JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks commonly associated with database usage:

* Making a connection to a database.
* Creating SQL statements.
* Executing that SQL queries in the database.
* Viewing & Modifying the resulting records.

JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as:

* Java Applications
* Java Applets
* Java Servlets
* Java ServerPages (JSPs)
* Enterprise JavaBeans (EJBs)

All of these different executables can use a JDBC driver to access a database and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

JDBC Architecture:

The JDBC API supports both two-tier and three-tier processing models for database access but in general JDBC Architecture consists of two layers:

* JDBC API: This provides the application-to-JDBC Manager connection.
* JDBC Driver API: This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.

The JDBC driver manager ensures that the correct driver is used to access each data source. The driver manager is capable of supporting multiple concurrent drivers connected to multiple heterogeneous databases.

Common JDBC Components:

The JDBC API provides the following interfaces and classes:

* DriverManager: This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication subprotocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database connection.
* Driver: This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manage objects of this type. It also abstracts the details associated with working with Driver objects
* Connection: This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
* Statement: You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
* ResultSet: These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
* SQLException: This class handles any errors that occur in a database application.

The java.sql and javax.sql are the primary packages for JDBC. It offers the main classes for interacting with your data sources.

## Creating JDBC Application:

Steps involved in building a JDBC application:

* Import the packages. Requires that you include the packages containing the JDBC classes needed for database programming. Most often, using *import java.sql.\** will suffice.
* Register the JDBC driver. Requires that you initialize a driver so you can open a communications channel with the database.
* Open a connection. Requires using the *DriverManager.getConnection()* method to create a Connection object, which represents a physical connection with the database.
* Execute a query. Requires using an object of type Statement for building and submitting an SQL statement to the database.
* Extract data from the result set. Requires that you use the appropriate *ResultSet.getXXX()* method to retrieve the data from the result set.
* Clean up the environment. Requires explicitly closing all database resources versus.

ORACLE🡪 ojdbc.jar

MYSQL🡪 MySQL-connection.5.18.jar

Class.forName(“com.mysql.jdbc.Driver”) 🡪 MySql

Class.forName(“oracle.jdbc.driver.OracleDriver”); 🡪 Oracle

Driver Manager:

Static connection getConnection(url,username,password) 🡪 is a factory method with 3 parameters which returns a connection objects. It is a static method.

String url = “jdbc:mysql://localhost:3306/sampledb”;

String username = “root”;

String password = ”root”;

Or

String url =”jdbc:oracle:thin:@localhost:1521:xe”;

String username = “system”;

String password =”manager”;

Conenction con = DriverManager.getConnection(url,username,password);

Statement: it is an interface ( we can create reference)

Int executeUpdate(String) //insert /update/delete

ResultSet executeQuery(String) // select

Connection:

Statement createStatement(); // Static Factory Method();

ORACLE🡪 ojdbc.jar

MYSQL🡪 MySQL-connection.5.18.jar

Class.forName(“com.mysql.jdbc.Driver”) 🡪 MySql

Class.forName(“oracle.jdbc.driver.OracleDriver”); 🡪 Oracle

Driver Manager:

Static connection getConnection(url,username,password) 🡪 is a factory method with 3 parameters which returns a connection objects. It is a static method.

String url = “jdbc:mysql://localhost:3306/sampledb”;

String username = “root”;

String password = ”root”;

Or

String url =”jdbc:oracle:thin:@localhost:1521:xe”;

String username = “system”;

String password =”manager”;

Conenction con = DriverManager.getConnection(url,username,password);

Statement: it is an interface ( we can create reference)

Int executeUpdate(String) //insert /update/delete

ResultSet executeQuery(String) // select

Connection:

Statement createStatement(); // Static Factory Method();

Interview Questions:

### 1. What are the steps to connect to the database in java?

The following steps are used in database connectivity.

* **Registering the driver class:**

The forName() method of the Class class is used to register the driver class. This method is used to load the driver class dynamically.

* Creating connection:

The getConnection() method of DriverManager class is used to establish the connection with the database

* Creating the statement:

The createStatement() method of Connection interface is used to create the Statement.

* Executing the queries:

The executeQuery() method of Statement interface is used to execute queries to the database. This method returns the object of ResultSet that can be used to get all the records of a table.

* Closing connection:

By closing connection, object statement and ResultSet will be closed automatically. The close() method of Connection interface is used to close the connection.

### 2. What are the JDBC API components?

### The java.sql package contains following interfaces and classes for JDBC API.

Interfaces:

* **Connection:** The Connection object is created by using getConnection() method of DriverManager class. DriverManager is the factory for connection.
* **Statement:** The Statement object is created by using createStatement() method of Connection class. The Connection interface is the factory for Statement.
* **PreparedStatement:** The PrepareStatement object is created by using prepareStatement() method of Connection class. It is used to execute the parameterized query.
* **ResultSet:** The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points before the first row. The executeQuery() method of Statement interface returns the ResultSet object.
* **ResultSetMetaData:** The object of ResultSetMetaData interface cotains the information about the data (table) such as numer of columns, column name, column type, etc. The getMetaData() method of ResultSet returns the object of ResultSetMetaData.
* **DatabaseMetaData:** DatabaseMetaData interface provides methods to get metadata of a database such as the database product name, database product version, driver name, name of the total number of tables, the name of the total number of views, etc. The getMetaData() method of Connection interface returns the object of DatabaseMetaData.
* **CallableStatement:** CallableStatement interface is used to call the stored procedures and functions. We can have business logic on the database through the use of stored procedures and functions that will make the performance better because these are precompiled. The prepareCall() method of Connection interface returns the instance of CallableStatement.

Classes:

* **DriverManager:** The DriverManager class acts as an interface between the user and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver. It contains several methods to keep the interaction between the user and drivers.
* **Blob:** Blob stands for the binary large object. It represents a collection of binary data stored as a single entity in the database management system.
* **Clob:** Clob stands for Character large object. It is a data type that is used by various database management systems to store character files. It is similar to Blob except for the difference that BLOB represent binary data such as images, audio and video files, etc. whereas Clob represents character stream data such as character files, etc.
* **SQLException** It is an Exception class which provides information on database access errors.

### 3. What are the differences between Statement and PreparedStatement interface?

Statement: The Statement interface provides methods to execute queries with the database. The statement interface is a factory of ResultSet; i.e., it provides the factory method to get the object of ResultSet. In the case of Statement, the query is compiled each time we run the program. In the case of Statement, the query is compiled each time we run the program. The Statement is mainly used in the case when we need to run the static query at runtime.

PreparedStatement: The PreparedStatement interface is a subinterface of Statement. It is used to execute the parameterized query. In the case of PreparedStatement, the query is compiled only once. PreparedStatement is used when we need to provide input parameters to the query at runtime.

### 4.  What are the differences between execute, executeQuery, and executeUpdate?

Execute: The execute method can be used for any SQL statements(Select and Update both). The execute method returns a boolean type value where true indicates that the ResultSet s returned which can later be extracted and false indicates that the integer or void value is returned.

executeQuery:The executeQuery method can be used only with the select statement. The executeQuery() method returns a ResultSet object which contains the data retrieved by the select statement.

ExecuteUpdate: The executeUpdate method can be used to update/delete/insert operations in the database. The executeUpdate() method returns an integer value representing the number of records affected where 0 indicates that the query returns nothing.

### 5. What are CLOB and BLOB data types in JDBC?

**BLOB:** Blob can be defined as the variable-length, binary large object which is used to hold the group of Binary data such as voice, images, and mixed media. It can hold up to 2GB of data on MySQL database and 128 GB on Oracle database. BLOB is supported by many databases such as MySQL, Oracle, and DB2 to store the binary data (images, video, audio, and mixed media).

**CLOB:** Clob can be defined as the variable-length, character-large object which is used to hold the character-based data such as files in many databases. It can hold up to 2 GB on MySQL database, and 128 GB on Oracle Database. A CLOB is considered as a character string.