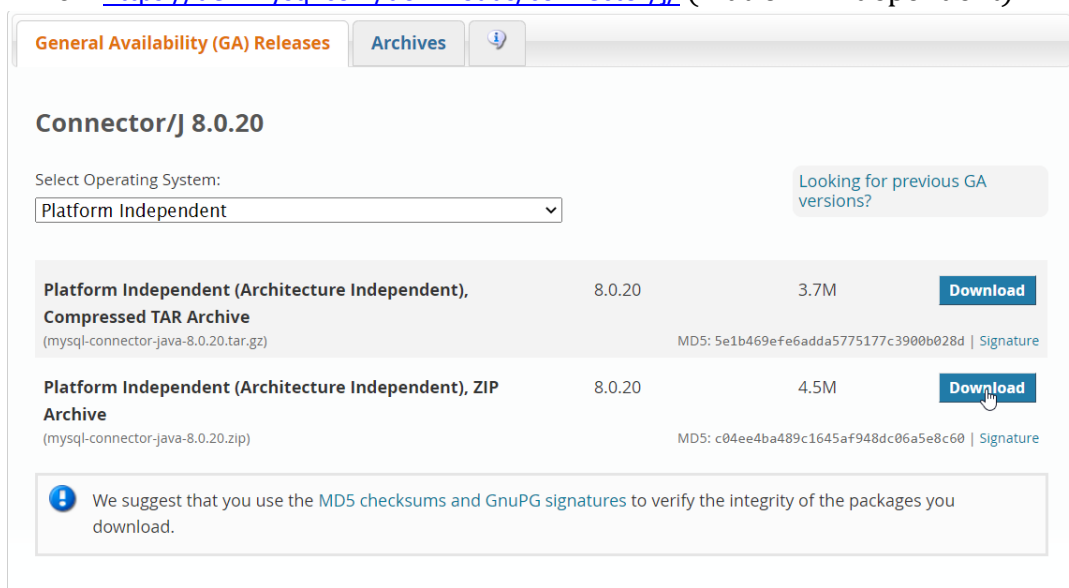


Abfragen einer MySQL Datenbank mit Java

Verwendet wird hierzu die Entwicklungsumgebung [Eclipse](#).

Bedingungen:

- Eine Installierte MySQL(!) Datenbank
 - <https://dev.mysql.com/downloads/>
- Der MySQL JDBC Treiber muss heruntergeladen werden (wird benötigt damit das Java Programm auf die Datenbank zugreifen kann)
 - <https://dev.mysql.com/downloads/connector/j/> (Platform Independent)



General Availability (GA) Releases Archives ⓘ

Connector/J 8.0.20

Select Operating System:
Platform Independent ▾

Looking for previous GA versions?

Platform Independent (Architecture Independent), Compressed TAR Archive <small>(mysql-connector-java-8.0.20.tar.gz)</small>	8.0.20	3.7M	Download
Platform Independent (Architecture Independent), ZIP Archive <small>(mysql-connector-java-8.0.20.zip)</small>	8.0.20	4.5M	Download

MD5: 5e1b469efe6adda5775177c3900b028d | [Signature](#)

MD5: c04ee4ba489c1645af948dc06a5e8c60 | [Signature](#)

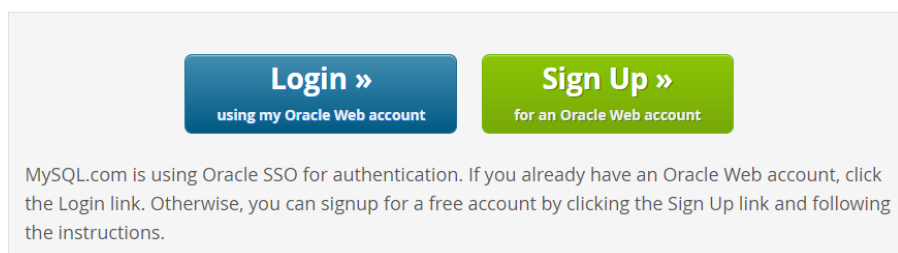
ⓘ We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system



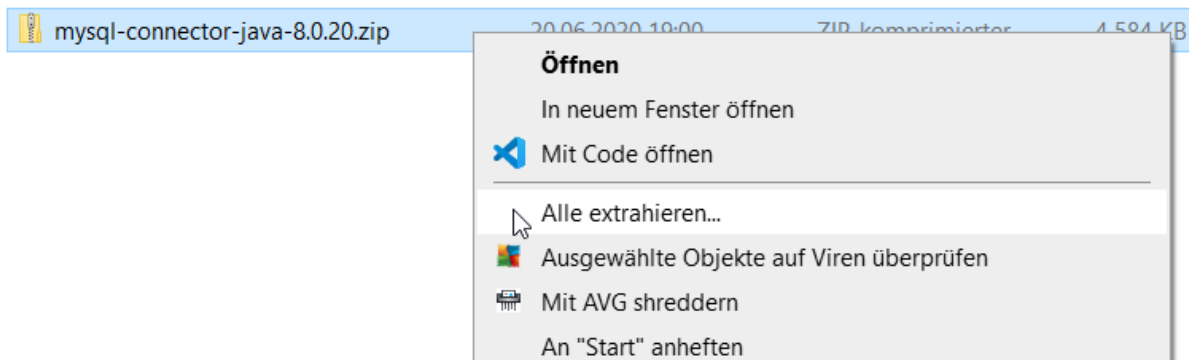
[Login »](#)
using my Oracle Web account

[Sign Up »](#)
for an Oracle Web account

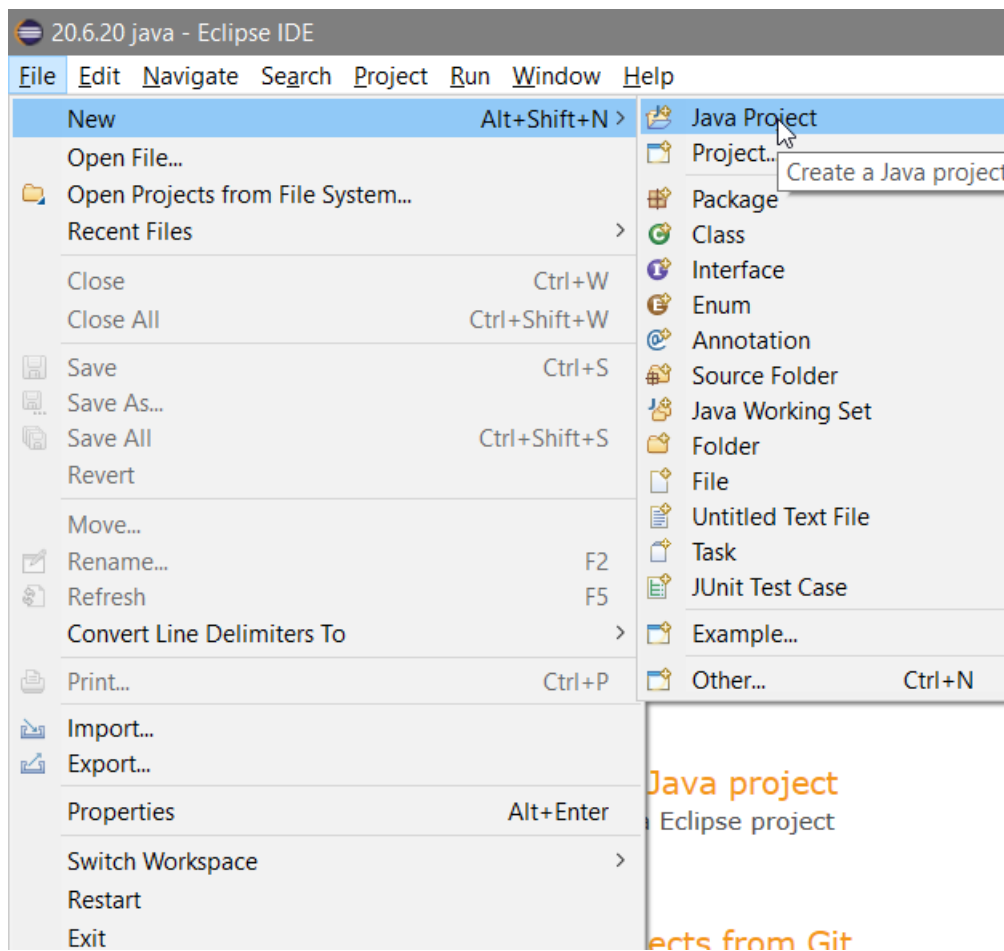
MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can sign up for a free account by clicking the Sign Up link and following the instructions.

[No thanks, just start my download.](#)

Danach im Ordner öffnen und extrahieren:

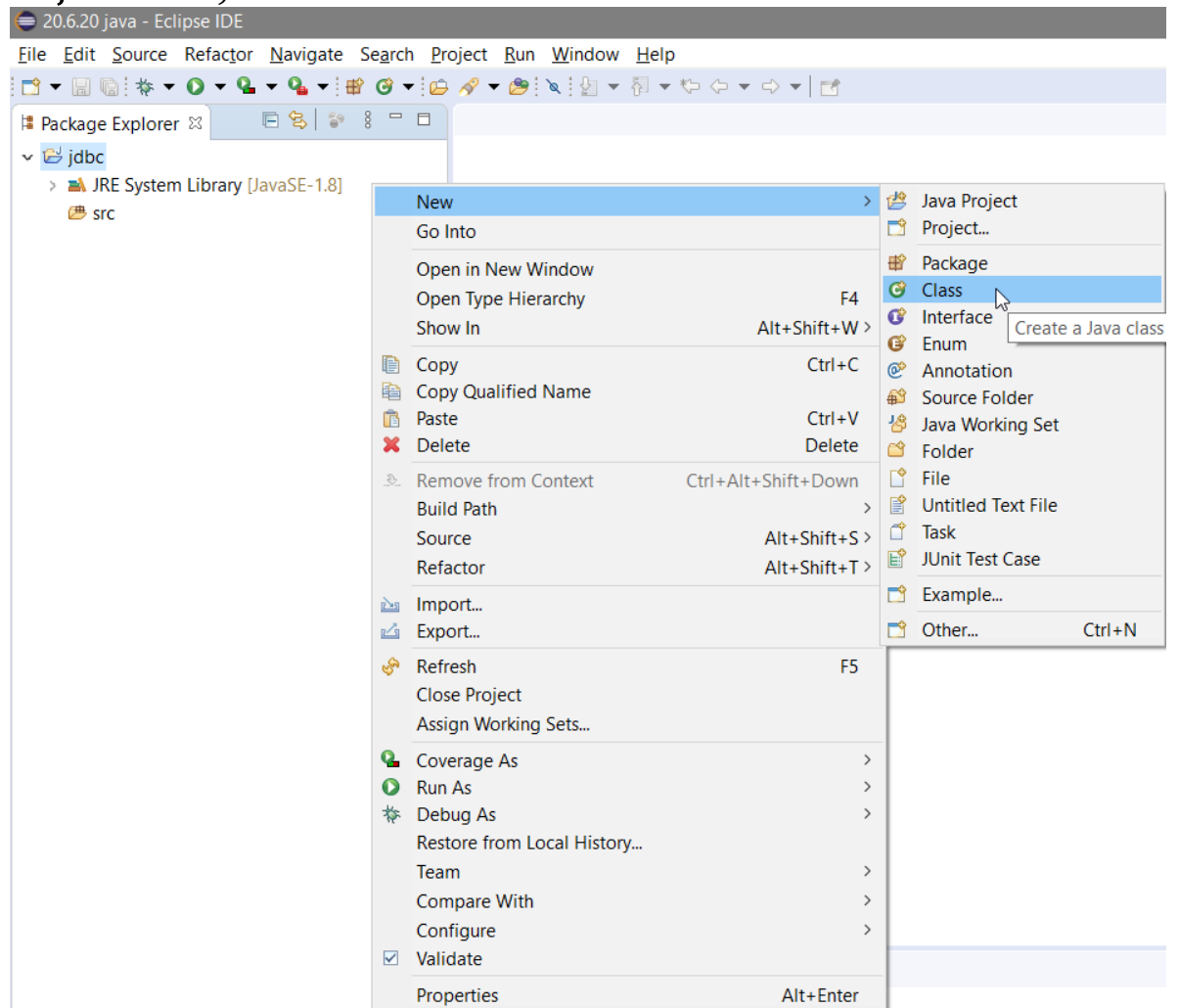


- Java Projekt in der Eclipse IDE erstellen:
(falls noch nicht installiert → <https://www.eclipse.org/downloads/>)

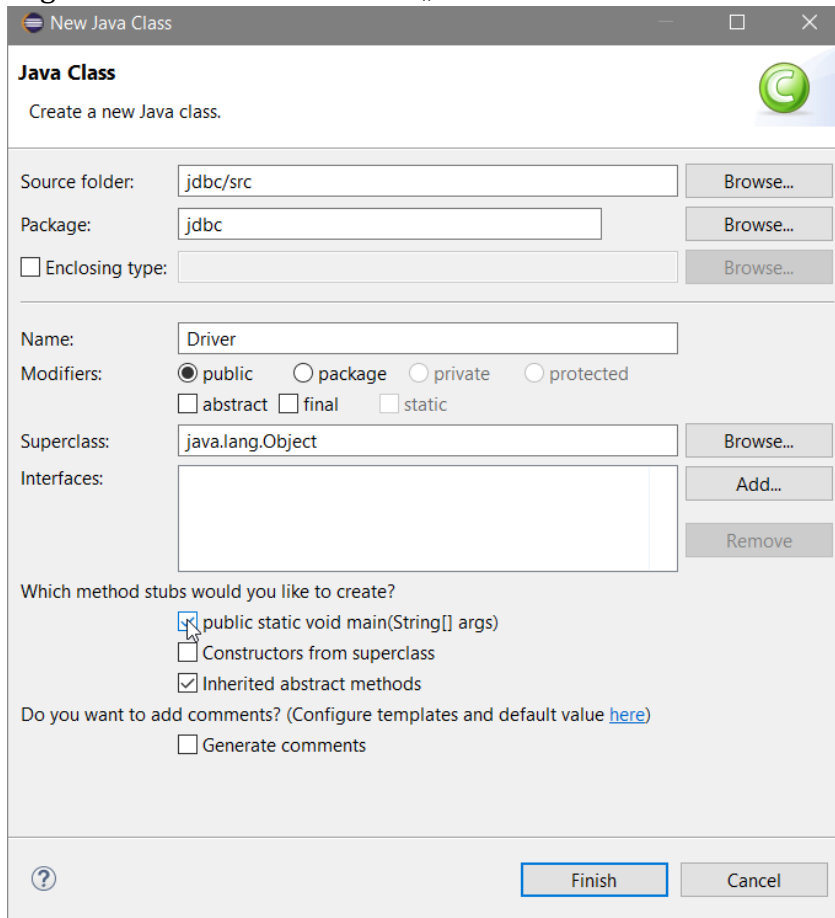


Im nächsten Fenster dann einen beliebigen Namen eingeben (in meinem Fall `jdbc`) und auf „Finish“ klicken.

Danach Rechtsklick auf den **src** Ordner (falls nicht zu sehen auf den kleinen Pfeil bei **jdbc** klicken) und eine Class erstellen.



Bei dem nachfolgenden Fenster wird ein Name eingetragen (in meinem Fall „Driver“) und für faule Menschen noch das „`public static void main(String[] args)`“ abgehackt. Danach wieder auf „Finish“.



New Java Class

Create a new Java class.

Source folder: jdbc/src Browse...

Package: jdbc Browse...

Enclosing type: Browse...

Name: Driver

Modifiers: public package private protected
 abstract final static

Superclass: java.lang.Object Browse...

Interfaces: Add...
Remove

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](#))

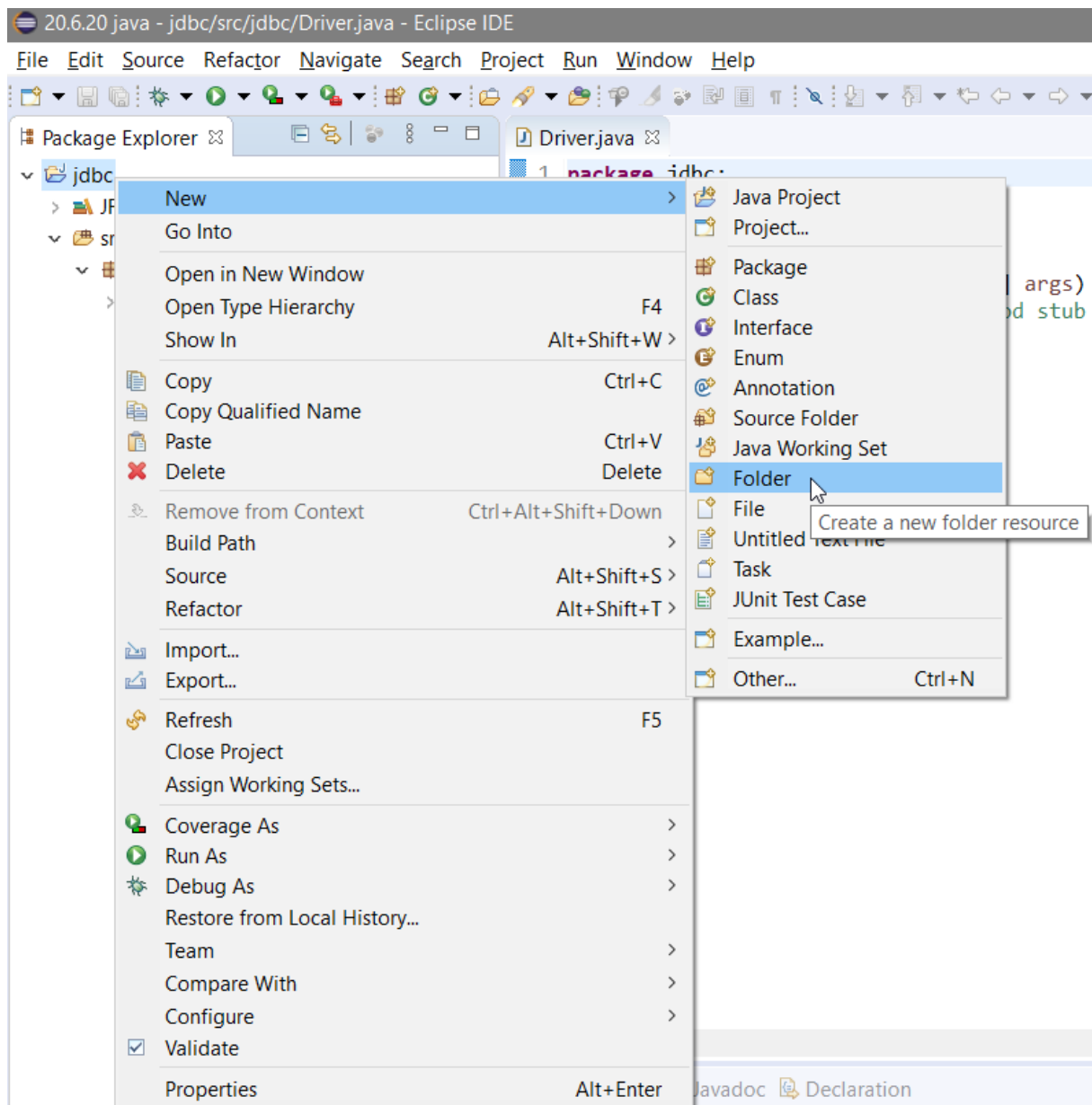
Generate comments

? Finish Cancel

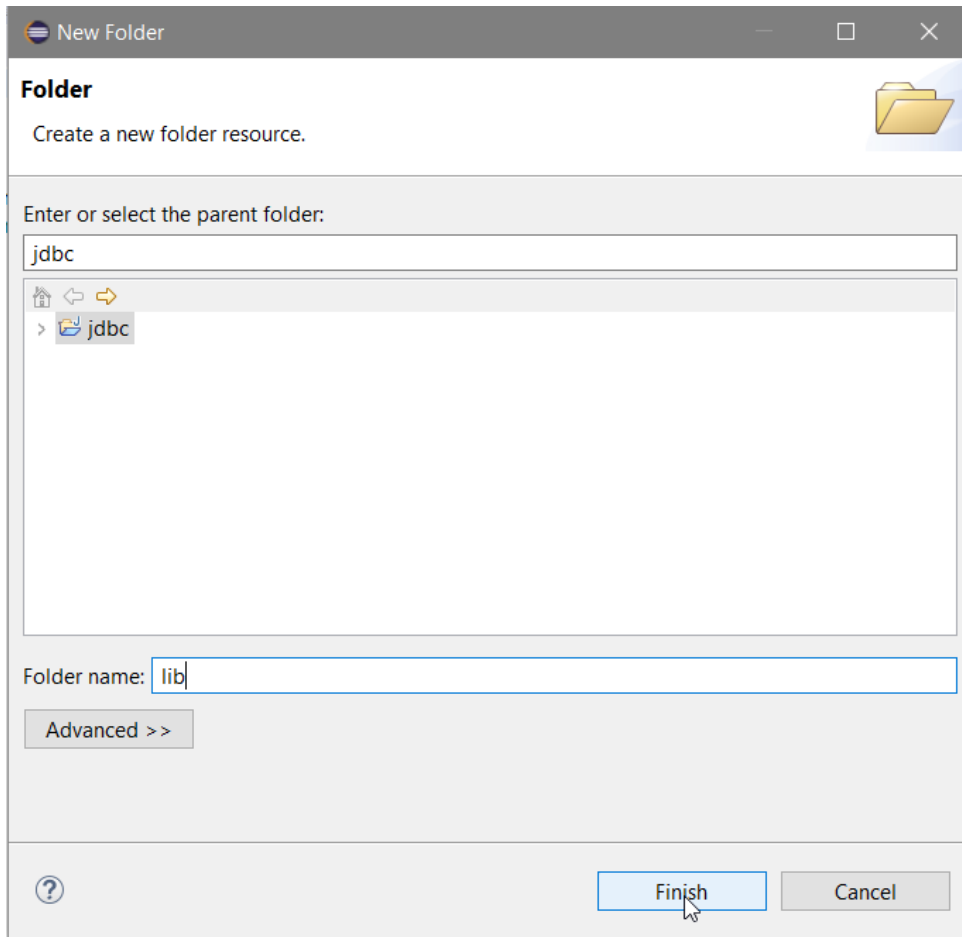
Entwicklungsablauf:

1. MySQL Datenbank Treiber zum „classpath“ hinzufügen
2. Verbindung herstellen
3. SQL-Abfrage erstellen
4. Ergebnis verarbeiten

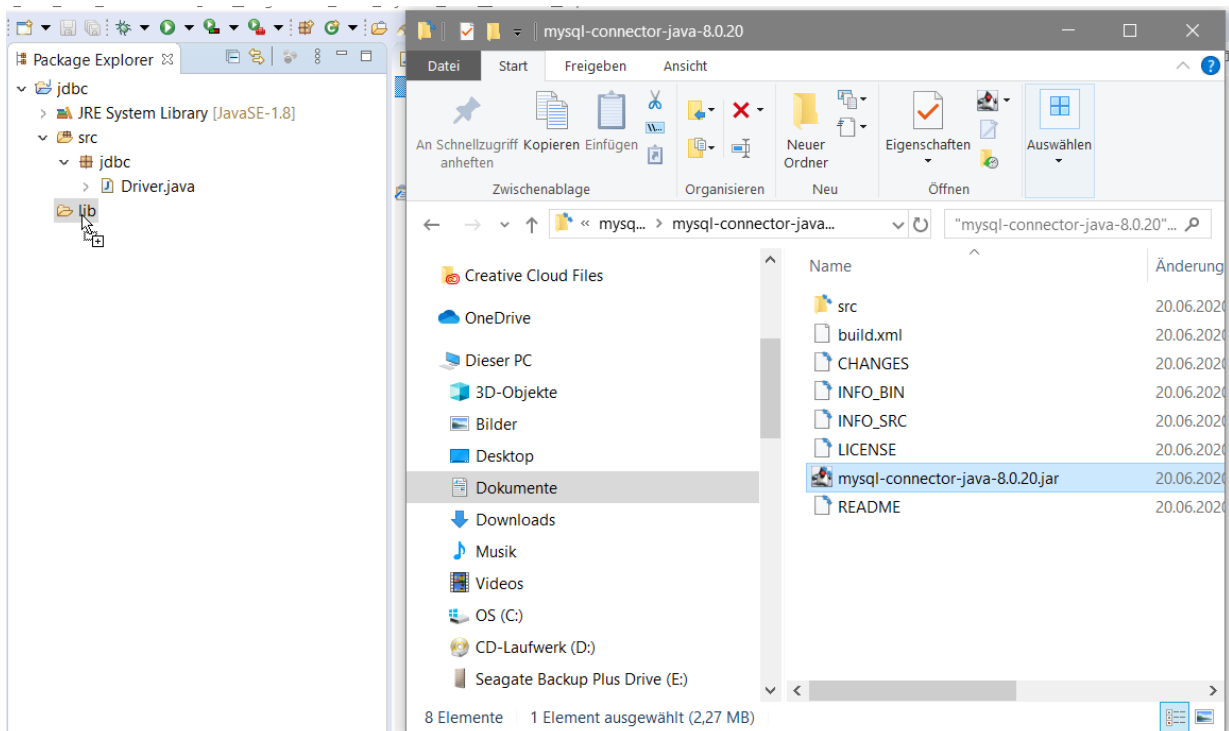
1. Rechtsklick auf „jdbc“ Reiter und neuen Folder erstellen.



Danach den Folder benennen (in meinem Fall „lib“ für library) und auf „finish“ klicken.

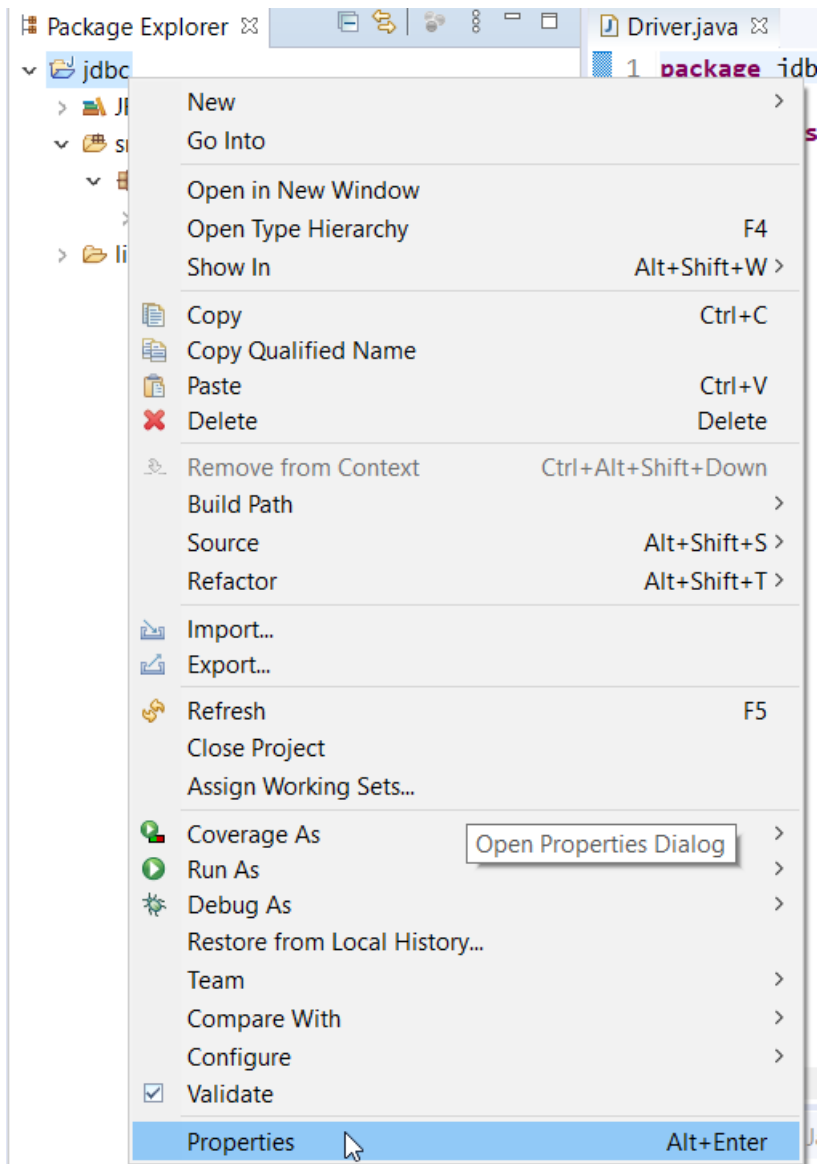


Per Drag&Drop wird nun aus dem zuvor entpackten Ordner die **jar** Datei genommen und in den ebenfalls zuvor erstellen Folder „**lib**“ gezogen.



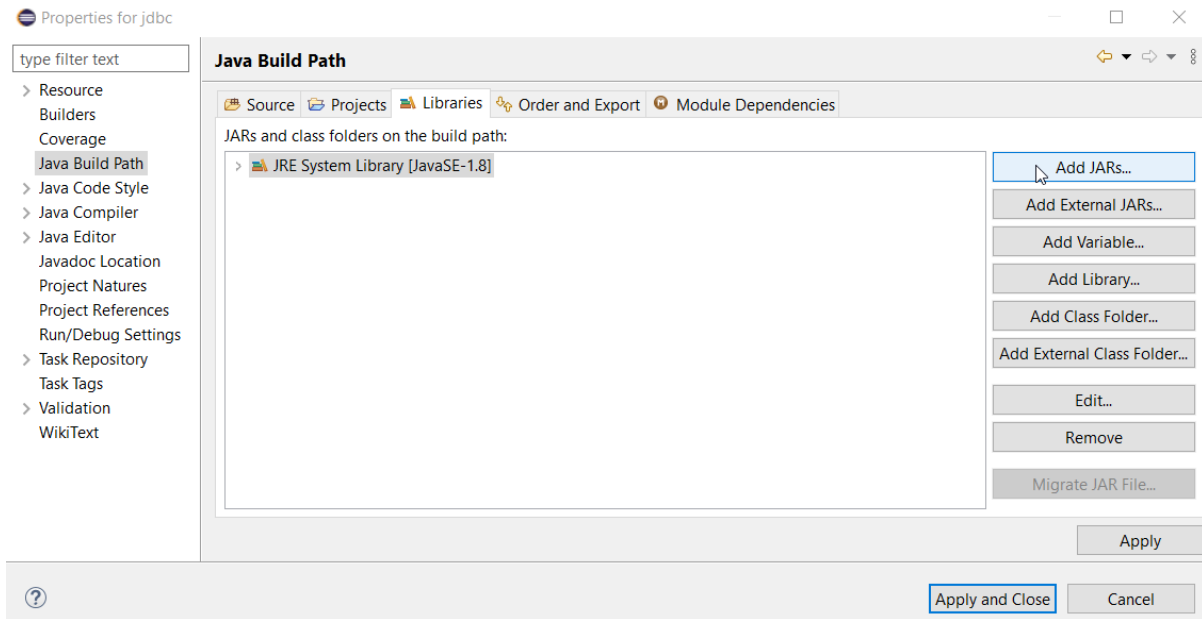
Danach auf „**Copy files**“ klicken.

Jetzt Rechtsklick auf den **oberen jdbc** Reiter und auf **Properties** klicken.

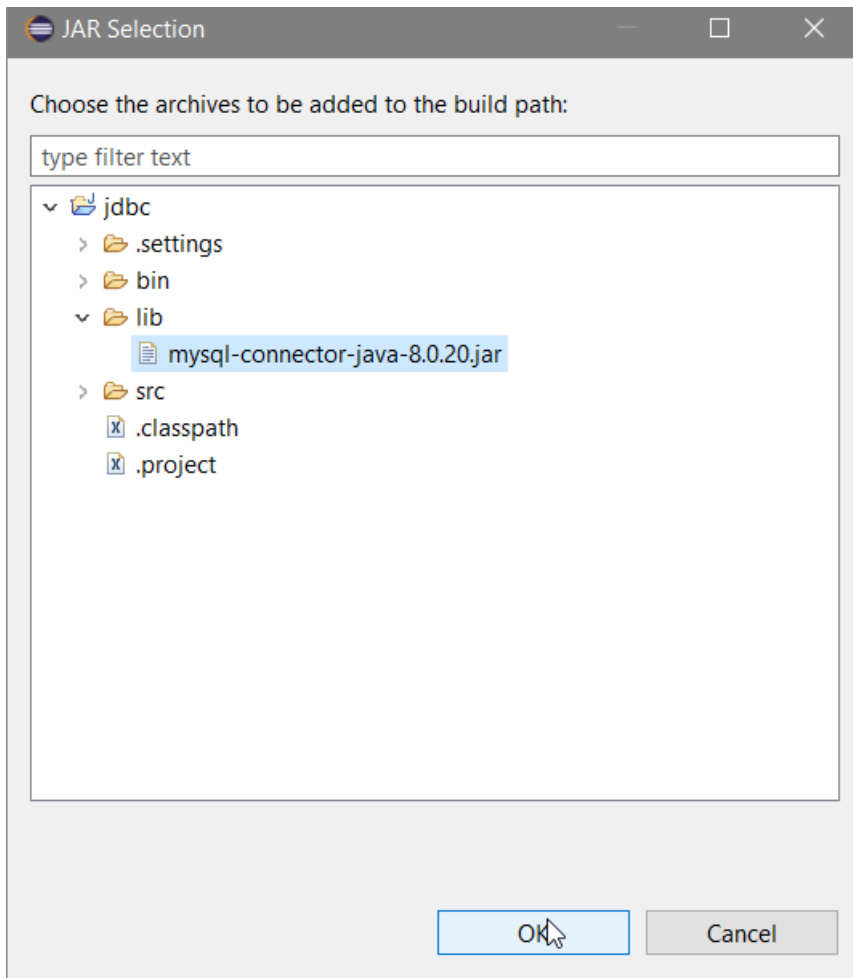


Dann folgendes auswählen:

Java Build Path → oben auf Libraries → Add JARs



Beim nachfolgenden Fenster wird dann die **.jar** Datei ausgewählt.



2. Code

2.0 Import und Exception

Als erstes wird das `java.sql.*` package importiert (da dort alle unsere Klassen und Interfaces drin sind die wir brauchen)
Ein try/catch Block wird benötigt, falls irgendwelche Exceptions auftreten.

```
package jdbc;

import java.sql.*;

public class Driver {

    public static void main(String[] args) {

        try {

        }

        catch (Exception exc) {
            exc.printStackTrace();
        }

    }

}
```

2.1 Verbindung zur Datenbank aufbauen

Dann werden die Daten für die Verbindung geschrieben. Es beginnt mit der Standardphrase „`jdbc:mysql://`“ danach wird die IP-Adresse der Datenbank eingesetzt. Sollte diese am selben Computer sein, kann auch einfach `localhost` eingesetzt werden, dann ein Doppelpunkt und die Portnummer 3306. Diese Portnummer wird meist für MySQL Datenbanken verwendet. (mehr dazu [hier](#))

Nun folgt nach einem `/` der Name der Datenbank und die jetzt schließenden `“`. In die letzten zwei Plätze kommt zuerst der Benutzername der Datenbank (falls man nie einen eingestellt hat, ist dieser Standardmäßig „`root`“) und das Passwort. (falls keines eingestellt einfach leer lassen)

Somit lautet der Aufbau wie folgt:

```
Connection myConn =
DriverManager.getConnection("jdbc:mysql://ipadresse/datenbankname",
"benutzername" , "passwort");
```

Und in meinem Fall:

```
Connection myConn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1/fuhrpark", "root" ,
"");
```

2.2 Statement erstellen

Ein Statement mit dem Namen `myStmt` wird erstellt und auf die oben erstellte Connection wird verwiesen.

```
Statement myStmt = myConn.createStatement();
```

2.3 SQL-Abfrage ausführen

`ResultSet` ist ein Objekt, welches einen Cursor beinhaltet, der auf die momentane Reihe von Daten zeigt. Der Cursor wird vor die erste Reihe gesetzt, da die nächste Methode den Cursor zur nächsten Reihe bewegt, und da er „false“ zurückliefert wenn es keine Reihen mehr im `ResultSet` Objekt gibt, kann es in einer while-Schleife verwendet werden, um die Ergebnismenge zu iterieren.

Dann wird auf das Statement verwiesen und mit `.executeQuery` ein `ResultSet` object returned. In die Klammern kommt dann die beliebige SQL-Abfrage.

```
ResultSet myRs = myStmt.executeQuery("SELECT * from fahrer left join route  
ON route.fahrer=fahrer.name");
```

2.4 Ergebnis verarbeiten und ausgeben

Erstellung einer while-Schleife, mit der Bedingung, erst aus der Schleife zu hüpfen, wenn mein `ResultSet` „false“ zurückliefert.

`getString` beinhaltet den Namen meiner Spalte die ausgegeben werden soll. Meine zwei auszugebenden Spalten sollen mit einem Beistrich und einem Leerzeichen getrennt sein.

```
while (myRs.next()) {  
    System.out.println(myRs.getString("fahrer.name") + ", " +  
myRs.getString("route.id"));  
}
```

Wenn man in Eclipse nun auf **Run** → **Run as** → **Java Application** geht, sollte ähnliches ausgegeben werden:



```

<terminated> Driver [Java Application] C:\Program Files\Java\jre1.8.0_181\bin\javaw.exe (21.06.2020, 01:19:14 – 01:19:15)
Paul Radauer, INN-KUF
Metin Doganer, INN-MÜN
Paul Radauer, INN-WIE
Marcel Miljak, INN-ZÜR
Manuel Fähndrich, SAL-WIE
Alexander Jäger, null
Armin Kadic, null
David Eder, null
Elias Dinkhauser, null
Kevin Gubitzer, null
Laurenz Preindl, null
Leonhard Schreiner, null
Michael Klingler, null
Simon Graber, null
Stefan Wegleiter, null

```

Gesamter Code:

```

package jdbc;

import java.sql.*;

public class Driver {

    public static void main(String[] args) {

        try {
            // 1. Verbindung zur Datenbank aufbauen
            Connection myConn =
DriverManager.getConnection("jdbc:mysql://127.0.0.1/fuhrpark", "root" , "");
            // 2. Statement erstellen
            Statement myStmt = myConn.createStatement();

            // 3. SQL Abfrage ausführen
            ResultSet myRs = myStmt.executeQuery("SELECT * from fahrer left
join route ON route.fahrer=fahrer.name");

            // 4. Ergebnis verarbeiten und ausgeben
            while (myRs.next()) {
                System.out.println(myRs.getString("fahrer.name") + ", " +
myRs.getString("route.id"));
            }

        }
        catch (Exception exc) {
            exc.printStackTrace();
        }

    }

}

```

Quellen:

<https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/getstring-method-java-lang-string-sqlserverresultset?view=sql-server-ver15>

<https://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html>

<https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html>

<https://javatutorialhq.com/java/util/scanner-class-tutorial/next-method-example/>

<https://docs.microsoft.com/en-us/sql/connect/jdbc/reference/getstring-method-java-lang-string-sqlserverresultset?view=sql-server-ver15>