java_programming \c
```

- If you open a single quote, without closing it, the new line prompt changes to '> instead of ->

```
mysql> select 'xxx
    '> '\c
```

- Use the up and down arrow keys to get previous or next commands from the command history.

- Enable the copy and paste functions of the Command Prompt if you are using Windows. To enable this function simply, click on the CMD icon -> Properties -> Options -> Edit Options -> Check “QuickEdit Mode”. You can now use right-click to copy and paste desired text.

Creating a new Database and a new Table, Inserting Records, Querying and Updating

A MySQL server contains many databases. In turn, a database contains many tables and a table contains many rows (records) and columns (fields).

We will now create a database called “student_db” and a table called “java_programming”. The table shall have three columns: id (of the type int), name (of the type varchar(50)) and grade (of the type float).

```
// Start a client with the new user "myuser"
// cd to the MySQL's bin directory
mysql -u myuser -p     // Windows
./mysql -u myuser -p   // Mac OS X

// Create a new database called 'student_db'
mysql> create database if not exists student_db;
Query OK, 1 row affected (0.08 sec)

// List all the databases on this server
mysql> show databases;
+--------------------+
```

```
| Database           |
+--------------------+
| ......             |
| student_db          |
| ......             |
+--------------------+
x rows in set (0.07 sec)

// Use "student_db" database as the default database
// You can refer to tables in the default database by the 'tablename' alone,
// instead of 'databasename.tablename'
mysql> use student_db;
Database changed

// Remove the table "java_programming" in the default database if it exists
mysql> drop table if exists java_programming;
Query OK, 0 rows affected, 1 warning (0.15 sec)


// Create a new table called "java_programming" in "student_db",
// with 3 columns of the specified types
mysql> create table java_programming (id int, name varchar(50), grade float);
Query OK, 0 rows affected (0.15 sec)

// List all the tables in "student_db"
mysql> show tables;
+---------------------+
| Tables_in_studentdb |
+---------------------+
| java_programming    |
+---------------------+
1 row in set (0.00 sec)

// Describe the "java_programming" table by listing its columns' definition
mysql> describe java_programming;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| id       | int(11)     | YES  |     | NULL    |       |
| name     | varchar(50) | YES  |     | NULL    |       |
| grade    | float       | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.04 sec)

// Insert a row into "java_programming" table.
// Strings are enclosed between single quotes.
// There are no quotes for int and float values.
mysql> insert into java_programming values (11, 'Sam Smith', 86);
Query OK, 1 row affected (0.03 sec)

// Insert another row into the "java_programming" table.
mysql> insert into java_programming values (22, 'Sarah Jones', 91);
Query OK, 1 row affected (0.03 sec)

// Select all columns and rows (*) from table "java_programming".
mysql> select * from java_programming;
```

```
+----+-------------+------+
| id | name        |grade |
+----+-------------+------+
| 11 | Sam Smith   |  86  |
| 22 | Sarah Jones |  91  |
+----+-------------+------+
2 rows in set (0.00 sec)

// Select some columns and rows from table "java_programming",
// that match certain the conditions
mysql> select name, grade from java_programming where grade > 90;
+-------------+-------+
| name        | grade |
+-------------+-------+
| Sarah Jones |  91   |
+-------------+-------+
1 rows in set (0.00 sec)

// Update the given field of the selected records
mysql> update java_programming set grade = 70 where name = 'Sam Smith';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from java_programming;
+----+-------------+-------+
| id | name        | grade |
+----+-------------+-------+
| 11 | Sam Smith   |  70   |
| 22 | Sarah Jones |  91   |
+----+-------------+-------+
2 rows in set (0.00 sec)

// Delete selected records
mysql> delete from java_programming where id = 22;
Query OK, 1 row affected (0.03 sec)

mysql> select * from java_programming;
+----+-------------+-------+
| id | name        | grade |
+----+-------------+-------+
| 11 | Sam Smith   |  70   |
+----+-------------+-------+
1 rows in set (0.00 sec)
```

Instead of entering commands one at a time, you can store a set of SQL commands in a file and run it. To do so:

- Use a text editor to create a new file called "mycommands.sql" that contains the following three SQL statements:

```
insert into java_programming values (33, 'Josh', 75);
insert into java_programming values (44, 'Kevin', 95);
Select * from java_programming;
```

- Save the file under "d:\myProject" for Windows or under "Documents" for Mac OS.

- After you created the file, you can use the following source command to run the SQL script:

```
mysql> source d:\myProject\mycommands.sql   // For Windows ONLY
mysql> source ~/Documents/mycommands.sql    // For Mac OS X ONLY
Query OK, 1 row affected (0.00 sec)         // INSERT command output
Query OK, 1 row affected (0.00 sec)         // INSERT command output
+------+-------------+-------+             // SELECT command output
| id   | name        | grade |
+------+-------------+-------+
|   11 | Sam Smith   |  70   |
|   33 | Josh        |  75   |
|   44 | Kevin       |  95   |
+------+-------------+-------+
3 rows in set (0.00 sec)
```

This was a short introduction to MySQL please use the **Reference Manual** to learn more.

Developing Database Applications Using JDBC

The JDBC API consists of classes and interfaces for establishing connections with databases, sending SQL statements to databases, and processing the results of SQL statements. To develop any database application using Java, four key interfaces are needed: Driver, Connection, Statement and ResultSet. These interfaces define the framework for generic SQL database access. The JDBC API defines these interfaces and the JDBC driver vendors provide the implementation for them. They are then used by programmers.

A JDBC application loads an appropriate driver using the Driver interface, connects to the database using the Connection interface, creates and executes SQL statements using the Statement interface, and processes the result using the ResultSet interface if the statement return results. Note, however, that some statements, such as SQL data definition statements and SQL data modification statements, do not return results.

In general, a JDBC program comprises of the following steps:

1. Connect to a database server by allocating a Connection object.
2. Allocate a Statement object, under the Connection object created, for holding a SQL command.
3. Using the Statement and Connection objects write a SQL query and execute it.
4. Process the query result.
5. Finally to free up resources, close the Statement and Connection objects.

Common JDBC Components

DriverManager: A class that manages the list of database drivers. The DriverManager matches connection requests from the java application with the correct database driver. This is done using communication sub-protocol. The first driver that recognises a specific subprotocol will be used under the JDBC will be used to establish a database connection.

Driver: An interface that handles the communication with the database server. Very rarely will you need to interact directly with Driver objects. You use DriverManager objects instead. DriverManager objects are used to manage Driver objects. These objects also abstracts the details associated with working with Driver objects.

Connection: An interface that is concerned with all methods for contacting a database. All communication with a database is done through a connection object.

Statement: An interface that is used to submit the SQL statements to the database.

ResultSet: ResultSet objects hold data retrieved from a database after executing a SQL query using Statement objects. It acts as an iterator to allow you to move through the retrieved data.

SQLException: A class that handles errors that occur in a database application.

Installing MySQL JDBC Driver

You need to install an appropriate JDBC driver to run your Java database programs. The MySQL's JDBC driver, "MySQL Connector/J", is available on the MySQL site.

To install the JDBC on Windows:

- Download the latest MySQL JDBC driver **here**.
    - Select "MySQL Connectors" -> "Connector/J"
    - Select "Platform Independent"-> "Zip Archive"
    - Select "No thanks, just start my download".
- UNZIP the download file into a temporary folder.
- From the temporary folder, copy the .jar file to your JDK's Extension Directory at "<JAVA_HOME>\jre\lib\ext" (where <JAVA_HOME> is the JDK installed directory), e.g., "c:\program files\java\jdk1.8.0_{xx}\jre\lib\ext".

To install the JDBC on Mac OS:

- Download the latest MySQL JDBC driver **here**.
    - Select "MySQL Connectors" -> "Connector/J"
    - Select "Platform Independent"-> Compressed TAR Archive
    - Select "No thanks, just start my download".
- Double-click on the downloaded TAR file to expand it.
- Copy the .jar file to your JDK's Extension Directory at "/Library/Java/Extensions"

Setting Up a Database

Before we can go any further, we need to set up a database. We will call this database "library_db" and create a table "books" within it with 4 columns: id, title, author, and qty.

id (int)	**title** (varchar(50))	**author** (varchar(50))	**qty** (int)
1001	Java for Beginners	John Holder	5
1002	Java	Sally Williams	5

	Fundamentals		
1003	A Cup of Java	Peter Jones	6
1004	Introduction to Java	Kumar Singh	6
1005	Advanced Java	Kelly Fields	7

- As shown previously, to create a database you need to start the MySQL server and start the MySQL client:

  For Windows:
  - Enter the following into your Command Prompt:

```
cd {path-to-mysql-bin}   // Check your MySQL installed directory
mysqld --console

mysql -u myuser -p
```

  For Mac:
  - To start the server select 'Apple' Icon -> System Preferences -> MySQL -> Start
  - Then enter the following into your terminal:

```
cd /usr/local/mysql/bin
./mysql -u myuser -p
```

  Note: remember to enter the password for the user "myuser" and not "root"

- Now, run the following SQL statements to create the database and table.

```
create database if not exists library_db;

use library_db;

drop table if exists books;
create table books (
   id int,
   title varchar(50),
   author varchar(50),
   qty int,
   primary key (id));

insert into books values (1001, 'Java for Beginners', 'John Holder', 5);
```

```
insert into books values (1002, 'Java Fundamentals', 'Sally Williams', 5);
insert into books values (1003, 'A Cup of Java', 'Peter Jones', 6);
insert into books values (1004, 'Introduction to Java', 'Kumar Singh', 6);
insert into books values (1005, 'Advanced Java', 'Kelly Fields', 7);

select * from books;
```

JDBC Programming

We will now demonstrate JDBC programming using some examples.

Issuing a SQL Select statement:

- Create a new Java source file called SelectTest.java. (Remember: the file name must be the same as the class name)
- Save the file in a directory of your choice.
- Enter the following code into your newly created file:

```
import java.sql.*;   // You need to use the 'Connection', 'Statement' and
                     // 'ResultSet' classes in java.sql package

public class SelectTest {

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

         // Step 1: Allocate a database 'Connection' object
         Connection conn = DriverManager.getConnection(
               // Database URL: "jdbc:mysql://hostname:port/databaseName",
               // "username", "password"
               "jdbc:mysql://localhost:3306/library_db?useSSL=false", "myuser",
"xxxx");


         // Step 2: Allocate a 'Statement' object in the Connection
         Statement stmt = conn.createStatement();
      ) {

         // Step 3: Execute a SQL SELECT query, the query result
         // is returned in a ResultSet object.
         String strSelect = "select title, qty from books";
         System.out.println("The SQL query is: " + strSelect);
         System.out.println();

         ResultSet rset = stmt.executeQuery(strSelect);

         // Step 4: Process the ResultSet by scrolling the cursor forward via next().
         // For each row, retrieve the contents of the cells with
```

```
         // getXxx(columnName).
         System.out.println("The records selected are:");
         int rowCount = 0;

         // Move the cursor to the next row, return false if no more row
         while(rset.next()) {
            String title = rset.getString("title");
            int    qty   = rset.getInt("qty");
            System.out.println(title + ", " + qty);
            ++rowCount;
         }

         System.out.println("Total number of records = " + rowCount);

      } catch(SQLException ex) {
         ex.printStackTrace();
      }

      // Step 5: Close the resources - Done automatically by try-with-resources
   }
}
```

- Compile the program by entering the following in a new Command Prompt or Terminal :

```
cd path-to-the-java-source-file
javac SelectTest.java
```

- Run the program by entering the following:

```
java SelectTest
```

Let's take a look at the program you just wrote. Firstly notice that there is not much actual programming involved in JDBC programming. You simply need to specify the database-URL, write the SQL query, and process the query result. The rest of the code can be thought of as a standard template for a JDBC program.

Now let's look at the following lines of code:

```
Connection conn = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/library_db?useSSL=false", "myuser", "xxxx");
```

jdbc:mysql://localhost:3306/library_db?useSSL=false is the database-URL and takes the following form:

```
jdbc:mysql://{host}:{port}/{database-name}
```

Port represents TCP port number of the MySQL server and database-name is the name of the database, which is in this instance library_db. “myuser” and “xxx” are the username and password of an authorised MySQL user.

Next let's examine this statement:

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

In this line, we are allocating a Statement object inside the Connection using conn.createStatement().

The next step in our process is to execute a SQL command (in this case Select). To do this we use the method stmt.executeQuery("SELECT …") which returns the query result in a ResultSet object called rset. The ResultSet models the table which is returned. This table can then be accessed using a row cursor. The cursor is initially positioned before the first row in ResultSet. It is then moved, using rset.next(), to the first row. rset,getXxx(column Name) can then be used to retrieve the value of the column for that row. Xxx corresponds to the data type of that column, for example, int, float, double and String. At the last row the rset.next() returns false, which terminates the while-loop. rset.getString(columnName) can also be used to retrieve all data types.

ResultSet columns within each row should be read in a left-to-right order, and each column should be read only once using the getXxx() methods. Issuing getXxx() to a cell more than once can cause an error.

A feature of JDK called try-with-resources automatically closes all the opened resources in the try-clause. It therefore closes the Connection and Statement objects automatically.

Issuing a SQL Update statement:

- Create a new Java source file called UpdateTest.java. (Remember: the file name must be the same as the class name)
- Save the file in a directory of your choice.
- Enter the following code into your newly created file:

```
import java.sql.*;

public class UpdateTest {    // Save as "UpdateTest.java"
   public static void main(String[] args) {
      try (
         // Step 1: Allocate a database 'Connection' object
         Connection conn = DriverManager.getConnection(
         "jdbc:mysql://localhost:3306/library_db?useSSL=false", "myuser", "xxxx");

         // Step 2: Allocate a 'Statement' object in the Connection
         Statement stmt = conn.createStatement();
      ) {
```

```
         // Step 3 & 4: Execute a SQL UPDATE via executeUpdate()
         // which returns an int indicating the number of rows affected.
         // Increase the qty by 5 for id=1001
         String strUpdate = "update books set qty = qty+5 where id = 1001";
         System.out.println("The SQL query is: " + strUpdate);
         int countUpdated = stmt.executeUpdate(strUpdate);
         System.out.println(countUpdated + " records affected.");

         // Step 3 & 4: Issue a SELECT to check the UPDATE.
         String strSelect = "select * from books where id = 1001";
         System.out.println("The SQL query is: " + strSelect);
         ResultSet rset = stmt.executeQuery(strSelect);
         while(rset.next()) {   // Move the cursor to the next row
            System.out.println(rset.getInt("id") + ", "
                    + rset.getString("author") + ", "
                    + rset.getString("title") + ", "
                    + rset.getInt("qty"));
         }
      } catch(SQLException ex) {
         ex.printStackTrace();
      }
      // Step 5: Close the resources - Done automatically by try-with-resources
   }
}
```

Unlike for Select, where we use executeQuery() to return a ResultSet object, Update, Insert and Delete returns an int value which indicates the number of records affected.

Issuing a SQL Insert and Delete statements:

- Create a new Java source file called InsertTest.java. (Remember: the file name must be the same as the class name)
- Save the file in a directory of your choice.
- Enter the following code into your newly created file:

```
import java.sql.*;


public class InsertTest {
   public static void main(String[] args) {
      try (

         // Step 1: Allocate a database 'Connection' object
         Connection conn = DriverManager.getConnection(
         "jdbc:mysql://localhost:3306/library_db?useSSL=false", "myuser", "xxxx");

         // Step 2: Allocate a 'Statement' object in the Connection
         Statement stmt = conn.createStatement();
      ) {
```

```
         // Step 3 & 4: Execute a SQL INSERT|DELETE statement using executeUpdate(),
         // which returns an int indicating the number of rows affected.

         // DELETE records with id>=3000 and id<4000
         String sqlDelete = "delete from books where id>=3000 and id<4000";
         System.out.println("The SQL query is: " + sqlDelete);
         int countDeleted = stmt.executeUpdate(sqlDelete);
         System.out.println(countDeleted + " records deleted.\n");

         // INSERT a record
         String sqlInsert = "insert into books "
               + "values (3001, 'Programming 101', 'Jane Doe', 1)";
         System.out.println("The SQL query is: " + sqlInsert);
         int countInserted = stmt.executeUpdate(sqlInsert);
         System.out.println(countInserted + " records inserted.\n");

         // INSERT multiple records
         sqlInsert = "insert into books values "
               + "(3002, 'Programming 101 2nd Edition', 'Jane Doe', 2),"
               + "(3003, 'Programming 101 3rd Edition', 'Jane Doe', 3)";
         System.out.println("The SQL query is: " + sqlInsert);
         countInserted = stmt.executeUpdate(sqlInsert);
         System.out.println(countInserted + " records inserted.\n");

         // INSERT a partial record
         sqlInsert = "insert into books (id, title, author) "
               + "values (3004, 'Java is Fun!', 'Kevin Peters')";
         System.out.println("The SQL query is: " + sqlInsert);
         countInserted = stmt.executeUpdate(sqlInsert);
         System.out.println(countInserted + " records inserted.\n");

         // Issue a SELECT to check the changes
         String strSelect = "select * from books";
         System.out.println("The SQL query is: " + strSelect);
         ResultSet rset = stmt.executeQuery(strSelect);

         // Move the cursor to the next row
         while(rset.next()) {
            System.out.println(rset.getInt("id") + ", "
                    + rset.getString("author") + ", "
                    + rset.getString("title") + ", "
                    + rset.getInt("qty"));
         }
      } catch(SQLException ex) {
         ex.printStackTrace();
      }
      // Step 5: Close the resources - Done automatically by try-with-resources
   }
}
```

Note that two records with the same primary key, in this case id, cannot be inserted. We therefore need to issue a Delete before Insert new record so that you can re-run the program. Also if you insert a partial record, the missing columns will

be set to their default values.

For more information on JDBC, you can find the homepage **here** or the online JDBC online tutorial **here**.

Compulsory Task 1

Follow these steps:

- Ensure that your environment is setup and you have followed all the steps outlined in this task.
- Using the MySQL client:
    - Insert the following 3 new rows into the java_programming table:

id	name	grade
55	Carl Davis	61
66	Dennis Fredrickson	88
77	Jane Richards	78

    - Select all records with a grade between 60 and 80.
    - Change Carl Davis's grade to 65.
    - Delete Dennis Fredrickson's row.
    - Change the grade of all people with an id greater than 55 to 80.
- After executing each instruction given above, take a screenshot of your console and send it to your mentor. Number your screenshots 1 to 5 in order of execution.
- Modify the Java program UpdateTest.java to set the qty for Introduction to Java to 0.
- Modify the Java program InsertTest.java to delete all books with id > 8000; and insert: (8001, 'Java ABC', 'Kevin Jones', 3) and (8002, 'Java XYZ', 'Kevin Jones', 5);

Compulsory Task 2

Follow these steps:

- Create a program that can be used by a bookstore clerk. The program should allow the clerk to:
    - enter new books into the database
    - update book information
    - delete books from the database
    - search the database to find a specific book.

- Create a database called ebookstore and a table called books. The table should have the following structure:

id	Title	Author	Qty
3001	A Tale of Two Cities	Charles Dickens	30
3002	Harry Potter and the Philosopher's Stone	J.K. Rowling	40
3003	The Lion, the Witch and the Wardrobe	C. S. Lewis	25
3004	The Lord of the Rings	J.R.R Tolkien	37
3005	Alice in Wonderland	Lewis Carroll	12

- Populate the table with the above values. You can also add your own values if you wish.

- The program should present the user with the following menu:

```
1. Enter book
2. Update book
3. Delete book
4. Search books
0. Exit
```

  The program should perform the function that the user selects. The

implementation of these functions are left up to you.

- Feel free to add more functionality and complexity to the program. This is your chance to show off all the programming concepts you have learnt so far!

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.