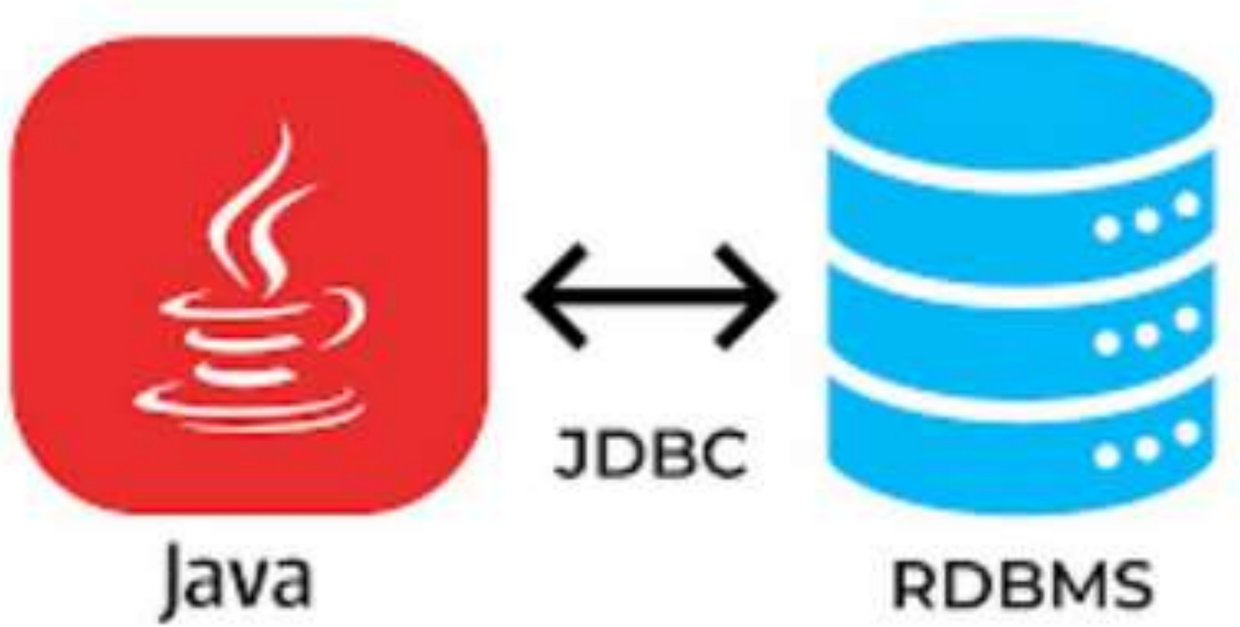
SB101

Day-13 & 14

JDBC: DML statement

JDBC

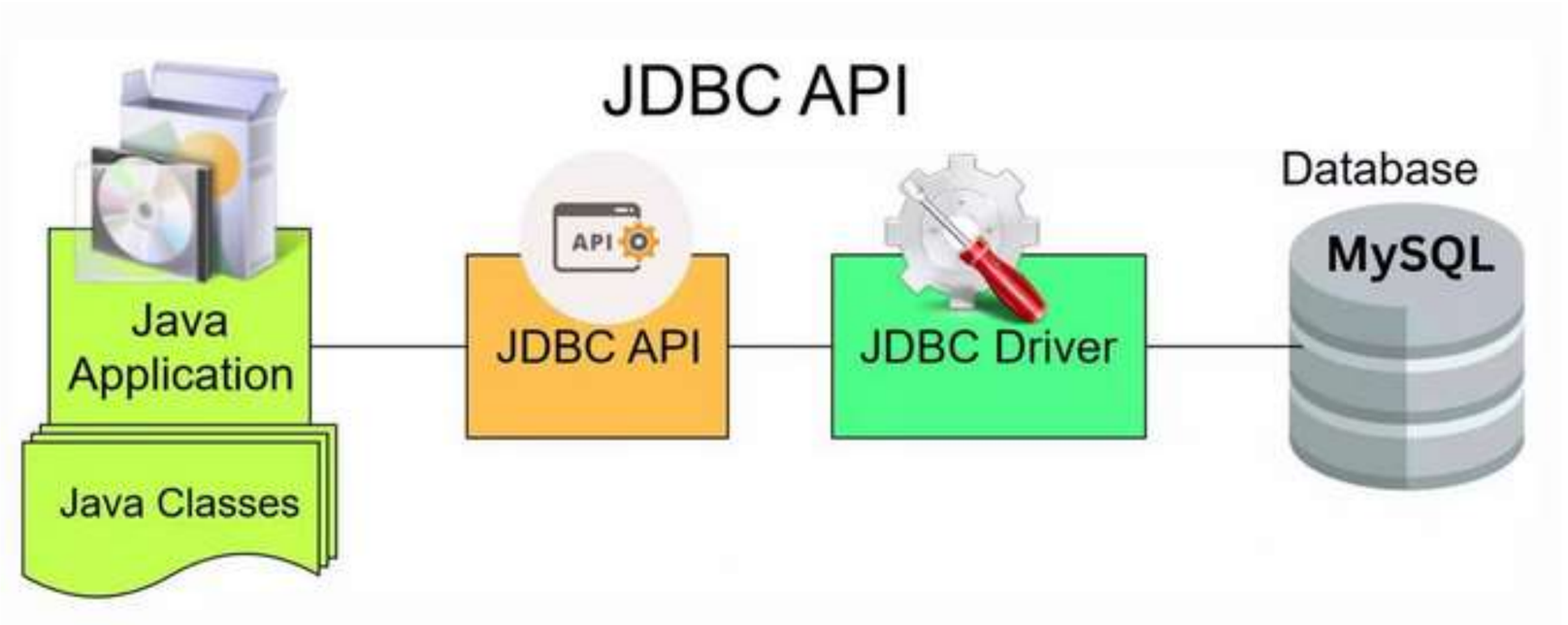
JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database.



JDBC Driver

JDBC Driver is a software component that enables java application to interact with the database.

A JDBC driver makes it possible to do three things: Establish a connection with a data source. Send queries and update statements to the data source.



Download & attach connector jar file

Open the following link-

https://dev.mysql.com/downloads/connector/j/

then go to Platform Independent (Architecture Independent), ZIP Archive

***or;* just go to**

https://dev.mysql.com/downloads/file/?id=513221

To import jar file in your Eclipse IDE, follow the steps given below.

1. **Right click on your project**
2. **Select Build Path**
3. **Click on Configure Build Path**
4. **Click on Libraries and select Add External JARs**
5. **Select the jar file from the required folder**
6. **Click and Apply and Ok**

The static block

- The static block will be executed before the main method of our application.
- As soon as our application is loaded into the memory, static blocks will be executed.
- Multiple static blocks would execute in the order they are defined in the class.
- A typical static block looks like

```
static{
        // code for static block
}
```

The non-static block

- A non-static block executes when the object is created, before the constructor
- Non-static blocks will never execute if we don't create an object of a class.
- In the case of multiple non-static blocks, the block executes the order in which it is defined in the class.
- All non-static block executes every time an object of the containing class is created.
- A typical non-static block looks like below

```
{
    // non static block
}
```

classes and interfaces for connectivity

1 **Register the driver class using following method of Class class**

```
public static void forName(String fqcn) throws ClassNotFoundException
```

```
try {
        Class.forName("com.mysql.cj.jdbc.Driver");
} catch (ClassNotFoundException e) {
        e.printStackTrace();
}
```

Following happens in the background-

a. The driver-related main class will be loaded into the primary memory.

b. Driver-related main class objects will be created.

c. That driver object will be registered with the DriverManager class.

classes and interfaces for connectivity

```
try {
        Driver d = new Driver();
        DriverManager.registerDriver(d);
} catch (SQLException e1) {
        e1.printStackTrace();
}
```

The above line of code is written inside the static block of every Driver class.

classes and interfaces for connectivity

2 **Connect to database using following method of DriverManager class**

public static Connection getConnection(String url)throws SQLException
public static Connection getConnection(String url, String name, String password) throws SQLException

- **A Connection is a session between a Java application and a database. It helps to establish a connection with the database.**
- **The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData, i.e., an object of Connection can be used to get the object of Statement and DatabaseMetaData.**
- **The Connection interface provide many methods for transaction management like commit(), rollback(), setAutoCommit(), setTransactionIsolation(), etc.**

classes and interfaces for connectivity

public Statement createStatement() throws SQLException
public PreparedStatement preparedStatement(String sql) throws SQLException
public void commit() throws SQLException
public void rollback() throws SQLException
public void close() throws SQLException
public boolean getAutoCommit() throws SQLException
Savepoint setSavepoint(String name) throws SQLException

classes and interfaces for connectivity

3.1 Using statement interface

The statement interface is used to create SQL basic statements in Java it provides methods to execute queries with the database.

The Statement interface accepts strings as SQL queries. Thus, the code becomes less readable when we concatenate SQL strings. Hence the statement interface is more suitable for the DDL statements and static SQL statements.

```
public int executeUpdate(String dml) throws SQLException;
public ResultSet executeQuery(String drl) throws SQLException;
public boolean execute(String any sql) throws SQLException
Connection getConnection() throws SQLException
void close() throws SQLException
```

classes and interfaces for connectivity

Limitation of the statement interface

- **The Statement interface can’t be used for storing and retrieving files and arrays.**
- **Complexities to concatenate the dynamic variables because it makes query unreadable**
- **Whenever we pass any query to the DB using the Statement object, the DB engine will perform the following tasks every time to execute that query:**

a. query compilation
b. query plan generation
c. query optimization

- **To tell the DB engine to perform the above 3 tasks only for once with the value or without the value (In case of without the value, make use of placeholders ?) and put that query inside the cache, next time onwards just add the dynamic values and execute that query.**

classes and interfaces for connectivity

3.2 Using PreparedStatement interface

The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized queries.

Before executing the query we need to bind the appropriate values to the corresponding placeholders by calling various types of setXXX(--) on the PreparedStatement object.

void setBoolean(int Index, boolean x)	void setDate(int Index, Date x)
void setDouble(int Index, double x)	void setInt(int Index, int x)
void setString(int Index, double x)	void setInt(int Index, int x)

After binding the appropriate placeholders we need to execute the query using executeQuery(), executeUpdate() and execute() method.

Note: Since Java 7, JDBC has ability to use try-with-resources statement to automatically close resources of type Connection, ResultSet, and Statement. It avoids explicit connection closing step.

QUERY?