1. **Introduction to JDBC**

Q1.WhatisJDBC(JavaDatabaseConnectivity)?

ANS :-

JDBC (Java Database Connectivity) is an API (Application Programming Interface) in Java that allows applications to interact with databases.

JDBC is a java API to connect and execute the query with the database.

It is a part of JavaSe (Java standard Edition).

There are four types of JDBC drivers :-

1) JDBC – ODBC Bridge driver

2) Native Driver

3) Network Protocol Driver

4) Thin driver

Q2.ImportanceofJDBCinJavaProgramming

ANS :-JDBC is essential in Java programming as it provides a **robust, secure, and efficient** way to connect Java applications with databases.

With JDBC, we can send SQL queries from Java code to store, update, delete, or get data from a database.

It is simple, secure, and works with many different types of databases.

Q3.JDBCArchitecture:DriverManager,Driver,Connection,Statement,andResult Set

- ➢ **Driver Manager** manages a list of database drivers and establishes connections.

- ➢ **Driver Manager** loads the JDBC driver and establishes a connection with the database.

- ➢ A JDBC **driver** acts as a bridge between the Java application and the database.

- ➢ Represents a **connection** between Java and the database.

- **Statements** Executes SQL queries and updates the database.

- **Result set** stores the results of SQL queries and allows iteration through the data.

2. JDBCDriverTypes
Q1.OverviewofJDBCDriverTypes:

- Type1:JDBC-ODBCBridge Driver
  - Uses **ODBC (Open Database Connectivity)** driver to communicate with databases.
  - The JDBC-ODBC driver uses an ODBC driver to connect to the database.
  - The JDBC-ODBC bridge driver converts JDBC methods calls into the ODBC function calls.
  - Oracle does not support the JDBC-ODBC bridge from java 8.
- Type2:Native-APIDriver
  - Uses **database-specific native libraries** (DLL files) to interact with the database.
  - The Native-API driver uses the client side libraries of the database.
  - The driver converts JDBC method calls into native calls of the database API.
- Type3:NetworkProtocolDriver
  - Uses a **middleware server** to convert JDBC calls into database-specific protocol calls.
- Type4:Thin Driver
  - Also called the **"Direct-to-Database" driver**.
  - Written **entirely in Java** and directly connects to the database **without middleware**.

Q2.ComparisonandUsageofEachDriverType

- **JDBC-ODBC Bridge Driver**
  - ✓ Working with **legacy** systems that require ODBC.
  - ✓ Not platform Independent.
  - ✓ Poor Performance
  - ✓ Low Security
  - ✓ Easy to use
  - ✓ Can be easily connected to any database.
- **Native-API Driver**
  - ✓ Application needs **database-specific features**.
  - ✓ Not platform Independent
  - ✓ Moderate Performance
  - ✓ Moderate Security
  - ✓ Performance upgraded than JDBC-ODBC bridge driver
- **Network Protocol Driver**
  - ✓ A **middleware server** is available to manage database connections.
  - ✓ Platform Independent

✓ Moderate Performance
✓ Moderate Security

➢ **Thin Driver**
✓ Using **Modern applications** (web, enterprise, cloud-based).
✓ Platform Independent (pure java)
✓ Excellent Performance
✓ High Security

3. StepsforCreatingJDBCConnections
Q1.Step-by-StepProcesstoEstablishaJDBCConnection:

1. ImporttheJDBCpackages
   • Import necessary JDBC classes from the java.sql package.
2. RegistertheJDBCdriver
   • Load the JDBC driver for the specific database you are using.
   • Class.forName("com.mysql.cj.jdbc.Driver");
3. Openaconnectiontothe database
   • Use DriverManager.getConnection() to establish a connection to the database.
   • Connection cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/java","root","");
4. Createastatement
   • A Statement object is used to execute SQL queries.
   • Three types of statements:
     ✓ **Statement**
     ✓ **PreparedStatement**
     ✓ **CallableStatement**
5. ExecuteSQLqueries
   • Use **executeQuery**() for SELECT statements (returns a **ResultSet**).
   • Use **executeUpdate**() for INSERT, UPDATE, and DELETE (returns an integer indicating affected rows).
6. Processtheresultset
   • Iterate through the ResultSet to fetch data.
7. Closetheconnection
   • Always **close** the ResultSet, Statement, and Connection.

4. TypesofJDBCStatements
Q1.Overview of JDBC Statements:

Statement:

✓ The statement provides methods to execute queries with the database.
✓ The statement interface is a factory of resultset.
✓ ExecutesimpleSQLquerieswithoutparameters.
✓ A Statement is used to execute **static SQL queries** that do not require input.

> Class.*forname(com.mysql.jdbc.driver);*
> Connection conn =DriverManager.getConnection(URL, USERNAME, PASSWORD);

```
Statement stmt = conn.createStatement();
ResultSetrs = stmt.executeQuery("SELECT * FROM employees");

while (rs.next()) {
System.out.println("ID: " + rs.getInt("id"));
}
```

Parameters :

➢ **PreparedStatement:**
- ✓ The preparedStatement is a sub interface of Statement.
- ✓ It is used to execute parameterized query.
- ✓ PrecompiledSQLstatementsforquerieswith parameters.
- ✓ Instead of Statement, use PreparedStatement for **better performance and security**.

➢ **CallableStatement:**
- ✓ Usedtocallstoredprocedures.

5. **JDBC CRUD Operations(Insert,Update,Select,Delete)**

➢ **Insert:**
- o Adding anewrecordtothedatabase.
- o With executeUpdate("SQL_QUERY") we insert the data.

➢ **Update:**
- o Modifyingexistingrecords.
- o Same as we had insert data with executeUpdate("SQL_QUERY") for updatation of data.

➢ **Select:**
- o Retrievingrecordsfromthedatabase.
- o While retrieving records from database we store the result in ResultSet.
- o ResultSetrs = stmt.executeQuery("SQL_QUERY");

➢ **Delete:**
- o Removingrecordsfromthedatabase.
- o Same as we used executeUpdate("SQL_QUERY") for deletion also.

6. **ResultSetInterface**
Q1.WhatisResultSetinJDBC?

➢ Result Set is an object that holds the results of a SQL query executed using a Statement or Prepared Statement.
➢ It provides a way to access and manipulate the data returned from the database.
➢ The object of resultset maintains a cursor pointing to a row of a tabl.
➢ Initially, the cursor points to the first row.

Q2.NavigatingthroughResultSet(first,last,next,previous)

➢ The Result Set interface provides several methods to navigate through the result set:
- ✓ **first():** Moves the cursor to the first row in the result set.

- ✓ **last():** Moves the cursor to the last row in the result set.
- ✓ **next():** Moves the cursor to the one row next from the current position in the result set.
- ✓ **previous():** Moves the cursor to the previous row in the result set.

Q3.Working with ResultSet to retrieve data from SQL queries

Use a Statement or Prepared Statement to execute the SQL query.

- ➢ Get the Result Set object from the Statement or Prepared Statement.
- ➢ Use navigation methods like next(), first(), last(), and previous() to move through the result set.
- ➢ Use getter methods like get String(), getInt(), and get Date() to retrieve data from the result set.

7. DatabaseMetadata
Q1.What is Database MetaData?

ANS :-An interface in JDBC that provides information about the database.

DatabaseMetaData is an **interface in JDBC** that provides detailed information about the **database** and its capabilities.

Q2.Importance of Database Metadata in JDBC

ANS :-Provides crucial information about the database structure schema.

Q3.MethodsprovidedbyDatabaseMetaData(getDatabaseProductName,getTables, etc.)

- ➢ **getDatabaseProductName():** database product name.
- ➢ **getDriverName**(): Returns the JDBC driver name.
- ➢ **get Tables():** Returns a Result Set containing information about the database tables.
- ➢ **getColumns()**: Returns a Result Set containing information about the columns in a table.
- ➢ **getPrimaryKeys():** Returns a Result Set containing information about the primary keys in a table.

8. ResultSetMetadata
Q1.What is ResultSet MetaData?

It provides information about the structure and properties of a Result Set, such as column names, data types, and column counts.

Q2.MethodsinResultSetMetaData(getColumnCount,getColumnName, getColumnType)

getColumnCount(): Returns the number of columns in the Result Set.

getColumnName(): Returns the name of a specific column.

getColumnType(): Returns the data type of a specific column.

9. PracticalSQLQueryExamples
Q1.Write SQL queries for:

- o Insertingarecordintoa table.
  - INSERT INTO table name (column1, column2, column3) VALUES ('value1', 'value2', 'value3');

- o Updatingspecificfieldsofarecord.
  - UPDATE table name SET column1 = 'new_value1', column2 = 'new_value2' WHERE condition;

- o Selectingrecordsbasedoncertain conditions.
  - SELECT column1, column2 FROM table name WHERE condition;

- o Deletingspecificrecords.
  - DELETE FROM table name WHERE condition;
  -

10. PracticalExample1:SwingGUIforCRUDOperations
Q1.Introduction to Java Swing for GUI development

ANS :-GUI toolkit for Java that provides components and tools for building desktop applications with graphical user interfaces.

Q2.How to integrate Swing component swith JDBC for CRUD operations

- o Create a Swing Frame
- o Add Swing Components
- o Establish a Database Connection
- o Perform CRUD Operations
- o Display Data
- o Handle Button Events