ITDBADM

CLASS SCHEDULE + MP POINTERS

Class Schedule (Tentative)

Week	Day	Date	Activity
Week 8	Tuesday	6/24/2025	• NO CLASS: MANILA DAY
	Friday	6/27/2025	• Intro to Triggers. MP Project. Java Connection.
Week 9	Tuesday	7/01/2025	• NO CLASS: Independent Learning Week
	Friday	7/04/2025	• NO CLASS: Independent Learning Week
Week 10	Tuesday	7/08/2025	• Triggers, Admin Privileges
	Friday	**7/11/2025**	• Transaction Management, DB Security • **Submit Proposal/Updates of your Final Project**
Week 11	Tuesday	7/15/2025	• Transaction Management, Quiz Reviewer
	Friday	**7/18/2025**	• **Quiz 2**: Triggers, Admin Privileges, Transaction Mgmt.
Week 12	Tuesday	7/22/2025	• DB Security (Back-up & Recovery)
	Friday	**7/25/2025**	• **Submission of Presentation and DB Script (ALL GROUPS)** • **Demo**: Presentation (4 Groups)
Week 13	Tuesday	7/29/2025	• **Demo**: Presentation (3 Groups)
	Friday	8/01/2025	• **Demo**: Presentation (3 Groups)
Week 14	Tuesday	8/05/2025	• Finals Exams Week (No Exam for our Class)
	Friday	8/08/2025	• Grade Submission and Consultation

Final Project Requirements

- No final exam. Instead, we will have a final project.
- Create an Online Store of your choosing.
  - Minimum Requirements
    - At least 6 tables
      - Users, Products, Orders, Order Items, Currencies, Payment/Transaction Logs)
    - Minimum of 10 stored procedures and triggers
    - Utilize the Virtual Machines to create users and grant privileges to access the database.
  - JULY 10 (Thurs): Submit a proposal on your proposed Online Store
  - JULY 24 (Thurs): Submission of the deliverables (Source Codes, PPT, ER Diagram, etc.)

Table descriptions

1. Users
•**Description**: Stores information about customers who create accounts and place orders.
•**Sample Fields**: user_id, name, email, password, created_at

2. Products/Services
•**Description**: Contains details about items available for sale.
•**Sample Fields**: product_id, name, description, price, stock_quantity, currency_id

3. Orders
•**Description**: Records each completed customer order, acting like a sales receipt.
•**Sample Fields**: order_id, user_id, order_date, total_amount, currency_id

4. Order_Items
•**Description**: Tracks the individual products included in each order.
•**Sample Fields**: order_item_id, order_id, product_id, quantity, price

5. Currencies
•**Description**: Stores supported currencies and their exchange rates on specific dates.
•**Sample Fields**: currency_id, currency_code, symbol, exchange_rate_to_usd

6. Transaction_Log
•**Description**: Logs payment activity for orders, useful for tracking and auditing.
•**Sample Fields**: transaction_id, order_id, payment_method, payment_status, amount, timestamp

Note: These table descriptions are just guidelines. You may modify or expand them depending on the features of your online store.

MP Requirements

- GUI FRONT-END (JAVA, PYTHON, PHP. Any is okay)
  - Login (based on roles) (At least 3 Roles: Admin, Staff, Customer)
  - Browse products/ Services (At least 6 products and/or services)
  - Currency handling (At least 3 currencies: PH Peso, US Dollar, Korean Won)
  - Admin panel (Add/edit products and services)
  - Add to cart and place an order
  - View order history
- DELIVERABLES
  1. SQL FILES: Schema, Stored Procedures, Triggers, etc.
     - Should include the 6 required tables
     - Should have at least 10 procedures and triggers
  2. GUI Source Codes (Python, Java, PHP, etc.)
  3. PPT PRESENTATION
     - ONLINE STORE INTRO, ERD DIAGRAM, DESCRIPTION OF PROCEDURES & TRIGGERS, SCREENSHOTS OF GUI, SQL CODE SNIPPETS, EXAMPLE OUTPUT
  - **FINAL PROJECT DEADLINE: JULY 24, 2025**

RUBRICS (100 points)

Criteria	Points	Description
1. Database Design	**25 pts**	Core tables (Users, Products, Orders, etc.) are correctly designed, normalized, and related using PK/FK.
2. Stored Procedures	**10 pts**	At least 5 working procedures (e.g., place order, cancel order, update stock).
3. Triggers	**10 pts**	At least 5 functional triggers for logging, validation, or updates.
4. Transactions + Logs	**10 pts**	Uses COMMIT/ROLLBACK properly; logs key actions (insert/update/delete) into a transaction log. Incorporate ACID properties.
5. Privileges	**10 pts**	Role-based access using GRANT/REVOKE (e.g., admin vs staff vs customer).
6. GUI + DB Integration	**15 pts**	GUI includes login, product browsing, ordering, and is connected to the MySQL database. + Design
7. Reporting & Demo	**15 pts**	Clear project report (ERD, code snippets, screenshots) + live demo or recorded video that explains functionality. (15 min demo, 5 min Q & A)
Bonus *(optional)*	**+5 pts**	Creativity and Extra features (e.g., product categories, search, analytics, user sessions, etc.). Or Additional data (Additional roles, stores, currencies, etc.)

Final Project Notes

- You may use Eclipse or any other IDE you prefer.
  What's important is that you are able to connect your MySQL Database to the GUI.
- If there are issues with your group or if you want to change/leave members, let me know by July 10.
- You will be asked to demo your system during the final presentation. Make sure your application runs smoothly and your database is connected.
- We will finalize the order/schedule of group presentations on July 10.

Proposal Guidelines (July 10)

- **Submit a 1-page proposal that includes the following:**
- **Online Store Name**
  Give your store a unique and creative name.
- **Product or Service Description**
  Briefly explain what your store sells (e.g., gadgets, clothing, digital services).
- **Target Customers**
  Who are your main buyers? (e.g., students, professionals, gamers)
- **Key Features**
  What features will your online store include?
- **Team Members**
  List group members with roles (e.g., frontend, backend, database admin).

Connecting MySQL to JAVA

1. **MySQL** Workbench is installed and running
2. Make sure you installed **Microsoft** OpenJDK (At least JDK17 or higher)
3. Install **Eclipse IDE** or any IDE to make the GUI
4. MySQL Connector/J JAR file

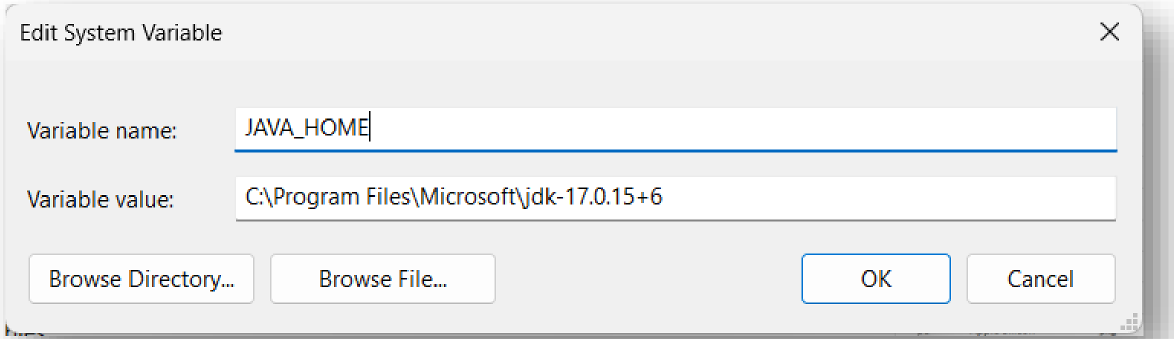
Note:

- Please try to connect MySQL to JAVA or your preferred IDE on your own.
- If you encounter issues, please get in touch with me.

Install JDK (17 or higher)

https://www.oracle.com/ph/java/technologies/downloads

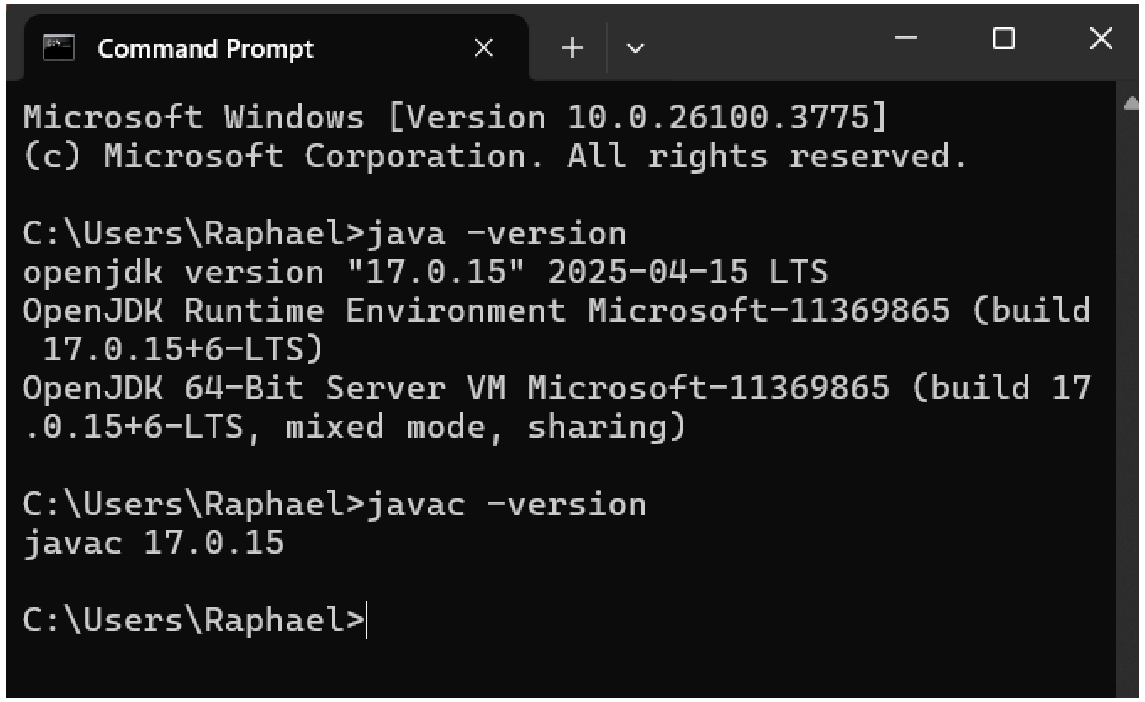
- Make sure Java is installed on your computer. (Check your terminal).
- Different steps for Windows and Mac.
- For windows, make sure Java is properly linked in the Environment Variables.



Variable name: JAVA_HOME

Variable value: C:\Program Files\Microsoft\jdk-17.0.15+6

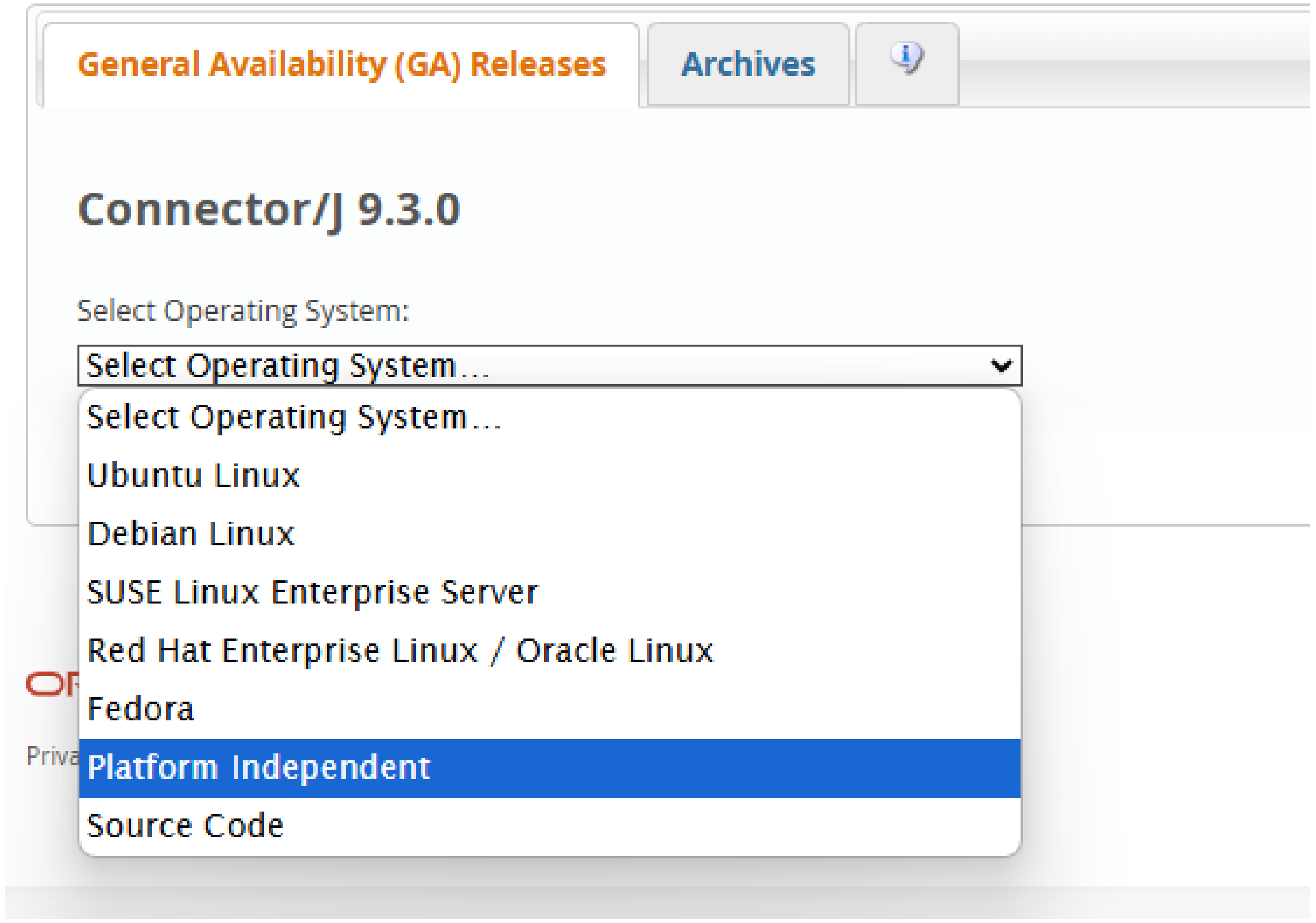
(For windows only)



```
Microsoft Windows [Version 10.0.26100.3775]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Raphael>java -version
openjdk version "17.0.15" 2025-04-15 LTS
OpenJDK Runtime Environment Microsoft-11369865 (build
 17.0.15+6-LTS)
OpenJDK 64-Bit Server VM Microsoft-11369865 (build 17
.0.15+6-LTS, mixed mode, sharing)

C:\Users\Raphael>javac -version
javac 17.0.15

C:\Users\Raphael>
```

(For windows only)

Install MySQL JDBC (Java Database Connectivity) Driver



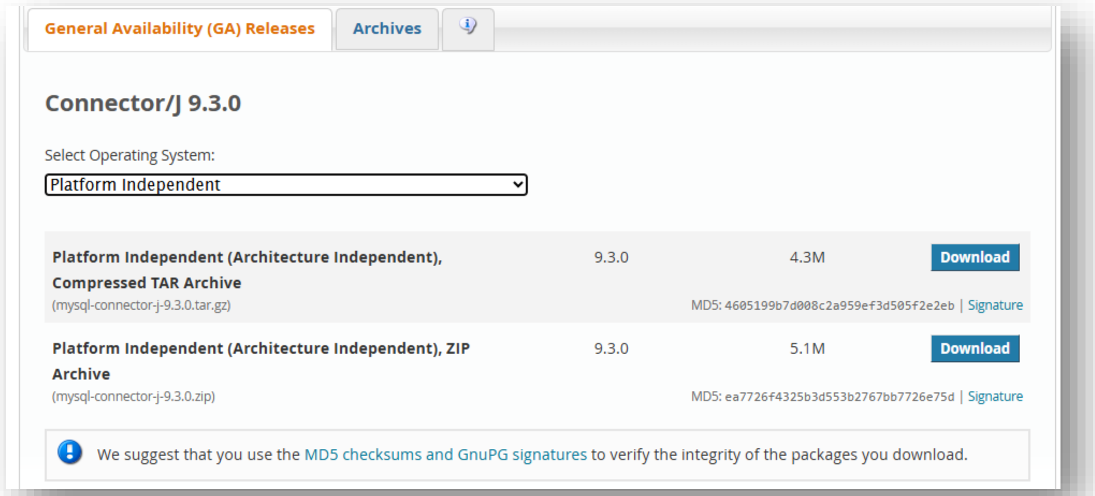
Connector/J 9.3.0

Select Operating System:

Select Operating System…

- Select Operating System…
- Ubuntu Linux
- Debian Linux
- SUSE Linux Enterprise Server
- Red Hat Enterprise Linux / Oracle Linux
- Fedora
- Platform Independent
- Source Code

- https://dev.mysql.com/downloads/connector/j/
- Save the file in a secure location to be used in Java



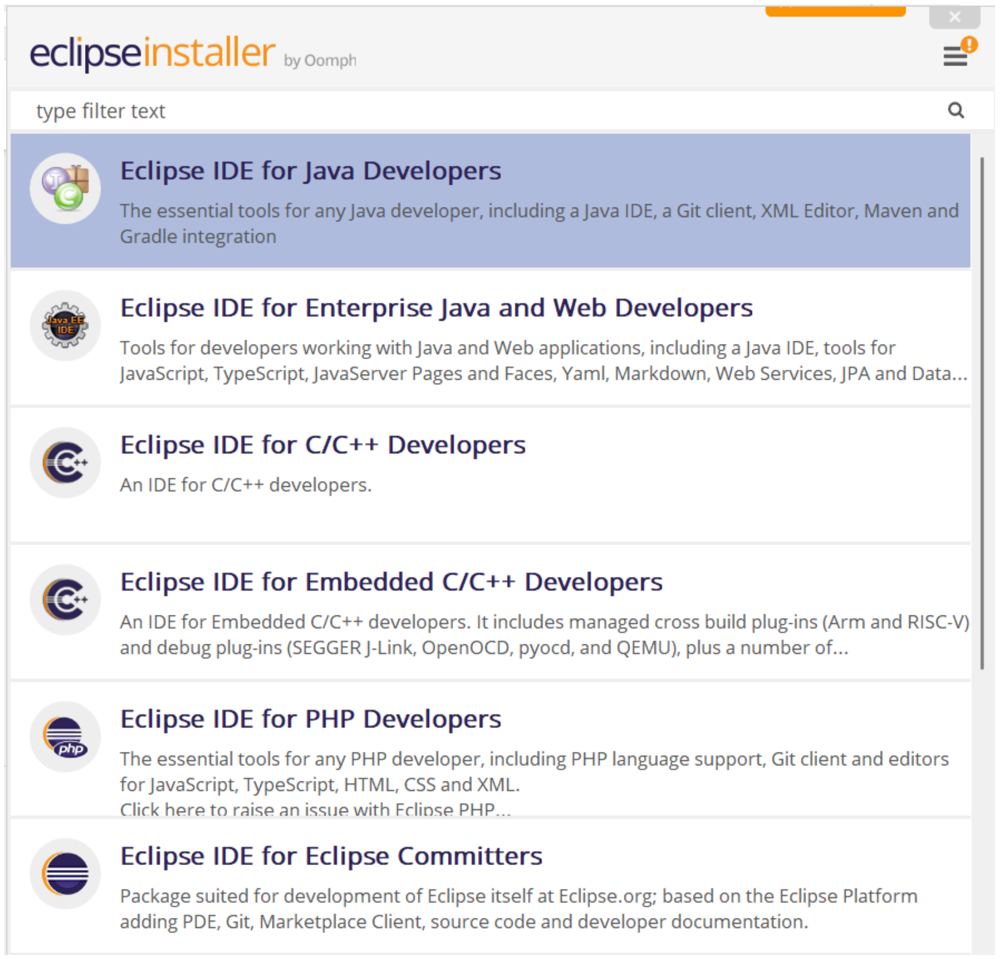
Connector/J 9.3.0

Select Operating System: Platform Independent

Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-j-9.3.0.tar.gz)	9.3.0	4.3M	Download MD5: 4605199b7d008c2a959ef3d505f2e2eb | Signature
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-j-9.3.0.zip)	9.3.0	5.1M	Download MD5: ea7726f4325b3d553b2767bb7726e75d | Signature

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

Install Eclipse IDE



eclipseinstaller by Oomph

type filter text

Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration

Eclipse IDE for Enterprise Java and Web Developers
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web Services, JPA and Data...

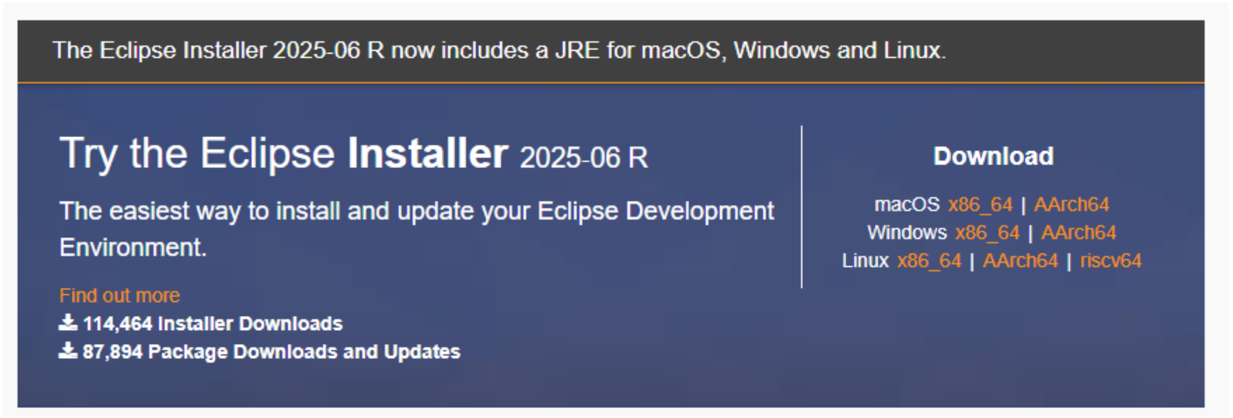
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers.

Eclipse IDE for Embedded C/C++ Developers
An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins (SEGGER J-Link, OpenOCD, pyocd, and QEMU), plus a number of...

Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client and editors for JavaScript, TypeScript, HTML, CSS and XML.
Click here to raise an issue with Eclipse PHP...

Eclipse IDE for Eclipse Committers
Package suited for development of Eclipse itself at Eclipse.org; based on the Eclipse Platform adding PDE, Git, Marketplace Client, source code and developer documentation.

- https://www.eclipse.org/downloads/packages/



The Eclipse Installer 2025-06 R now includes a JRE for macOS, Windows and Linux.

Try the Eclipse **Installer** 2025-06 R

The easiest way to install and update your Eclipse Development Environment.

Find out more

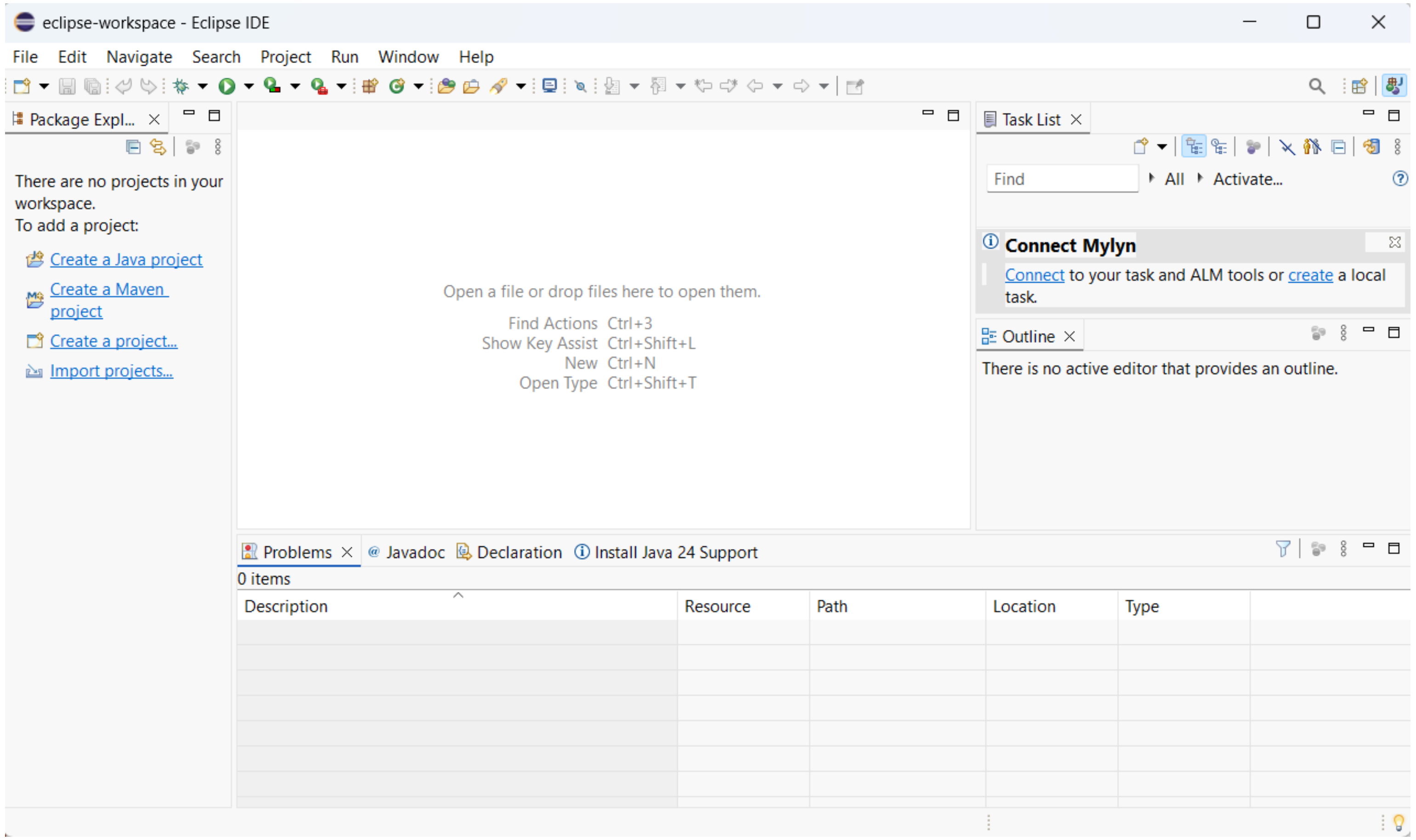
114,464 Installer Downloads

87,894 Package Downloads and Updates

Download

macOS x86_64 | AArch64
Windows x86_64 | AArch64
Linux x86_64 | AArch64 | riscv64

Eclipse Interface



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

Create a Java project
Create a Maven project
Create a project...
Import projects...

Open a file or drop files here to open them.

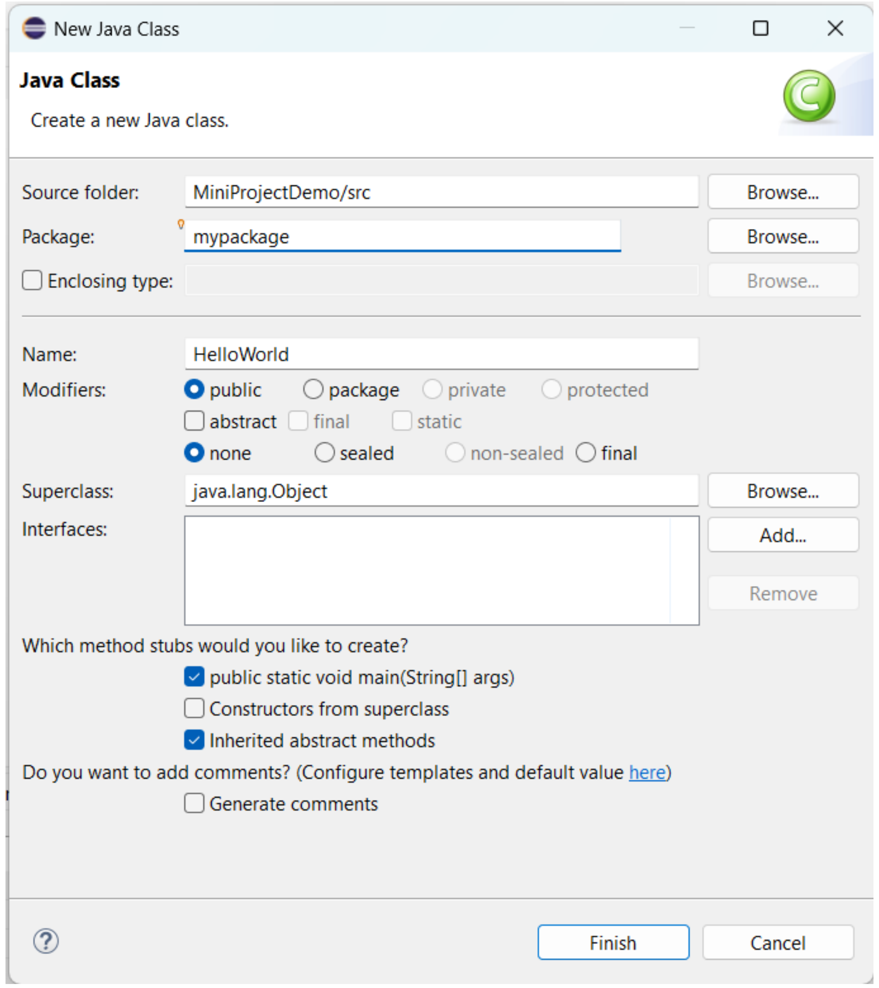
```
     Find Actions  Ctrl+3
  Show Key Assist  Ctrl+Shift+L
              New  Ctrl+N
        Open Type  Ctrl+Shift+T
```

Connect Mylyn
Connect to your task and ALM tools or create a local task.

There is no active editor that provides an outline.

0 items

Description	Resource	Path	Location	Type



New Java Class

Java Class
Create a new Java class.

Source folder: MiniProjectDemo/src
Package: mypackage
Enclosing type: [ ]

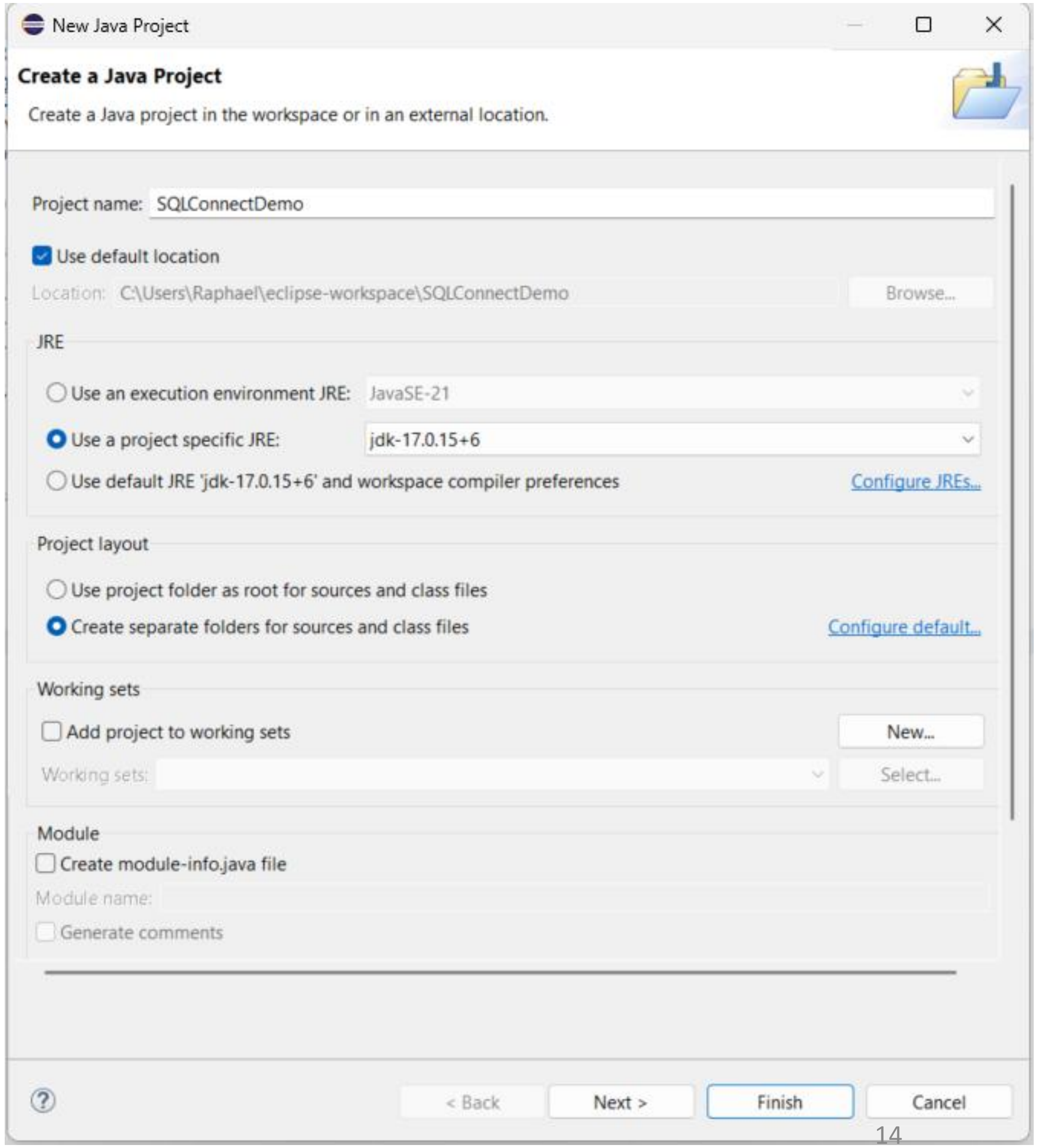
Name: HelloWorld
Modifiers: [✓ public] package private protected
[ ] abstract [ ] final [ ] static
[✓ none] sealed non-sealed final
Superclass: java.lang.Object
Interfaces:

Which method stubs would you like to create?
[✓] public static void main(String[] args)
[ ] Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
[ ] Generate comments

Eclipse IDE

- **Step 1: Create a New Java Project in Eclipse**

    1. Open Eclipse
    2. Go to File > New > Java Project
    3. Enter project name (e.g., **SQLConnectDemo**)
    4. IMPORTANT: Uncheck "Create module-info.java file"
    5. Click Finish


New Java Project

Create a Java Project

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

Project name: SQLConnectDemo

[✓] Use default location

Location: C:\Users\Raphael\eclipse-workspace\SQLConnectDemo

JRE
- ○ Use an execution environment JRE: JavaSE-21
- ● Use a project specific JRE: jdk-17.0.15+6
- ○ Use default JRE 'jdk-17.0.15+6' and workspace compiler preferences

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:

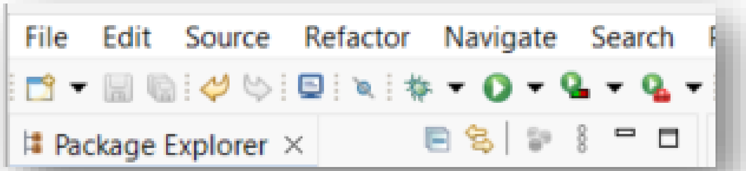
Module
- [ ] Create module-info.java file

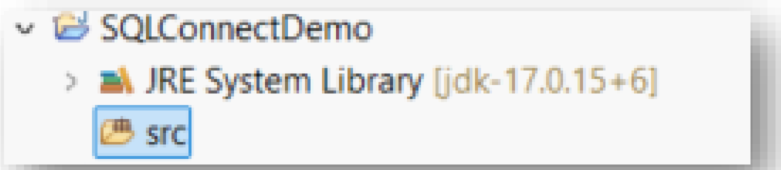
Module name:
- [ ] Generate comments

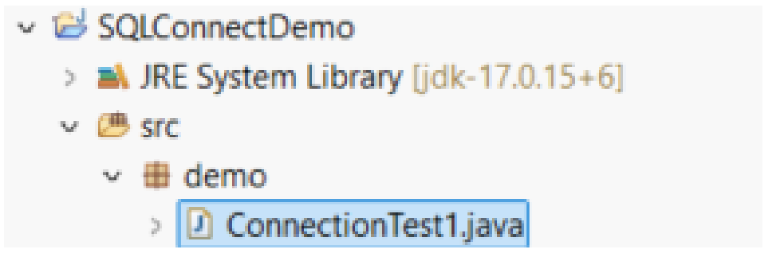
Eclipse IDE

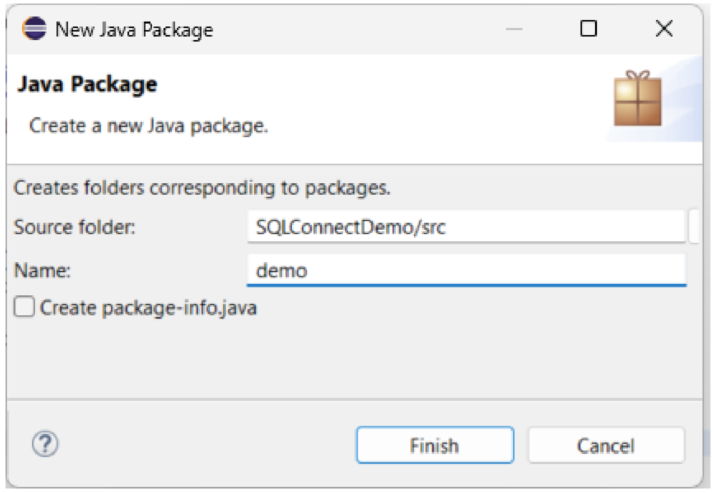
- **Step 2: Create Package and Java Class**

1. Go to Package Explorer in the Left Panel of Eclipse IDE
2. Expand the Java Project (e.g. SQLConnectDemo)
3. Right-click `src` > New > Package > Name it `*demo*` > Finish
4. Right-click the `demo` package > New > Class
   - Name it `*ConnectionTest1*`
   - **Check the box: `public static void main(String[] args)`**
   - Click Finish








New Java Package

Java Package

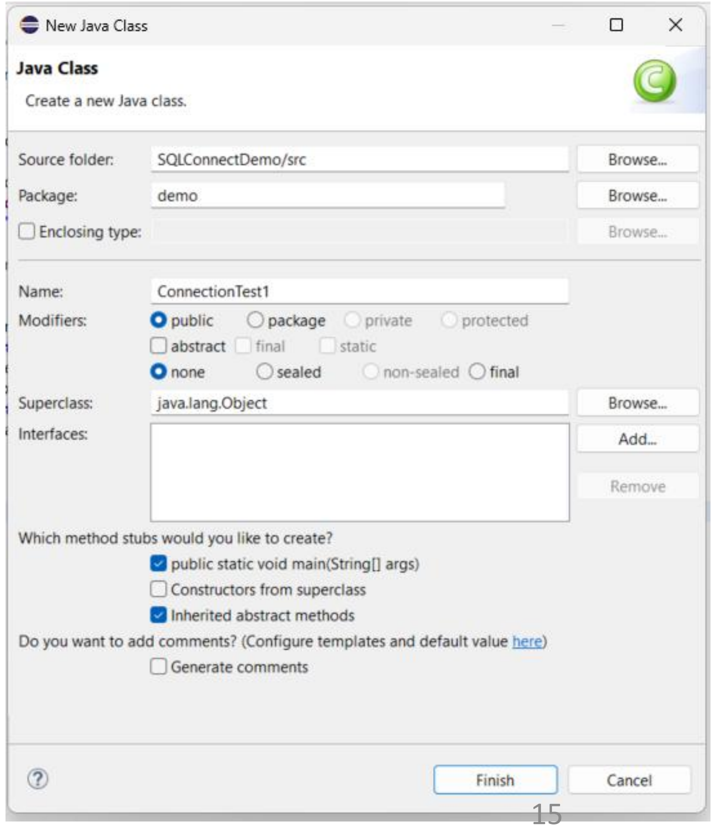
Create a new Java package.

Creates folders corresponding to packages.

Source folder: SQLConnectDemo/src

Name: demo

☐ Create package-info.java


New Java Class

Java Class

Create a new Java class.

Source folder: SQLConnectDemo/src

Package: demo

☐ Enclosing type:

Name: ConnectionTest1

Modifiers: [● public] ○ package ○ private ○ protected

☐ abstract ☐ final ☐ static

[● none] ○ sealed ○ non-sealed ○ final

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

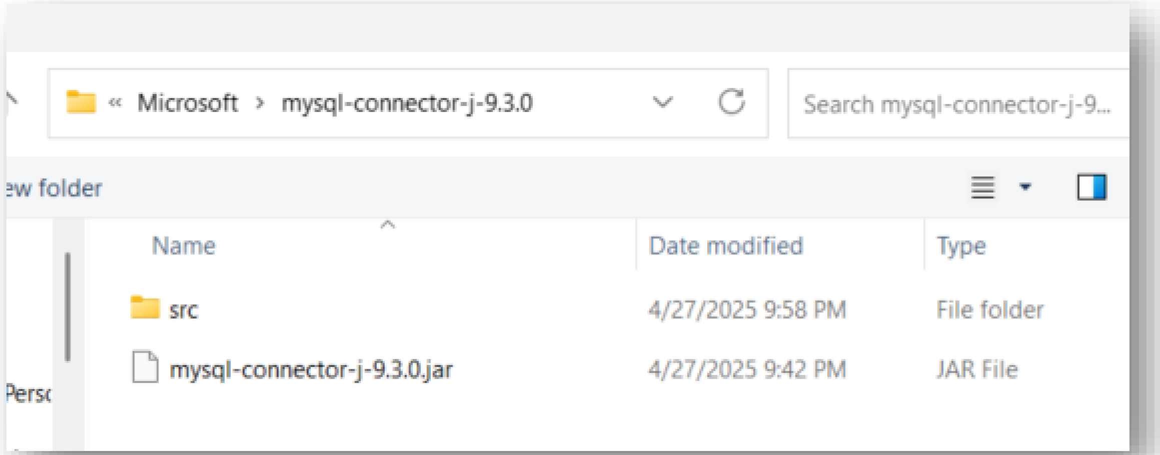
Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

Eclipse IDE

- **Step 3: Add the MySQL JAR to Eclipse**

1. Right-click your project (e.g. SQLConnectDemo) > Build Path > Configure Build Path
2. Go to the Libraries tab > Click Classpath
3. Click Add External JARs
4. Select the `mysql-connector-j-xxx.jar` file
5. Make sure it appears under the **Classpath**, NOT Modulepath
6. Click Apply and Close



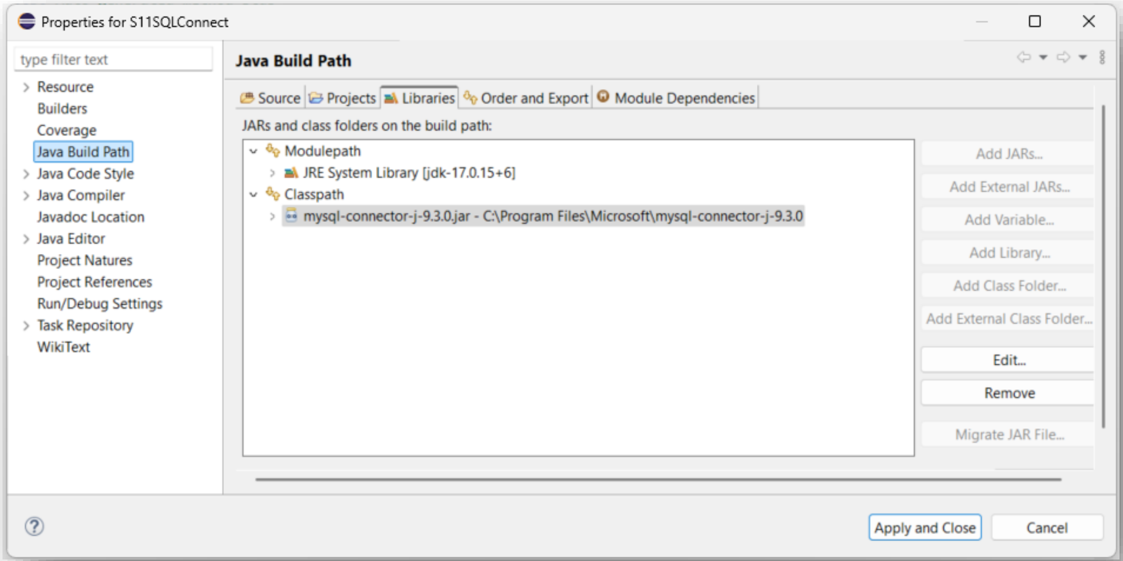
« Microsoft › mysql-connector-j-9.3.0

Search mysql-connector-j-9...

ew folder

Name	Date modified	Type
src	4/27/2025 9:58 PM	File folder
mysql-connector-j-9.3.0.jar	4/27/2025 9:42 PM	JAR File

Persc



Properties for S11SQLConnect

type filter text

- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
- Java Compiler
- Javadoc Location
- Java Editor
- Project Natures
- Project References
- Run/Debug Settings
- Task Repository
- WikiText

Java Build Path

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

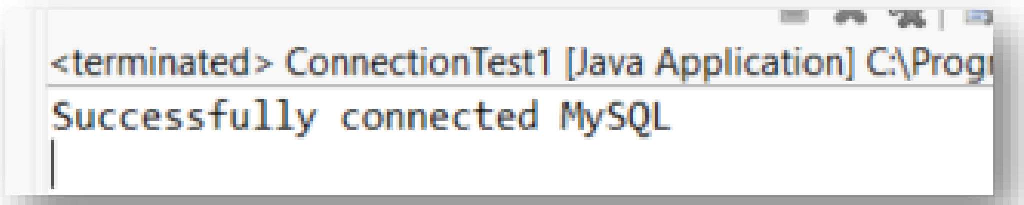
JARs and class folders on the build path:

```
Modulepath
  JRE System Library [jdk-17.0.15+6]
Classpath
  mysql-connector-j-9.3.0.jar - C:\Program Files\Microsoft\mysql-connector-j-9.3.0
```

Eclipse IDE

- **Step 3: Add the MySQL JAR to Eclipse**

1. Paste the following code to "ConnectionTest1.java"
2. Right-click `ConnectionTest.java`
3. Select Run As > Java Application



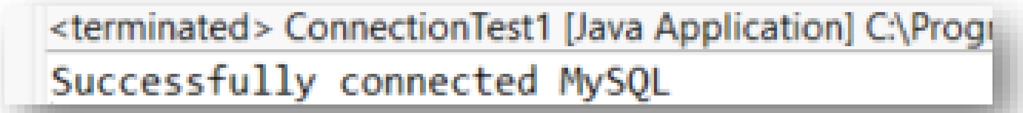
```
<terminated> ConnectionTest1 [Java Application] C:\Progr
Successfully connected MySQL
```

```
package demo;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnectionTest1 {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/sakila";
        String user = "root";
        String password = "Dlsu1234!";

        try {
            Connection conn = DriverManager.getConnection(url,
user, password);
            System.out.println("Successfully connected MySQL");
            conn.close();
        } catch (SQLException e) {
            System.out.println("Connection failed.");
            e.printStackTrace();
        }
    }
}
```

Local Connection (Your own computer) OR Virtual Machine

```java
package demo;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


public class ConnectionTest {
public static void main(String[] args) {
String url = "jdbc:mysql://localhost:3306/sakila";
String user = "root";
String password = "Dlsu1234!";


try {
Connection conn = DriverManager.getConnection(url,
user, password);
System.out.println("Successfully connected MySQL");
conn.close();
} catch (SQLException e) {
System.out.println("Connection failed.");
e.printStackTrace();
}
}
}
```

```java
package demo;


import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;


public class ConnectionTest1 {
public static void main(String[] args) {
// Use the localhost tunnel from Workbench (Virtual Machine)
String url = "jdbc:mysql://127.0.2.2:3306/sakila"; // <-- change this
String user = "root"; // from your MySQL connection (not SSH username)
String password = "Dlsu1234!"; // replace with the VM password


try {
Connection conn = DriverManager.getConnection(url, user, password);
System.out.println("Successfully connected MySQL");
conn.close();
} catch (SQLException e) {
System.out.println("Connection failed.");
e.printStackTrace();
}
}
}
```



```
<terminated> ConnectionTest1 [Java Application] C:\Progı
Successfully connected MySQL
```

Triggers

- A **trigger** is a set of SQL instructions that **automatically** runs **when a specific event happens** on a table (like INSERT, UPDATE, or DELETE).

```
-- Sample basic trigger syntax
CREATE TRIGGER trigger_name
BEFORE INSERT ON table_name
FOR EACH ROW
BEGIN
  -- actions to perform
END;
```

Timing	Description
BEFORE	Runs **before** the action happens
AFTER	Runs **after** the action happens

Event	Description
INSERT	When a new row is added
UPDATE	When a row is updated
DELETE	When a row is deleted

Triggers (pbb_collab DB): Eviction Log

```
-- 1. Create a new schema and table
CREATE DATABASE pbb_collab;
USE pbb_collab;

CREATE TABLE housemates (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(100),
   status ENUM('Active', 'Evicted') DEFAULT 'Active’
);

INSERT INTO housemates (name) VALUES
('Michael'), ('Emilio'), ('Josh’);

-- 2. Create a log table for the trigger
CREATE TABLE eviction_log (
   id INT AUTO_INCREMENT PRIMARY KEY,
   housemate_name VARCHAR(100)
);
```

```
-- 3. Create the trigger for evicting a housemate
DELIMITER $$
CREATE TRIGGER eviction_trigger
AFTER UPDATE ON housemates
FOR EACH ROW – To execute the trigger once for every row
BEGIN
   IF NEW.status = 'Evicted' AND OLD.status <> 'Evicted' THEN
      INSERT INTO eviction_log (housemate_name)
      VALUES (NEW.name);
   END IF;
END
$$ DELIMITER ;
```

```
-- 4. Remove safety updates when updating tables
SET SQL_SAFE_UPDATES = 0;

-- 5. When updating the housemates, a trigger will occur
UPDATE housemates SET status = 'Evicted' WHERE name = 'Emilio';
SELECT * FROM eviction_log;
```

Admin Privileges (Sakila DB)

'username'@'host/ip address' → 'admin'@'%'

- Grant privileges for Admin and Staff

```
-- Creating an ADMIN user AND granting full access to the whole database
CREATE USER 'admin'@'%' IDENTIFIED BY ‘Dlsu1234!’; -- Change the name and password
GRANT ALL PRIVILEGES ON *.* TO 'admin'@'%' WITH GRANT OPTION;
FLUSH PRIVILEGES;

-- Create STAFF user
CREATE USER 'staff'@'%' IDENTIFIED BY ‘Dlsu1234!’;
-- Grant partial access to STAFF (e.g. read-only access to film and actor tables in sakila)
GRANT SELECT ON sakila.film TO 'staff'@'%';
GRANT SELECT ON sakila.actor TO 'staff'@'%';
-- Grant INSERT/UPDATE on a specific table (e.g., rental)
GRANT SELECT, INSERT, UPDATE ON sakila.rental TO 'staff'@'%’;
FLUSH PRIVILEGES; -- To reload the user account info to make changes effective. Clears the cache.

SELECT user, host FROM mysql.user; -- See list of users created in MySQL
SHOW GRANTS FOR 'staff'@'%’; -- Show the privileges for each user (e.g. staff)
```

Transaction Management

A **transaction** is a set of SQL operations executed as a single unit.

- Managed using: START TRANSACTION, COMMIT, ROLLBACK
- In SQL transactions, we try to incorporate the ACID properties.

Property	Sample
Atomicity	Both inserts happen together or not at all (START + ROLLBACK)
Consistency	If one insert fails, database remains unchanged
Isolation	This transaction won’t interfere with others until committed
Durability	Once committed, changes are permanent, even if server crashes

Transaction Management Syntax

Command	Description
START TRANSACTION;	Begins a transaction block
COMMIT;	Saves all changes made in the transaction
ROLLBACK;	Cancels all changes made in the transaction

Online Store Shopping Cart: Proceed with Order (COMMIT)

```
START TRANSACTION;

UPDATE products
SET stock_quantity = stock_quantity - 1
WHERE product_id = 101;

COMMIT;
```

Online Store Shopping Cart: Cancel Order (ROLLBACK)

```
START TRANSACTION;

UPDATE products
SET stock_quantity = stock_quantity - 1
WHERE product_id = 101;

ROLLBACK;
```