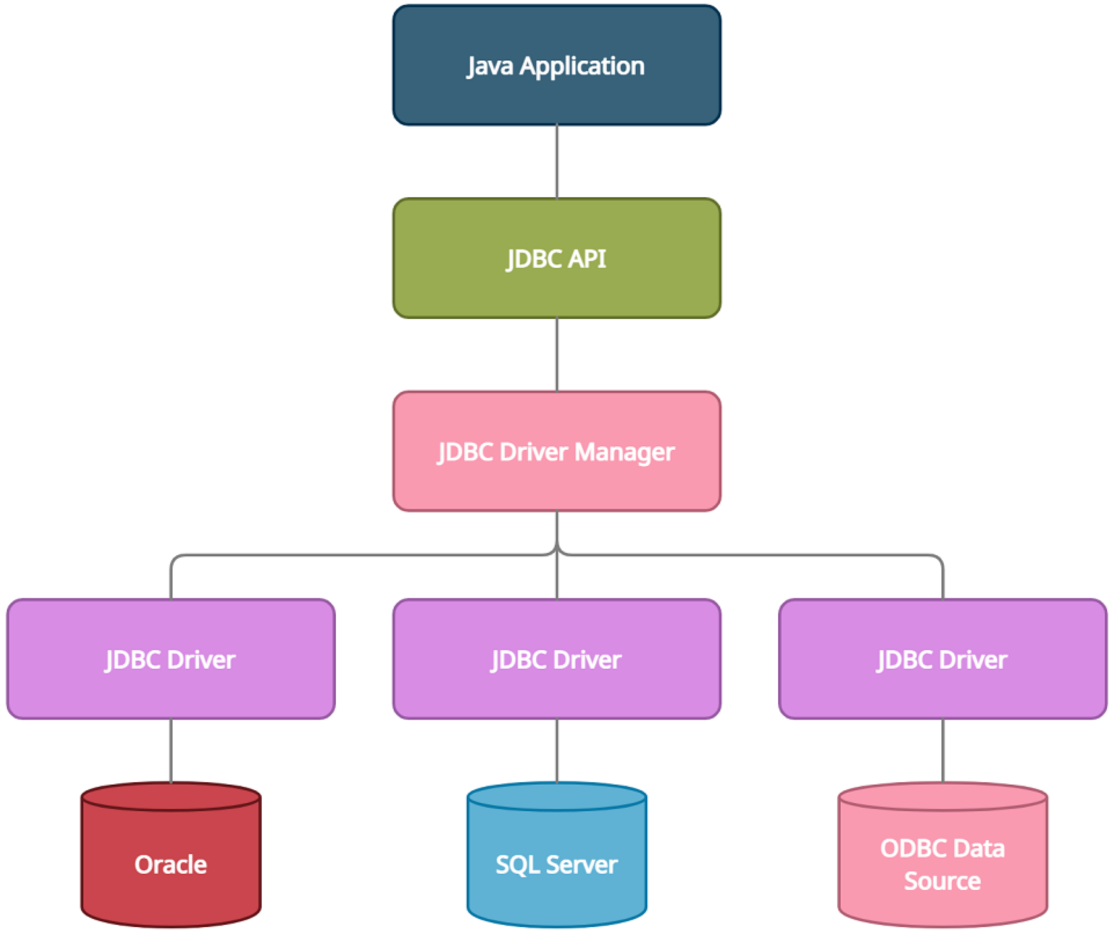
JDBC IN JAVA

JDBC allows Java applications to connect to database.

The Architecture of JDBC in Java



JDBC IN JAVA

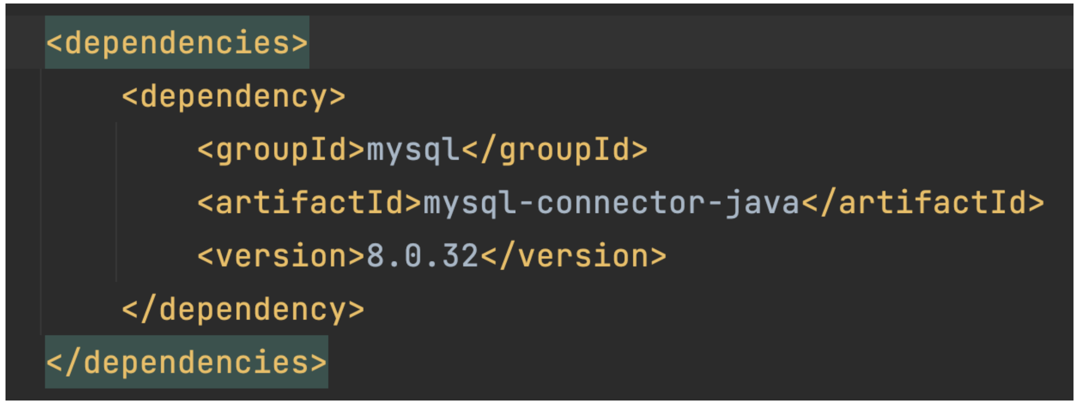
Java Database Connectivity

Pre- requisites:

1- Java JDDK
2- IDE (eclipse or Inteelij)
3- Driver (mysql, oracle, ms-sql, postgresDb)
https://dev.mysql.com/downloads/connector/j/

POM.xml dependency

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>8.0.32</version>
</dependency>
```

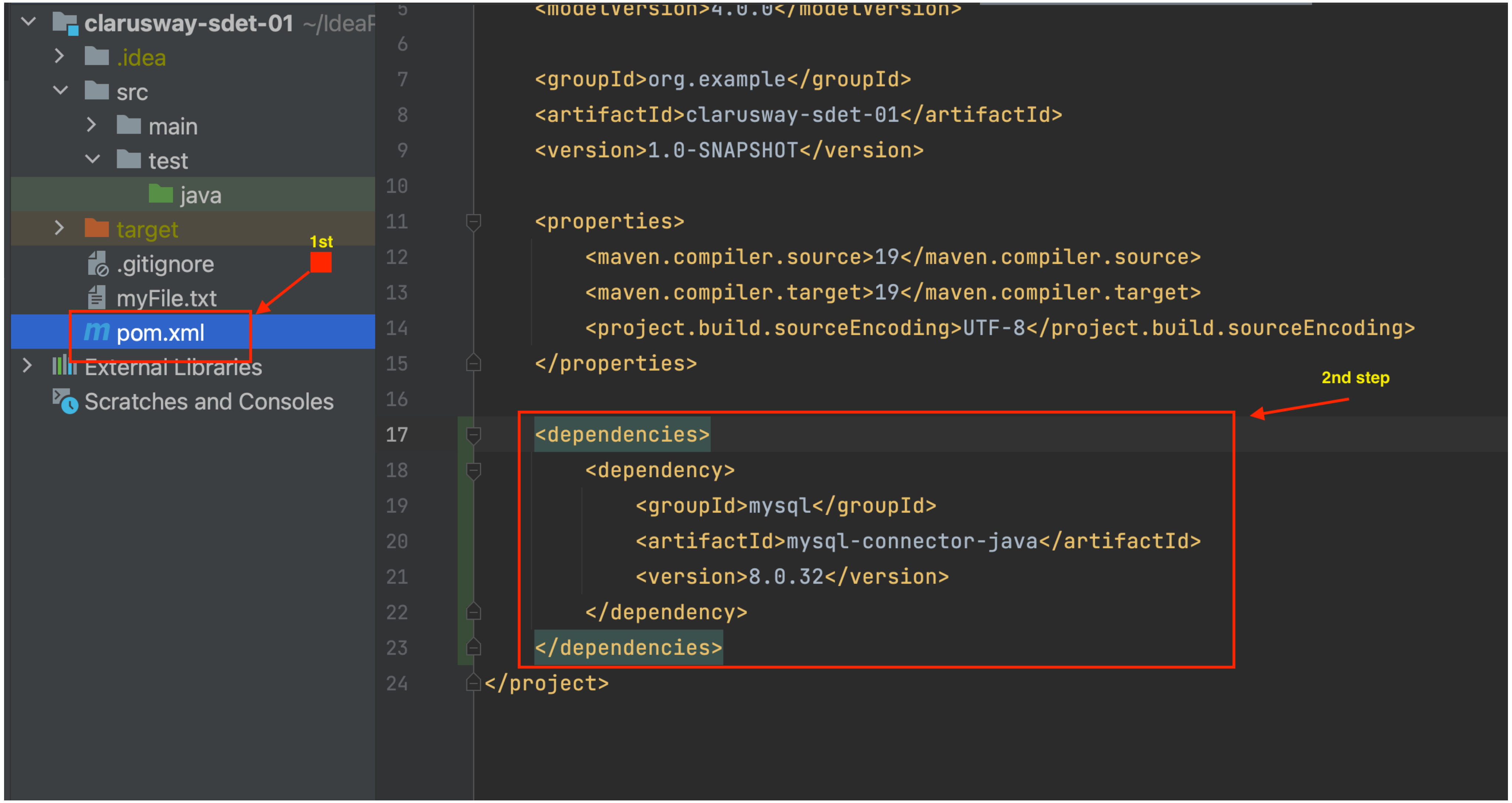


```
<dependencies>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.32</version>
    </dependency>
</dependencies>
```

Java™

CLARUSWAY
WAY TO REINVENT YOURSELF

JDBCIN JAVA



```
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>clarusway-sdet-01</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>19</maven.compiler.source>
        <maven.compiler.target>19</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.32</version>
        </dependency>
    </dependencies>
</project>
```

JDBC IN JAVA

JDBC Driver Manager

DriverManager helps to connect, and application based on the database connection string.

The JDBC drivers are automatically loaded based on the classpath in newer versions.

JDBC API

Key classes:

- java.sql.DriverManager
- java.sql.Connection
- java.slq.Statment
- java.sql.ResultSet

JDBC IN JAVA

Development Process

Insert-Update-Delete (DML)

1- Create Connection

2- Create Statement/Query

3- Execute Statement/Query

4- Close Connection

Select statement (DQL)

1- Create Connection

2- Create Statement/Query

3- Execute Statement/Query

4- Store results in ResultSet

5- Close Connection

Step 1: Create connection to a database

For connecting a DB: Need a connection string in form of JDBC URL

Basic sysntax: jdbc:<driver_protocol>:<driver_connection_details>

Example:

Database	JDBC Host URL
MS SQL Server	jdbs:odbc:DemoDatabase
Oracle	jdbc:oracle:thin@myserver_name:3306:DemoDatabase
MySQL	jdbc:mysql://localhost:3306/DemoDatabase

Step 1: Create connection to a database

```
String host =“jdbc:mysql://localhost:3306/DemoDatabase”;
String user= “root”;
String password= “rootuser”;

Connection con = DriverManager.getConnection(host, user, password);
```

Note: If connection fails, it will throw an exception and due to that we need to handle
The exception using try-catch block.

Step 2: Create Statement object

The Statement object is based on the connection and later we will use it for execution.

```
String host =“jdbc:mysql://localhost:3306/DemoDatabase”;
String user= “root”;
String password= “rootuser”;

Connection con = DriverManager.getConnection(host, user, password);
```

Statement st = con.createStatement();

JDBC IN JAVA

Step 3: Execute SQL query

```
String host =“jdbc:mysql://localhost:3306/DemoDatabase”;
String user= “root”;
String password= “rootuser”;

Connection con = DriverManager.getConnection(host, user, password);

Statement st = con.createStatement();

ResultSet rs = st.executeQuery(“select * from animals”);
```

Step 4: Process the result (ResultSet.next())

```
String host =“jdbc:mysql://localhost:3306/DemoDatabase”;
String user= “root”;
String password= “rootuser”;

Connection con = DriverManager.getConnection(host, user, password);

Statement st = con.createStatement();

ResultSet rs = st.executeQuery(“select * from animals”);

While(rs.next()){
            int id= rs.getInt(“id”);
            String animalName= rs.getString(“name”);
            System.out.println(“id : “+id+” “+ “Animal Name : “ +animalName);

}
```

PreparedStatements

- What are Prepared Statements
- Create PreparedStatements
- Setting parameter values
- Executing a PreparedStatement
- Reusing a PreparedStatements

A Prepared Statement is a precompiled SQL statement.

Benefits:

Makes it easier to set SQL parameter values
Prevent SQL injection attacks.
Possible performance increment.

JDBC IN JAVA

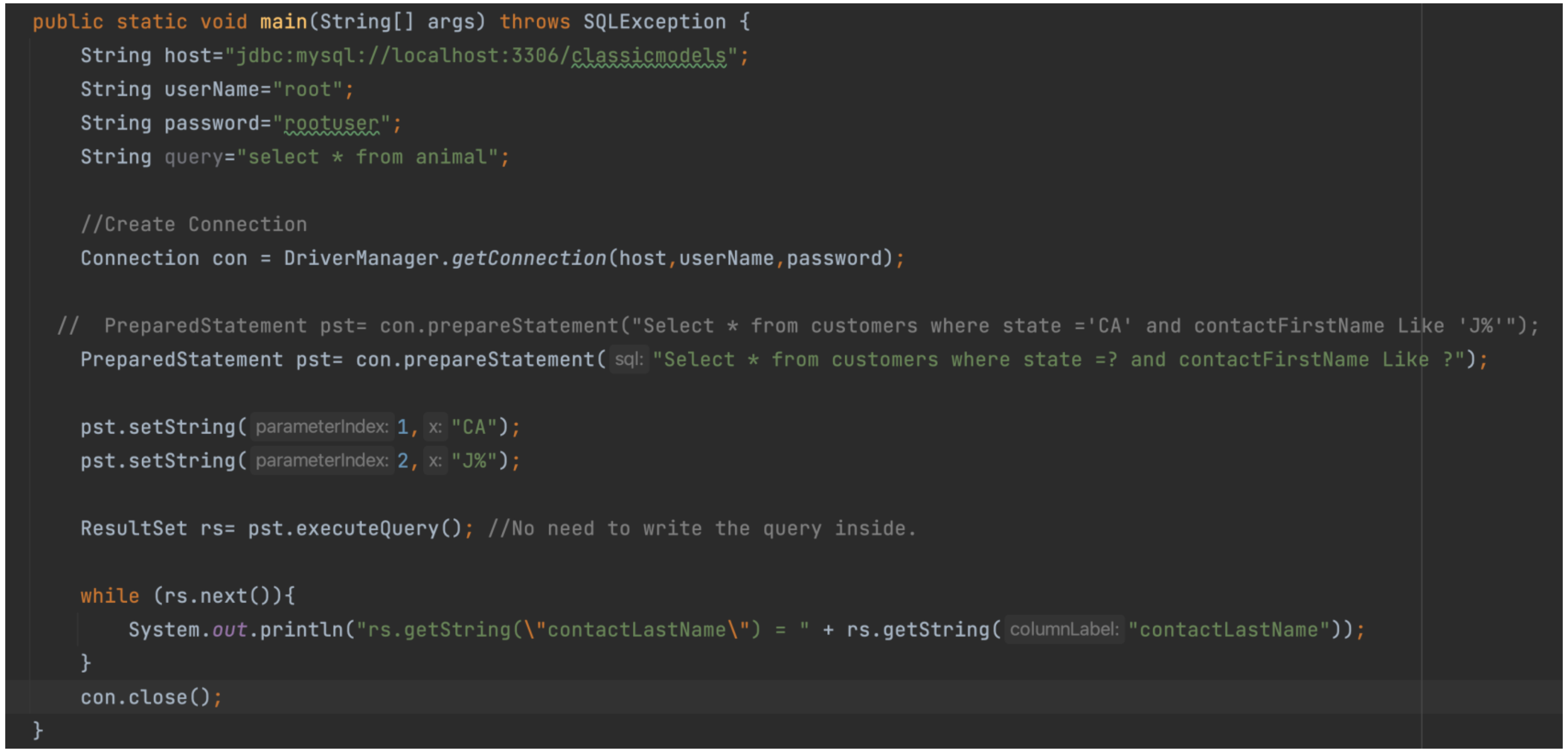
Using Prepared Statements

Instead of hard coding your SQL values:

Select * from customers where state ='CA' and contactFirstName Like 'J%';

Set patameters placeholders using question mark.

Select * from customers where state =? and contactFirstName Like ? ;



```
public static void main(String[] args) throws SQLException {
    String host="jdbc:mysql://localhost:3306/classicmodels";
    String userName="root";
    String password="rootuser";
    String query="select * from animal";

    //Create Connection
    Connection con = DriverManager.getConnection(host,userName,password);

  //  PreparedStatement pst= con.prepareStatement("Select * from customers where state ='CA' and contactFirstName Like 'J%'");
    PreparedStatement pst= con.prepareStatement( sql: "Select * from customers where state =? and contactFirstName Like ?");

    pst.setString( parameterIndex: 1, x: "CA");
    pst.setString( parameterIndex: 2, x: "J%");

    ResultSet rs= pst.executeQuery(); //No need to write the query inside.

    while (rs.next()){
        System.out.println("rs.getString(\"contactLastName\") = " + rs.getString( columnLabel: "contactLastName"));
    }
    con.close();
}
```

JDBC IN JAVA

Statement vs Prepared statement

```
String sql = "SELECT * FROM users WHERE name = '" + name + "'";
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(sql);
```

vs

```
String sql = "SELECT * FROM users WHERE name = ?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, name);
ResultSet rs = pstmt.executeQuery();
```

```
//Create the prepared statement once, outside of any loops or functions
String sql = "INSERT INTO users (name, email) VALUES (?, ?)";
PreparedStatement pstmt = conn.prepareStatement(sql);

//Then later, in a loop or function, you can reuse the prepared statement by setting new parameter values:
for (int i = 0; i < list.size(); i++) {
   pstmt.setString(1, list.get(i).getName());
   pstmt.setString(2, list.get(i).getEmail());
   pstmt.executeUpdate();
}
```