**📘 JDBC with MySQL – Complete Topics**

**🔹 1. What is JDBC?**

**JDBC** (Java Database Connectivity) is an API that allows Java applications to connect to relational databases (like MySQL), execute SQL queries, and retrieve/manipulate data.

**🔹 2. JDBC Architecture**

* **Driver Manager** – Manages a list of database drivers.
* **Driver** – Interface between Java app and the DB.
* **Connection** – Session with the DB.
* **Statement/PreparedStatement** – Executes SQL.
* **ResultSet** – Holds the result of a query.
* **SQLException** – Handles DB-related errors.

**🔹 3. JDBC Setup with MySQL**

**✅ Steps:**

1. Add **MySQL JDBC driver** to your project:

xml

CopyEdit

<!-- pom.xml for Maven -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.0.33</version>

</dependency>

1. Load the driver:

java

CopyEdit

Class.forName("com.mysql.cj.jdbc.Driver");

1. Connect to DB:

java

CopyEdit

Connection conn = DriverManager.getConnection(

"jdbc:mysql://localhost:3306/telecom", "root", "password");

**🔹 4. Statement vs PreparedStatement**

**🔸 Statement (for static queries)**

java

CopyEdit

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM customers");

**🔸 PreparedStatement (for dynamic, safe queries)**

java

CopyEdit

PreparedStatement ps = conn.prepareStatement(

"SELECT \* FROM customers WHERE customer\_id = ?");

ps.setString(1, "C001");

ResultSet rs = ps.executeQuery();

✔️ Avoids SQL injection  
✔️ Faster with caching

**🔹 5. CRUD Operations with JDBC**

**➕ INSERT**

java

CopyEdit

PreparedStatement ps = conn.prepareStatement(

"INSERT INTO customers VALUES (?, ?, ?, ?)");

ps.setString(1, "C007");

ps.setString(2, "Asha");

ps.setString(3, "9876543210");

ps.setString(4, "Gold");

ps.executeUpdate();

**✏️ UPDATE**

java

CopyEdit

PreparedStatement ps = conn.prepareStatement(

"UPDATE customers SET plan\_type=? WHERE customer\_id=?");

ps.setString(1, "Platinum");

ps.setString(2, "C007");

ps.executeUpdate();

**❌ DELETE**

java

CopyEdit

PreparedStatement ps = conn.prepareStatement(

"DELETE FROM customers WHERE customer\_id=?");

ps.setString(1, "C007");

ps.executeUpdate();

**🔹 6. ResultSet Navigation**

java

CopyEdit

while (rs.next()) {

System.out.println(rs.getString("name"));

}

* rs.getInt(), rs.getString() used to extract column values.
* rs.next() moves cursor forward.

**🔹 7. Batch Processing**

Executes multiple SQL statements as a batch (faster):

java

CopyEdit

PreparedStatement ps = conn.prepareStatement(

"INSERT INTO calls VALUES (?, ?, ?, ?, ?)");

for (...) {

ps.setString(1, callId);

ps.setString(2, customerId);

ps.setDate(3, callDate);

ps.setInt(4, duration);

ps.setDouble(5, charge);

ps.addBatch();

}

ps.executeBatch();

**🔹 8. Transaction Management**

java

CopyEdit

conn.setAutoCommit(false);

try {

// 1. deduct balance

// 2. insert into payments

conn.commit();

} catch (SQLException e) {

conn.rollback();

}

Used to maintain **ACID** properties in apps like billing.

**🔹 9. CallableStatement (Stored Procedures)**

java

CopyEdit

CallableStatement cs = conn.prepareCall("{call getCustomerBill(?)}");

cs.setString(1, "C001");

ResultSet rs = cs.executeQuery();

Useful if you move logic to the MySQL database.

**🔹 10. Handling Exceptions**

java

CopyEdit

try {

...

} catch (SQLException e) {

System.out.println("Error: " + e.getMessage());

}

Always **close resources** in finally or use **try-with-resources**.

**🔹 11. JDBC Best Practices**

* Use PreparedStatement always
* Close Connection, Statement, ResultSet
* Use **connection pooling** (HikariCP, Apache DBCP)
* Log queries and exceptions
* Normalize database and validate inputs

**🔹 12. Sample JDBC Utility Class**

java

CopyEdit

public class DBUtil {

private static final String URL = "jdbc:mysql://localhost:3306/telecom";

private static final String USER = "root";

private static final String PASSWORD = "root";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(URL, USER, PASSWORD);

}

}