DATABASE SYSTEM – LAB 10

CONNECTING TO A DATABASE SERVER FROM A PROGRAM

I. Objective

In this practice lab, you will learn how to connect to a database server (MS SQL) from a program written in java language and perform basic operations such as getting data from the server, inserting data, updating data and deleting data from the server.

There are many ways to connect to a database server from a java program, in this practice lab we will be using java JDBC (**Java Database Connectivity**) which is the standard and the traditional way to connect and interact with a database server from a java program.

By the end of this lab, you will know how to:

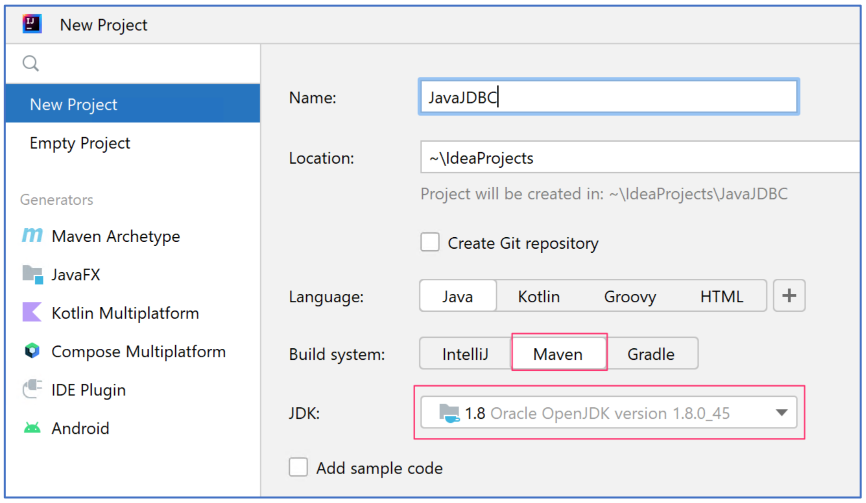
- Use maven tool to manage a java program dependencies
- Connect to a database server
- Perform CRUD (Create-Read-Update-Delete) tasks on the server

II. Instruction

2.1. Environment Setup

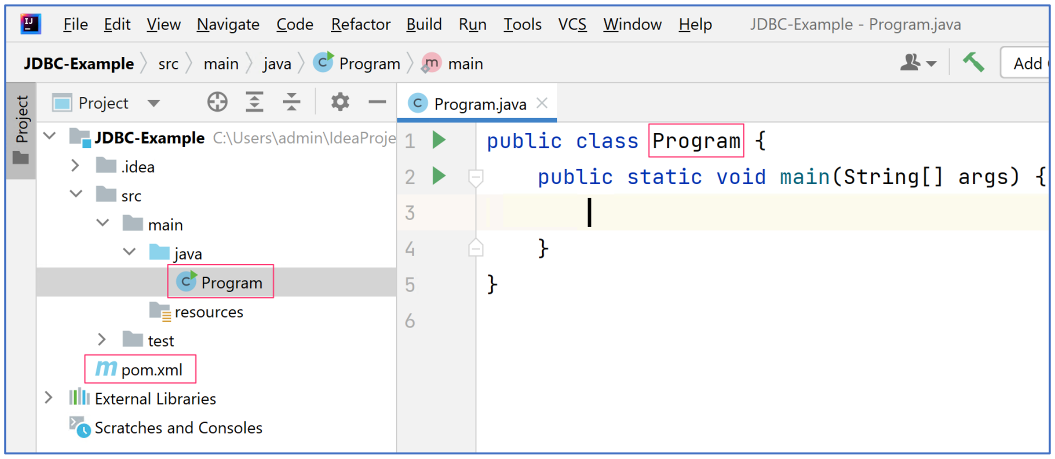
For this lab, I recommend using **Intellij Idea** which is one of the most powerful IDE for java development in the market at the moment to do the exercises. Intellij Idea offers two versions: the ultimate version (paid product) and the **community version** (free product). You can download the community version from this url: https://www.jetbrains.com/idea/download.

After downloading and successfully installing the application, we will create a maven project by choosing **Maven** from the build system choices as demonstrated in the following image. Also make sure that **JDK 1.8** is selected from the JDK dropdown menu.



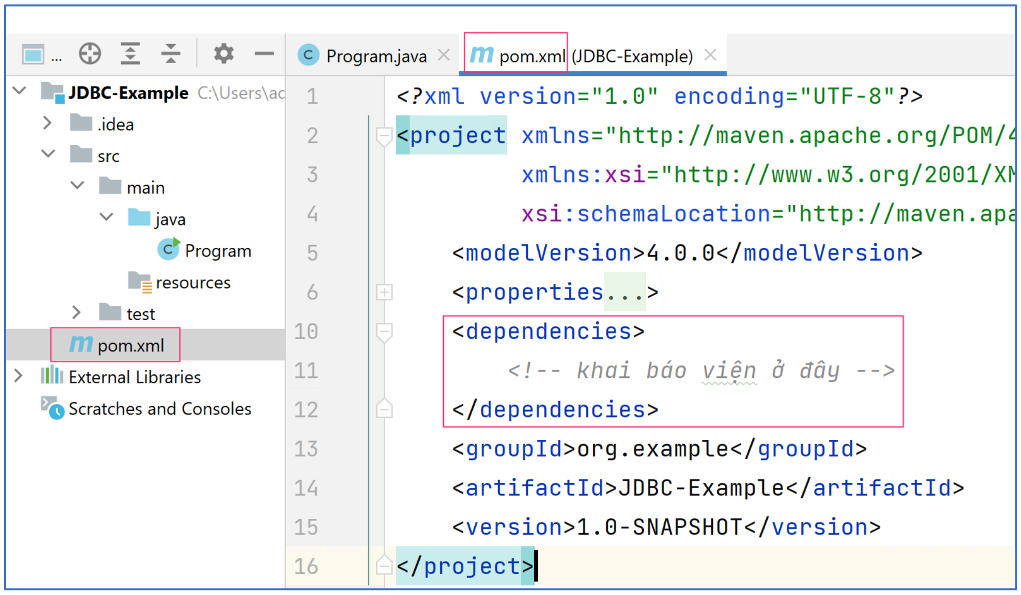
Name: JavaJDBC
Location: ~\IdeaProjects
Project will be created in: ~\IdeaProjects\JavaJDBC
Create Git repository
Language: Java Kotlin Groovy HTML
Build system: IntelliJ [✓ Maven] Gradle
JDK: 1.8 Oracle OpenJDK version 1.8.0_45
Add sample code

After creating the project, you should see a project structure like this:


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help JDBC-Example - Program.java
JDBC-Example src main java Program main
Add
Project
Program.java
JDBC-Example C:\Users\admin\IdeaProje
.idea
src
main
java
Program
resources
test
pom.xml
External Libraries
Scratches and Consoles

```java
public class Program {
    public static void main(String[] args) {

    }
}
```

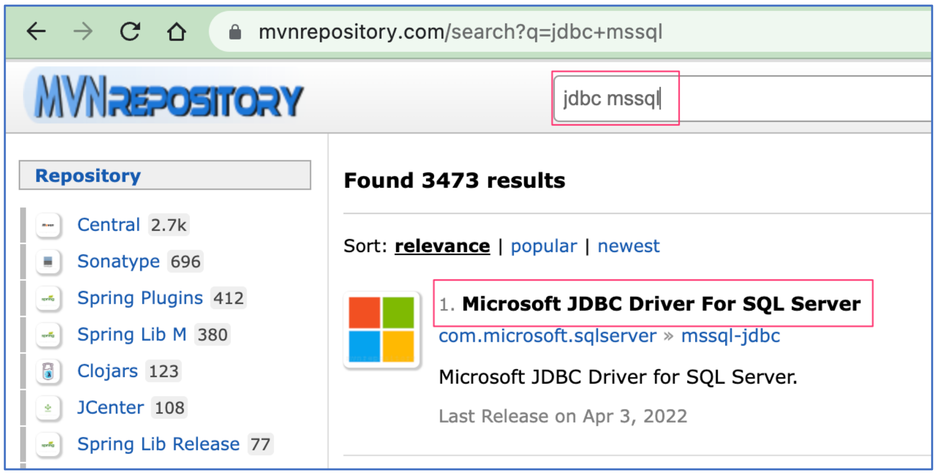
The src/main/java folder is the folder where we put all the java source files. The pom.xml file is the maven central configuration file where all dependencies (libraries) should be declared using the <dependency> tag.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4
         xmlns:xsi="http://www.w3.org/2001/XM
         xsi:schemaLocation="http://maven.apa
    <modelVersion>4.0.0</modelVersion>
    <properties...>
    <dependencies>
        <!-- khai báo viện ở đây -->
    </dependencies>
    <groupId>org.example</groupId>
    <artifactId>JDBC-Example</artifactId>
    <version>1.0-SNAPSHOT</version>
</project>
```

2.2 Importing JDBC dependency (library)

For our application to be able to work with JDBC, we first need to include JDBC library files in to our project. Maven will help us to simplify the process of importing a library and managing a library version. All we have to do is to visit the maven repository website, search for the 'jdbc mssql' keyword and select the appropriate library compatible to your database server.


mvnrepository.com/search?q=jdbc+mssql

jdbc mssql

Repository

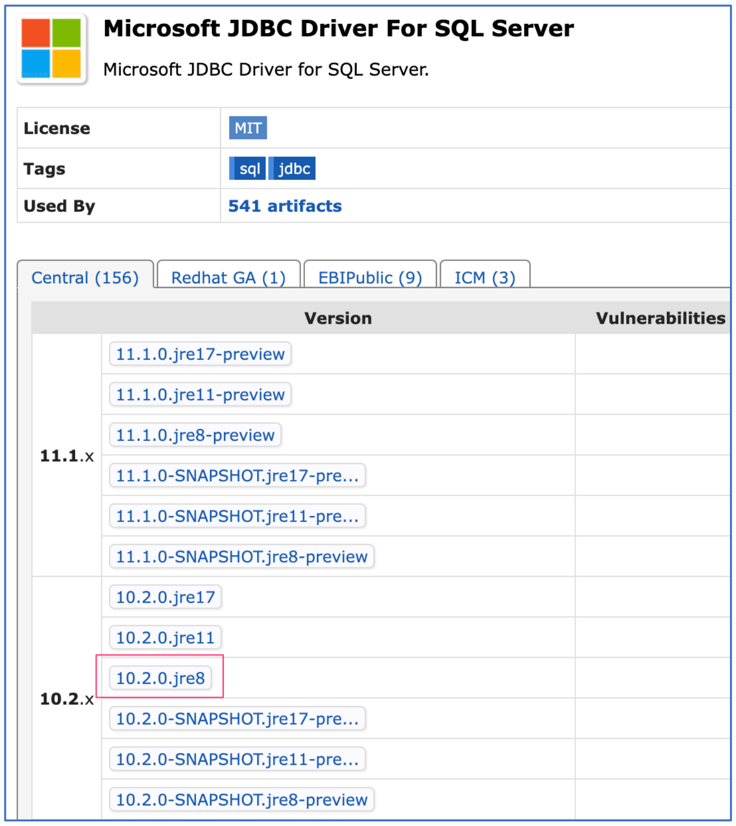
Central	2.7k
Sonatype	696
Spring Plugins	412
Spring Lib M	380
Clojars	123
JCenter	108
Spring Lib Release	77

Found 3473 results

Sort: relevance | popular | newest

1. Microsoft JDBC Driver For SQL Server
com.microsoft.sqlserver » mssql-jdbc
Microsoft JDBC Driver for SQL Server.
Last Release on Apr 3, 2022

After that select the appropriate version which compatible to your database server and the installed jdk version on your machine.


Microsoft JDBC Driver For SQL Server

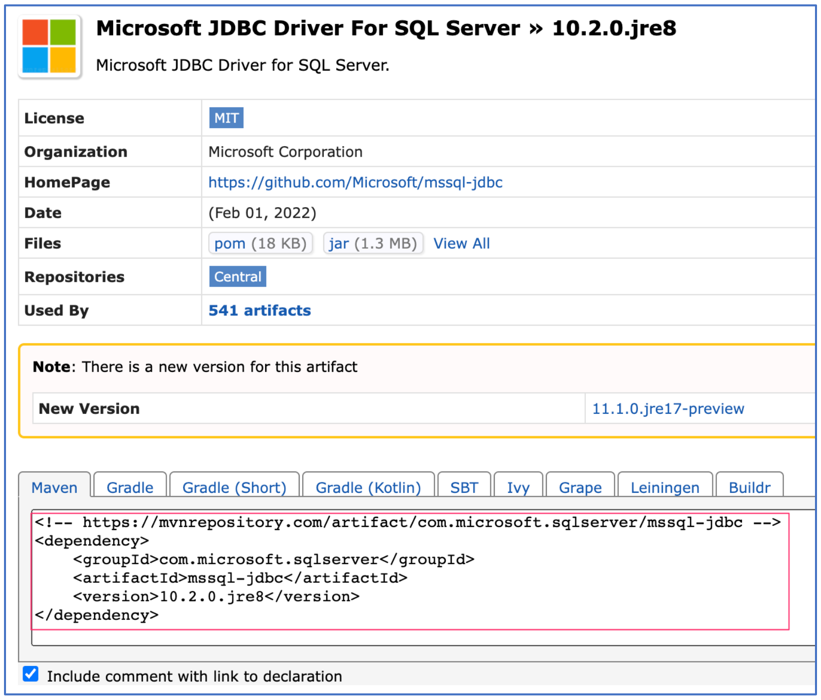
Microsoft JDBC Driver for SQL Server.

License	MIT
Tags	sql jdbc
Used By	541 artifacts

Central (156) | Redhat GA (1) | EBIPublic (9) | ICM (3)

	Version	Vulnerabilities
11.1.x	11.1.0.jre17-preview	
	11.1.0.jre11-preview	
	11.1.0.jre8-preview	
	11.1.0-SNAPSHOT.jre17-pre...	
	11.1.0-SNAPSHOT.jre11-pre...	
	11.1.0-SNAPSHOT.jre8-preview	
10.2.x	10.2.0.jre17	
	10.2.0.jre11	
	10.2.0.jre8	
	10.2.0-SNAPSHOT.jre17-pre...	
	10.2.0-SNAPSHOT.jre11-pre...	
	10.2.0-SNAPSHOT.jre8-preview	

In the next step, copy the <dependency> xml tag from the Maven section at the first tab.



Microsoft JDBC Driver For SQL Server » 10.2.0.jre8

Microsoft JDBC Driver for SQL Server.

License	MIT
Organization	Microsoft Corporation
HomePage	https://github.com/Microsoft/mssql-jdbc
Date	(Feb 01, 2022)
Files	pom (18 KB) jar (1.3 MB) View All
Repositories	Central
Used By	541 artifacts

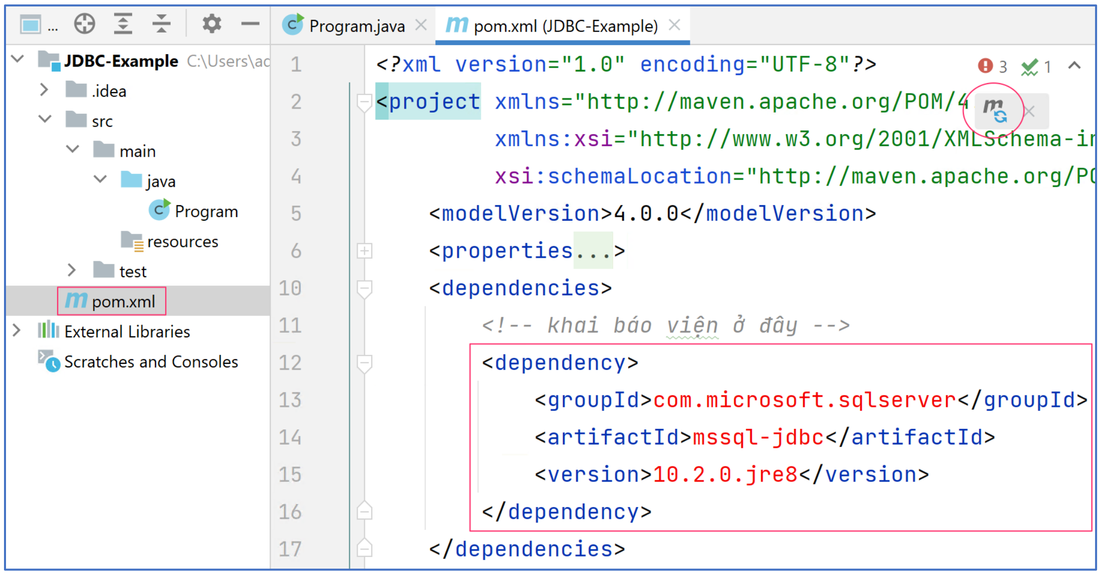
Note: There is a new version for this artifact

New Version	11.1.0.jre17-preview

```xml
<!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->
<dependency>
    <groupId>com.microsoft.sqlserver</groupId>
    <artifactId>mssql-jdbc</artifactId>
    <version>10.2.0.jre8</version>
</dependency>
```

☑ Include comment with link to declaration

and then paste it inside the <dependencies> section inside the pom.xml file at your project then click the sync popup button to download the library.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
         xsi:schemaLocation="http://maven.apache.org/PO
    <modelVersion>4.0.0</modelVersion>
    <properties...>
    <dependencies>
        <!-- khai báo viện ở đây -->
        <dependency>
            <groupId>com.microsoft.sqlserver</groupId>
            <artifactId>mssql-jdbc</artifactId>
            <version>10.2.0.jre8</version>
        </dependency>
    </dependencies>
```

2.3. Connect to the database server

Before doing anything to the server, we must establish an sql connection using the DriverManager.getConnection() method. This method receives a string parameter which is called the connection string. The connection string contains information such as server name (or ip address), username and password for logging in to the server, the default database and other useful information. Here is a sample connection string for connecting to the local database server on a tdtu student pc:

jdbc:sqlserver://507-20;user=admin;password=111111;databaseName=Lab10;encrypt=true;trustServerCertificate=true;

Make sure to change the text in red color according to the database server you want to connect to. To create a sql connection, you should write the code similar to this:

```
try {
    String dbURL = "jdbc:sqlserver://507-GV;user=admin;password=111111;databaseName=Truonghoc;
    Connection conn = DriverManager.getConnection(dbURL);

    System.out.println("Connected to server");
}catch (SQLException e){
    System.out.println("Can not connect: " + e.getMessage());
}
```

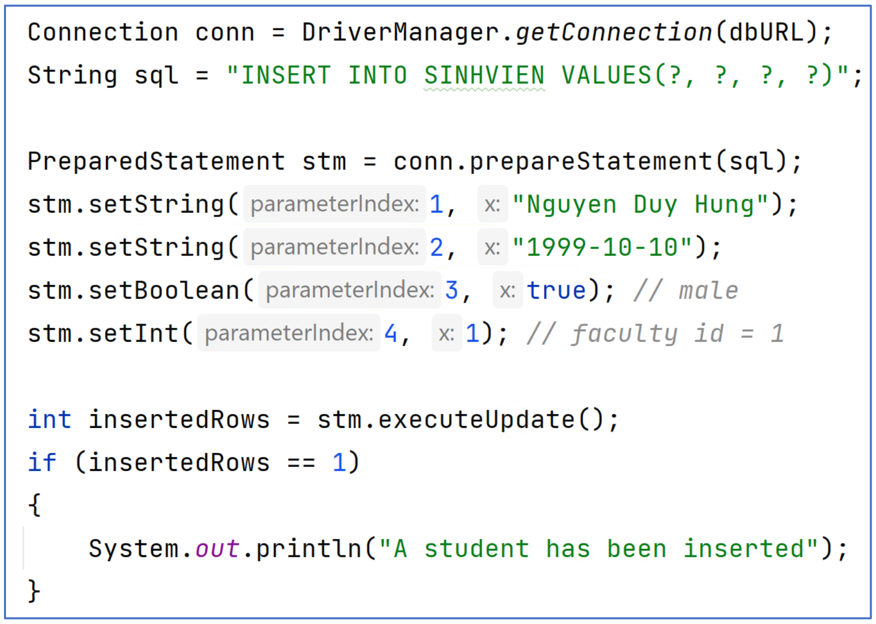
2.4. CRUD Examples

2.4.1 Query that does not return any value

Queries that do not return any value are query such as create a table, drop a table, insert a record, update a record, delete a record... Mostly, they are queries that do not contain the select clause.

```
try {
    Connection conn = DriverManager.getConnection(dbURL);
    String sql = "CREATE TABLE LOG(ID INT PRIMARY KEY, " +
            "DESC NVARCHAR(255) NOT NULL )";

    Statement stm = conn.createStatement();
    if (stm.execute(sql))
    {
        System.out.println("Table has been created");
    }
}catch (SQLException e){
    // lỗi cú pháp, hoặc lỗi permission, hoặc lỗi bảng/db không tồn tại
    System.out.println("Eror: " + e.getMessage());
}
```

Example of executing a sql query to create a table

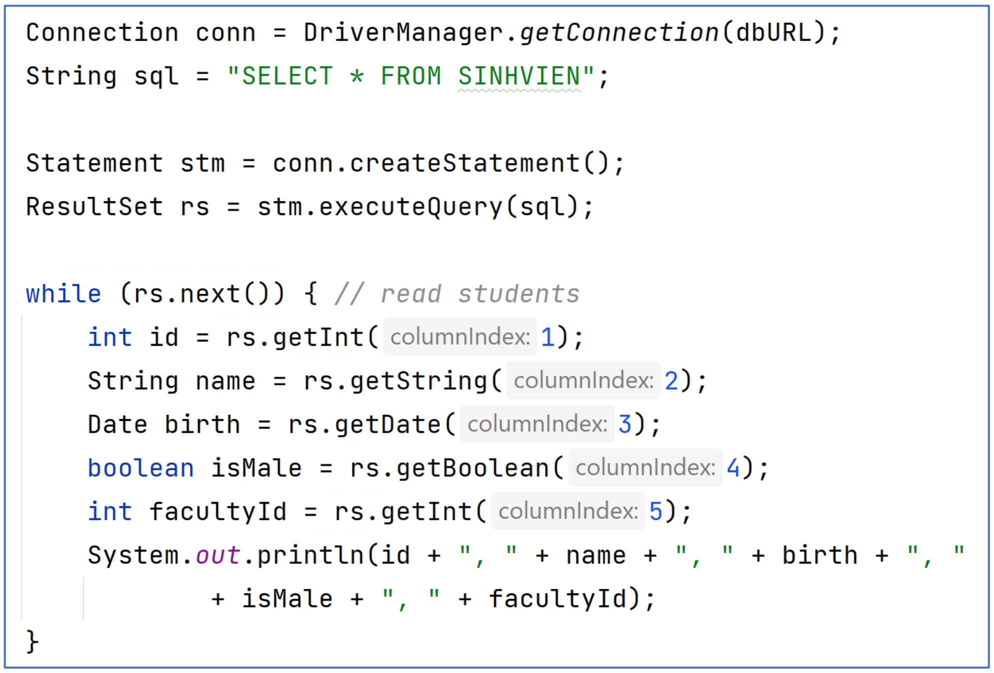


```
Connection conn = DriverManager.getConnection(dbURL);
String sql = "INSERT INTO SINHVIEN VALUES(?, ?, ?, ?)";

PreparedStatement stm = conn.prepareStatement(sql);
stm.setString( parameterIndex: 1,  x: "Nguyen Duy Hung");
stm.setString( parameterIndex: 2,  x: "1999-10-10");
stm.setBoolean( parameterIndex: 3,  x: true); // male
stm.setInt( parameterIndex: 4,  x: 1); // faculty id = 1

int insertedRows = stm.executeUpdate();
if (insertedRows == 1)
{
    System.out.println("A student has been inserted");
}
```

Example of executing a sql query to insert data to a table

2.4.2 Query that returns value

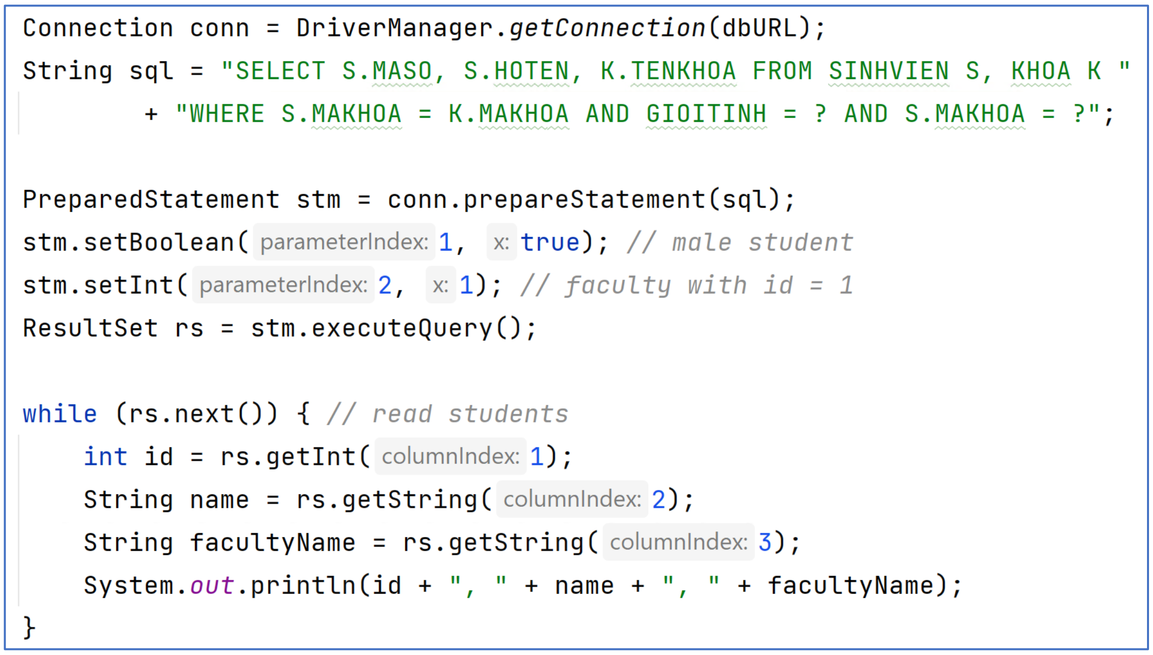
Queries that return value usually start with the select clause. Here is a simple query which read all student from a database table



```
Connection conn = DriverManager.getConnection(dbURL);
String sql = "SELECT * FROM SINHVIEN";

Statement stm = conn.createStatement();
ResultSet rs = stm.executeQuery(sql);

while (rs.next()) { // read students
    int id = rs.getInt( columnIndex: 1);
    String name = rs.getString( columnIndex: 2);
    Date birth = rs.getDate( columnIndex: 3);
    boolean isMale = rs.getBoolean( columnIndex: 4);
    int facultyId = rs.getInt( columnIndex: 5);
    System.out.println(id + ", " + name + ", " + birth + ", "
            + isMale + ", " + facultyId);
}
```

Example of executing a sql query to read all rows from a table

In real world project, you also want to read students based on some condition, for example only read student who are male and from the faculty with id = 1.



```java
Connection conn = DriverManager.getConnection(dbURL);
String sql = "SELECT S.MASO, S.HOTEN, K.TENKHOA FROM SINHVIEN S, KHOA K "
        + "WHERE S.MAKHOA = K.MAKHOA AND GIOITINH = ? AND S.MAKHOA = ?";

PreparedStatement stm = conn.prepareStatement(sql);
stm.setBoolean( parameterIndex: 1,  x: true); // male student
stm.setInt( parameterIndex: 2,  x: 1); // faculty with id = 1
ResultSet rs = stm.executeQuery();

while (rs.next()) { // read students
    int id = rs.getInt( columnIndex: 1);
    String name = rs.getString( columnIndex: 2);
    String facultyName = rs.getString( columnIndex: 3);
    System.out.println(id + ", " + name + ", " + facultyName);
}
```

Example of executing a sql query to read data from table with parameters

III. EXERCISE

- **On the server side**: create a file called **lab10.sql** which contains sql commands to create a database called **lab10**, then create the **product** table which has the following columns: <u>**id**</u>, name, price, color and description.
- **On the client side**: Create a **Product** class which coresponds to the product table in the database. Write a program to do the following tasks:
  - Connect to the database server
  - Insert at least 10 products
  - Delele 2 products
  - Read all products and print them to the console
  - Update 2 products
  - Read all products again and print them to the console.
- For this exercise, you should submit both the **sql file** and the java **source files**.