JDBC (java database connectivity) is an API packaged with java that makes it possible to standardize and simplify the process of connecting java applications to external RDBMS.

java supports logical programming. but does not store data persistently.

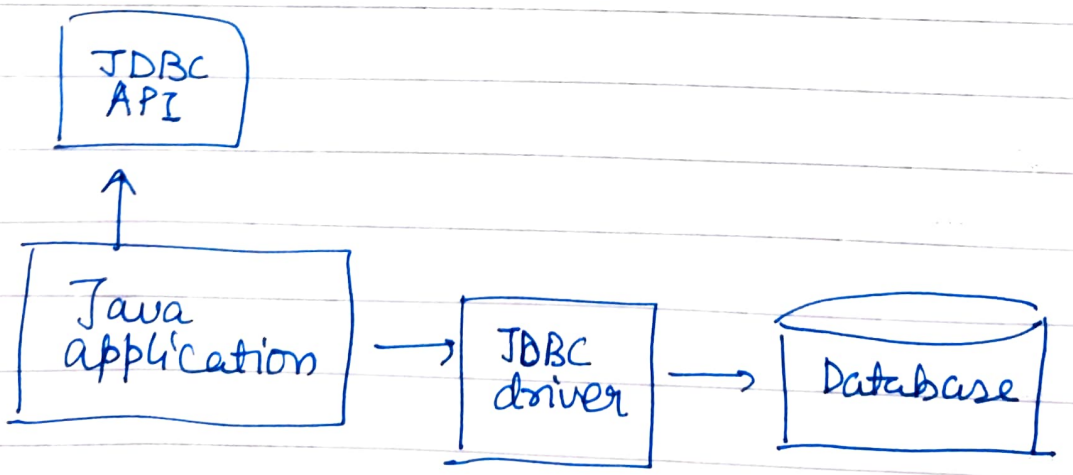
Data persistance is typically delegated to NoSQL databases such as MongoDB & cassandra or RDB's such as IBM's DB2 or, ~~HHE~~ MYSQL.

JDBC is a java API to connect and execute the query with the database. ~~It is a part of~~

① JDBC ~~API~~ API uses JDBC drivers to connect with the database.

② to access tabular data stored in any relational db.

③ Update, delete, fetch data from db.



```
import java.sql.*;
import java.util.Date;
```

→ import java.sql.Connection;
.sql.DriverManager;
• PreparedStatement
• ResultSet
• SQLException
• Statement.

```
public class FirstExample {
```

JDBC driver name & db URL (MYSQL connectivity)

```
static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
static final String DB_URL = "jdbc:mysql://localhost/EMP";
```

↳ jdbc:mysql://hostname/dbname

```
static final String USER = "username";
static final String PASS = "password";

public static void main(String args[]) {

    Connection conn = null;
    Statement stmt = null;

try {

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

↓

method used to dynamically load the driver's class file into memory, which automatically registers it. This method allows you to make the driver registration configurable and portable.

other methods like getInstance()
registerDriver() can be used
↓
for non jdk compliant

Interfaces

open a connection

S.O.p ("connecting to db");

Conn = DriverManager.getConnection(DB_URL, USER, PASS);

↓ method for establishing connection.

3 formats getConnection (String URL) → URL already has username & pwd

(String URL, properties prop)

(String URL, String USER, String PWD)

Properties obj holds a set of keyword value pairs.
It is used to pass driver properties to the driver.

↳ Properties info = new properties();
info.put ("user", "username");
info.put ("password", "password");
getconnection (URL, info);

query execution

System.out.println("statements")

→ Stmt = Conn.~~CCSE~~ createStatement();

String SqL;

SqL = "SELECT id, first, last, age FROM Employees";

→ ResultSet rs = Stmt.executeQuery (SqL);

↓ contains result of query execution.

Statement objects are required to execute an SQL query. They are created using CreateStatement()

Statement obj can then be used to execute SQL by using either of the following methods:

① boolean execute(String SQL): for DDL, returns true/false.

② int executeUpdate(String SQL): return no of rows affected by execution of SQL st. eg insert, update, delete

③ ResultSet executeQuery(String SQL): in case of select.

→ result set method to move to next row of result set, making it current row.

```
while (rs.next()) {
  int id = rs.getInt("id");
  Int age = rs.getInt("age");
  String first = rs.getString("first");
  String last = rs.getString("last");

display values

   S.O.p("Id:" + id);
  S.O.p(", Age:" + age);
  S.O.p(", first" + first);
  S.O.p(", last " + last);
}
```

rs.close();
Stmt.close();
conn.close(); } catch(SQLException se) {
se.printStackTrace();
}

```
catch (Exception e) {
        se.printStackTrace();
} finally {
    try {
        if (stmt != null)
            stmt.close();
    } catch (SQLException se2) { }

try {
    if (conn != null)
        conn.close();
} catch (SQLException se) {
        se.printStackTrace();
    }
}
}
```

Three important interfaces

Statement : for general purpose access to database.
Can not accept parameters
connection objects createstatement() method is used to create a statement obj.
Should be ~~cnct~~ closed properly to save db resources and ensure proper cleanup.

Prepared Statement : use this when you plan to use the sql statements many times.
This interface accepts input parameters at runtime.

This interface extends the statement i/f.
Gives you the flexibility of supplying arguments dynamically

eg

```
PreparedStatement pstmt = null;
try {
    String SQL = " update Employee SET age = ? where
                  id = ? " ;
    pstmt = conn.prepareStatement(SQL);

    pstmt.close
```

(? : parameter refers)

Callable Statement

↳ use when you want to access the database stored procedures. Can accept runtime input parameters