



**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 JDBC.



Get in touch

**Connect for support**

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

The best way to get help is to login to [www.hyperiondev.com/portal](https://www.hyperiondev.com/portal) to start a chat with a code reviewer. You can also schedule a call or get support via email.

Our expert code reviewers are 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 JAVA DATABASE CONNECTIVITY

JDBC is the Java API for developing Java database applications. Liang (2011, p. 1286) defines JDBC as providing a uniform interface to Java programmers for accessing and manipulating a wide range of relational databases. “Using the JDBC API, applications written in the Java programming language can execute SQL statements, retrieve results, present data in a user-friendly interface, and propagate changes back to the database” (pp. 1286-1287). This makes data handling an easy and user-friendly process. To use JDBC, you need to download a JDBC driver to link to the database you want to work with. Each database will have its own specific JDBC driver, such as the MySQL JDBC driver to access the MySQL database.

In summary with the JDBC API, you are able to (Ciubotaru, & Muntean, 2013, p. 136):

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 set up 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:

1. Download MySQL Installer from <https://dev.mysql.com/downloads/mysql/> and execute it.
2. Choose the appropriate Setup Type for your system. Typically you will choose Developer Default to install MySQL. If a default option is not provided, complete the following:
  - a. Click on the *General Availability (GA) Releases* tab.
  - b. Under *Select Operating System*, select *Microsoft Windows* from the dropdown menu.



General Availability (GA) Releases Archives

## MySQL Community Server 8.0.19

Select Operating System:  
Microsoft Windows

Looking for previous GA versions?

**Recommended Download:**

All MySQL Products. For All Windows Platforms. In One Package.

Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

**Windows (x86, 32 & 64-bit), MySQL Installer MSI** [Go to Download Page >](#)

**Other Downloads:**

Download	Version	Size	Action
Windows (x86, 64-bit), ZIP Archive (mysql-8.0.19-winx64.zip)	8.0.19	187.8M	<a href="#">Download</a>
MD5: f52c52e7b499958acc5f08ce0a869cab   <a href="#">Signature</a>			
Windows (x86, 64-bit), ZIP Archive Debug Binaries & Test Suite (mysql-8.0.19-winx64-debug-test.zip)	8.0.19	406.7M	<a href="#">Download</a>
MD5: 9b885558e74cc4629af77cbe75d11631   <a href="#">Signature</a>			

**i** We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

- c. Click on the *Go to Download Page* button and Download the *Windows (x86, 32-bit), MSI Installer* (The bigger one. For the 8.0.19 version it was 398.9 MB).

General Availability (GA) Releases Archives

## MySQL Installer 8.0.19

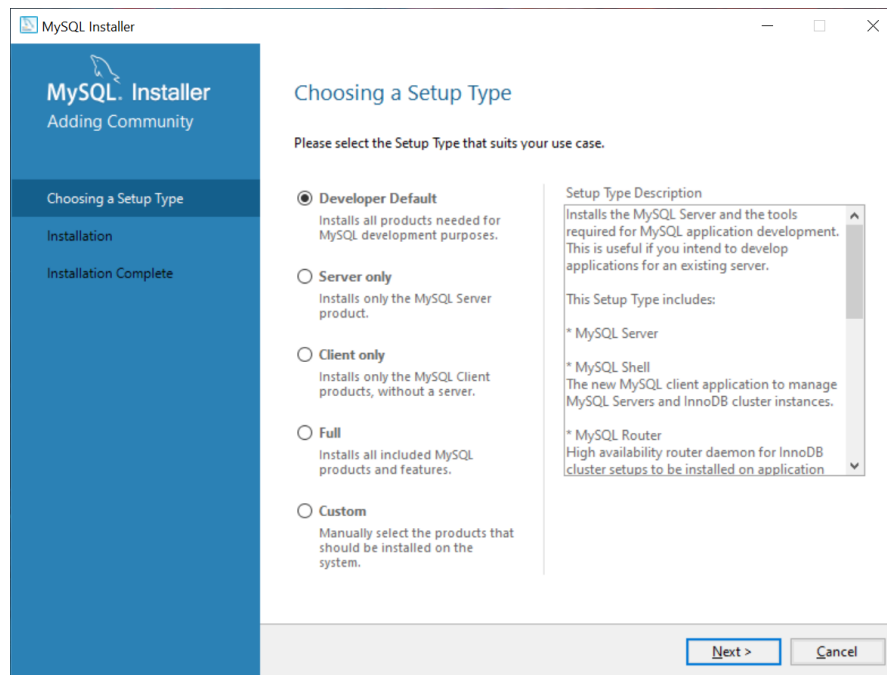
Select Operating System:  
Microsoft Windows

Looking for previous GA versions?

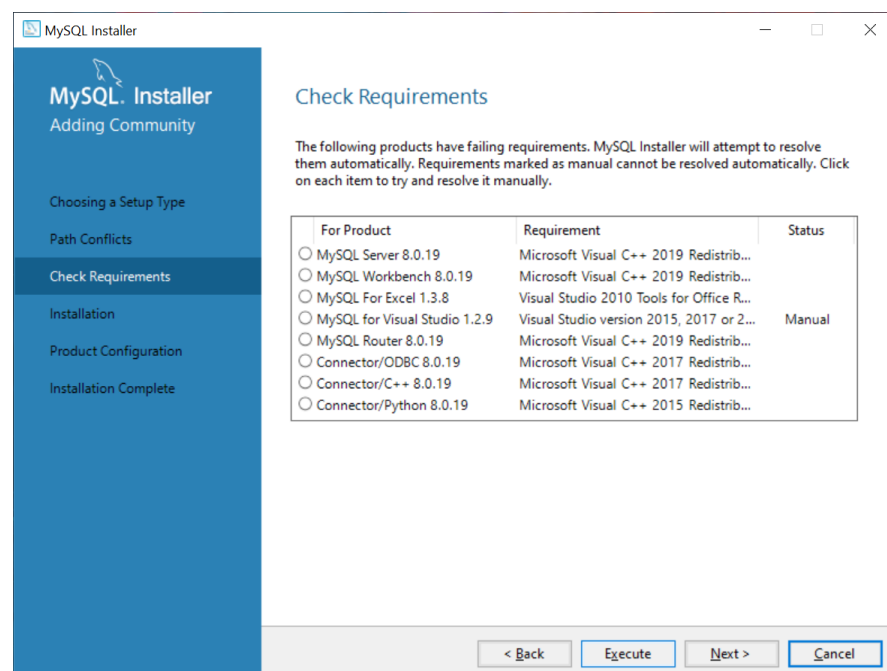
Download	Version	Size	Action
Windows (x86, 32-bit), MSI Installer (mysql-installer-web-community-8.0.19.0.msi)	8.0.19	18.6M	<a href="#">Download</a>
MD5: 32043776cb2239db45fddaa86dc0ad61   <a href="#">Signature</a>			
Windows (x86, 32-bit), MSI Installer (mysql-installer-community-8.0.19.0.msi)	8.0.19	398.9M	<a href="#">Download</a>
MD5: 1a882015da7fb93f20c4717e63b6817c   <a href="#">Signature</a>			

**i** We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

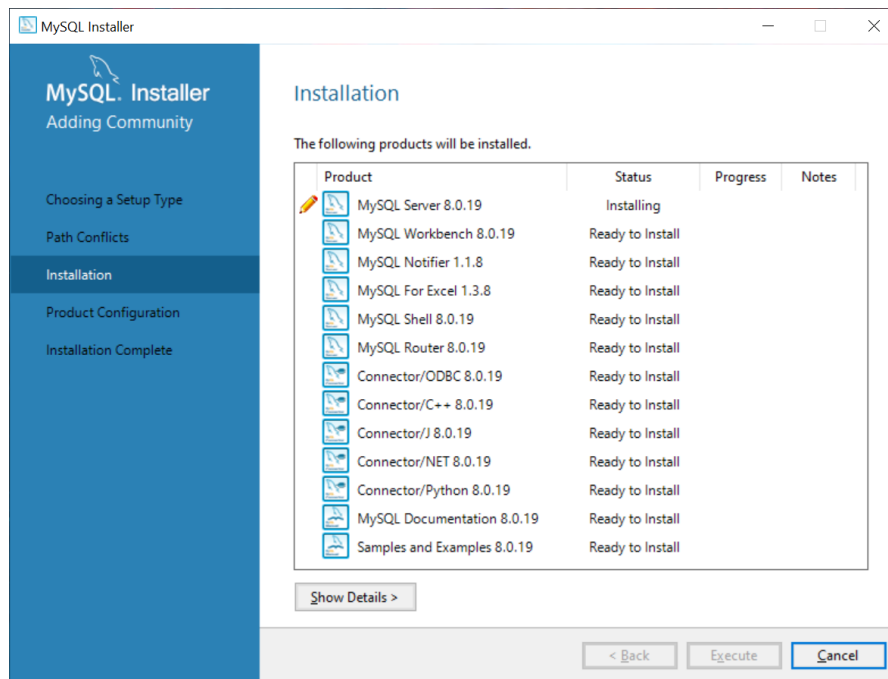
- It is not necessary to login or sign up. Simply click *No thanks, just start my download* to continue the installation process.
- Once it is downloaded, open the installer. Allow MySQL to make changes to your device.
- Select *Develop Default* as your *Setup Type* and click *Next*.



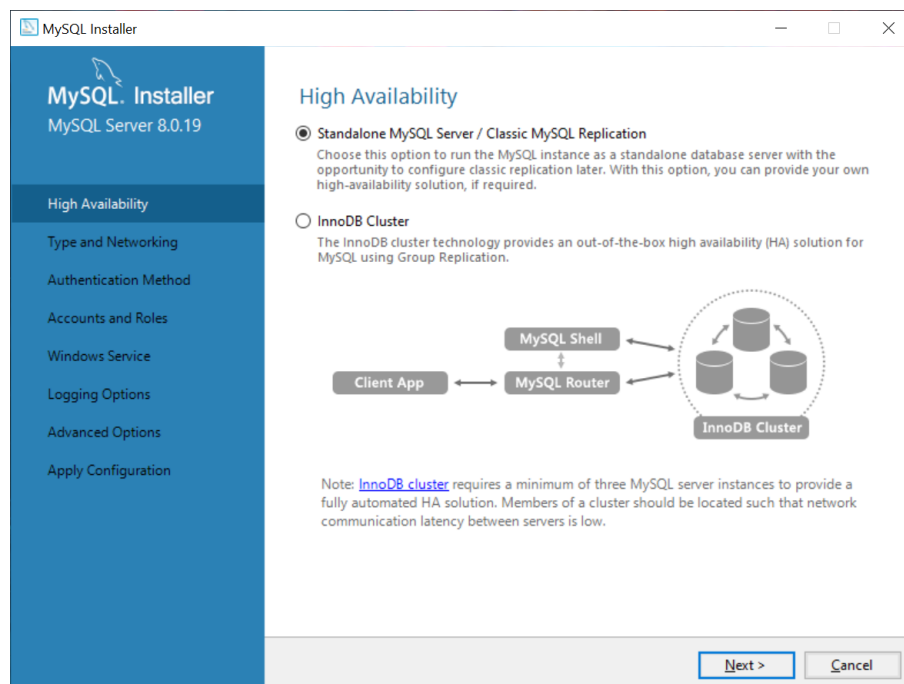
- When you get to the *Check Requirements* step, simply click *Execute* at the bottom. Allow all the required programs to be installed



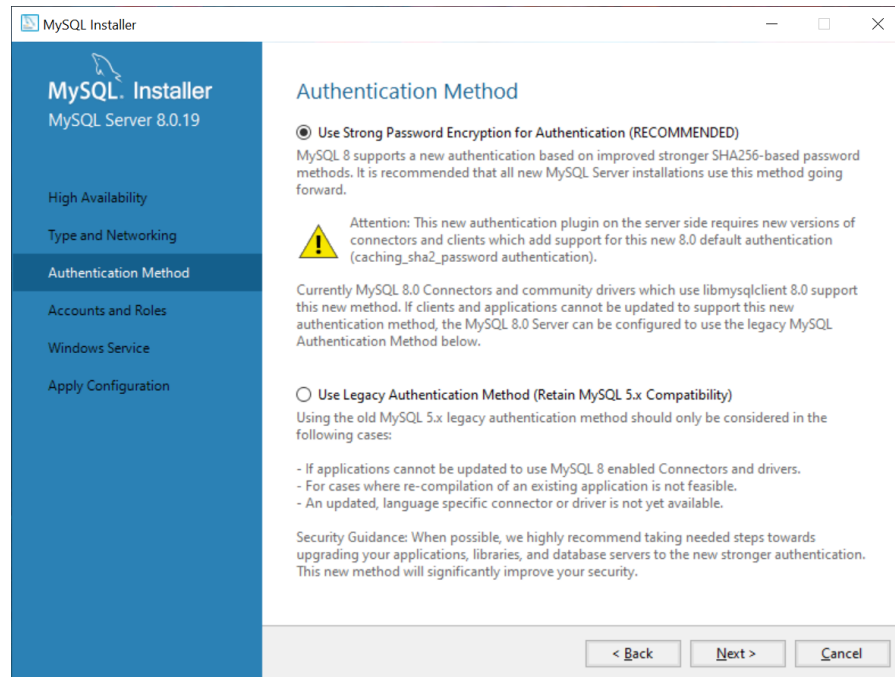
- Once the downloads are completed, click *Next*
- When you get to *Installation*, click *Execute* and wait for everything to download. Once that is completed, click *Next*



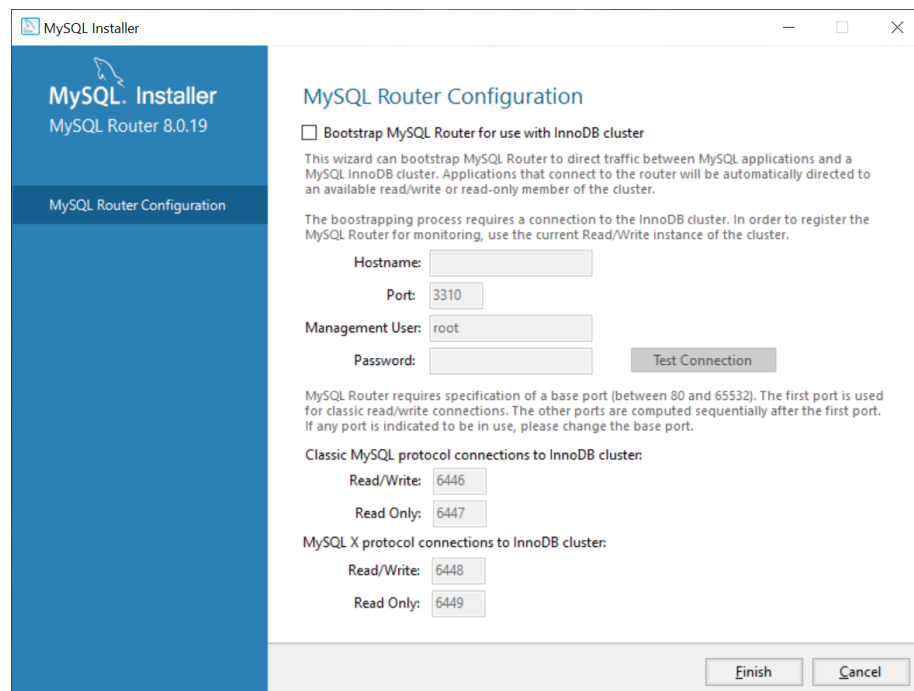
9. On the *Product Configuration* page, click *Next*. When asked about *High Availability*, choose *Standalone MySQL /Classic MySQL Replication* and click *Next*



10. On the *Authentication Method* page, select *Use Strong Password Encryption for Authentication* and click *Next*



11. **When asked to add a password, be sure to make a note of it so you don't forget it!** You can keep *root* as the user.



12. When asked, type in your created password for root and click *Next*
13. Finally, to *Apply Configuration*, click *Execute*. Once that is completed click *Finish*
14. Once you get to *Installation Complete*, click *Finish*. Your MySQL console and Workbench should open automatically.

## For Mac:

1. Download the *MySQL DMG Archive* from [MySQL website](#). Your operating system should be automatically detected, however, if it is not:
  - a. Select the *General Available (GA) Releases* tab.
  - b. In *Select Operating System* choose *macOS*
  - c. It is not necessary to login or sign up. Simply click *No thanks, just start my download* to continue the installation process
2. Install MySQL:
  - a. Go to *Downloads* and Double-click the ".dmg" file.
  - b. Double-click the downloaded MySQL download.
  - c. Follow the on-screen instructions to install MySQL.
  - d. 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!
  - e. MySQL will be installed in `/usr/local/mysql`. Take note of this installed directory!
  - f. Eject the ".dmg" file
3. If you make a mistake you need to:
  - a. stop the server by clicking on the 'Apple' Icon → System Preferences → MySQL → Stop
  - b. remove the directories `/usr/local/mysql-5.7.{xx}...`
  - c. re-run the installation
  - d. restart the server by clicking on the 'Apple' Icon → System Preferences → MySQL → Start
  - e. You may also need to reboot your machine.
4. Remember to take note of your MySQL installed directory.

## For Linux:

1. Check if your distribution includes MySQL in the native repository:
  - a. Open a terminal
  - b. Run **apt show install mysql-server** (as superuser if necessary)
  - c. Unless apt show returns "E: No packages found", skip to "Install MySQL from native repository".
2. Download the DEB package from the MySQL [website](#):
  - a. It is not necessary to login or sign up. Simply click *No thanks, just start my download* to continue the installation process
3. Install MySQL from download:
  - a. Open a terminal in the folder where you saved the DEB package
  - b. Run **dpkg -i: "dpkg -i [name of DEB package]"**



- c. At some point during installation, you will be prompted to enter a *root* password. **It is critical you remember this!** It is difficult to reset.
  - d. Congratulations! Installation is complete.
4. Install MySQL from native package manager:
  - a. From the terminal, run **sudo apt-get install mysql-server**
  - b. At some point during installation, you will be prompted to enter a *root* password. It is critical you remember this! It is difficult to reset.
  - c. After the server finishes installing, run **sudo apt-get install mysql-client**
  - d. Congratulations! Installation is complete.

## USING MYSQL

We will now briefly discuss the basics of using MySQL. Some of the examples below are based on Hock-Chuan of Nanyang Technological University's (2020a) guide to 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. To start up the server you need to do the following:

#### For Windows:

- If you used the MSI installer, MySQL should be running once your computer starts.

#### 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.

#### For Linux:

- From the terminal, run **service mysql start**
- This may prompt you for an admin password, after which it will start up.

## 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.

### For Linux:

- From the terminal, run **service mysql stop**
- This may prompt you for an admin password, after which it will shut down.

## 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.

We will now start a command-line client with the superuser "root".

### For Windows:

- Firstly, you need to find the directory path for the MySQL bin. The path should be C:\Program Files\MySQL\MySQL Server 8.0\bin
- Start a Command Prompt and enter the following:

```
cd C:\Program Files\MySQL\MySQL Server 8.0\bin

// Start a client as "root"
mysql -u root -p
Enter password:
    // Enter the root's password
    // Nothing will be shown when typing the password
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 8.0.19 MySQL Community Server - GPL

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
// You can now enter SQL commands.
```

### For Mac:

- Firstly, make sure you know the directory path for the MySQL bin folder.
- Open a new terminal and change the directory to your MySQL bin.

```
cd /usr/local/mysql/bin

// Start a client with "root"
./mysql -u root -p
Enter password:
    // Enter the root's password
    // Nothing will be shown when typing the password

Welcome to the MySQL monitor.  Commands end with ; or \g.
.....
mysql>
// You can now enter SQL commands.
```

### For Linux:

- From the terminal, run `mysql -u root -p`
- If access is denied even if the password is correct, it may be necessary to run this as superuser by using `sudo mysql -u root -p`.

## CHANGING THE PASSWORD FOR ROOT

The superuser "root" can do anything to the databases. This includes deleting all of them. Therefore, security is imperative when it comes to the "root" superuser. This means that you need 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:

```
// You need to replace XXXXX with your chosen password
// 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
mysql> quit
Bye
```

## 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 "otheruser". To create a new user you need to start a client with the "root" superuser as shown above.

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

// We can give otheruser the password swordfish
mysql> create user 'otheruser'@'localhost' identified by 'swordfish';
Query OK (0.01 sec)

// The newly created user has no privileges.
// Let's grant otheruser all privileges to all databases and tables

mysql> grant all on *.* to 'otheruser'@'localhost';
Query OK (0.01 sec)

mysql> quit

// Now otheruser has all the same privileges as root, except being able
to grant privileges
```

## CLIENT SESSION TIPS

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

- Terminate your command with a semicolon (;) like in Java.
- A single command can span many lines. To show continuation after pressing Enter, the new line prompt changes to ->. Once you are finished with the command add the ; at the end.
- Use \c to cancel (abort) the current command
- If you open a single quote without closing it the new line prompt changes to '>' instead of ->
- Use the up and down arrow keys to get previous or next commands from the command history.

## 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 “dogs\_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 weight (of the type float).

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

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

// List all the databases on this server
mysql> show databases;
+-----+
| Database |
+-----+
| .....  |
| dogs_db |
| .....  |
+-----+
x rows in set (0.07 sec)

// Use "song_db" database as the default database
mysql> use dogs_db;
Database changed
```

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

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

// List all the tables in "dogs_db"
mysql> show tables;
+-----+
| Tables_in_dogs_db |
+-----+
| 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 (1, 'Fluffy', 12.2);
Query OK, 1 row affected (0.03 sec)

// Insert another row into the "java_programming" table.
mysql> insert into java_programming values (2, 'Rover', 42);
Query OK, 1 row affected (0.03 sec)

// Select all columns and rows (*) from table "java_programming".
mysql> select * from java_programming;
+----+-----+-----+
| id | name   | weight |
+----+-----+-----+
| 1  | Fluffy | 12.2   |
| 2  | Rover  | 42     |
+----+-----+-----+
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 weight < 20;
+-----+-----+
| name   | weight |
+-----+-----+
```

```

+-----+-----+
| Fluffy      | 12.2 |
+-----+-----+
1 rows in set (0.00 sec)

// Update the given field of the selected records
mysql> update java_programming set weight = 36.4 where name = 'Rover';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from java_programming;
+----+-----+-----+
| id | name      | weight |
+----+-----+-----+
| 1  | Fluffy    | 12.2   |
| 2  | Rover     | 36.4   |
+----+-----+-----+
2 rows in set (0.00 sec)

// Delete selected records
mysql> delete from java_programming where id = 2;
Query OK, 1 row affected (0.03 sec)

mysql> select * from java_programming;
+----+-----+-----+
| id | name      | weight |
+----+-----+-----+
| 1  | Fluffy    | 12.2   |
+----+-----+-----+
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 (3, 'Patch', 6);
insert into java_programming values (4, 'Denzel', 54);
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 | weight |
+-----+-----+-----+
| 1 | Fluffy | 12.2 |
| 3 | Patch | 6 |
| 4 | Denzel | 54 |
+-----+-----+-----+
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

According to Liang (2011, p. 1287), “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.” This means that a program written in Java can access any database by using SQL.

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 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 abstract 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. MySQL's JDBC driver, "MySQL Connector/J", is available on the MySQL site, but is also automatically downloaded with the MSI installer if you have Windows. The information below is based on Hock-Chuan's explanations and guides (2020b).

### To install the JDBC on Windows:

If you used the MSI installer, you should have Connector J in your *Program Files* (x86) folder. If not, follow the following steps:

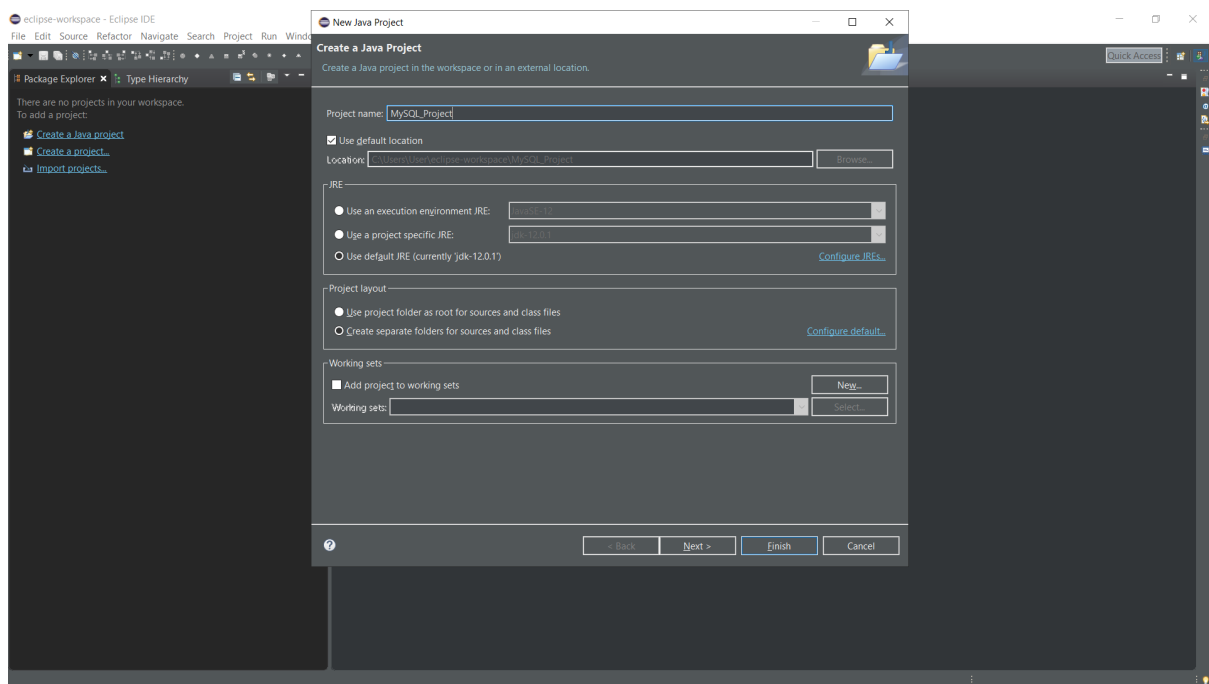
- Download the latest MySQL JDBC driver [here](#).
  - Select *MySQL Connectors* → *Connector/J*
  - Select *Platform Independent* → *Zip Archive*
  - Select *No thanks, just start my download*.
- UNZIP the download file into a temporary folder.
- From the temporary folder, copy the .jar file to your 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:

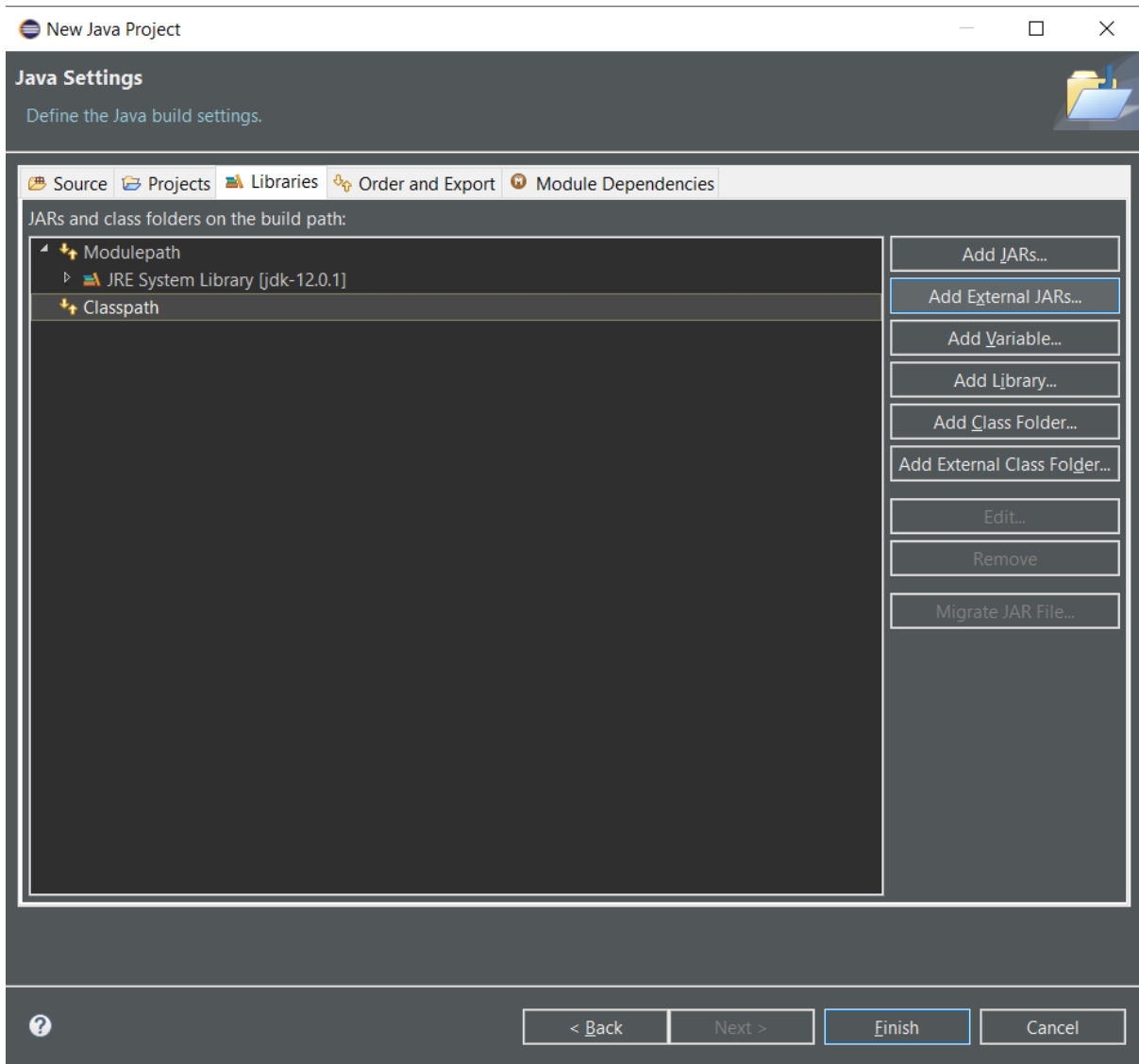
- Download the latest MySQL JDBC driver [here](#).
  - Select *MySQL Connectors* → *Connector/J*
  - Select *Platform Independent* → *Compressed TAR Archive*
  - Select *No thanks, just start my download*
- Double-click on the downloaded TAR file to expand it.
- Copy the .jar file to your JDK's Extension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

## To connect your Java Application to your MySQL server:

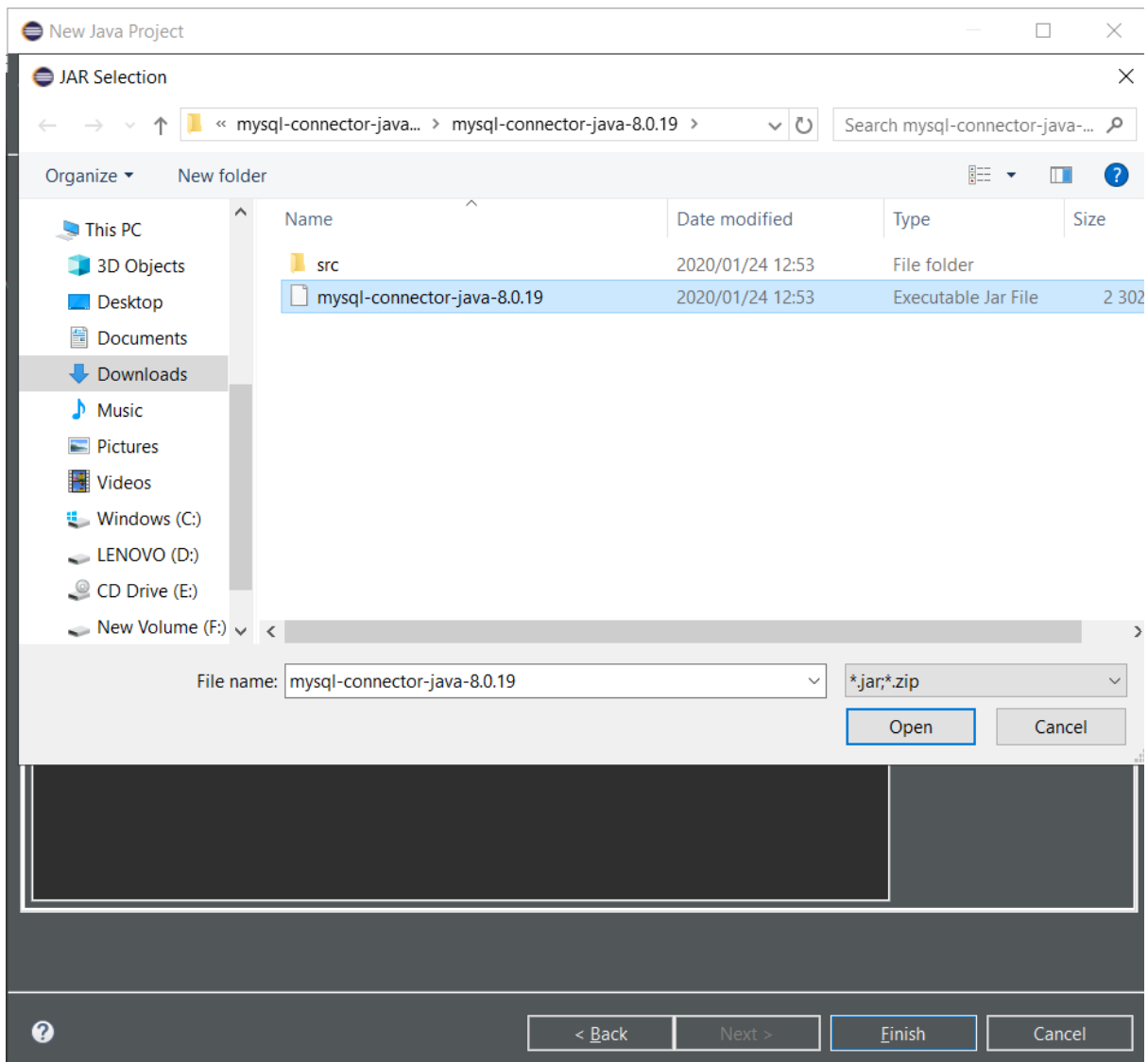
- Create a project in Eclipse



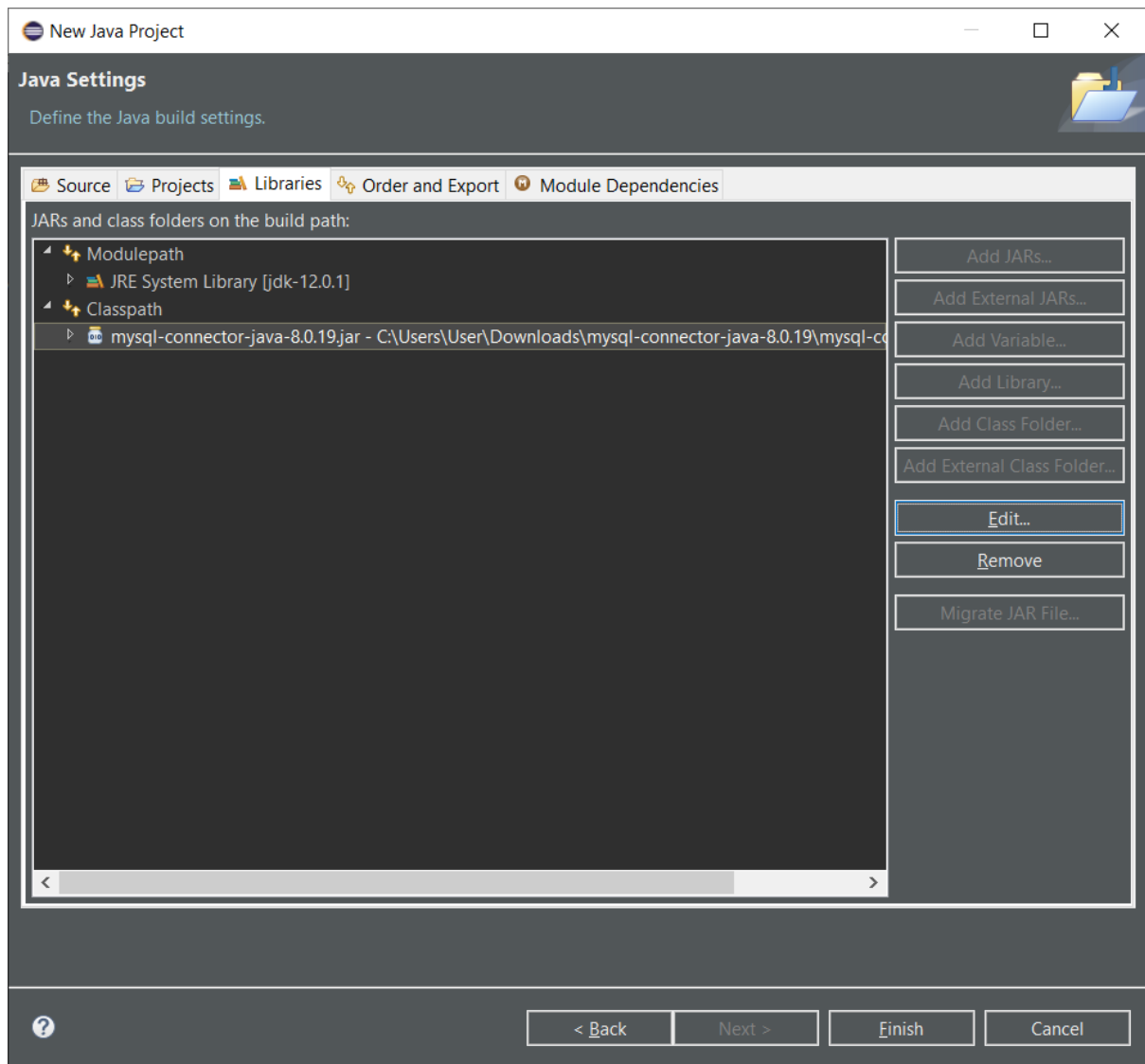
- Instead of clicking the finish option to take the default setup, click the “Next” button, which will allow for some further project setup.
- On the next screen, from the tabs on top, select the “Libraries” tab, and select the “Classpath” option.
- From here, select the “Add External JARS” option.



- Click on it and find the .jar file you've downloaded earlier. At the time of writing, the file was named Mysql-connector-java-8.0.19.jar
- Click the "Open" option.



- After that, click the “Finish” button and your project should be able to interface with your Server already.



## 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.

<b>id</b> (int)	<b>title</b> (varchar(50))	<b>author</b> (varchar(50))	<b>qty</b> (int)
1001	Java for Beginners	John Holder	5
1002	Java Fundamentals	Sally Williams	5

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
mysql -u otheruser -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 otheruser -p
```

Note: remember to enter the password for the user "otheruser" 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 (Hock-Chuan, 2020b). The code below shows how to select, insert, update and delete entries in a database.

### Issuing SQL statement:

- Create a new Java source file called **Main.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 Main {

    public static void main(String[] args) {
        try {
            // Connect to the library_db database, via the jdbc:mysql:
            // channel on localhost (this PC)
            // Use username "otheruser", password "swordfish".
            Connection connection = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/library_db?useSSL=false",
                "otheruser",
                "swordfish"
            );
            // Create a direct line to the database for running our queries
            Statement statement = connection.createStatement();
            ResultSet results;
            int rowsAffected;
            // Set up finished, do some stuff:

            // executeQuery: runs a SELECT statement and returns the
            results.
            results = statement.executeQuery("SELECT title, qty FROM
            books");
            // Loop over the results, printing them all.
            while (results.next()) {
```

```

        System.out.println(results.getString("title") + ", " +
+results.getInt("qty"));
    }

    // Add a new book:
    rowsAffected = statement.executeUpdate(
        "INSERT INTO books VALUES (3001, 'Programming 101',
'Jane Doe', 1)"
    );
    System.out.println("Query complete, " + rowsAffected + " rows
added.");
    printAllFromTable(statement);

    // Change a book:
    rowsAffected = statement.executeUpdate(
        "UPDATE books SET qty=500 WHERE id=1001"
    );
    System.out.println("Query complete, " + rowsAffected + " rows
updated.");
    printAllFromTable(statement);

    // Clear a book:
    rowsAffected = statement.executeUpdate(
        "DELETE FROM books WHERE id=3001"
    );
    System.out.println("Query complete, " + rowsAffected + " rows
removed.");
    printAllFromTable(statement);

    // Close up our connections
    results.close();
    statement.close();
    connection.close();

    } catch (SQLException e) {
        // We only want to catch a SQLException - anything else is
off-limits for now.
        e.printStackTrace();
    }
}

/**
 * Method printing all values in all rows.
 * Takes a statement to try to avoid spreading DB access too far.
 *
 * @param a statement on an existing connection
 * @throws SQLException
 */

```



```

    public static void printAllFromTable(Statement statement) throws
SQLException{

        ResultSet results = statement.executeQuery("SELECT id, title,
author, qty FROM books");
        while (results.next()) {
            System.out.println(
                results.getInt("id") + ", "
                + results.getString("title") + ", "
                + results.getString("author") + ", "
                + results.getInt("qty")
            );
        }
    }
}

```



### Take note:

You may encounter an error when trying to run your code that looks something like this: "SQLException: Access denied for user 'otheruser'@'localhost'". To solve this, from the MySQL console, execute "ALTER USER 'otheruser'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'your\_password\_here';" to activate the mysql\_native\_password plugin. Then do "FLUSH PRIVILEGES;" to reset.

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. In this example, our username is "otheruser" and our password is "swordfish", but you should change those to your MySQL username and password:

```

Connection connection = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/library_db?useSSL=false",
    "otheruser",
    "swordfish"
);

```

`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 the TCP port number of the MySQL server and database-name is the name of the database, which is in this instance `library_db`.

Next let's examine this statement:

```
Statement statement = connection.createStatement();
```

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

## Select

The next step in our process is to execute a SQL command (in this case `Select`). To do this we use the method `statement.executeQuery("SELECT ...")`, which returns the query result in a `ResultSet` object called `results`. 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 `results.next()`, to the first row. `results.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 `results.next()` returns `false`, which terminates the while-loop. `results.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.

## Insert and Delete

The insert and delete actions are fairly straightforward, as you can see from the code above. It is worth noting that you cannot add an entry if an existing entry has the same primary key; you will first have to delete the existing entry.

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

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 set up 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 **Main.java** to set the qty for Introduction to Java to 0.
- Modify the Java program **Main.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 is 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!

## Completed the task(s)?

Ask an expert to review your work!

[Review work](#)



Rate us

**Share your thoughts**

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

Think that 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.



## References:

Ciubotaru, B., & Muntean, G. (2013). *Advanced network programming – Principles and techniques: Network application programming with Java* (p. 136). London: Springer Science & Business Media.

Hock-Chuan, C. (2020a). Java Tutorial - An Introduction to Java Database Programming (JDBC) by Examples with MySQL. Retrieved 19 February 2020, from [https://www3.ntu.edu.sg/HOME/EHCHUA/PROGRAMMING/java/JDBC\\_Basic.html](https://www3.ntu.edu.sg/HOME/EHCHUA/PROGRAMMING/java/JDBC_Basic.html)

Hock-Chuan, C. (2020b). An Introduction to Java Database Programming (JDBC) by Examples. Retrieved 20 February 2020, from [https://www3.ntu.edu.sg/HOME/EHCHUA/PROGRAMMING/java/JDBC\\_Basic2.html](https://www3.ntu.edu.sg/HOME/EHCHUA/PROGRAMMING/java/JDBC_Basic2.html)

Liang, Y. (2011). *Introduction to Java programming: Comprehensive version* (8th ed., pp. 1273-1308). New Jersey: Prentice Hall.