LESSON

# Java Database Programming, Testing & Documentation

Visit our website

Copyright © 2018 Hyperion Development. All rights reserved.

# Introduction

## WELCOME TO THE INTRODUCTION TO THE JDBC TASK!

In this lesson, we learn how to connect to a MySQL database with JDBC, and cover procedures for testing and documentation. At the end there is a task for you to put into practice what you have learned.

## INTENDED LEARNING OUTCOMES

You will know that you have succeeded at this task if, by the time you complete it, you are able to:

- Write program code for database access for a computer application using SQL
- Test programs for a computer application that accesses a database using SQL
- Document programs for a computer application that accesses a database using SQL
- Install and run MySQL servers and clients, and use the JDBC to create java applications that interact with these

Refer to the self-assessment table provided after the programming task near the end of the document for a checklist to assess your own degree of achievement of each of these learning outcomes. Try to gauge your own learning and address any weak areas before submitting your task, which will become part of your formative assessment portfolio.

**This lesson addresses Unit Standard 114048: Create database access for a computer application using structured query language.**

**Specific Outcomes:**

1. **SO1: Review the requirements for database access for a computer application using SQL (Range: Piecemeal development, Data sharing, Integration, Abstraction, Data Independence, Data models, Data Definition Language, Data Manipulation Language, Data Control, Language (at least 4))**
2. **SO2: Design database access for a computer application using SQL. (Range: Simple indexes, Multi-level indexes, Index-sequential file, Tree structures, SQL CREATE INDEX and DROP INDEX statements (at least 3))**
3. **SO3: Write program code for database access for a computer application using SQL. (Range: Row types, User-defined types, User**

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

**defined routines, Reference types, Collection types, Support for large objects, Stored procedures, Triggers (at least 3))**

4. **SO4: Test programs for a computer application that accesses a database using SQL. (Range: Logic tests; Access functions; Debugging.)**
5. **SO5: Document programs for a computer application that accesses a database using SQL. (Range: Clarity; Simplicity; Ease of Use)**

## PROGRAM CODE FOR DATABASE ACCESS FOR A COMPUTER APPLICATION USING SQL (SO3)

In this lesson, you're going to extend your knowledge of SQL using the MySQL variant, and get further practice writing code, testing your code, and documenting your code using comments.

## 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**.

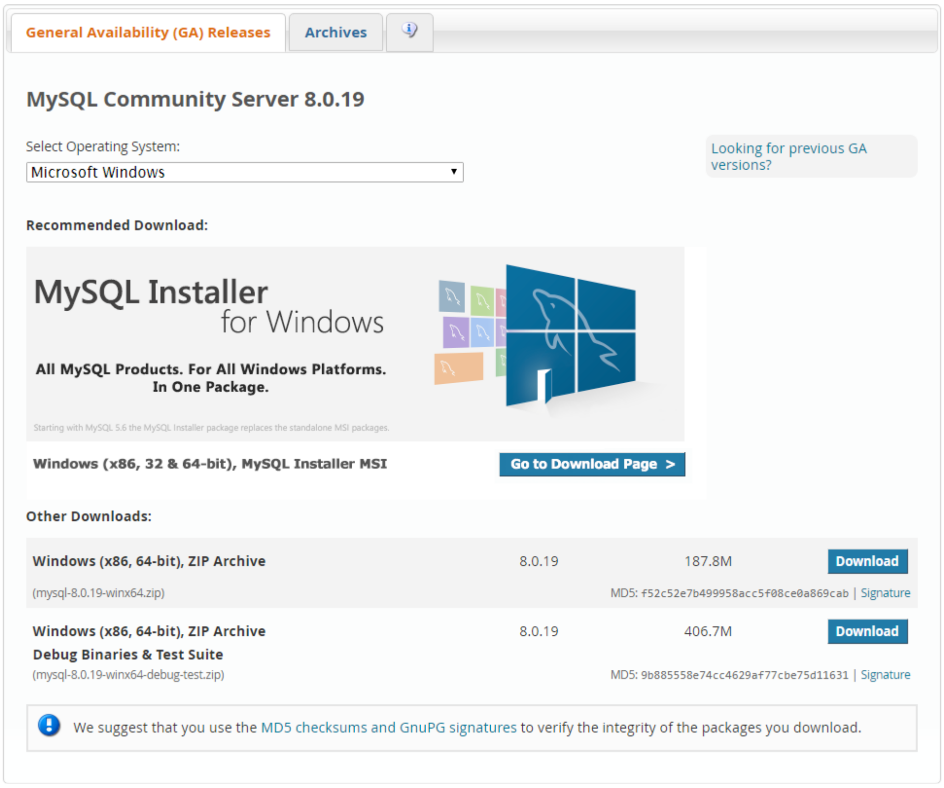
We will now explain the steps to download and install MySQL.

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

## DOWNLOAD AND INSTALL MYSQL

**For Windows:**
Download MySQL Installer from **https://dev.mysql.com/downloads/mysql/** and execute it.

1. 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:**

MySQL Installer
for Windows
All MySQL Products. For All Windows Platforms.
In One Package.
Starting with MySQL 5.6 the MySQL Installer package replaces the standalone MSI packages.

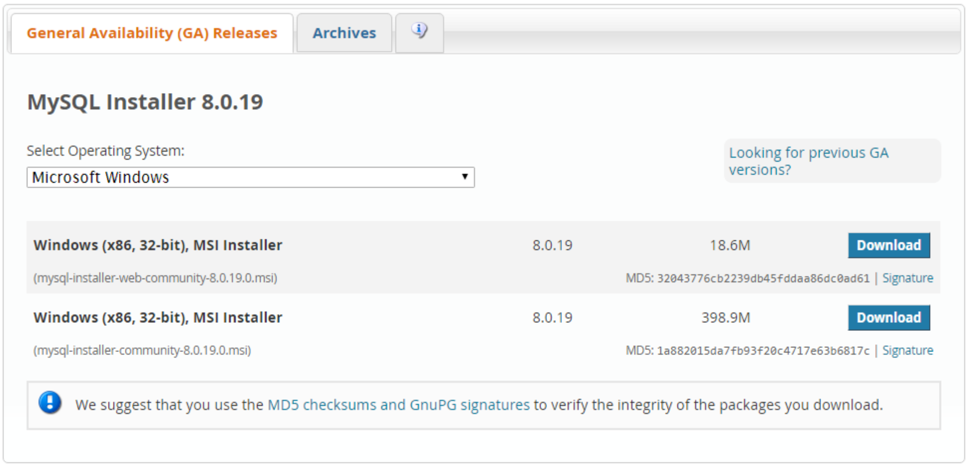
**Windows (x86, 32 & 64-bit), MySQL Installer MSI** — Go to Download Page >

**Other Downloads:**

| | | | |
|---|---|---|---|
| **Windows (x86, 64-bit), ZIP Archive** | 8.0.19 | 187.8M | Download |
| (mysql-8.0.19-winx64.zip) | | MD5: f52c52e7b499958acc5f08ce0a869cab \| Signature | |
| **Windows (x86, 64-bit), ZIP Archive Debug Binaries & Test Suite** | 8.0.19 | 406.7M | Download |
| (mysql-8.0.19-winx64-debug-test.zip) | | MD5: 9b885558e74cc4629af77cbe75d11631 \| Signature | |

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).



**MySQL Installer 8.0.19**

Select Operating System:
Microsoft Windows

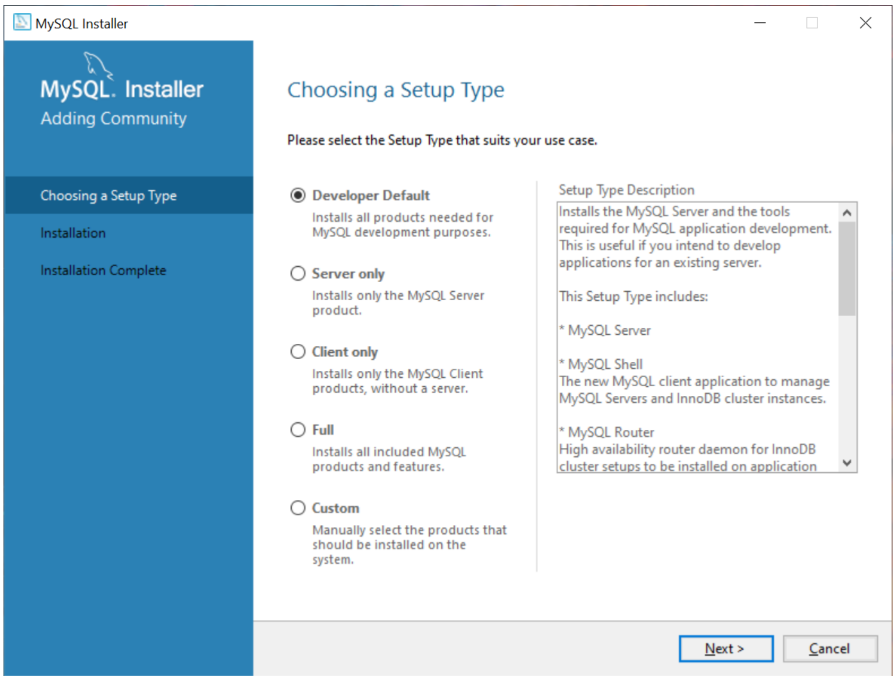
Looking for previous GA versions?

| | | | |
|---|---|---|---|
| **Windows (x86, 32-bit), MSI Installer** | 8.0.19 | 18.6M | Download |
| (mysql-installer-web-community-8.0.19.0.msi) | | MD5: 32043776cb2239db45fddaa86dc0ad61 \| Signature | |
| **Windows (x86, 32-bit), MSI Installer** | 8.0.19 | 398.9M | Download |
| (mysql-installer-community-8.0.19.0.msi) | | MD5: 1a882015da7fb93f20c4717e63b6817c \| Signature | |

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

Copyright © 2018 Hyperion Development. All rights reserved.

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



Choosing a Setup Type

Please select the Setup Type that suits your use case.

- [✓ Developer Default] Installs all products needed for MySQL development purposes.
- Server only: Installs only the MySQL Server product.
- Client only: Installs only the MySQL Client products, without a server.
- Full: Installs all included MySQL products and features.
- Custom: Manually select the products that should be installed on the system.

Setup Type Description

Installs the MySQL Server and the tools required for MySQL application development. This is useful if you intend to develop applications for an existing server.

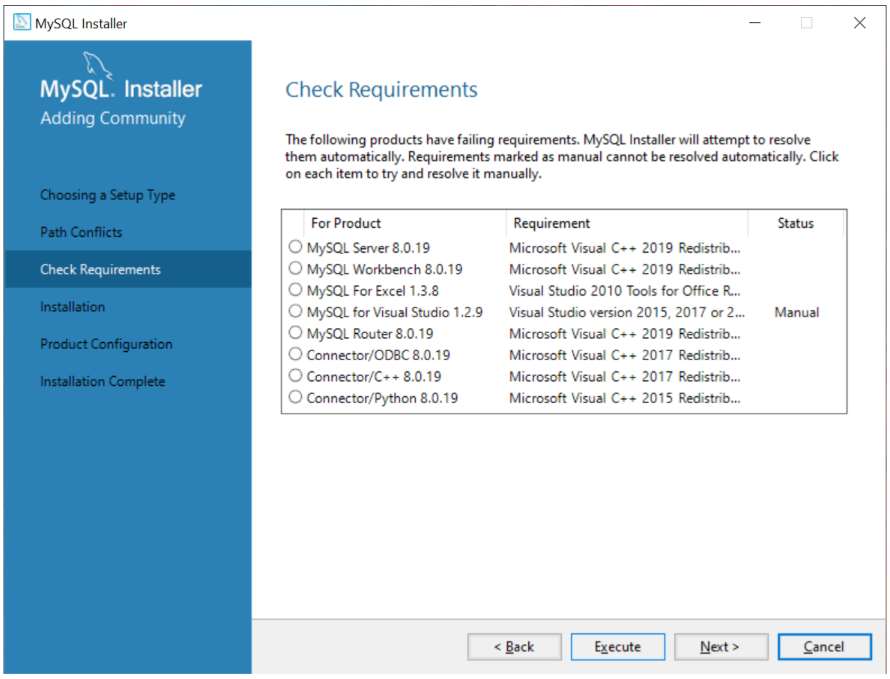
This Setup Type includes:

* MySQL Server

* MySQL Shell
The new MySQL client application to manage MySQL Servers and InnoDB cluster instances.

* MySQL Router
High availability router daemon for InnoDB cluster setups to be installed on application

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



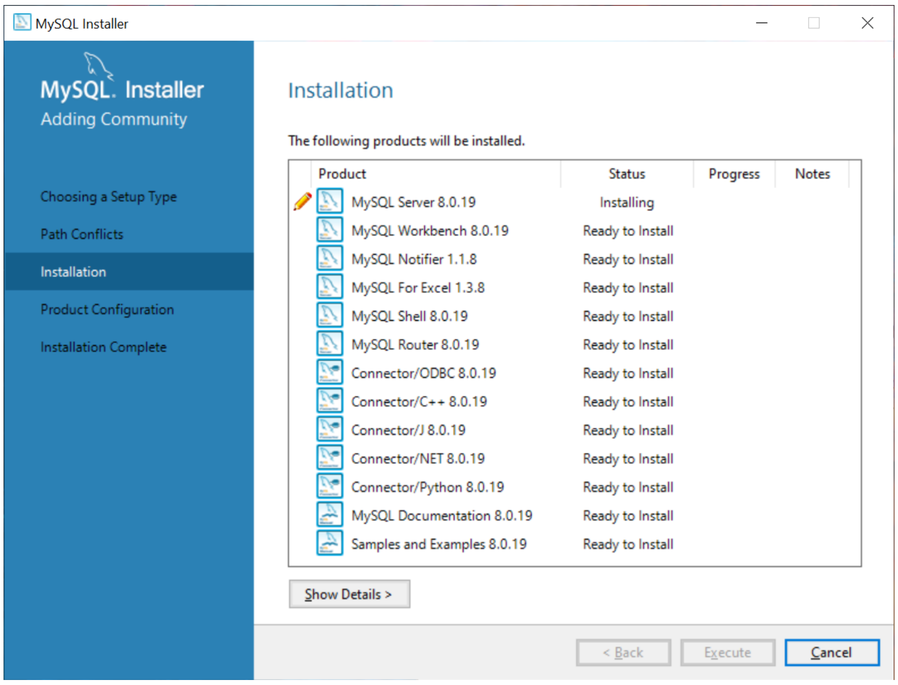
Check Requirements

The following products have failing requirements. MySQL Installer will attempt to resolve them automatically. Requirements marked as manual cannot be resolved automatically. Click on each item to try and resolve it manually.

| For Product | Requirement | Status |
|---|---|---|
| MySQL Server 8.0.19 | Microsoft Visual C++ 2019 Redistrib... | |
| MySQL Workbench 8.0.19 | Microsoft Visual C++ 2019 Redistrib... | |
| MySQL For Excel 1.3.8 | Visual Studio 2010 Tools for Office R... | |
| MySQL for Visual Studio 1.2.9 | Visual Studio version 2015, 2017 or 2... | Manual |
| MySQL Router 8.0.19 | Microsoft Visual C++ 2019 Redistrib... | |
| Connector/ODBC 8.0.19 | Microsoft Visual C++ 2017 Redistrib... | |
| Connector/C++ 8.0.19 | Microsoft Visual C++ 2017 Redistrib... | |
| Connector/Python 8.0.19 | Microsoft Visual C++ 2015 Redistrib... | |

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

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

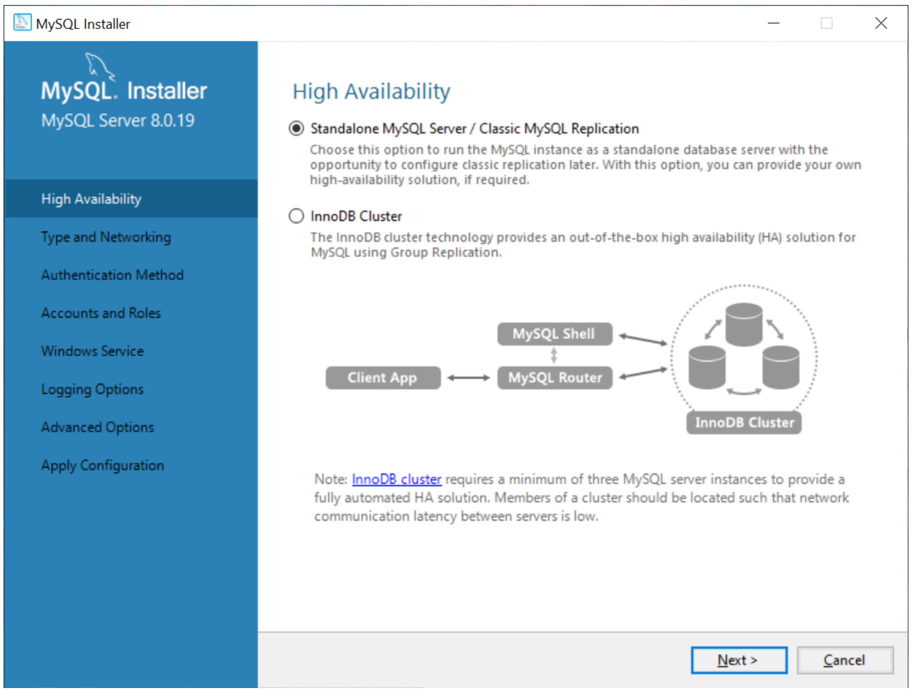

MySQL Installer

Installation

The following products will be installed.

| Product | Status | Progress | Notes |
|---|---|---|---|
| MySQL Server 8.0.19 | Installing | | |
| MySQL Workbench 8.0.19 | Ready to Install | | |
| MySQL Notifier 1.1.8 | Ready to Install | | |
| MySQL For Excel 1.3.8 | Ready to Install | | |
| MySQL Shell 8.0.19 | Ready to Install | | |
| MySQL Router 8.0.19 | Ready to Install | | |
| Connector/ODBC 8.0.19 | Ready to Install | | |
| Connector/C++ 8.0.19 | Ready to Install | | |
| Connector/J 8.0.19 | Ready to Install | | |
| Connector/NET 8.0.19 | Ready to Install | | |
| Connector/Python 8.0.19 | Ready to Install | | |
| MySQL Documentation 8.0.19 | Ready to Install | | |
| Samples and Examples 8.0.19 | Ready to Install | | |

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


MySQL Installer

High Availability

(●) Standalone MySQL Server / Classic MySQL Replication

Choose this option to run the MySQL instance as a standalone database server with the opportunity to configure classic replication later. With this option, you can provide your own high-availability solution, if required.

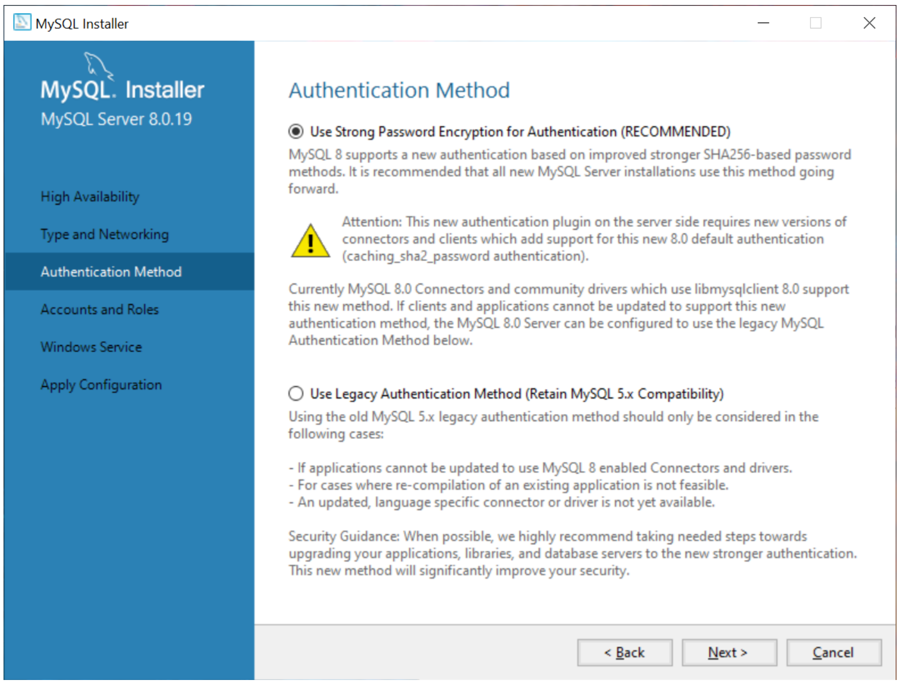
( ) InnoDB Cluster

The InnoDB cluster technology provides an out-of-the-box high availability (HA) solution for MySQL using Group Replication.

Note: InnoDB cluster requires a minimum of three MySQL server instances to provide a fully automated HA solution. Members of a cluster should be located such that network communication latency between servers is low.

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

Copyright © 2018 Hyperion Development. All rights reserved.


MySQL Installer

MySQL Installer
MySQL Server 8.0.19

High Availability
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration

Authentication Method

● Use Strong Password Encryption for Authentication (RECOMMENDED)

MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.

Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).

Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.

○ Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)

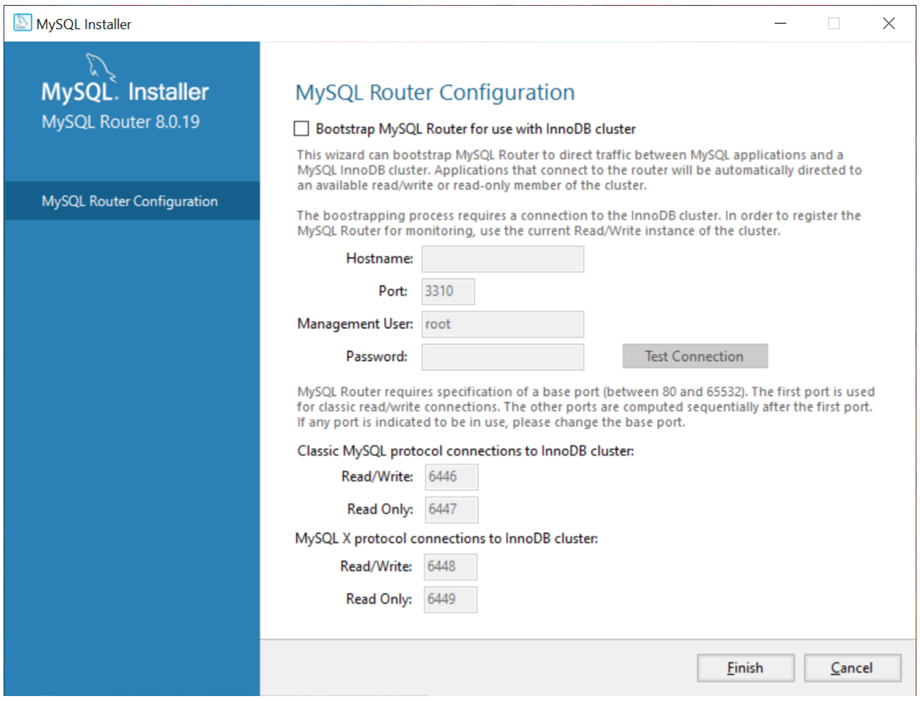
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:

- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.

Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.

< Back | Next > | Cancel

10. **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.


MySQL Installer

MySQL Installer
MySQL Router 8.0.19

MySQL Router Configuration

MySQL Router Configuration

☐ Bootstrap MySQL Router for use with InnoDB cluster

This wizard can bootstrap MySQL Router to direct traffic between MySQL applications and a MySQL InnoDB cluster. Applications that connect to the router will be automatically directed to an available read/write or read-only member of the cluster.

The boostrapping process requires a connection to the InnoDB cluster. In order to register the MySQL Router for monitoring, use the current Read/Write instance of the cluster.

Hostname:
Port: 3310
Management User: root
Password:
Test Connection

MySQL Router requires specification of a base port (between 80 and 65532). The first port is used for classic read/write connections. The other ports are computed sequentially after the first port. If any port is indicated to be in use, please change the base port.

Classic MySQL protocol connections to InnoDB cluster:
Read/Write: 6446
Read Only: 6447

MySQL X protocol connections to InnoDB cluster:
Read/Write: 6448
Read Only: 6449

Finish | Cancel

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

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

**For Mac:**

1. Download the *MySQL DMG Archive* from the **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`

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

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
......
```

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

```
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.
```

Copyright © 2018 Hyperion Development. All rights reserved.

```
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 a 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
```

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

```
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.

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

## 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   |
+---------------------+
```

Copyright © 2018 Hyperion Development. All rights reserved.

```
| 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    |
```

Copyright © 2018 Hyperion Development. All rights reserved.

```
|  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.

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

**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**.

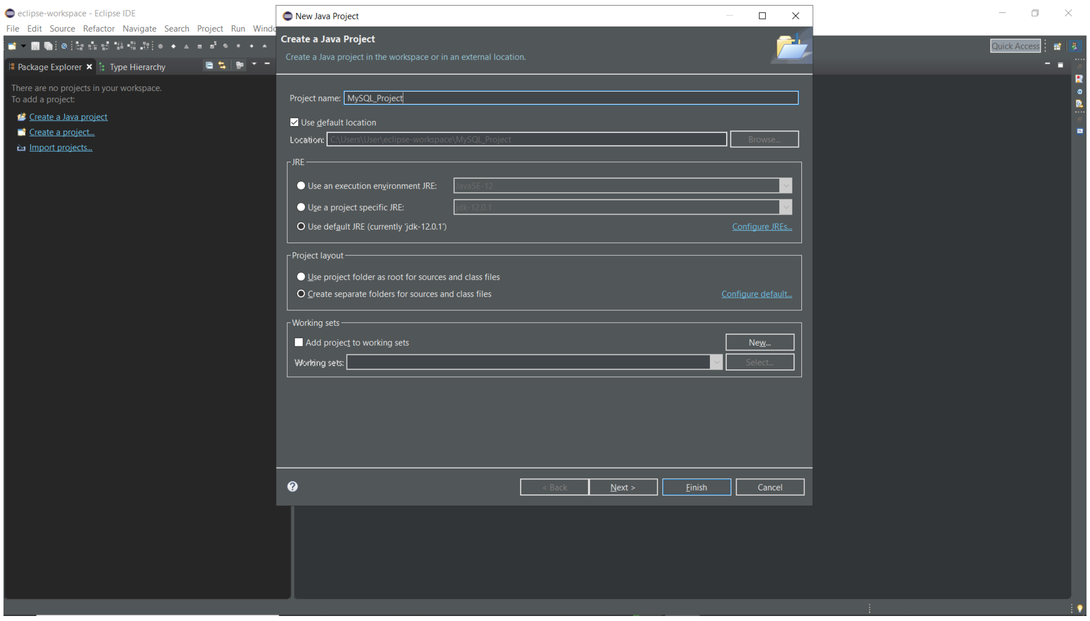
- Select *MySQL Connectors → Connector/J*
- Select *Platform Independent → Compressed TAR Archive*
- Select *No thanks, just start my download*

- 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

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.


eclipse-workspace - Eclipse IDE

Package Explorer | Type Hierarchy

There are no projects in your workspace.
To add a project:
Create a Java project
Create a project...
Import projects...

New Java Project

**Create a Java Project**

Create a Java project in the workspace or in an external location.

Project name: MySQL_Project

☑ Use default location

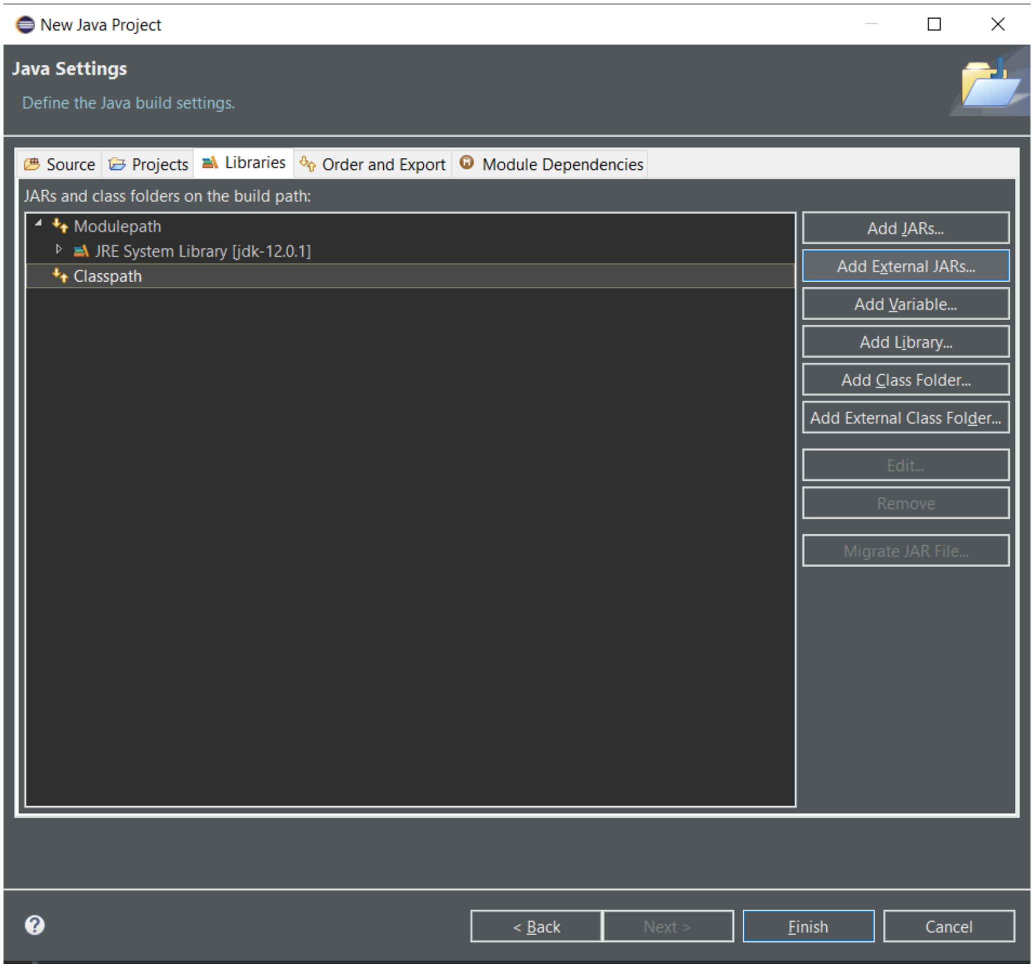
Location: C:\Users\User\eclipse-workspace\MySQL_Project

JRE
- Use an execution environment JRE: JavaSE-12
- Use a project specific JRE: jdk-12.0.1
- Use default JRE (currently 'jdk-12.0.1')

Project layout
- Use project folder as root for sources and class files
- Create separate folders for sources and class files

Working sets
- ☐ Add project to working sets
- Working sets:

- 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.


New Java Project

**Java Settings**

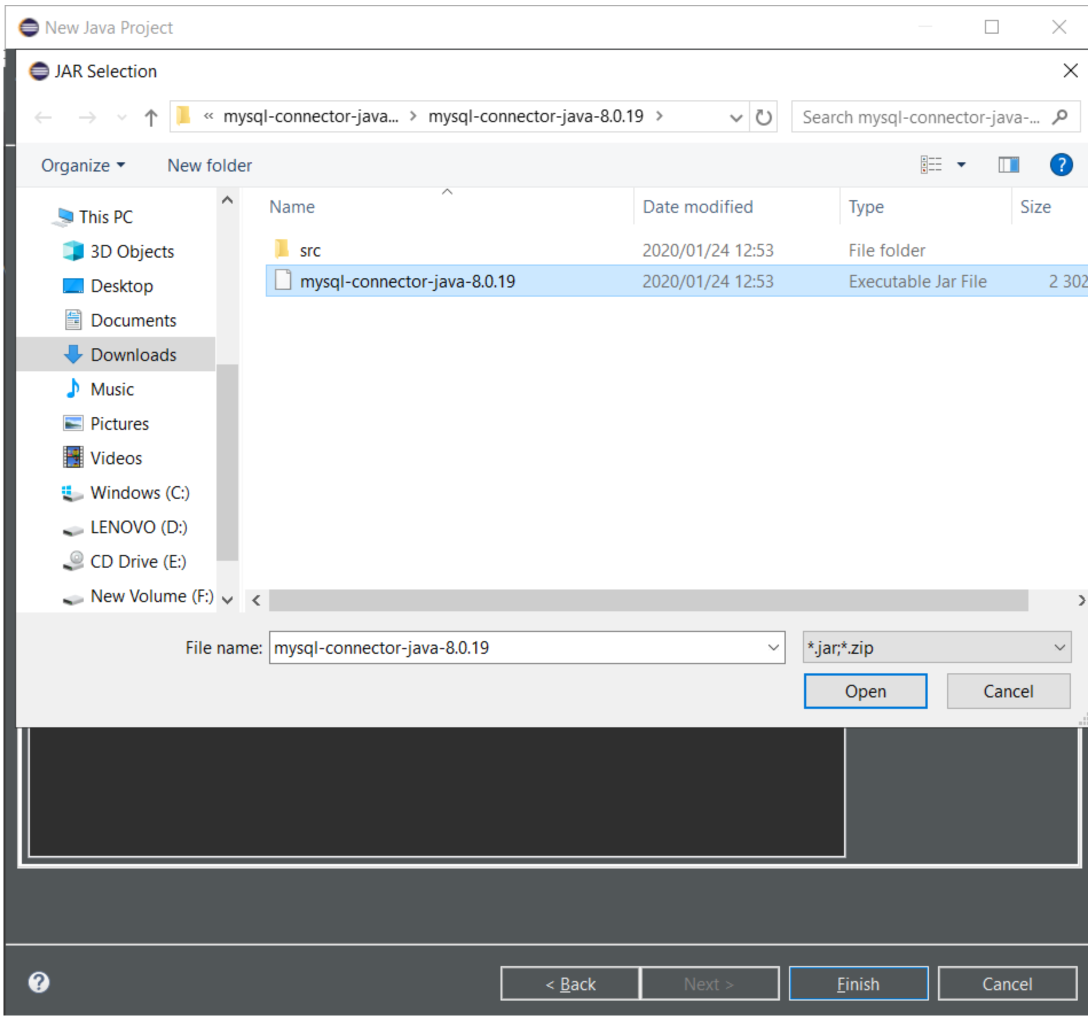
Define the Java build settings.

Source | Projects | Libraries | Order and Export | Module Dependencies

JARs and class folders on the build path:

- Modulepath
  - JRE System Library [jdk-12.0.1]
- Classpath

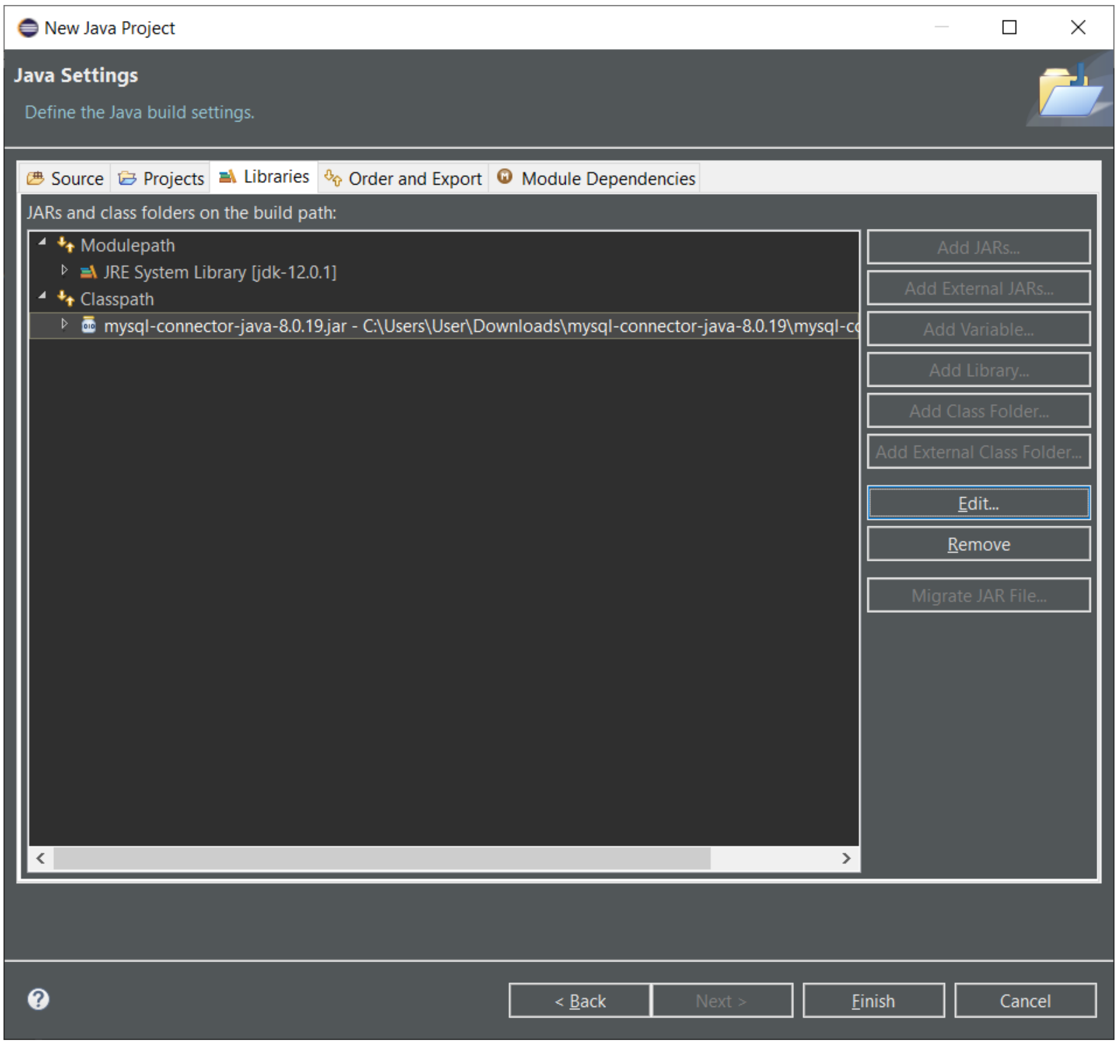
- Click on it and find the .jar file you've downloaded earlier. At the time of writing, the file was named Mysql-connector-java-8.0.19.jar
- Click the "Open" option.



| Name | Date modified | Type | Size |
|---|---|---|---|
| src | 2020/01/24 12:53 | File folder | |
| mysql-connector-java-8.0.19 | 2020/01/24 12:53 | Executable Jar File | 2 302 |

File name: mysql-connector-java-8.0.19
*.jar;*.zip

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



JARs and class folders on the build path:

```
Modulepath
  JRE System Library [jdk-12.0.1]
Classpath
  mysql-connector-java-8.0.19.jar - C:\Users\User\Downloads\mysql-connector-java-8.0.19\mysql-c
```

## 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 called “books” within it with 4 columns: id, title, author, and qty.

| **id**<br>(int) | **title**<br>(varchar(50)) | **author**<br>(varchar(50)) | **qty**<br>(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 |

Copyright © 2018 Hyperion Development. All rights reserved.

- 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. Not that this time you will set up the table to use the ID field as the index, i.e. primary key.

```
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;
```

Copyright © 2018 Hyperion Development. All rights reserved.

## 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 statements:**

- 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")
```

Copyright © 2018 Hyperion Development. All rights reserved.

```
                );
            }
        }
}
```

A note from our coding mentor
**Melanie**

*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:

```
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. “otheruser” and “swordfish” are the username and password of an authorised MySQL user.

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.

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.

**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.

**Update**
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**.

## TESTING AND DEBUGGING (SO4)

Normally the first step in debugging is to attempt to reproduce the problem. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Also, specific user environments and usage history can make it difficult to reproduce the problem.

After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. For example, a bug in a compiler can make it crash when parsing some large source file. However, after simplification of the test case, only a few lines from the original source file can be sufficient to reproduce the same crash. Such simplification can be made manually, using

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

a divide-and-conquer approach. The programmer will try to remove some parts of the original test case and check if the problem still exists. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.

After the test case is sufficiently simplified, a programmer can use a debugger tool to examine program states (values of variables, plus the call stack) and track down the origin of the problem(s). Alternatively, tracing can be used. In simple cases, tracing is just a few print statements, which output the values of variables at certain points of program execution.

- **Print debugging** (or tracing) is the act of watching (live or recorded) trace statements, or print statements, that indicate the flow of execution of a process.

- **Remote debugging** is the process of debugging a program running on a system different from the debugger. To start remote debugging, a debugger connects to a remote system over a network. The debugger can then control the execution of the program on the remote system and retrieve information about its state.

- **Post-mortem debugging** is debugging of the program after it has already crashed. Related techniques often include various tracing techniques, and/or analysis of memory dump (or core dump) of the crashed process. The dump of the process could be obtained automatically by the system (for example, when the process has terminated due to an unhandled exception), or by a programmer-inserted instruction, or manually by the interactive user.

- **"Wolf fence" algorithm**: Edward Gauss described this simple but very useful and now famous algorithm in a 1982 article for communications of the ACM as follows: "There's one wolf in Alaska; how do you find it? First build a fence down the middle of the state, wait for the wolf to howl, determine which side of the fence it is on. Repeat the process on that side only, until you get to the point where you can see the wolf." This is implemented e.g. in the Git version control system as the command git bisect, which uses the above algorithm to determine which commit introduced a particular bug.

- **Delta Debugging** - technique of automating test case simplification

- **Saff Squeeze** - technique of isolating failure within the test using progressive inlining of parts of the failing test.

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

## DOCUMENTATION (SO5)

The documents associated with a software project and the system being developed have a number of associated requirements:

1. They should act as a communication medium between members of the development team.
2. They should be a system information repository to be used by maintenance engineers.
3. They should provide information for management to help them plan, budget, and schedule the software development process.
4. Some of the documents should tell users how to use and administer the system.

### Product documentation

Product documentation is concerned with describing the delivered software product. Unlike most process documentation, it has a relatively long life. It must evolve in step with the product which it describes. Product documentation includes user documentation which tells users how to use the software product and system documentation which is principally intended for maintenance engineers.

### User Documentation

Users of a system are not all the same. The producer of documentation must structure it to cater for different user tasks and different levels of expertise and experience. It is particularly important to distinguish between end-users and system administrators:

1. End-users use the software to assist with some task. This may be flying an aircraft, managing insurance policies, writing a book, etc. They want to know how the software can help them.

2. System administrators are responsible for managing the software used by end-users. This may involve acting as an operator if the system is a large mainframe system, as a network manager in the system involves a network of workstations or as a technical guru who fixes end-users software problems and who liaises between users and the software supplier.

To cater for these different classes of user and different levels of user expertise, there are at least 5 documents (or perhaps chapters in a single document) which should be delivered with the software system (Figure1).

The *functional description* of the system outlines the system requirements and briefly describes the services provided. This document should provide an overview of the system. Users should be able to read this document with an

Copyright © 2018 Hyperion Development. All rights reserved.

introductory manual and decide if the system is what they need.

The *system installation document* is intended for system administrators. It should provide details of how to install the system in a particular environment. It should contain a description of the files making up the system and the minimal hardware configuration required. The permanent files which must be established, how to start the system and the configuration dependent files which must be changed to tailor the system to a particular host system should also be described. The use of automated installers for PC software has meant that some suppliers see this document as unnecessary. In fact, it is still required to help system managers discover and fix problems with the installation.

The *introductory manual* should present an informal introduction to the system, describing its 'normal' usage. It should describe how to get started and how end-users might make use of the common system facilities. It should be liberally illustrated with examples. Inevitably beginners, whatever their background and experience, will make mistakes. Easily discovered information on how to recover from these mistakes and restart useful work should be an integral part of this document.

A more general *system administrator's guide* should be provided for some types of system such as command and control systems. This should describe the messages generated when the system interacts with other systems and how to react to these messages. If system hardware is involved, it might also explain the operator's task in maintaining that hardware. For example, it might describe how to clear faults in the system console, how to connect new peripherals, etc.

As well as manuals, other, easy-to-use documentation might be provided. A quick reference card listing available system facilities and how to use them is particularly convenient for experienced system users. On-line help systems, which contain brief information about the system, can save the user spending time in consultation of manuals although should not be seen as a replacement for more comprehensive documentation.

**System Documentation**

System documentation includes all of the documents describing the system itself from the requirements specification to the final acceptance test plan. Documents describing the design, implementation and testing of a system are essential if the program is to be understood and maintained. Like user documentation, it is important that system documentation is structured, with overviews leading the reader into more formal and detailed descriptions of each aspect of the system.

For large systems that are developed to a customer's specification, the system documentation should include:

1. The requirements document and an associated rationale.

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

2. A document describing the system architecture.
3. For each program in the system, a description of the architecture of that program.
4. For each component in the system, a description of its functionality and interfaces.
5. Program source code listings. These should be commented where the comments should explain complex sections of code and provide a rationale for the coding method used. If meaningful names are used and a good, structured programming style is used, much of the code should be self-documenting without the need for additional comments. This information is now normally maintained electronically rather than on paper with selected information printed on demand from readers.
6. Validation documents describing how each program is validated and how the validation information relates to the requirements.

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

## 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:

  - Create a **new** table in the dogs_db database called student (id int, name varchar(50), grade float);
  - Insert the following 3 new rows into the student 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 place it in a document to send in for review. 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);

Copyright © 2018 Hyperion Development. All rights reserved.

## Compulsory Task 2

Follow these steps:

- Create a program that can be used by a bookstore clerk. Use the comment functionality to document what you have done making the code easily understandable to anyone reading your program. 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 (note that the id field is the primary key):

| 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.

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.

- Include comments to explain complex sections of code and provide a rationale for the coding method used.
- Test and debug your code.
- 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!

## Self-assessment table

| **Intended learning outcome (SAQA US 114048)** | Assess your own proficiency level for each intended learning outcome. Check each box when the statement becomes true- **I can successfully**: | | | | |
|---|---|---|---|---|---|
| Write program code for database access for a computer application using SQL (SO 3) | Create and read data | Update data | Delete data | Use criteria to select a limited subset of data searched or on which specific operations are performed | |
| Test programs for a computer application that accesses a database using SQL (SO 4) | Understand and be able to apply the testing and debugging process for code | | Explain and apply including different types of and approaches to debugging (e.g. print, remote, post-mortem, wolf-fence, delta, Saff squeeze) | | |
| Document programs for a computer application that accesses a database using SQL (SO 5) | Understand and be able to create product documentation | Understand and be able to create user documentation | Understand and be able to create system documentation | | |
| **Intended learning outcomes (Hyperion extension)** | Assess your own proficiency level for each intended learning outcome. Check each box when the statement becomes true- **I can successfully**: | | | | |
| Install and run MySQL servers and clients, and use the JDBC to create java | Download and install MySQL | Run a MySQL server and | Create MySQL users and | Connect a java applica | Within a MySQL database, create new |

Hyperiondev
Copyright © 2018 Hyperion Development. All rights reserved.

| | | | | | |
|---|---|---|---|---|---|
| applications that interact with these and run MySQL code | and the JDK | client sessions | databases | tion to a MySQL server | tables,insert records, query, update, and delete |

Rate us

## Share your thoughts

Hyperion 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**

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**

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

Hyperiondev Copyright © 2018 Hyperion Development. All rights reserved.